Ege Ozan Özyedek
EEE102 – 02
LabWork 05
Lab Report

- Design Methodology

  I set out to create a sort of calculator in this week's lab. This calculator gets two 6-bit binary inputs and adds or subtracts them (which the user can decide with a switch) from each other, then shows the calculation in its octal value on the 7-segment display on BASYS 3. This is achieved by using 5 modules, the details of these modules can be found below.

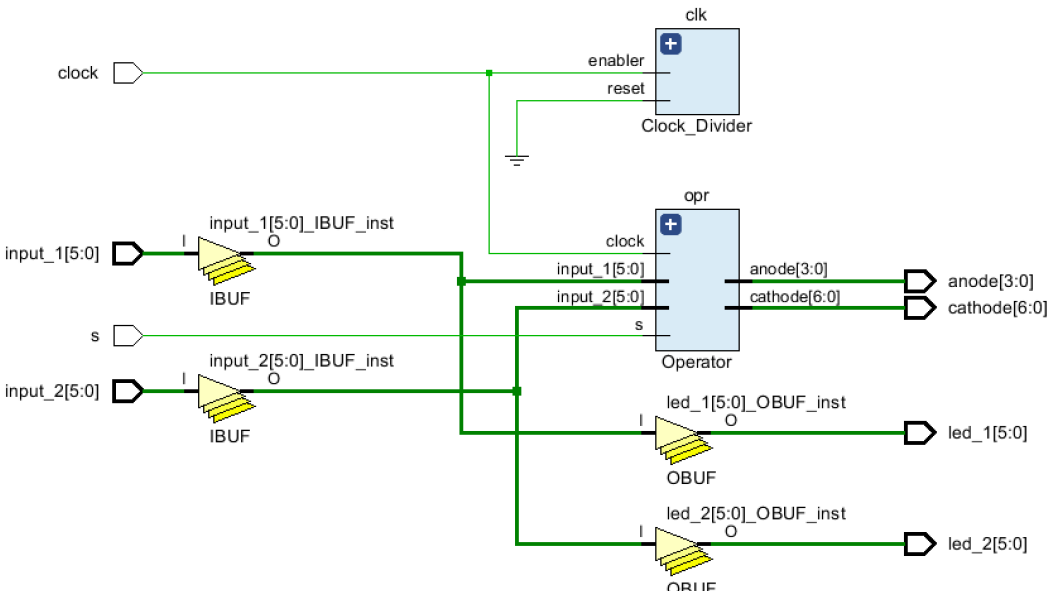

Figure 1: The RTL Schematic

  - Main (Top Module)
    This module is the top module, hence there are no operations inside it. It uses two components (Operator and Clock_Divider) and makes everything work. Its inputs and outputs are:
    - input_1: 6-bit binary input that are obtained from the first 6 switches of BASYS 3
    - input_2: 6-bit binary input that are obtained from the last 6 switches of BASYS 3
    - led_1 and led_2: outputs to shows the values of switches on the LED's on top of them.
    - s: the operation switches, if '1' the operation is subtraction otherwise its summation
    - anode: the 4-bit anode output, determines the display which lights up on the 7-segment display
    - cathode: the 7-bit cathode output which determines the value or sign that will show on the display

  - Operator
    This module is where all operations happen. In this operation, the values of the outputs from summation and subtraction turn into their octal values with

the use of decoders. This module is also where the anode and cathode "sync" happens, the anode and the cathode value that correspond to each other gets coupled and shows the value on the 7-segment display. The inputs of this module are the same as the Main module (except for the LED outputs). The module uses 2 types of components, one of them being Decoder (4 of them) and the other being Arithmetic_Operator.

