Lab 5 – Seven Segment Display

Purpose

The aim of this lab is to get students along with the seven segment display of the Basys 3. This is done by displaying hexadecimal numbers in increasing form, by changing the value of anodes of the display and using the phenomenon persistence of vision.

Design

Two inputs and two outputs are used in this lab, one for restarting from the leftmost FPGA switch and one for counting process called clock from 100 MHz internal frequency. One of the outputs saves the number to be displayed and the other one is for picking the anode to turn off.

Questions on the Lab Manual

1) It is 100 MHz, however the outputted frequency can be changed.
2) Producing a slower clock is doable by changing the refreshing signal's frequency to become higher as this means that the outputted value's frequency changes.
3) This is only possible by creating a new value in terms of 10/2^n MHz, n being an integer.

Methodology, Results

After creating the input and outputs on the port section I created a signal called 'ledd' that saves the value of which number in the display to be changed, also to pick 'anodeactivator'. 'refresh_counter' is of 20 bits and its most significant 2 bits are used for the phenomenon of persistence of vision, the change of the MSB bits can be observed well with the human eye. 'led_count' which is this 2 bits picks which of the number sections of the seven segment display to change and uses the necessary 4 bits of the 16 bit signal 'value'. A specific decoder is built in order to find out the 'anodeactivator' from 'value', and a multiplexer is prepared to pick the outputted digits of 'output'. The LEDs work with the principle of zeros turning on the desired parts of the seven segments, as the parts are anodes and their cathodes are assigned with the value '1' and a voltage difference is being required. The counting system resets itself as the 'restarter' becomes '1' and displays '0'.

I later created the RTL Schematics to observes the mux with the whole mechanism. I then wrote a testbench to observe a simulation.

