Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

**Abstract / Objective:**

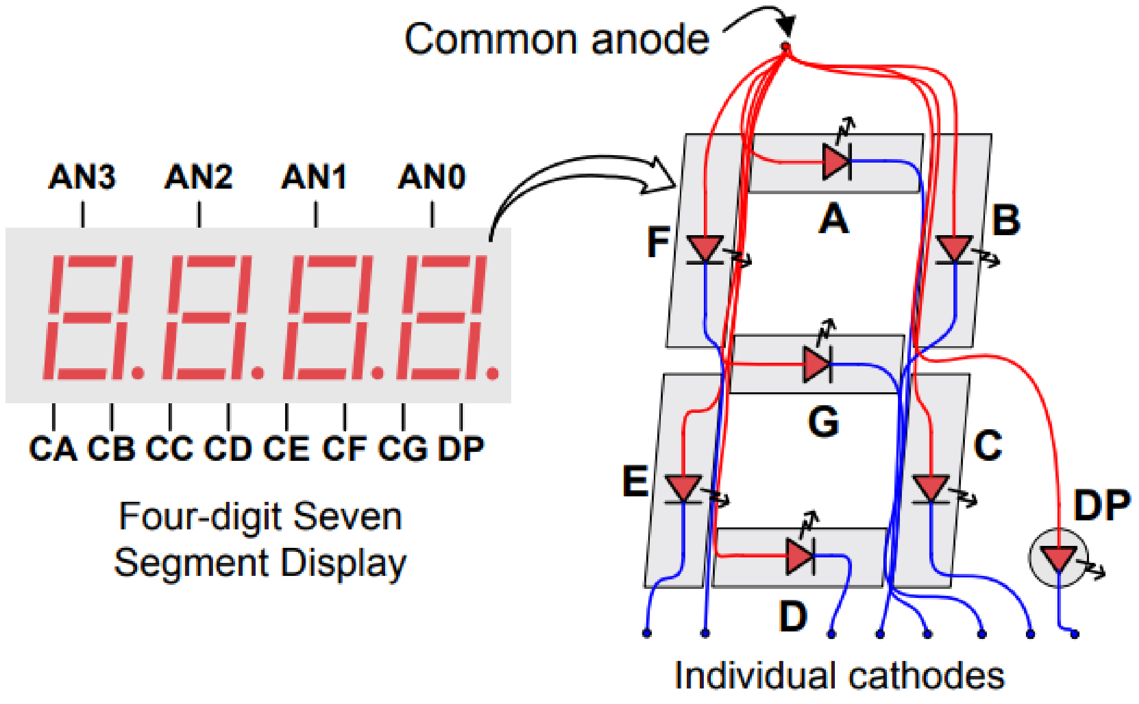
In this lab we will be using multiple 7 segment displays. To do this we will be both using both decoders and clock which will broaden our understanding of both of them. I will make a binary to octa decimal converter. To do this I will use 3 to 7 decoders for the 7 segment displays and a clock to show multiple digits at the same time.

**Design Specification Plan:**

Like I said I will be using 3 to 7 decoders to convert a 3 bit binary number to an image of an octa decimal number on 7 bit segment. To do so I will draw a truth table for the 7 leds on the display for the inputs then I will draw 7 K maps for the 7 leds of the display then find their logic equations.