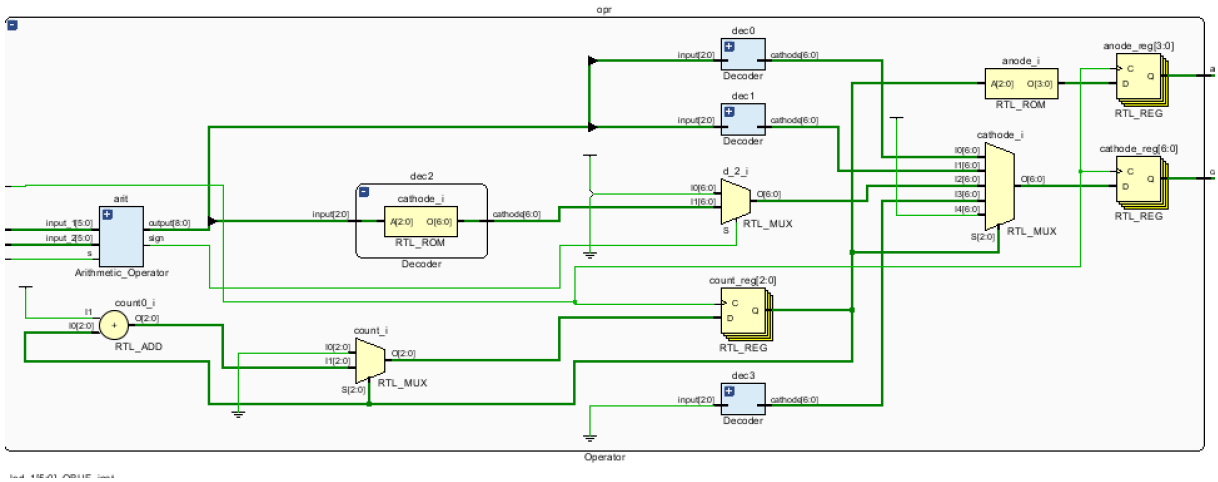

Figure 2: The RTL Schematic of Operator Module

- Decoder
  The Decoder is used as a way to give the correct value to the cathode output. It has a case statement; its input is a 3-bit binary value and its output is the 7-bit cathode value.

- Clock_Divider
  The clock divider enables the program to change the input signal into another output signal with a different frequency. This is used to change the frequency to one that the programmer finds fitting to the program.

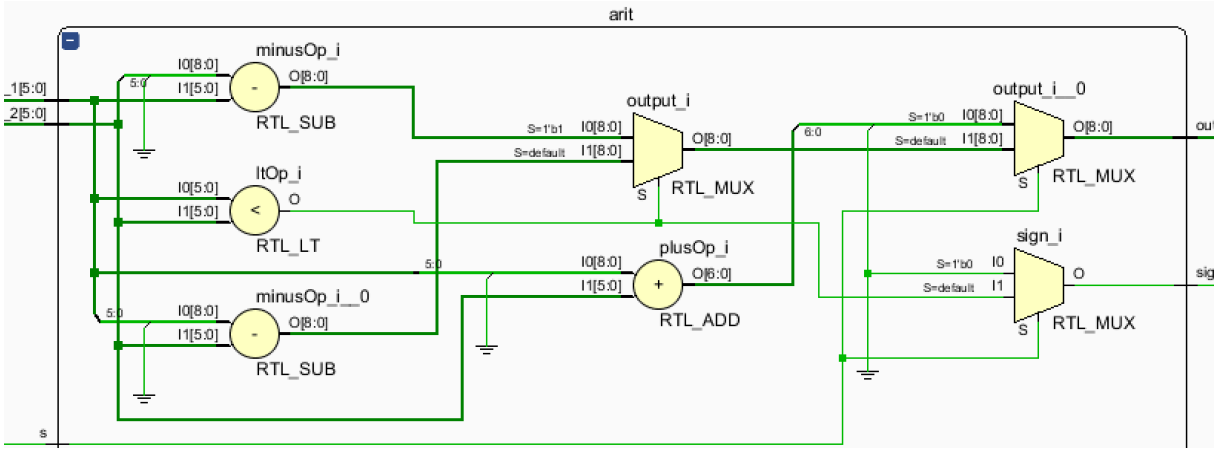

Figure 3: The RTL Schematic of Arithmetic_Operator Module

- Arithmetic_Operator
  This module is where all the arithmetic operations take place. It inputs the two 6-bit inputs that are assigned on the top module and also gets the operator switcher s from there too. It outputs a 9-bit vector, its 9-bits to easily divide the output to 3 groups and convert them to their cathode values. The module uses operators (+ and -) to sum and subtract the two binary inputs, these operators are defined in the std_logic_arith library.

