Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

Design Methodology:

In this lab work, we needed to design chronometer with some features. For this purpose, I used my seven segment displays code from Lab4 with minor changes such as displaying decimal numbers instead of hexa-decimal numbers. Then, I used 4-bit binary number to represent each digit of seven segment display. In this lab, display max values must be 59:59. Hence, the maximum value that second and minute digits can take is ‘9’ and in binary, “1001”; maximum value that tenseconds and tenminutes digits can take is ‘5’ and in binary, “0101”. After that, I designed counter module under the top module. The other three modules are for seven segment displays. In counter module, firstly, I wrote clock divider to acquire 1 Hz clock which has a period of one second. And with using 1 Hz clock in the process that I wrote chronometer, three pushbuttons worked as I expected. But for the pushbuttons to toggle pause/resume feature and forwards/backwards feature, I needed to convert pushbuttons to switchs. When I click once to pushbutton, it must toggle on and when I click again, it must toggle off. So, for this purpose I used FSM.

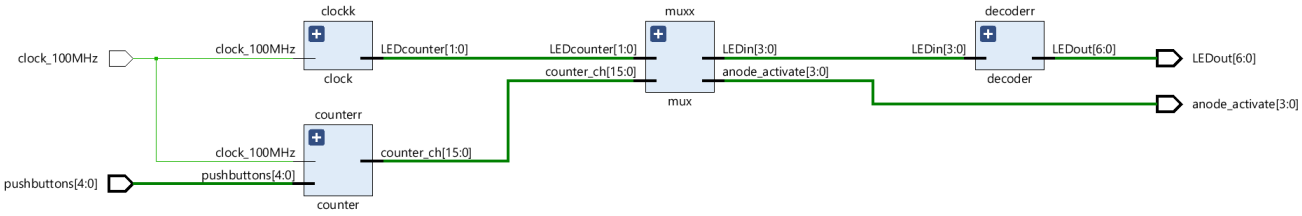

Figure 1: RTL Schematic

Pushbuttons	P.B. Working Type	Features
Pushbutton Up	Switch	Pause / Resume Chr.
Pushbutton Down	Switch	Change Counting Mode
Pushbutton Center	Button	Reset Chronometer
Pushbutton Right	Button	Add 10 Seconds
Pushbutton Left	Button	Subtract 10 Seconds

Table 1: Pushbuttons and Features

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

Results:

After Synthesis and Implementation, I checked for RTL schematics. Schematic of counter module is very long because of the if statements in VHDL code. Schematics are:

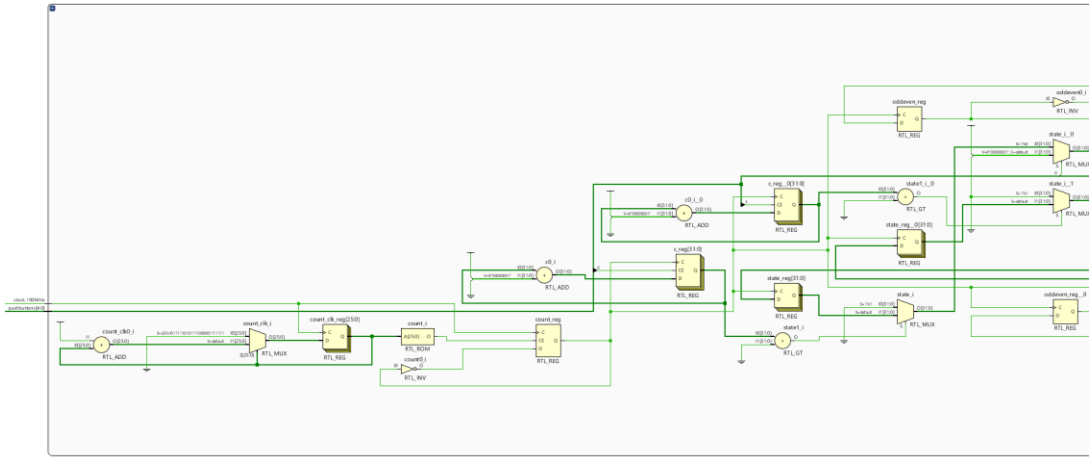

Figure 2: Schematic of Counter Module Part 1

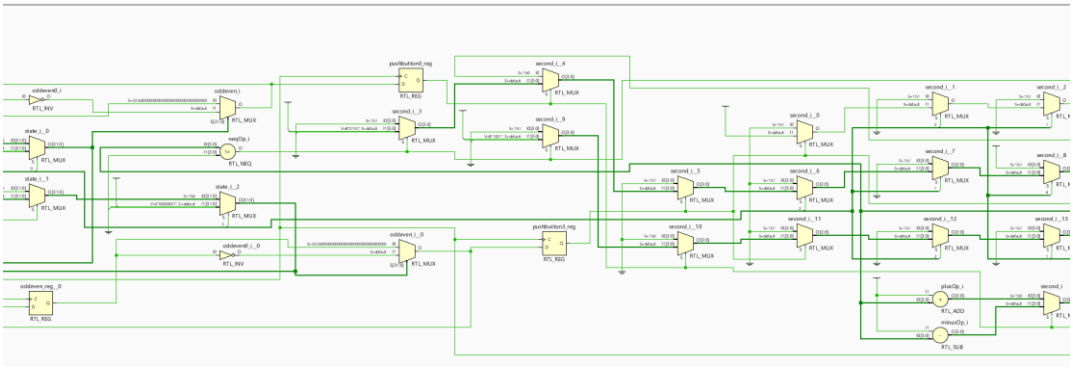

Figure 3: Schematic of Counter Module Part 2

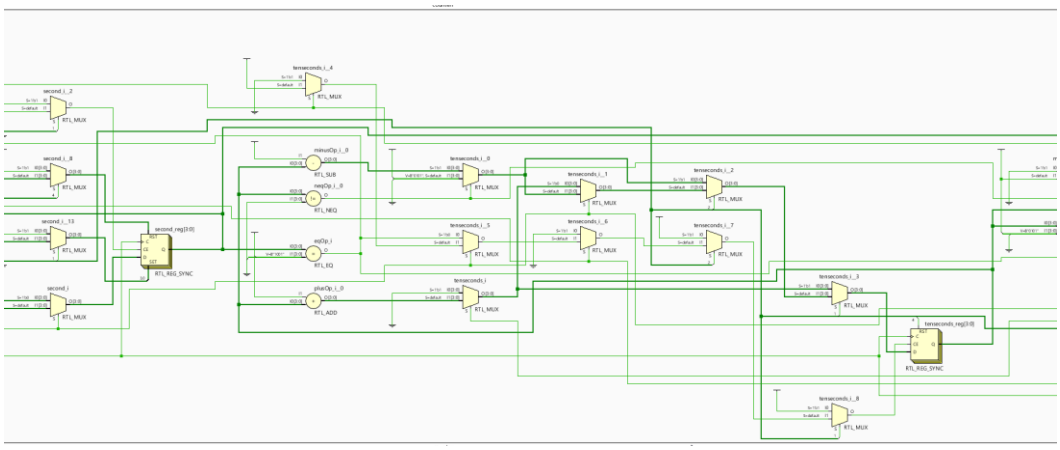

Figure 4: Schematic of Counter Module Part 3

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