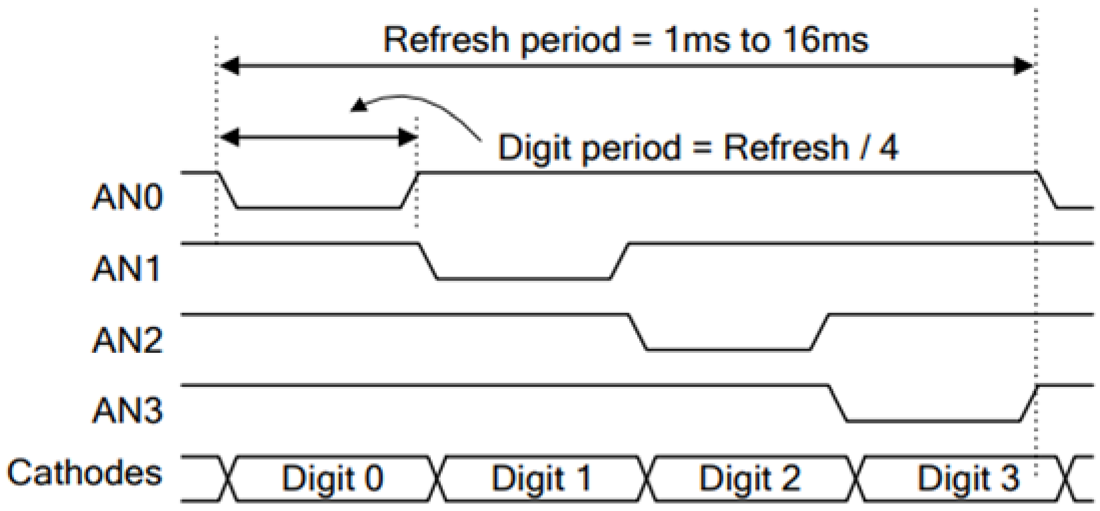
**Proposed Design Methodology:**

the 7 Segment Display has 7 leds in it.

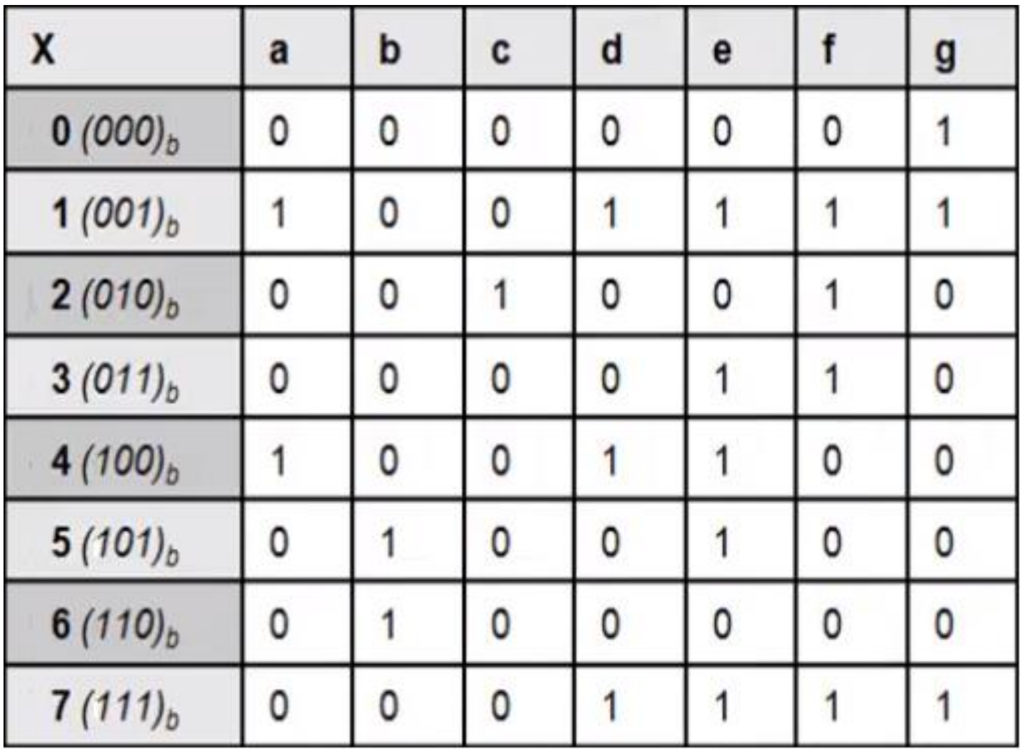


Displays have common anodes but separate cathodes but the 4 displays cathodes are shared. To give an example when we want to light up led A we light up all 4 Led A's on the display. To show 4 different numbers at the same time, we the 4 numbers one after the other one based on a clock so fast that the human eye cannot see the leds go dark.

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019



This is a graph of the {an0, an1, an2, an3} outputs for us to display 4 different digits at the same time. Or at least look like we are displaying 4 digits at the same time.



| X | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| 0 $(000)_b$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 $(001)_b$ | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 $(010)_b$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 $(011)_b$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 $(100)_b$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 $(101)_b$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 $(110)_b$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 $(111)_b$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

This is the truth table for the leds on the seven segment display.