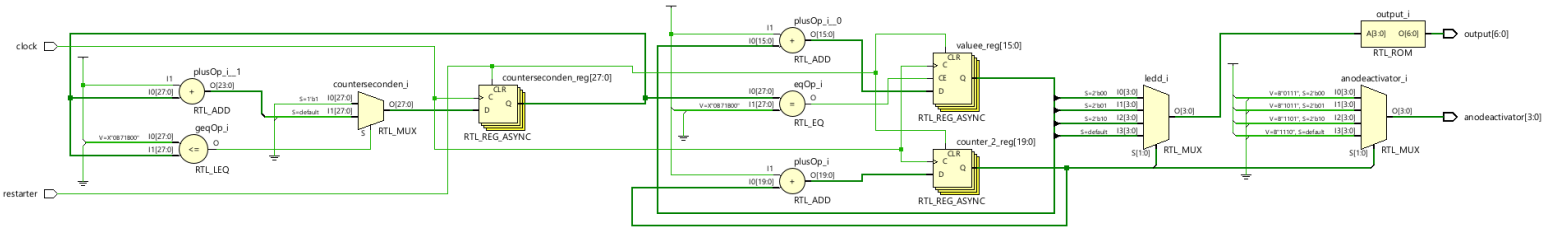

Figure 1: RTL Schematics

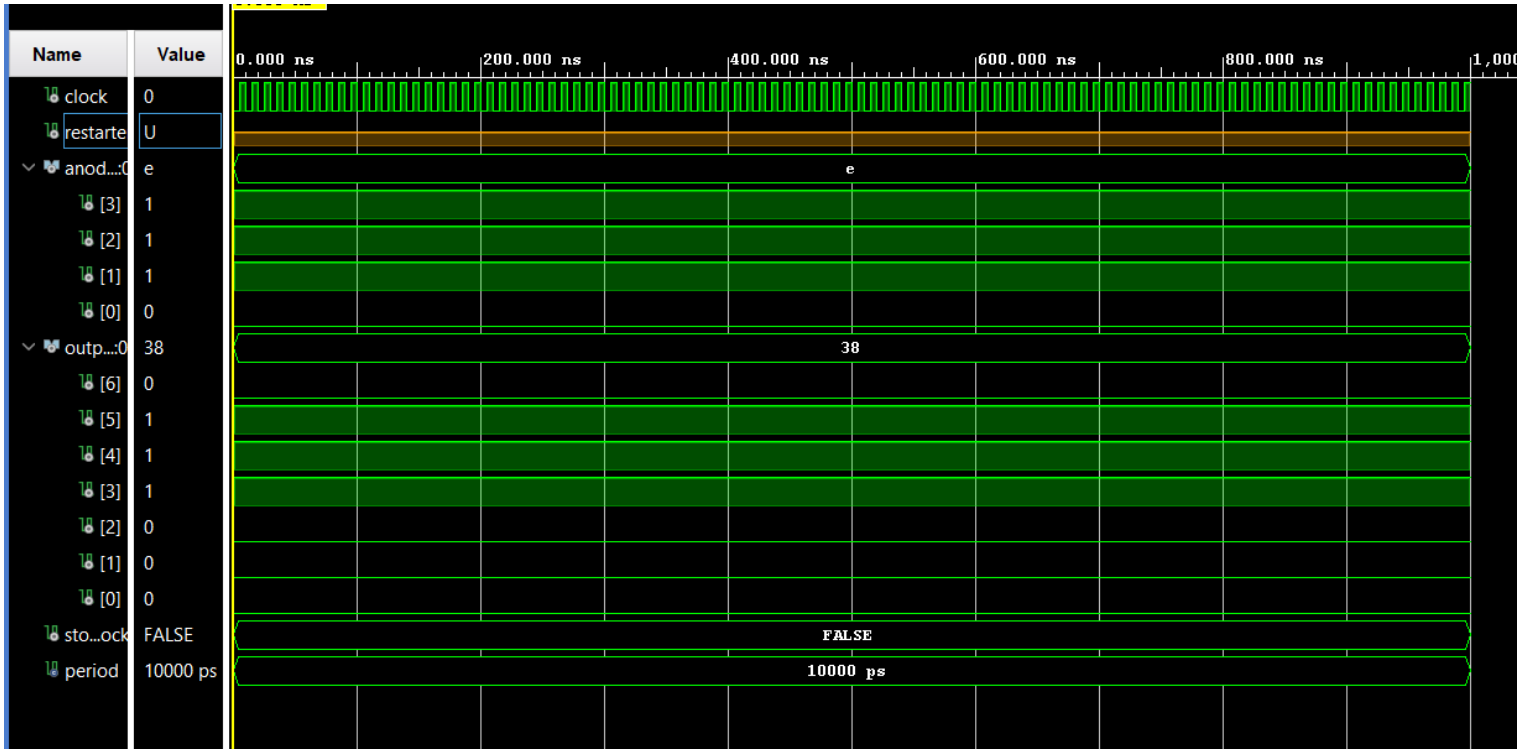

Figure 2: Simulation

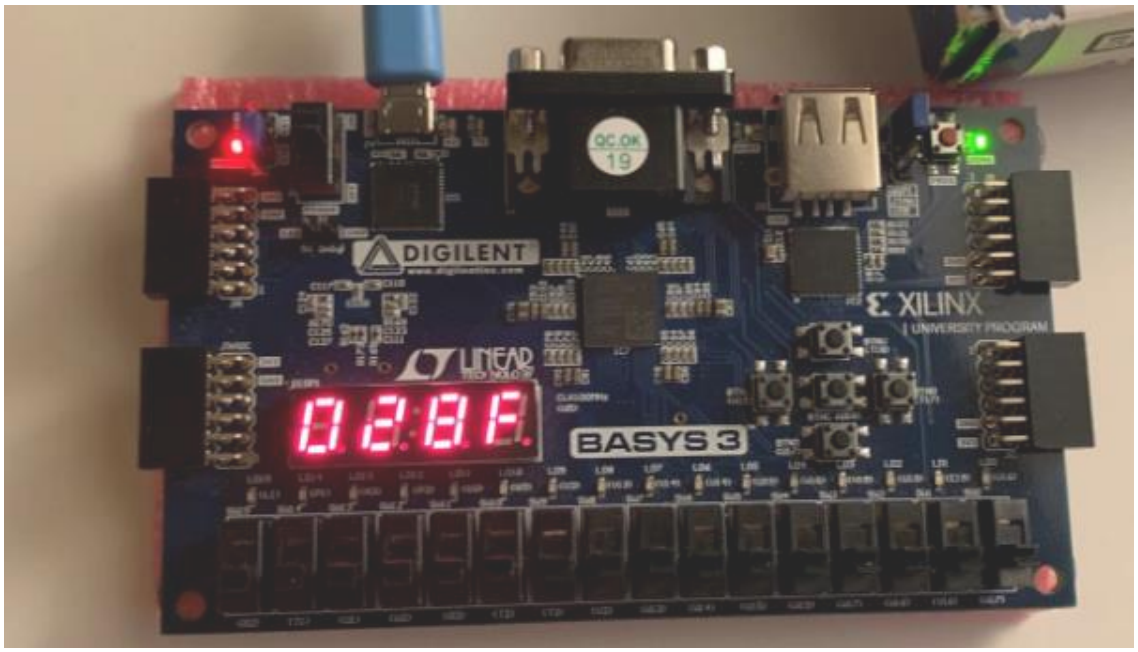

Figure 3: Beginning the Counting Process

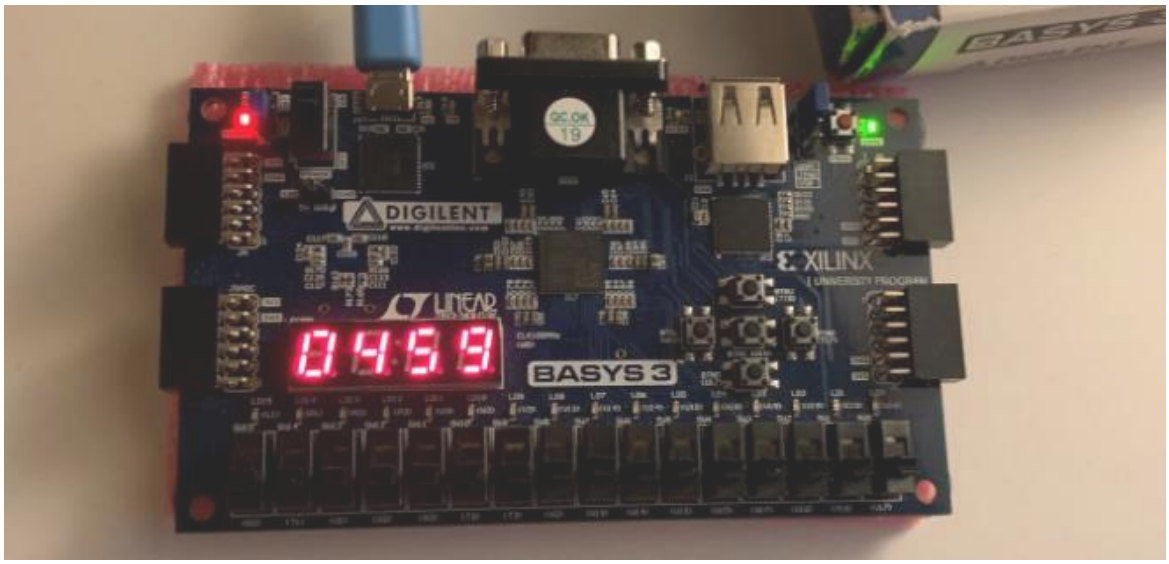

Figure 4: Increasing in the Value

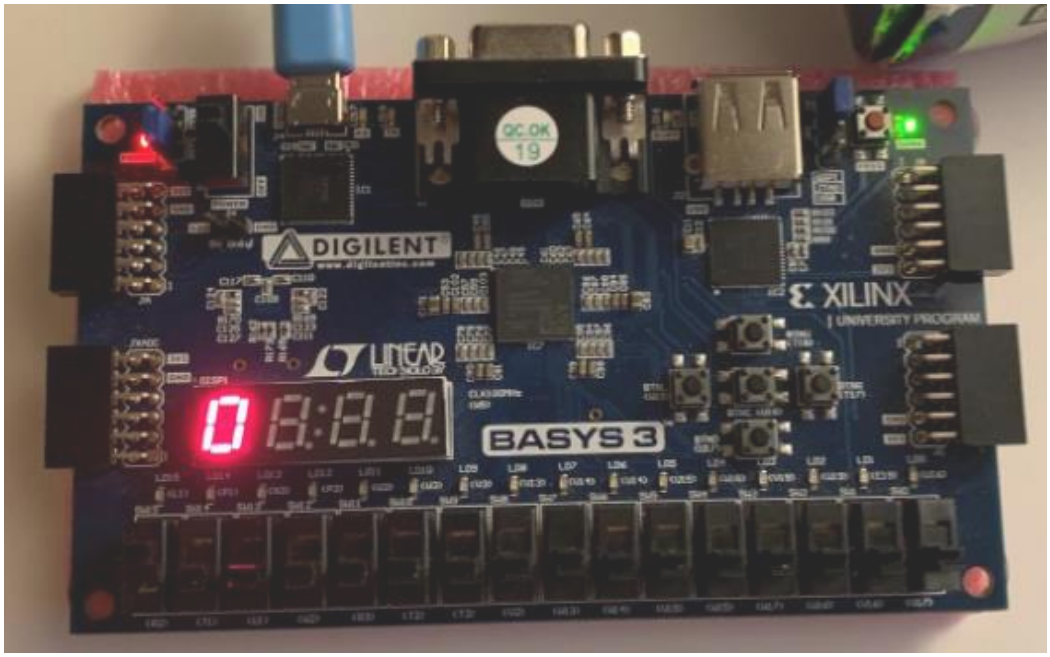

Figure 5: Reset

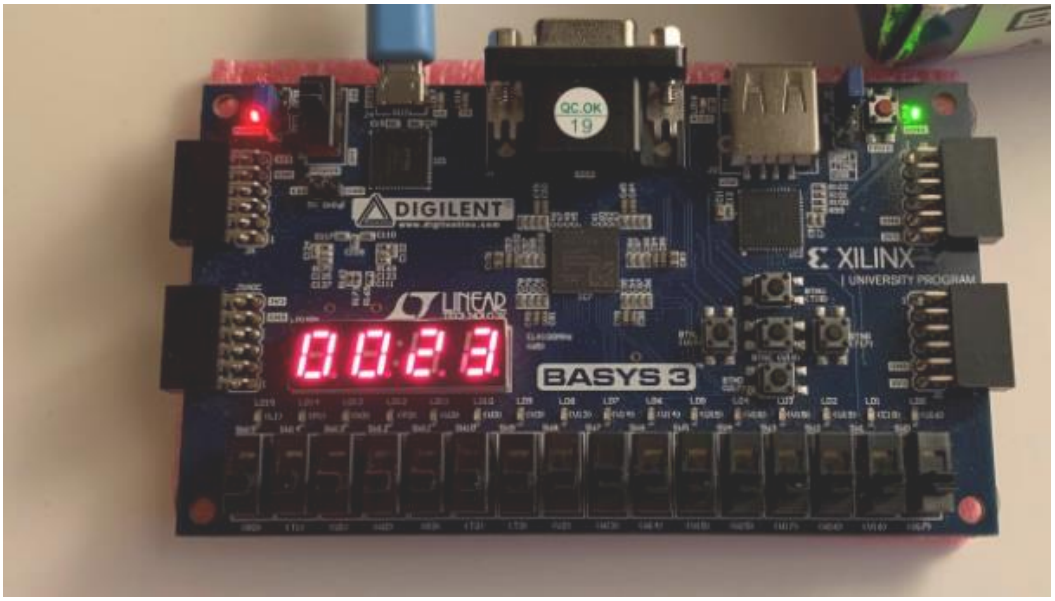

Figure 6: Counting Continues

Conclusion

The purpose of this lab session was to get students along with the seven segment display and clocks, using counters and be more into decoders and multiplexers. I had to research online to understand counters and seven segment display, and had to understand clock dividers truthfully before continuing. I had to analyze code online and used a reset for the first time. I also used Boolean values for the first time on my test bench. While