LabWork04
Ege Ozan Özyedek
EEE102 – 02

- The Design Methodology

  In this week's lab, I have used the BASYS 3 FGPA board to compare two 6-bit unsigned binary numbers, take the smaller one and then to convert it to its octal value. The octal value was then shown on the 7-segment display. To achieve this, I used top module design. My modules were as follows;
  - Top Module: The top module does not contain any operations other than the one used to light up the LED's above the switches. A multiplexer component and a clock divider are declared. The 6-bit inputs are taken here and transferred into the port map of the multiplexer.
  - Multiplexer: The multiplexer enables the program to display the final octal value by changing the anode and cathode values. By changing these we can properly see the value on the 7-segment display.
  - Decoder: Decoder changes the cathode to its corresponding 7-bit binary value.
  - Comparator: This module compares the two 6-bit inputs and outputs two 3-bit outputs which are grouped values of the smaller input. By using these we can properly convert it via the decoder.
  - Clock Divider: The clock divider produces an output signal which makes it possible for the 7-segment display to work
- Result

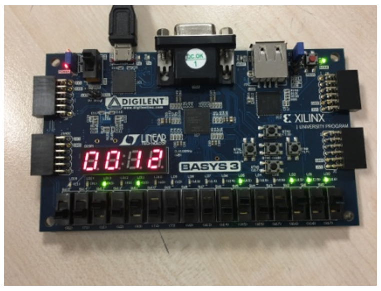
*Figure 1: f: (12), s: (47) and the output is the smaller (12)*

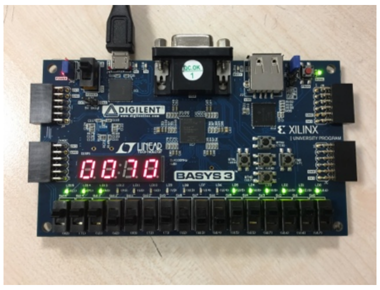
*Figure 2: f: (70), s: (77) and the output is the smaller (70)*

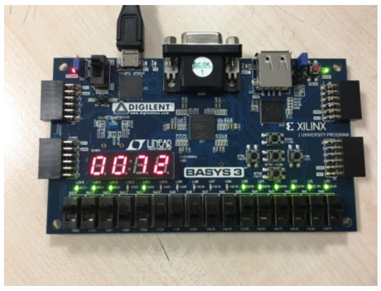
*Figure 3: f: (72), s: (77) and the output is the smaller (72)*