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

A = (X2*X1'*X0')+(X2'*X1'*X0)

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |

B = (X2*X1'*X0)+(X2*X1*X0')

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |

C = (X2'*X1*X0')

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |

D = (X2*X1'*X0')+(X2'*X1'*X0)+(X2*X1*X0)

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

E = (X2*X1')+X0

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |

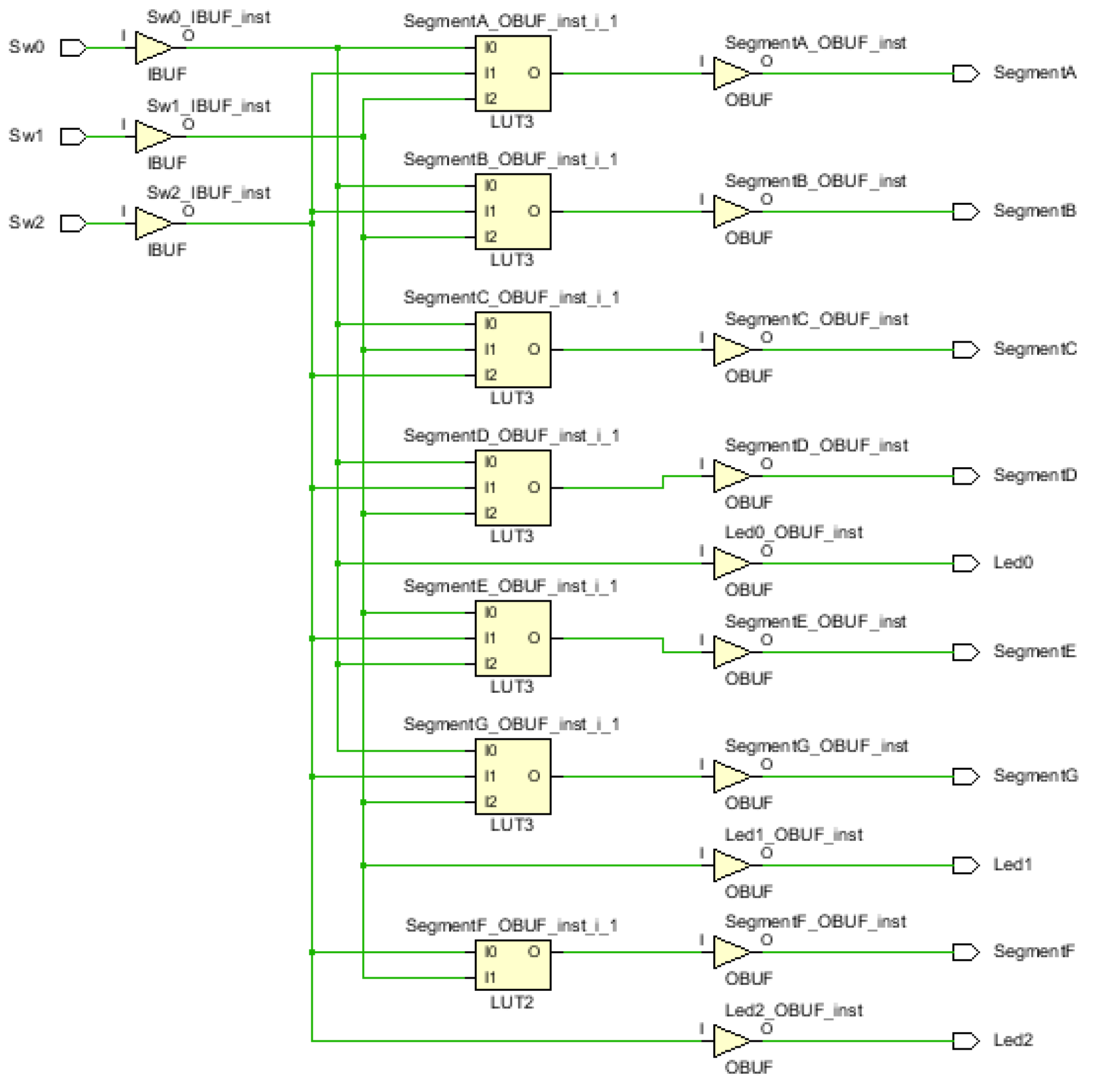
F = (X2'*X0)+(X2'*X1)+(X1*X0)

| X2 \ X1X0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |

G = (X2'*X1')+(X2*X1*X0)