implementing the code on Basys 3, I saw the value 8888, meaning I had a problem with my frequency as the numbers were increasing way too quickly. This lab was useful in terms of learning new sides of digital design, clock mechanisms and the seven segment display, also the persistence of vision phenomenon was a new eye opening reality for me. I also improved in researching capabilities in this lab.

Appendices

ssds.vhd

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

entity ssds is
   Port(
      clock: in STD_LOGIC;
      restarter: in STD_LOGIC;
      anodeactivator: out STD_LOGIC_VECTOR (3 downto 0);
      output: out STD_LOGIC_VECTOR (6 downto 0));
end ssds;

architecture Behavioral of ssds is

signal counterseconden: STD_LOGIC_VECTOR (27 downto 0);
```

```
signal counter_1sec_enable: std_logic;
signal ledd: STD_LOGIC_VECTOR (3 downto 0);
signal counter_2: STD_LOGIC_VECTOR (19 downto 0);
signal ledcount: std_logic_vector(1 downto 0);
signal valuee: STD_LOGIC_VECTOR (15 downto 0);

begin
process(ledd)
begin
   case ledd is
      when "0000" => output <= "0000001";
      when "0001" => output <= "1001111";
      when "0010" => output <= "0010010";
      when "0011" => output <= "0000110";
      when "0100" => output <= "1001100";
      when "0101" => output <= "0100100";
      when "0110" => output <= "0100000";
      when "0111" => output <= "0001111";
      when "1000" => output <= "0000000";
      when "1001" => output <= "0000100";
      when "1010" => output <= "0000010";
      -- a
      when "1011" => output <= "1100000";
      -- b
      when "1100" => output <= "0110001";
      -- c
      when "1101" => output <= "1000010";
      -- d
      when "1110" => output <= "0110000";
```