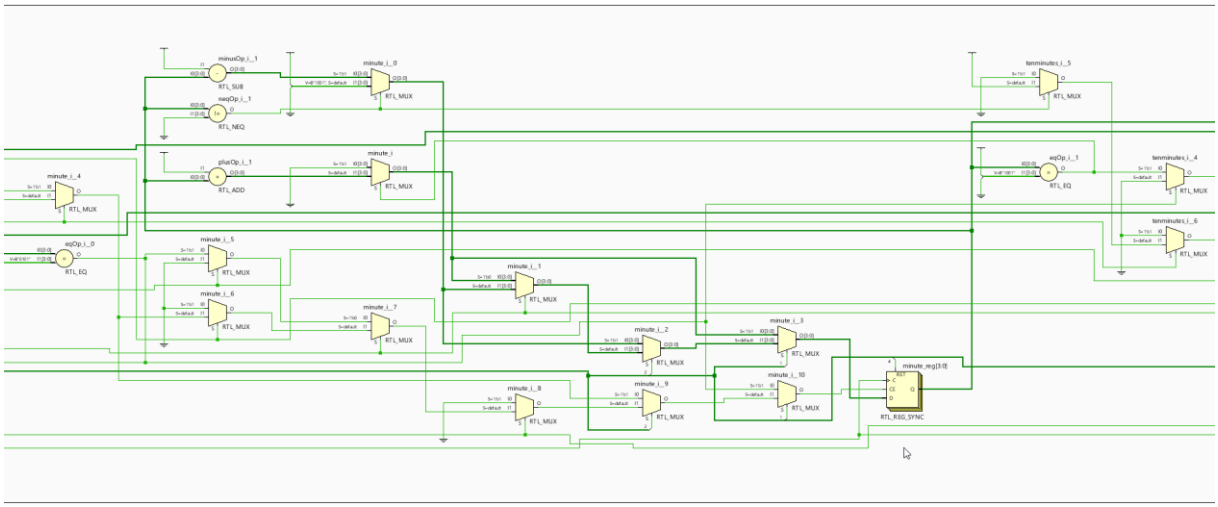

Figure 5: Schematic of Counter Module Part 4

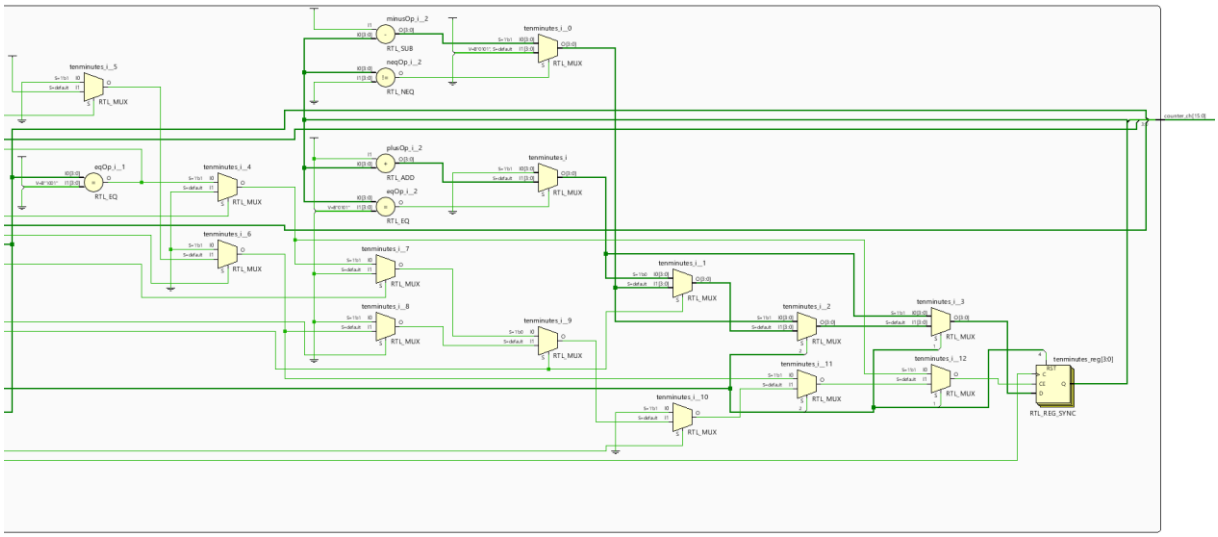

Figure 6: Schematic of Counter Module Part 5

After generating bitstream and programming FPGA device, our demo results are as we expected:

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

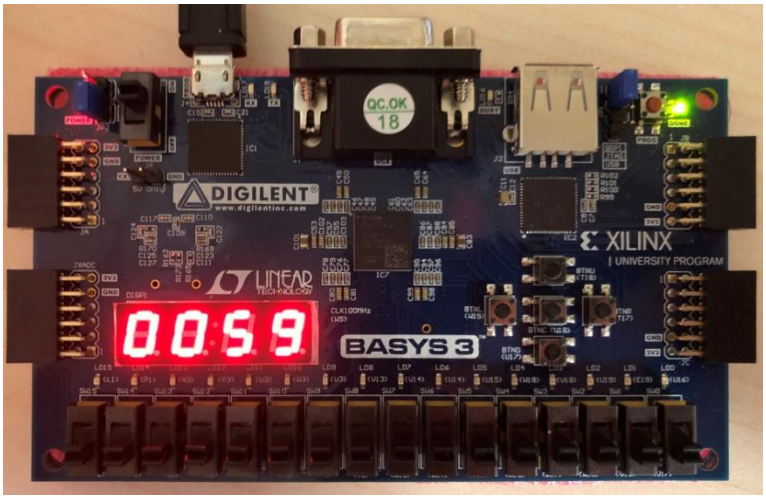

Figure 7: Demo 1 – Forwards Counting Mode and Pause

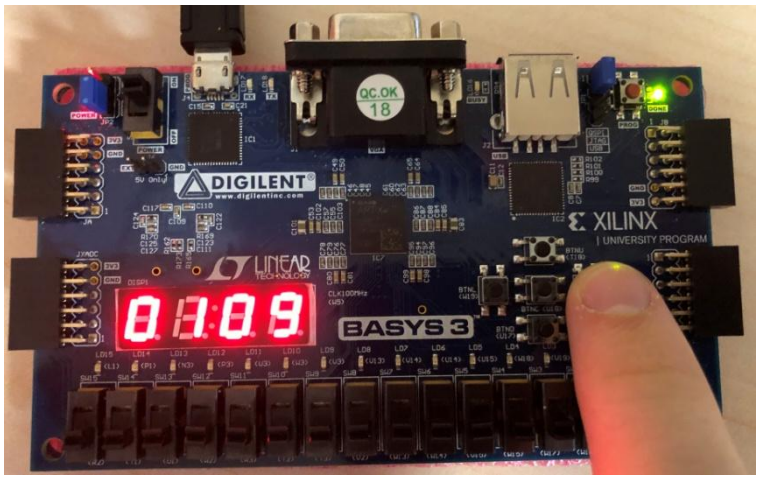

Figure 8: Demo 2 – Add 10 Seconds

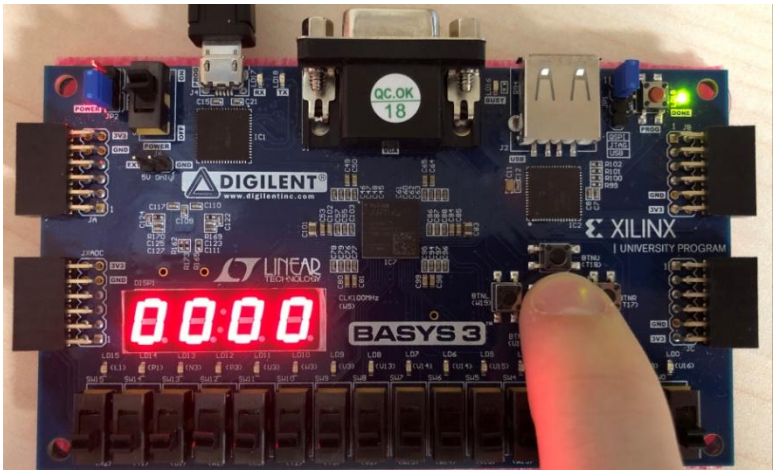

Figure 9: Demo 3 – Reset Chronometer

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

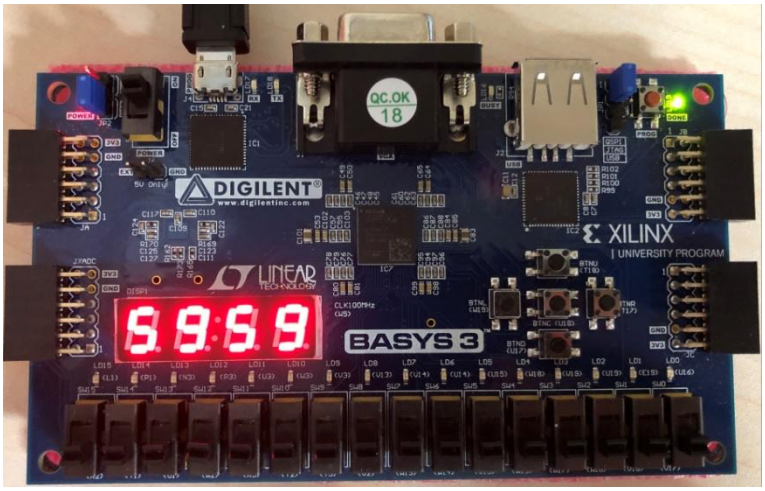

Figure 10: Demo 4 – Resume and Backwards Counting Mode

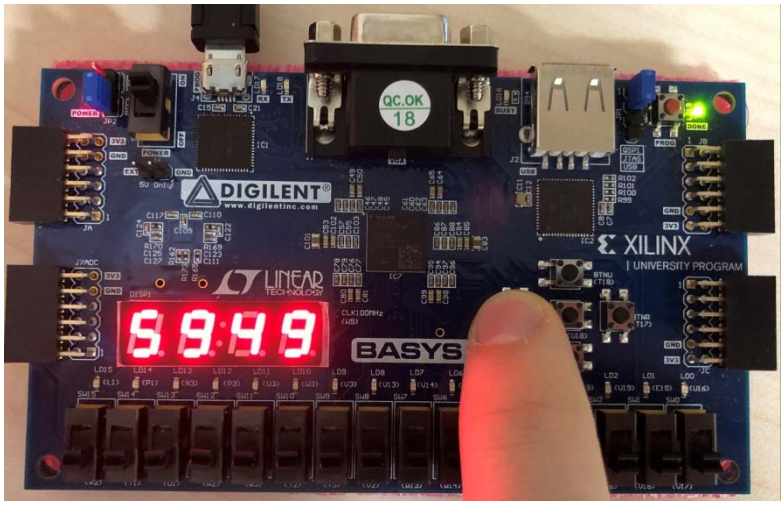

Figure 11: Demo 5 – Subtract 10 Seconds

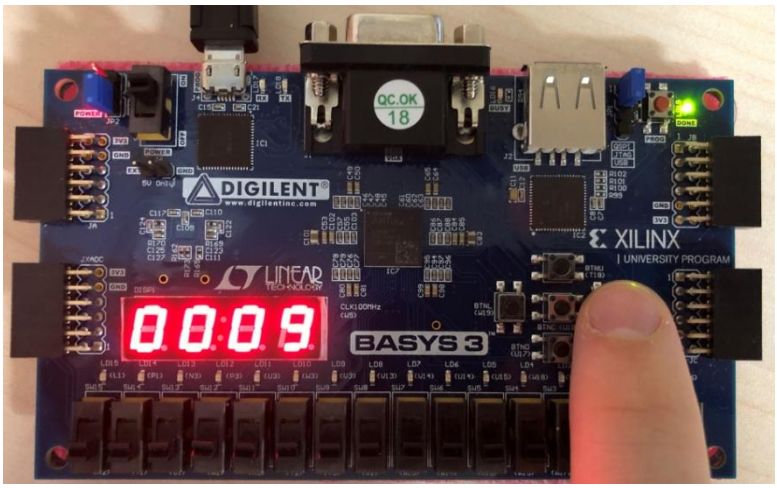

Figure 12: Demo 6 – Add 10 Second (Clicked twice)

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

Conclusion:

In this lab experiment I learned to design with memory and using 100 MHz clock to acquire real time values such as second. Also, I realized that using pushbuttons is a bit challenging because if we use fast clock, when we push to pushbutton, it can count 2 or 3 counts. So, we can use D FF's, FSM or slow clock. To use pushbuttons with features of reset, add 10 seconds, subtract 10 second; I use clock divider to acquire slow clock which is 1 Hz (I could use 4 Hz clock instead but since I am counting seconds, I did not wanted to write more if statements such as if count = 5 then second <= second +1;) And also I needed to use two pushbutton for pause/resume feature and change counting mode feature. In these features, I needed to convert pushbuttons to switchs then I used Finite State Machine for this situation. I defined oddeven variable and I changed this value at every click.

```
process(count) -- pushbutton 0 fsm to use as switch
    variable c: integer :=0;
    variable state: integer :=0;
    variable oddeven: std_logic :='0';
    begin
    if rising_edge(count) then
        if pushbuttons(0) = '0' then
            if c > 0 then
                state:= 0;
            end if;
        else
            state := 1;
            c := c+1;
        end if;
        if state = 0 then
            oddeven := oddeven;
        else
            oddeven := NOT oddeven;
        end if;
        if oddeven = '0' then
            pushbutton0 <= '0';
        else
            pushbutton0 <= '1';
        end if;

    end if;
    end process;
```