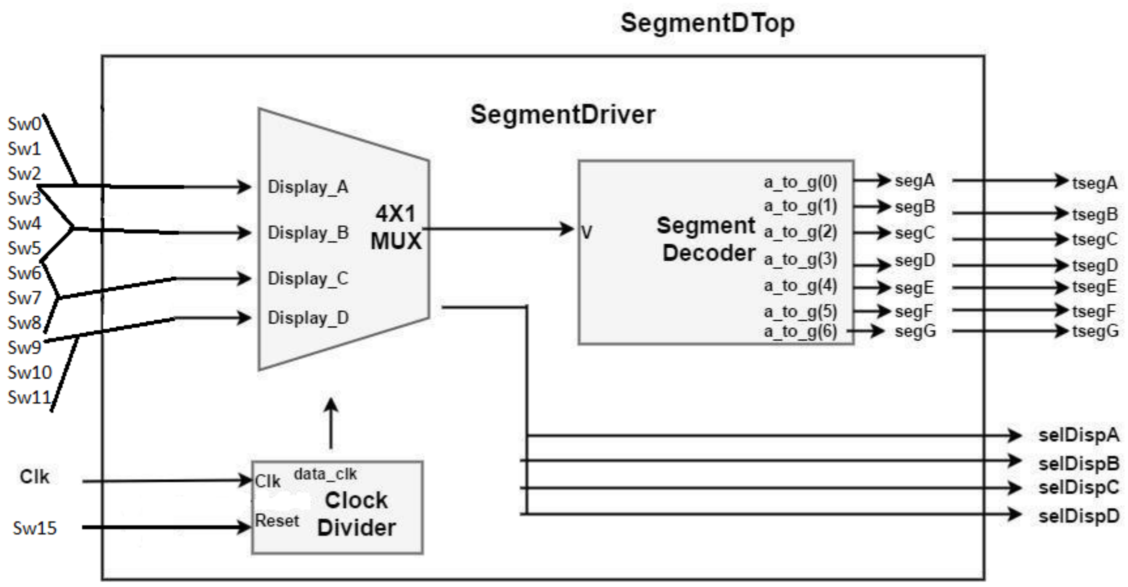
This are the K maps and the logic equations for the leds on the seven segment display.

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019



This is the logic gate schematic of the 3 to 7 decoder for the display. (Leds are only there to show the input signals work) (I also wanted Vivado to draw this schematic with AND and OR gates but it did not do it.)

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019



This is a schematic of the whole system

**Proposed Design Methodology:**

***Main source:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.std_logic_unsigned.all;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;


-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;


entity main is

   Port ( Sw0 : in STD_LOGIC;
        Sw1 : in STD_LOGIC;
        Sw2 : in STD_LOGIC;
        Sw3 : in STD_LOGIC;
        Sw4 : in STD_LOGIC;
        Sw5 : in STD_LOGIC;
        Sw6 : in STD_LOGIC;
        Sw7 : in STD_LOGIC;
        Sw8 : in STD_LOGIC;
        Sw9 : in STD_LOGIC;
        Sw10: in STD_LOGIC;
        Sw11: in STD_LOGIC;
        an0 : out STD_LOGIC;
        an1 : out STD_LOGIC;
        an2 : out STD_LOGIC;
        an3 : out STD_LOGIC;
        SegmentA : out STD_LOGIC;
        SegmentB : out STD_LOGIC;
        SegmentC : out STD_LOGIC;
        SegmentD : out STD_LOGIC;
        SegmentE : out STD_LOGIC;
        SegmentF : out STD_LOGIC;
        SegmentG : out STD_LOGIC;
        clock_100Mhz : in STD_LOGIC;-- 100Mhz clock on Basys 3 FPGA board
        reset : in STD_LOGIC); -- reset

        end main;

architecture Behavioral of main is

signal x0: std_logic;
signal x1: std_logic;

signal x2: std_logic;


-- my variables

signal refresh_counter: STD_LOGIC_VECTOR (19 downto 0);

-- creating 10.5ms refresh period


signal counter_for_LEDs: std_logic_vector(1 downto 0);

component decoder

port(

   Dx0 : in STD_LOGIC;

   Dx1 : in STD_LOGIC;

   Dx2 : in STD_LOGIC;

   DSegmentA : out STD_LOGIC;

   DSegmentB : out STD_LOGIC;

   DSegmentC : out STD_LOGIC;

   DSegmentD : out STD_LOGIC;

   DSegmentE : out STD_LOGIC;

   DSegmentF : out STD_LOGIC;