```
        -- E
        when others => output <= "0111000";
        -- F
    end case;
end process;

process(clock,restarter)
begin

if(restarter='1') then
    counter_2 <= (others => '0');
elsif(rising_edge(clock)) then
    counter_2 <= counter_2 + 1;
end if;
end process;

ledcount <= counter_2(19 downto 18);

process(ledcount)
begin
    case ledcount is
        when "00" =>
        anodeactivator <= "0111";
        ledd <= valuee(15 downto 12);
        when "01" =>
        anodeactivator <= "1011";
        ledd <= valuee(11 downto 8);
        when "10" =>
        anodeactivator <= "1101";
```

```
        ledd <= valuee(7 downto 4);
        when others =>
        anodeactivator <= "1110";
        ledd <= valuee(3 downto 0);
    end case;
end process;



process(clock, restarter)
begin
if(restarter='1') then
    counterseconden <= (others => '0');
elsif(rising_edge(clock)) then
    if(counterseconden >= x"B71B00") then
        counterseconden <= (others => '0');
    else
        counterseconden <= counterseconden + "0000001";
    end if;
end if;
end process;



counter_1sec_enable <= '1' when counterseconden = x"B71B00" else '0';


process(clock, restarter)
begin
if(restarter = '1') then
    valuee <= (others => '0');
elsif(rising_edge(clock)) then
```

```
    if(counter_1sec_enable = '1') then
    valuee <= valuee + x"0001";
    end if;
end if;
end process;

end Behavioral;
```

ssds.tb

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;
entity seventb is
end;

architecture bench of seventb is
component ssds
    Port(
        clock: in STD_LOGIC;
        restarter: in STD_LOGIC;
        anodeactivator: out STD_LOGIC_VECTOR (3 downto 0);
        output: out STD_LOGIC_VECTOR (6 downto 0));
end component;

signal clock: STD_LOGIC;
signal restarter: std_logic;
signal anodeactivator: std_logic_vector (3 downto 0);
signal output: std_logic_vector (6 downto 0) ;
```

```
constant period: time := 10 ns;
signal stop_clock: boolean;
begin

seven_func: ssds port map(
clock => clock,
restarter => restarter,
anodeactivator => anodeactivator,
output => output);
work_clock: process

begin
stop_clock <= false;
while not stop_clock loop
clock <= '0', '1' after period/2;

wait for period;
end loop;
wait;
end process;
end;
```

pins.xdc

```
set_property PACKAGE_PIN W5 [get_ports clock]
   set_property IOSTANDARD LVCMOS33 [get_ports clock]
set_property PACKAGE_PIN R2 [get_porestarterrestarter]
   set_property IOSTANDARD LVCMOS33 [grestarterorts restarter]
set_property PACKAGE_PIN W7 [get_ports {output[6]}]
```

```
   set_property IOSTANDARD LVCMOS33 [get_ports {output[6]}]
set_property PACKAGE_PIN W6 [get_ports {output[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {output[5]}]
set_property PACKAGE_PIN U8 [get_ports {output[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {output[4]}]
set_property PACKAGE_PIN V8 [get_ports {output[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {output[3]}]
set_property PACKAGE_PIN U5 [get_ports {output[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {output[2]}]
set_property PACKAGE_PIN V5 [get_ports {output[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {output[1]}]
set_property PACKAGE_PIN U7 [get_ports {output[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {output[0]}]
set_property PACKAGE_PIN U2 [get_ports {anodeactivator[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anodeactivator[0]}]
set_property PACKAGE_PIN U4 [get_ports {anodeactivator[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anodeactivator[1]}]
set_property PACKAGE_PIN V4 [get_ports {anodeactivator[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anodeactivator[2]}]
set_property PACKAGE_PIN W4 [get_ports {anodeactivator[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anodeactivator[3]}]
```

Works Cited

1 - https://www.fpga4student.com/2017/09/seven-segment-led-display-controller-basys3-fpga.html

2 - https://www.fpga4student.com/2017/09/vhdl-code-for-seven-segment-display.html