- Results

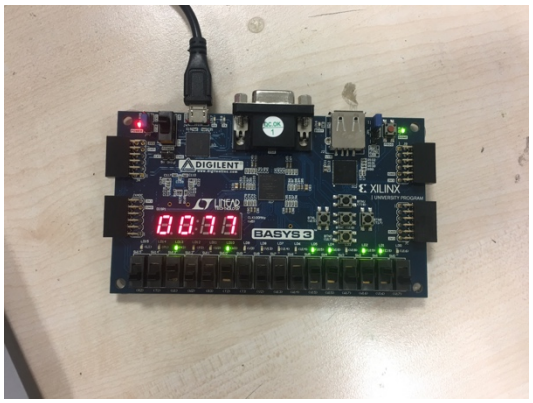

Figure 4: $(001001)_2 + (110110)_2 = (77)_8$

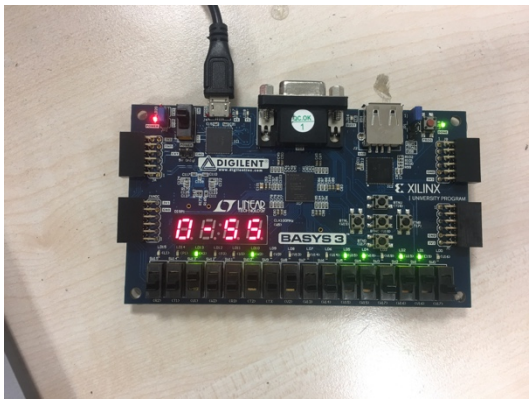

Figure 5: $(001001)_2 - (110110)_2 = (-55)_8$

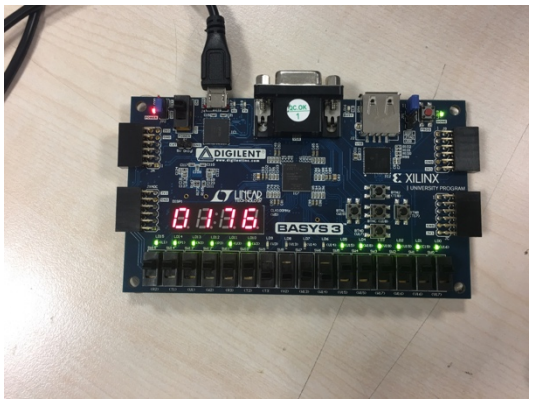

Figure 6: $(111111)_2 + (111111)_2 = (176)_8$

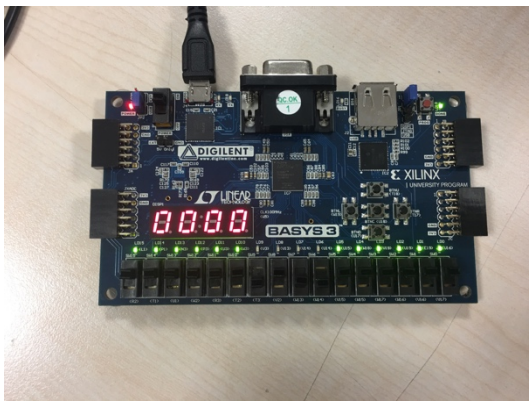

Figure 7: $(111111)_2 - (111111)_2 = (0)_8$

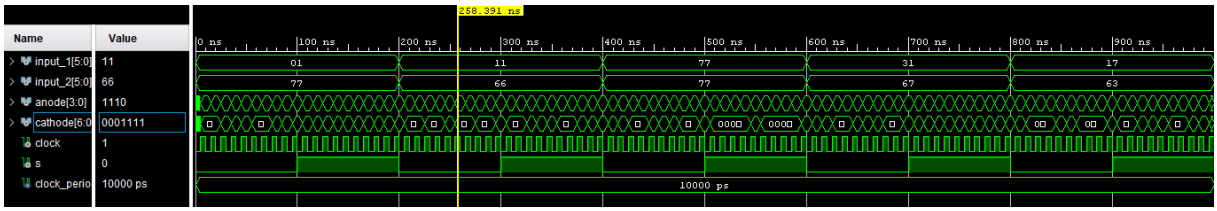

Figure 5: Test Bench zoomed out

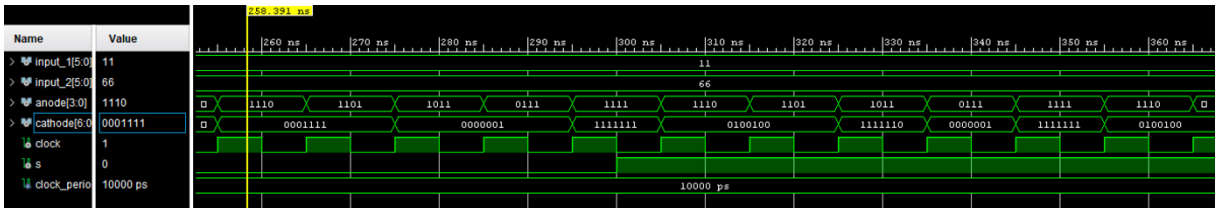

Figure 6: Test Bench zoomed in, octal values 11 and 66 subtraction and addition

The results of the experiment can be seen above, both on the test bench and on BASYS 3. The input values can be observed via the LED's above the switches. Thankfully, there seems to be no errors on both the test bench and BASYS 3 itself. The 7-segment display shows the value in its octal value and the inputs are observed and processed without any complications.

- Discussion and Conclusion

The RTL schematic is pretty different from the one I designed on the preliminary report. I did kind of expect this, mostly because while coding some of my initial logic changed but also because there are many different operators on VHDL that I did not quite think of beforehand. For example, there are operands for if and else, which were not on my initial schematic. However, my initial schematic of the general circuit is similar to the general RTL schematic. This is because there are not many operations on the top module so it's much more easier to guess how the circuit design might be.

The FPGA that we are using, BASYS 3, does have hardware limitations. For now this is not a big issue, but as we progress further into our projects it might be. From my observation, for this lab I utilized 3% of the FPGA. For such few operations, this seems like a huge number and this may limit bigger projects.

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	51	20800	0.25
FF	29	41600	0.07
IO	37	106	34.91
BUFG	1	32	3.13