   DSegmentG : out STD_LOGIC);


end component;


component multiplexer

port(

   Mcounter_for_LEDs : in std_logic_vector(1 downto 0);

   MSw0 : in STD_LOGIC;

   MSw1 : in STD_LOGIC;

   MSw2 : in STD_LOGIC;

   MSw3 : in STD_LOGIC;
```

```
    MSw4 : in STD_LOGIC;
    MSw5 : in STD_LOGIC;
    MSw6 : in STD_LOGIC;
    MSw7 : in STD_LOGIC;
    MSw8 : in STD_LOGIC;
    MSw9 : in STD_LOGIC;
    MSw10: in STD_LOGIC;
    MSw11: in STD_LOGIC;
    Man0 : out STD_LOGIC;
    Man1 : out STD_LOGIC;
    Man2 : out STD_LOGIC;
    Man3 : out STD_LOGIC;
    Mx0 : out STD_LOGIC;
    Mx1 : out STD_LOGIC;
    Mx2 : out STD_LOGIC);

end component;

component Clock
port(
   Cclock_100Mhz : in STD_LOGIC;
   Creset : in STD_LOGIC;
   Ccounter_for_LEDs: out std_logic_vector(1 downto 0));

end component;



begin

uut1 : decoder port map(
    Dx0 => x0,
    Dx1 => x1,
    Dx2 => x2,
    DSegmentA => SegmentA,
    DSegmentB => SegmentB,
    DSegmentC => SegmentC,
    DSegmentD => SegmentD,
    DSegmentE => SegmentE,
    DSegmentF => SegmentF,
    DSegmentG => SegmentG);

uut2 : multiplexer port map(
    Mcounter_for_LEDs => counter_for_LEDs,
    MSw0 => Sw0,
    MSw1 => Sw1,
    MSw2 => Sw2,
    MSw3 => Sw3,
    MSw4 => Sw4,
    MSw5 => Sw5,
    MSw6 => Sw6,
    MSw7 => Sw7,
    MSw8 => Sw8,
    MSw9 => Sw9,
    MSw10 => Sw10,
    MSw11 => Sw11,
    Man0 => an0,
    Man1 => an1,
    Man2 => an2,
    Man3 => an3,
    Mx0 => x0,
    Mx1 => x1,
    Mx2 => x2);

uut3 : Clock port map(
    Cclock_100Mhz => clock_100Mhz,
    Creset => reset,
    Ccounter_for_LEDs => counter_for_LEDs);

end Behavioral;
```

<u>*** decoder source:</u>

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity decoder is
port(
   Dx0 : in STD_LOGIC;
   Dx1 : in STD_LOGIC;
   Dx2 : in STD_LOGIC;
   DSegmentA : out STD_LOGIC;
```

```
    DSegmentB : out STD_LOGIC;
    DSegmentC : out STD_LOGIC;
    DSegmentD : out STD_LOGIC;
    DSegmentE : out STD_LOGIC;
    DSegmentF : out STD_LOGIC;
    DSegmentG : out STD_LOGIC);
end decoder;

architecture Behavioral of decoder is

begin

DSegmentA <= (Dx2 and (not Dx1) and (not Dx0)) or ((not Dx2) and (not Dx1) and Dx0);
DSegmentB <= (Dx2 and (not Dx1) and Dx0) or (Dx2 and Dx1 and (not Dx0));
DSegmentC <= ((not Dx2) and Dx1 and (not Dx0));
DSegmentD <= (Dx2 and (not Dx1) and (not Dx0)) or ((not Dx2) and (not Dx1) and Dx0) or (Dx2 and
Dx1 and Dx0);
DSegmentE <= (Dx2 and (not Dx1)) or Dx0;
DSegmentF <= ((not Dx2) and Dx0) or ((not Dx2) and Dx1) or (Dx1 and Dx0);
DSegmentG <= ((not Dx2) and (not Dx1)) or (Dx2 and Dx1 and Dx0);

end Behavioral;
```