Questions:

1) <u>Do you use any type of memory when taking inputs from buttons? If so, please explain.</u>

Yes, I used memory in the code part above. To avoid multiple clicks, I defined variable "state" and to use pushbutton as switch, I defined variable "oddeven". Memory keeps

values of these variables. For example until I press button again, oddeven keeps its value and my chronometer's counting mode is not changing. To be more specific, I used positive edge triggered D-type Flip-Flops I think. I also added little part of code to below:

```
if rising_edge(count) then
                    ...
if state = 0 then
    oddeven := oddeven;
else
    oddeven := NOT oddeven;
```

2) <u>What type of memory do you think your chronometer uses (latches, flip-flops etc.)? Can you show which part of your code results in this kind of memory?</u>

I think chronometer uses positive edge triggered D-type Flip-Flops. Because I used process with rising_edge of 1 Hz clock signal. Also "second <= second" or "second <= second+1" usage inside process with clock shows us it uses memory and FF. Some part of code can be found below:

```
process(count) --chronometer process
begin
if rising_edge(count) then

        -- ...  I cut some codes here

if pushbutton3 = '1' then -- pause and resume chronometer
    second <= second;

else

    if pushbutton0 = '0' then -- forward counting mode
        second <= second +1;
        if second = "1001" then
            second <= "0000";
            tenseconds <= tenseconds +1;

        -- ...  I cut some codes here

    else                    -- backward counting mode
        if second /= "0000" then
        second <= second -1;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

Appendices:

Top Module - Chronometer:

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


entity chronometer is
    Port (
        clock_100MHz: in std_logic;
        pushbuttons: in std_logic_vector(4 downto 0);
        LEDout: out std_logic_vector (6 downto 0);
        anode_activate: out std_logic_vector (3 downto 0)
        );
end chronometer;
architecture Behavioral of chronometer is
component counter is
Port (
    clock_100MHz: in std_logic;
    pushbuttons: in std_logic_vector(4 downto 0);
    counter_ch: out std_logic_vector (15 downto 0)
    );
end component;
component clock is
Port (
    clock_100MHz: in std_logic;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
    LEDcounter: out std_logic_vector (1 downto 0)
    );
end component;
component mux is
Port (
    LEDcounter: in std_logic_vector (1 downto 0);
    counter_ch: in std_logic_vector (15 downto 0);
    LEDin: out std_logic_vector (3 downto 0);
    anode_activate: out std_logic_vector (3 downto 0)
    );
end component;
component decoder is
Port (
    LEDin: in std_logic_vector(3 downto 0);
    LEDout: out std_logic_vector (6 downto 0)
    );
end component;
signal LEDcounter: std_logic_vector (1 downto 0);
signal LEDin: std_logic_vector (3 downto 0);
signal counter_ch: std_logic_vector(15 downto 0);
begin
clockk: clock port map ( clock_100MHz => clock_100MHz, LEDcounter => LEDcounter);
decoderr: decoder port map ( LEDin => LEDin, LEDout => LEDout);
muxx: mux port map ( LEDcounter => LEDcounter, counter_ch => counter_ch, LEDin =>
LEDin, anode_activate => anode_activate);
```

```
counterr: counter port map ( clock_100MHz => clock_100MHz, pushbuttons => pushbuttons, counter_ch => counter_ch);

end Behavioral;
```

Counter Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.std_logic_unsigned.all;


entity counter is
    Port (
    clock_100MHz: in std_logic;
    pushbuttons: in std_logic_vector(4 downto 0);
    counter_ch: out std_logic_vector (15 downto 0)
    );
end counter;
architecture Behavioral of counter is
    signal counter2: integer:= 0;
    signal count  : std_logic;
    signal second :std_logic_vector(3 downto 0);
    signal tenseconds  : std_logic_vector (3 downto 0);
    signal minute  : std_logic_vector (3 downto 0);
    signal tenminutes  : std_logic_vector (3 downto 0);
    signal pushbutton3 : std_logic;
    signal pushbutton0 : std_logic;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
begin