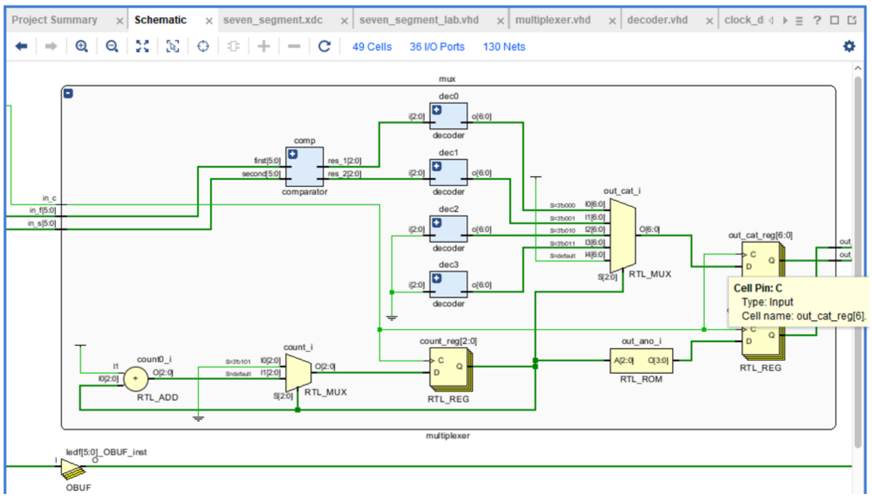

*Figure 4: RTL Schematic of the inside of the multiplexer*

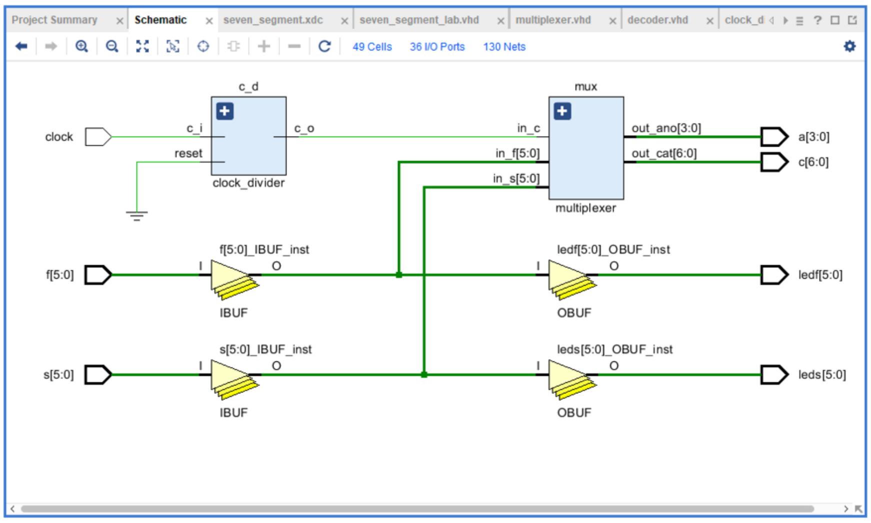

*Figure 5: The RTL schematic*

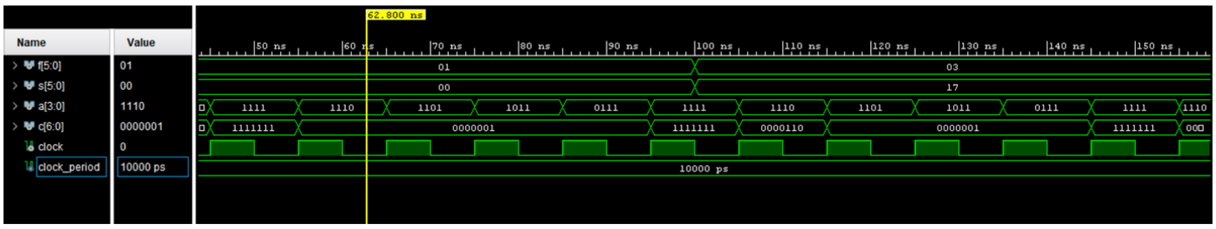

*Figure 6: Test Bench close-up*

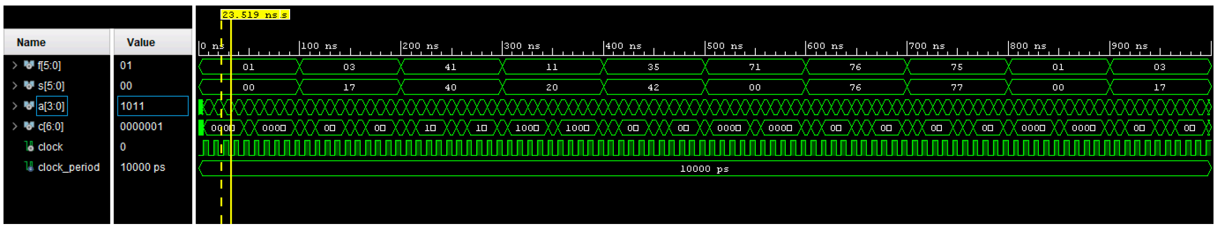

*Figure 7: Test Bench zoomed out*

The results were as expected. I did; however, encounter some problems along the way. My test bench code did not work at first since I inputed the wrong clock signal into the multiplexer.

- Conclusion

This labs goal was to familiarize ourselves with the 7-segment display and its confusing nature. I did learn a lot, both about the 7-segmen display and about VHDL. I learned how to use and change the values on the display using its cathode and anode values. I also learned much more about VHDL and was able to use some past coding skills to my advantage. I also learned some operations and types that are avalaible on VHDL such as if, else, for, signal and constant. I also learned how to code a clock divider, multiplexer and decoder and how to use them with each other. One final thing that was beneficial was our introduction to top module design.

- Appendices

    - VHDL code of Top Module (seven_segment_lab.vhd)

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

entity seven_segment_lab is
    Port ( f, s : in STD_LOGIC_VECTOR (5 downto 0);
    ledf, leds : out std_logic_vector (5 downto 0);
   clock : in std_logic;
   a : out std_logic_vector (3 downto 0):= "0000";
   c : out std_logic_vector (6 downto 0):= "0000000");
end seven_segment_lab;

architecture Behavioral of seven_segment_lab is

component multiplexer is
Port ( in_f: in std_logic_vector (5 downto 0);
  in_s : in std_logic_vector (5 downto 0);
  in_c: in std_logic;
  a: out std_logic_vector (3 downto 0);
  c: out std_logic_vector (6 downto 0)
   );
end component;

component clock_divider is
Port ( c_i : in std_logic;
  c_o : out std_logic;
  reset: in std_logic);
end component;

signal clocksignal: std_logic;

begin

c_d: clock_divider port map (clock, clocksignal, '0');
mux : multiplexer port map (f, s, clock,  a,  c);