*** multiplexer source:

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
```

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;


entity multiplexer is
   port(
   Mcounter_for_LEDs : in std_logic_vector(1 downto 0);
   MSw0 : in STD_LOGIC;
   MSw1 : in STD_LOGIC;
   MSw2 : in STD_LOGIC;
   MSw3 : in STD_LOGIC;
   MSw4 : in STD_LOGIC;
   MSw5 : in STD_LOGIC;
   MSw6 : in STD_LOGIC;
   MSw7 : in STD_LOGIC;
   MSw8 : in STD_LOGIC;
   MSw9 : in STD_LOGIC;
   MSw10: in STD_LOGIC;
   MSw11: in STD_LOGIC;
   Man0 : out STD_LOGIC;
   Man1 : out STD_LOGIC;
   Man2 : out STD_LOGIC;
   Man3 : out STD_LOGIC;
   Mx0 : out STD_LOGIC;
   Mx1 : out STD_LOGIC;
   Mx2 : out STD_LOGIC);
end multiplexer;
```

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

```
architecture Behavioral of multiplexer is


begin

process(Mcounter_for_LEDs)

begin

   case Mcounter_for_LEDs is

   when "00" =>

      Man0 <= '0';

      Man1 <= '1';

      Man2 <= '1';

      Man3 <= '1';

      -- activate LED1 and Deactivate LED2, LED3, LED4

      Mx0 <= MSw0;

      Mx1 <= MSw1;

      Mx2 <= MSw2;

      -- the first hex digit of the 12-bit number

   when "01" =>

      Man0 <= '1';

      Man1 <= '0';

      Man2 <= '1';

      Man3 <= '1';

      -- activate LED2 and Deactivate LED1, LED3, LED4

      Mx0 <= MSw3;

      Mx1 <= MSw4;

      Mx2 <= MSw5;

      -- the second hex digit of the 12-bit number

   when "10" =>

      Man0 <= '1';

      Man1 <= '1';

      Man2 <= '0';
```

```
        Man3 <= '1';
        -- activate LED3 and Deactivate LED2, LED1, LED4
        Mx0 <= MSw6;
        Mx1 <= MSw7;
        Mx2 <= MSw8;
        -- the third hex digit of the 12-bit number
    when "11" =>
        Man0 <= '1';
        Man1 <= '1';
        Man2 <= '1';
        Man3 <= '0';
        -- activate LED4 and Deactivate LED2, LED3, LED1
        Mx0 <= MSw9;
        Mx1 <= MSw10;
        Mx2 <= MSw11;
        -- the fourth hex digit of the 12-bit number
    end case;
end process;

end Behavioral;
```

*** clock source:

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
```

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

```
-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;


entity Clock is

port(

   Cclock_100Mhz : in STD_LOGIC;

   Creset : in STD_LOGIC;

   Ccounter_for_LEDs: out std_logic_vector(1 downto 0));




end Clock;


architecture Behavioral of Clock is


signal refresh_counter: STD_LOGIC_VECTOR (19 downto 0);


begin


process(Cclock_100Mhz,Creset) --this part is the clock that generates the anode activating signals

begin

   if(Creset='1') then

      refresh_counter <= (others => '0');

   elsif(rising_edge(Cclock_100Mhz)) then

      refresh_counter <= refresh_counter + 1;

   end if;

end process;

 Ccounter_for_LEDs <= refresh_counter(19 downto 18);
```

end Behavioral;

**Constrans**

```
# Switches

set_property PACKAGE_PIN V17 [get_ports {Sw0}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw0}]

set_property PACKAGE_PIN V16 [get_ports {Sw1}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw1}]

set_property PACKAGE_PIN W16 [get_ports {Sw2}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw2}]

set_property PACKAGE_PIN W17 [get_ports {Sw3}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw3}]

set_property PACKAGE_PIN W15 [get_ports {Sw4}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw4}]

set_property PACKAGE_PIN V15 [get_ports {Sw5}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw5}]

set_property PACKAGE_PIN W14 [get_ports {Sw6}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw6}]

set_property PACKAGE_PIN W13 [get_ports {Sw7}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw7}]

set_property PACKAGE_PIN V2 [get_ports {Sw8}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw8}]

set_property PACKAGE_PIN T3 [get_ports {Sw9}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw9}]

set_property PACKAGE_PIN T2 [get_ports {Sw10}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw10}]

set_property PACKAGE_PIN R3 [get_ports {Sw11}]

   set_property IOSTANDARD LVCMOS33 [get_ports {Sw11}]


# Clock signal

set_property PACKAGE_PIN W5 [get_ports clock_100Mhz]
```

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

```
 set_property IOSTANDARD LVCMOS33 [get_ports clock_100Mhz]
set_property PACKAGE_PIN R2 [get_ports reset]
       set_property IOSTANDARD LVCMOS33 [get_ports reset]