Figure 7: Post Implementation Utilization

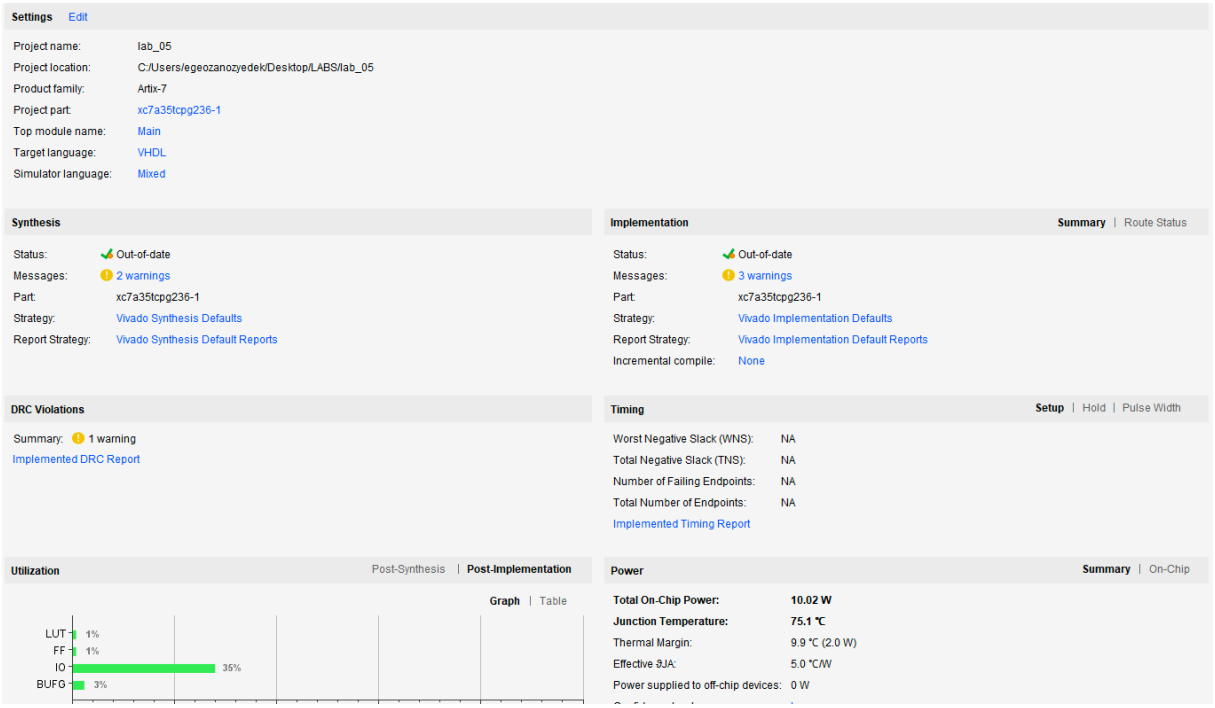

Figure 8: Project Summary

This lab, in conclusion, has helped me to increase my knowledge and understanding in how the 7-segment display, switches and other components on BASYS 3 work. This is beneficial for me since I will be using most of these components for the project, mostly the switches and the display. While creating this lab I did encounter some problems, I struggled with using the arithmetic library at first and couldn't represent negative values properly. I also accidentally forgot to reset the counter on the clock divider which caused the 7-segment display to show every input on top of each other. Thankfully, I was able to fix these issues and finish the lab properly.

- Appendix
  - **VHDL code for Main (Top Module)**

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

entity Main is
    Port ( input_1 : in STD_LOGIC_VECTOR (5 downto 0);
           input_2 : in STD_LOGIC_VECTOR (5 downto 0);
           led_1 : out STD_LOGIC_VECTOR (5 downto 0);
           led_2 : out STD_LOGIC_VECTOR (5 downto 0);
           s : in STD_LOGIC;
           clock : in STD_LOGIC;
           anode : out STD_LOGIC_VECTOR (3 downto 0);
           cathode : out STD_LOGIC_VECTOR (6 downto 0));
end Main;
```

```
architecture Behavioral of Main is

component Operator is
Port ( input_1 : in STD_LOGIC_VECTOR (5 downto 0);
     input_2 : in STD_LOGIC_VECTOR (5 downto 0);
     s : in STD_LOGIC;
     clock : in STD_LOGIC;
     anode : out STD_LOGIC_VECTOR (3 downto 0);
     cathode : out STD_LOGIC_VECTOR (6 downto 0));
end component;

component Clock_Divider is
Port ( enabler : in STD_LOGIC;
     reset : in STD_LOGIC;
     clock : out STD_LOGIC);
end component;

signal clk_out : STD_LOGIC:= '0';

begin

clk: Clock_Divider port map (clock, '0', clk_out);
opr: Operator port map (input_1, input_2, s, clock, anode, cathode);

process (input_1, input_2) --process which enables me to set the led's above the
switches to light up accordingly
begin
   for i in 0 to 5 loop
      led_1(i) <= input_1(i);
      led_2(i) <= input_2(i);
   end loop;
end process;

end Behavioral;