clk_div : process (clock_100MHz) -- 100MHz to 1Hz, 50% duty cycle, clock divider
variable count_clk : integer range 0 to 49999999 := 0;  -- (100.000.000/1)/2 - 1
begin
   if (rising_edge(clock_100MHz)) then
            if (count_clk = 49999999) then
               count <= not count; -- to obtain a 50% duty cycle
               count_clk := 0;
            else
               count_clk := count_clk + 1;
                   end if;
          end if;
end process clk_div;


   process(count) -- pushbutton 3 fsm to use as switch
   variable c: integer :=0;
   variable state: integer :=0;
   variable oddeven: std_logic;
   begin
   if rising_edge(count) then
      if pushbuttons(3) = '0' then
         if c > 0 then
            state:= 0;
```

```
      end if;
   else
      state := 1;
      c := c+1;
   end if;
   if state = 0 then
      oddeven := oddeven;
   else
      oddeven := NOT oddeven;
   end if;
   if oddeven = '0' then
      pushbutton3 <= '0';
   else
      pushbutton3 <= '1';
   end if;
end if;
end process;

process(count) -- pushbutton 0 fsm to use as switch
variable c: integer :=0;
variable state: integer :=0;
variable oddeven: std_logic :='0';
begin
if rising_edge(count) then
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
    if pushbuttons(0) = '0' then
        if c > 0 then
            state:= 0;
        end if;
    else
        state := 1;
        c := c+1;
    end if;
    if state = 0 then
        oddeven := oddeven;
    else
        oddeven := NOT oddeven;
    end if;
    if oddeven = '0' then
        pushbutton0 <= '0';
    else
        pushbutton0 <= '1';
    end if;
end if;
end process;

process(count) --chronometer process
begin
if rising_edge(count) then -- reset chronometer
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
if pushbuttons(4) = '1' then
second <= "0000";
tenseconds <= "0000";
minute <= "0000";
tenminutes <= "0000";

elsif pushbuttons(1) = '1' then -- add 10 seconds
   tenseconds <= tenseconds +1;
   if tenseconds = "0101" then
      tenseconds <= "0000";
      minute <= minute +1;
         if minute = "1001" then
            minute <= "0000";
            tenminutes <= tenminutes +1;
            if tenminutes = "0101" then
               tenminutes <= "0000";
            end if;
         end if;
   end if;

elsif pushbuttons(2) = '1' then --subtract 10 seconds
   if tenseconds /= "0000" then
   tenseconds <= tenseconds - 1;
   else
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
        tenseconds <= "0101";
        if minute /= "0000" then
        minute <= minute -1;
        else
           minute <= "1001";
           if tenminutes /= "0000" then
              tenminutes <= tenminutes -1;
           else
              tenminutes <= "0101";
           end if;
        end if;
     end if;


  else
     if pushbutton3 = '1' then -- pause and resume chronometer
        second <= second;
     else
     if pushbutton0 = '0' then -- forward counting mode
     second <= second +1;
        if second = "1001" then
        second <= "0000";
        tenseconds <= tenseconds +1;
           if tenseconds = "0101" then
              tenseconds <= "0000";
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
            minute <= minute +1;
               if minute = "1001" then
                  minute <= "0000";
                  tenminutes <= tenminutes +1;
                  if tenminutes = "0101" then
                     tenminutes <= "0000";
                  end if;
               end if;
         end if;
      end if;

   else                    -- backward counting mode
      if second /= "0000" then
         second <= second -1;
      else
         second <= "1001";
         if tenseconds /= "0000" then
            tenseconds <= tenseconds - 1;
         else
            tenseconds <= "0101";

            if minute /= "0000" then
            minute <= minute -1;
            else
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 6
18.04.2021

```
                    minute <= "1001";
                    if tenminutes /= "0000" then
                       tenminutes <= tenminutes -1;
                    else
                       tenminutes <= "0101";
                    end if;
                  end if;
               end if;
         end if;
         end if;
         end if;
      end if;
   end if;
   end process;

counter_ch(3 downto 0) <= second;
counter_ch(7 downto 4) <= tenseconds;
counter_ch(11 downto 8) <= minute;
counter_ch(15 downto 12) <= tenminutes;



end Behavioral;
```