#7 segment display
set_property PACKAGE_PIN W7 [get_ports {SegmentA}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentA}]
set_property PACKAGE_PIN W6 [get_ports {SegmentB}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentB}]
set_property PACKAGE_PIN U8 [get_ports {SegmentC}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentC}]
set_property PACKAGE_PIN V8 [get_ports {SegmentD}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentD}]
set_property PACKAGE_PIN U5 [get_ports {SegmentE}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentE}]
set_property PACKAGE_PIN V5 [get_ports {SegmentF}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentF}]
set_property PACKAGE_PIN U7 [get_ports {SegmentG}]
   set_property IOSTANDARD LVCMOS33 [get_ports {SegmentG}]


set_property PACKAGE_PIN U2 [get_ports {an0}]
   set_property IOSTANDARD LVCMOS33 [get_ports {an0}]
set_property PACKAGE_PIN U4 [get_ports {an1}]
   set_property IOSTANDARD LVCMOS33 [get_ports {an1}]
set_property PACKAGE_PIN V4 [get_ports {an2}]
   set_property IOSTANDARD LVCMOS33 [get_ports {an2}]
set_property PACKAGE_PIN W4 [get_ports {an3}]
   set_property IOSTANDARD LVCMOS33 [get_ports {an3}]
```

**Stimulation:**

Batu Arda Düzgün
Lab 4 Preliminary Work
21802633
10.29.2019

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

USE ieee.std_logic_unsigned.all;

Lab 2 experiment 7.10.2019

USE ieee.numeric_std.ALL;

ENTITY simple_testbench IS

END simple_testbench;

 ARCHITECTURE behavior OF simple_testbench IS

 -- Component Declaration for the Unit Under Test (UUT)

COMPONENT

lab_4

PORT(

        Sw0 : in STD_LOGIC;

        Sw1 : in STD_LOGIC;

        Sw2 : in STD_LOGIC;

        Sw3 : in STD_LOGIC;

        Sw4 : in STD_LOGIC;

        Sw5 : in STD_LOGIC;

        Sw6 : in STD_LOGIC;

        Sw7 : in STD_LOGIC;

        Sw8 : in STD_LOGIC;

        Sw9 : in STD_LOGIC;

        Sw10: in STD_LOGIC;

        Sw11: in STD_LOGIC;

        an0 : out STD_LOGIC;

        an1 : out STD_LOGIC;

        an2 : out STD_LOGIC;

        an3 : out STD_LOGIC;

        SegmentA : out STD_LOGIC;

        SegmentB : out STD_LOGIC;
        SegmentC : out STD_LOGIC;
        SegmentD : out STD_LOGIC;
        SegmentE : out STD_LOGIC;
        SegmentF : out STD_LOGIC;
        SegmentG : out STD_LOGIC;
        clock_100Mhz : in STD_LOGIC;
        reset : in STD_LOGIC);
END COMPONENT;
        Signal Sw0 : in STD_LOGIC;
        Signal Sw1 : in STD_LOGIC;
        Signal Sw2 : in STD_LOGIC;
        Signal Sw3 : in STD_LOGIC;
        Signal Sw4 : in STD_LOGIC;
        Signal Sw5 : in STD_LOGIC;
        Signal Sw6 : in STD_LOGIC;
        Signal Sw7 : in STD_LOGIC;
        Signal Sw8 : in STD_LOGIC;
        Signal Sw9 : in STD_LOGIC;
        Signal Sw10: in STD_LOGIC;
        Signal Sw11: in STD_LOGIC;
        Signal an0 : out STD_LOGIC;
        Signal an1 : out STD_LOGIC;
        Signal an2 : out STD_LOGIC;
        Signal an3 : out STD_LOGIC;
        Signal SegmentA : out STD_LOGIC;
        Signal SegmentB : out STD_LOGIC;
        Signal SegmentC : out STD_LOGIC;
        Signal SegmentD : out STD_LOGIC;
        Signal SegmentE : out STD_LOGIC;
        Signal SegmentF : out STD_LOGIC;
        Signal SegmentG : out STD_LOGIC;


UUT)

uut: lab_2 PORT MAP

(Sw0 => Sw0, Sw1 => Sw1, … ); -- in this part I assign the ports with the signals

 -- Stimulus process

stim_proc: process

        begin


            # #Here I will write all the possible inputs for the circuit.


        end process;

        END
```