```

- **VHDL code for Operator**

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

entity Operator is
   Port ( input_1 : in STD_LOGIC_VECTOR (5 downto 0);
        input_2 : in STD_LOGIC_VECTOR (5 downto 0);
        s : in STD_LOGIC;
        clock : in STD_LOGIC;
        anode : out STD_LOGIC_VECTOR (3 downto 0);
        cathode : out STD_LOGIC_VECTOR (6 downto 0));
end Operator;

architecture Behavioral of Operator is
```

Ege Ozan Özyedek
EEE102 – 02
LabWork 05
Lab Report

```
component Decoder is port ( input : in STD_LOGIC_VECTOR (2 downto 0);
        cathode : out STD_LOGIC_VECTOR (6 downto 0));
        end component;

component Arithmetic_Operator is port ( input_1 : in STD_LOGIC_VECTOR (5 downto 0);
        input_2 : in STD_LOGIC_VECTOR (5 downto 0);
        s : in STD_LOGIC;
        output : out STD_LOGIC_VECTOR (8 downto 0);
        sign : out STD_LOGIC);
        end component;

signal sign: std_logic;
signal d_0: std_logic_vector (6 downto 0);
signal d_1: std_logic_vector (6 downto 0);
signal d_2: std_logic_vector (6 downto 0);
signal d_2s: std_logic_vector (6 downto 0);
signal d_3: std_logic_vector (6 downto 0);
signal output: std_logic_vector (8 downto 0);
signal count: integer range 0 to 4;

begin

arit : arithmetic_operator port map (input_1, input_2, s, output, sign);
dec0 : decoder port map ( output(2 downto 0) ,d_0);
dec1 : decoder port map ( output(5 downto 3) ,d_1);
dec2 : decoder port map ( output(8 downto 6) ,d_2s);
dec3 : decoder port map ( "000" ,d_3);

process (sign)
begin
   if (sign = '1') then
      d_2 <= "1111110";
      else
      d_2 <= d_2s;
   end if;
end process;

process( clock)
begin

if (rising_edge(clock)) then

   case count is
      when 0 => anode <= "1110";
            cathode <= d_0;
      when 1 => anode <= "1101";
            cathode <= d_1;
      when 2 => anode <= "1011";
            cathode <= d_2;
      when 3 => anode <= "0111";
```

```
            cathode <= d_3;
      when others => anode <= "1111";
                  cathode <= "1111111";
   end case;

   count <= count + 1;
   if (count = 4) then

      count <= 0;

    end if;

end if;
end process;

end Behavioral;
```