process (f, s) --process which enables me to set the led's above the switches to light up
accordingly
begin
    for i in 0 to 5 loop
        ledf(i) <= f(i);
        leds(i) <= s(i);
    end loop;
end process;
end Behavioral;
```

- VHDL code of Multiplexer

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

entity multiplexer is
  Port ( in_f: in std_logic_vector (5 downto 0);
  in_s : in std_logic_vector (5 downto 0);
  in_c: in std_logic;
  a: out std_logic_vector (3 downto 0);
  c: out std_logic_vector (6 downto 0)
   );
end multiplexer;

architecture Behavioral of multiplexer is

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_3: std_logic_vector (6 downto 0);
signal comp_1: std_logic_vector (2 downto 0);
signal comp_2: std_logic_vector (2 downto 0);
signal count: integer range 0 to 4;

Component decoder is
Port (
i : in std_logic_vector ( 2 downto 0);
c : out std_logic_vector (6 downto 0));
end Component;

Component comparator is
Port (
first : in STD_LOGIC_VECTOR (5 downto 0);
second : in STD_LOGIC_VECTOR (5 downto 0);
res_1 : out STD_LOGIC_VECTOR (2 downto 0);
res_2 : out STD_LOGIC_VECTOR (2 downto 0));

end Component;

begin

comp : comparator port map (in_f, in_s, comp_1, comp_2);
dec0 : decoder port map (comp_1 ,d_0);
dec1 : decoder port map (comp_2 ,d_1);
dec2 : decoder port map ("000" ,d_2);
dec3 : decoder port map ("000",d_3);

process( in_c)
begin

if (rising_edge(in_c)) then
```

```
  case count is
     when 0 => a <= "1110";
            c <= d_0;
     when 1 => a <= "1101";
            c <= d_1;
     when 2 => a <= "1011";
            --c <= d_2;
     when 3 => a <= "0111";
            --c <= d_3;
     when others => a <= "1111";
                 c <= "1111111";

  end case;

  count <= count + 1;

  if (count = 4) then

     count <= 0;

   end if;

end if;
end process;

end Behavioral;
```

- VHDL code of Decoder

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

entity decoder is
  Port ( i : in std_logic_vector ( 2 downto 0);
  c : out std_logic_vector (6 downto 0));

end decoder;

architecture Behavioral of decoder is

begin

with i select c <=
   "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 of Comparator (compares two 6-bit values and outputs two 3 bit values)

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

entity comparator is
   Port ( first : in STD_LOGIC_VECTOR (5 downto 0);
        second : in STD_LOGIC_VECTOR (5 downto 0);
        res_1 : out STD_LOGIC_VECTOR (2 downto 0);
        res_2 : out STD_LOGIC_VECTOR (2 downto 0));
end comparator;

architecture Behavioral of comparator is

begin

process( first, second)

begin

if ( to_integer( unsigned( first)) < to_integer( unsigned( second))) then

  for i in 0 to 5 loop

   if ( i < 3) then
         res_1(i) <= first(i);
   else
         res_2(i - 3) <= first(i);
   end if;

  end loop;

else

  for i in 0 to 5 loop

   if ( i < 3) then
      res_1(i) <= second(i);
   else
      res_2(i - 3) <= second(i);
   end if;

  end loop;

end if;

end process;

end Behavioral;
```

LabWork04
Ege Ozan Özyedek
EEE102 – 02

- VHDL code of Clock Divider

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


entity clock_divider is
  Port ( c_i : in std_logic;
  c_o : out std_logic;
  reset: in std_logic := '0' );
end clock_divider;

architecture Behavioral of clock_divider is

signal indicator: std_logic;
signal count: integer range 0 to 10000 :=0;

begin

process(c_i, reset)

begin

    if( reset = '1') then

        indicator <= '0';
        count <= 0;
    elsif rising_edge(c_i) then

        if (count = 9999) then

            indicator <= not indicator;
            count <= 0;
        else

            count <= count + 1;

        end if;

     end if;

end process;

c_o <= indicator;

end Behavioral;
```

- VHDL code of Test Bench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

LabWork04
Ege Ozan Özyedek
EEE102 – 02

```
entity test_bench is
end test_bench;

architecture Behavioral of test_bench is

component seven_segment_lab is
port (f, s : in STD_LOGIC_VECTOR (5 downto 0);
  ledf, leds : out std_logic_vector (5 downto 0);
  clock : in std_logic;
  a : out std_logic_vector (3 downto 0);
  c : out std_logic_vector (6 downto 0));
end component;

signal f : std_logic_vector (5 downto 0) := "000000";
signal s : std_logic_vector (5 downto 0) := "000000";
signal a : std_logic_vector (3 downto 0) :="0000";
signal c : std_logic_vector (6 downto 0) := "0000000";
signal clock : std_logic;
constant clock_period : time := 10 ns;

begin

uut: seven_segment_lab port map ( f => f, s => s, clock => clock, a => a, c => c);


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

stimulus_process: process

begin

f <= "000001";
s <= "000000";
wait for clock_period*10;

f <= "000011";
s <= "001111";
wait for clock_period*10;

f <= "100001";
s <= "100000";
wait for clock_period*10;

f <= "001001";
s <= "010000";
wait for clock_period*10;

f <= "011101";
s <= "100010";
```

```
wait for clock_period*10;

f <= "111001";
s <= "000000";
wait for clock_period*10;

f <= "111110";
s <= "111110";
wait for clock_period*10;

f <= "111101";
s <= "111111";
wait for clock_period*10;

end process;

end Behavioral;
```