- **VHDL code for Decoder**

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

entity Decoder is
   Port ( input : in STD_LOGIC_VECTOR (2 downto 0);
        cathode : out STD_LOGIC_VECTOR (6 downto 0));
end Decoder;

architecture Behavioral of Decoder is

begin

with input select cathode  <=
   "0000001" when "000",
   "1001111" when "001",
   "0010010" when "010",
   "0000110" when "011",
   "1001100" when "100",
   "0100100" when "101",
   "0100000" when "110",
   "0001111" when "111",
   "0111000" when others;

end Behavioral;
```

- **VHDL code for Clock_Divider**

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

entity Clock_Divider is
```

```
    Port ( enabler : in STD_LOGIC;
           reset : in STD_LOGIC;
           clock : out STD_LOGIC);
end Clock_Divider;

architecture Behavioral of Clock_Divider is

signal out_signal: STD_LOGIC;
signal counter : integer range 0 to 9999:= 0;

begin

process( enabler, reset)
begin

   if (reset = '1') then
      out_signal <= '0';
      counter <= 0;

   elsif rising_edge(enabler) then
      if (counter = 9999) then
         out_signal <= not out_signal;
         counter <= 0;
      else
         counter <= counter + 1;
      end if;
    end if;
end process;

clock <= out_signal;

end Behavioral;
```

- **VHDL code for Arithmetic_Operator**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
--use IEEE.Numeric_std.ALL;


entity Arithmetic_Operator is
    Port ( input_1 : in STD_LOGIC_VECTOR (5 downto 0);
           input_2 : in STD_LOGIC_VECTOR (5 downto 0);
           s : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (8 downto 0);
           sign : out STD_LOGIC);
end Arithmetic_Operator;

architecture Behavioral of Arithmetic_Operator is
```

```
begin

process (input_1, input_2, s)
begin
   if ( s = '0') then
      sign <= '0';
      output <= "000" & unsigned( input_1) + unsigned( input_2);
   else
      if ( unsigned(input_1) < unsigned(input_2)) then
         sign <= '1';
         output <= "000" & unsigned( input_2) - unsigned( input_1);
      else
      sign <= '0';
         output <= "000" & unsigned( input_1) - unsigned( input_2);
      end if;
   end if;
end process;

end Behavioral;
```

- **VHDL code for the Test Bench**

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

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is

component Main is
 Port ( input_1, input_2 : in STD_LOGIC_VECTOR (5 downto 0);
       s : in STD_LOGIC;
       clock : in STD_LOGIC;
       anode : out STD_LOGIC_VECTOR (3 downto 0);
       cathode : out STD_LOGIC_VECTOR (6 downto 0));
       end component;

signal input_1: STD_LOGIC_VECTOR (5 downto 0);
signal input_2: STD_LOGIC_VECTOR (5 downto 0);
signal anode: STD_LOGIC_VECTOR (3 downto 0);
signal cathode: STD_LOGIC_VECTOR (6 downto 0);
signal clock: STD_LOGIC:= '0';
signal s: STD_LOGIC:= '0';
constant clock_period: time:= 10ns;


begin
```

```
uut: Main port map ( input_1 => input_1, input_2 => input_2, s => s, clock => clock,
anode => anode, cathode => cathode);

clock_process :process
begin

clock <= '0';
wait for clock_period/2;
clock <= '1';
wait for clock_period/2;
end process;

switch_process: process
begin

s <= '0';
wait for clock_period*10;
s <= '1';
wait for clock_period*10;
end process;

stimulus_process: process
begin

input_1 <= "000001";
input_2 <= "111111";
wait for clock_period*10;
input_1 <= "000001";
input_2 <= "111111";
wait for clock_period*10;

input_1 <= "001001";
input_2 <= "110110";
wait for clock_period*10;
input_1 <= "001001";
input_2 <= "110110";
wait for clock_period*10;

input_1 <= "111111";
input_2 <= "111111";
wait for clock_period*10;
input_1 <= "111111";
input_2 <= "111111";
wait for clock_period*10;

input_1 <= "011001";
input_2 <= "110111";
wait for clock_period*10;
input_1 <= "011001";
input_2 <= "110111";
wait for clock_period*10;
```

```
input_1 <= "001111";
input_2 <= "110011";
wait for clock_period*10;
input_1 <= "001111";
input_2 <= "110011";
wait for clock_period*10;

input_1 <= "111111";
input_2 <= "111100";
wait for clock_period*10;
input_1 <= "111111";
input_2 <= "111100";
wait for clock_period*10;
end process;

end Behavioral;
```

- **VHDL code for constraints**

```
set_property PACKAGE_PIN R2 [get_ports {input_1[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_1[5]}]
set_property PACKAGE_PIN T1 [get_ports {input_1[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_1[4]}]
set_property PACKAGE_PIN U1 [get_ports {input_1[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_1[3]}]
set_property PACKAGE_PIN W2 [get_ports {input_1[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_1[2]}]
set_property PACKAGE_PIN R3 [get_ports {input_1[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_1[1]}]
set_property PACKAGE_PIN T2 [get_ports {input_1[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_1[0]}]


set_property PACKAGE_PIN L1 [get_ports { led_1[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports { led_1[5]}]
set_property PACKAGE_PIN P1 [get_ports { led_1[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports { led_1[4]}]
set_property PACKAGE_PIN N3 [get_ports { led_1[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports { led_1[3]}]
set_property PACKAGE_PIN P3 [get_ports { led_1[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports { led_1[2]}]
set_property PACKAGE_PIN U3 [get_ports { led_1[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports { led_1[1]}]
set_property PACKAGE_PIN W3 [get_ports { led_1[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports { led_1[0]}]

set_property PACKAGE_PIN V15 [get_ports {input_2[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_2[5]}]
set_property PACKAGE_PIN W15 [get_ports {input_2[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_2[4]}]
```

```
set_property PACKAGE_PIN W17 [get_ports {input_2[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_2[3]}]
set_property PACKAGE_PIN W16 [get_ports {input_2[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_2[2]}]
set_property PACKAGE_PIN V16 [get_ports {input_2[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_2[1]}]
set_property PACKAGE_PIN V17 [get_ports {input_2[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {input_2[0]}]

set_property PACKAGE_PIN U15 [get_ports {led_2[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led_2[5]}]
set_property PACKAGE_PIN W18 [get_ports {led_2[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led_2[4]}]
set_property PACKAGE_PIN V19 [get_ports {led_2[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led_2[3]}]
set_property PACKAGE_PIN U19 [get_ports {led_2[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led_2[2]}]
set_property PACKAGE_PIN E19 [get_ports {led_2[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led_2[1]}]
set_property PACKAGE_PIN U16 [get_ports {led_2[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led_2[0]}]

set_property PACKAGE_PIN V2 [get_ports {s}]
      set_property IOSTANDARD LVCMOS33 [get_ports {s}]

set_property PACKAGE_PIN W5 [get_ports {clock}]
   set_property IOSTANDARD LVCMOS33 [get_ports {clock}]

set_property PACKAGE_PIN W7 [get_ports {cathode[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[6]}]
set_property PACKAGE_PIN W6 [get_ports {cathode[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[5]}]
set_property PACKAGE_PIN U8 [get_ports {cathode[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[4]}]
set_property PACKAGE_PIN V8 [get_ports {cathode[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[3]}]
set_property PACKAGE_PIN U5 [get_ports {cathode[2]}]
         set_property IOSTANDARD LVCMOS33 [get_ports {cathode[2]}]
set_property PACKAGE_PIN V5 [get_ports {cathode[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[1]}]
set_property PACKAGE_PIN U7 [get_ports {cathode[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[0]}]

set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {anode[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {anode[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {anode[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anode[3]}]
```

Ege Ozan Özyedek
EEE102 – 02
LabWork 05
Lab Report