Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

Design Methodology:

In this lab work, we needed to design circuit with ALU. After some research, I realized that usage of Arithmetic-Logic Unit in VHDl is very easy. If I use unsigned and numeric libraries, I can do any kind of Arithmetic-Logic operation in PROCESS and IF with only using signs such as +, AND, > etc. We needed to do at least 8 operations and so I did these operations:

000 – ADDITION	001 – SUBTRACTION
010 – ADD WITH CARRY	011 – COMPARISON
100 – AND	101 – SHIFT LEFT
110 – SHIFT RIGHT	111 – ROTATE RIGHT

Table 1: Operations

In order to show my design's outputs, I decided to use seven segment displays. In the previous lab, we learned to use seven segment displays and I used these codes with some enhancement in this lab work. As seen below, I used 3-bit binary numbers as inputs (A and B) and I defined these inputs with 3 switches each. Also I needed to "Select" input and I defined this 3-bit binary input with 3 switches. As output, I used seven segment displays' last three digits to show 3-bit binary output for all operations except "add with carry". In the "add with carry" operation, I used 4-bit binary as output and used all 4 digits of seven segment displays. Therefore, I have an "error LED" which shows us there is a problem about operation. I used "error LED" in only two operations that are "addition" and "subtraction". In the addition op., error LED shows there is a carry out digit which we cannot show in 3-bit binary output; in the subtraction op., error LED shows there is needed carry in.

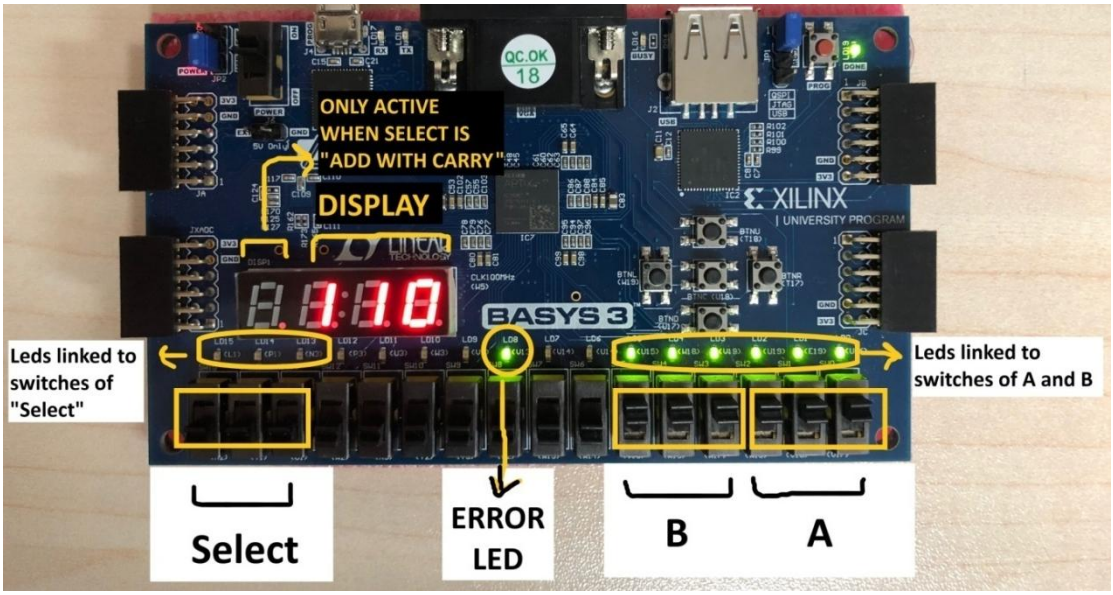

Figure 1: Locations of inputs and outputs

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

Results:

After writing the four modules and top module, codes worked as I expected. There are RTL schematics for design and each module:

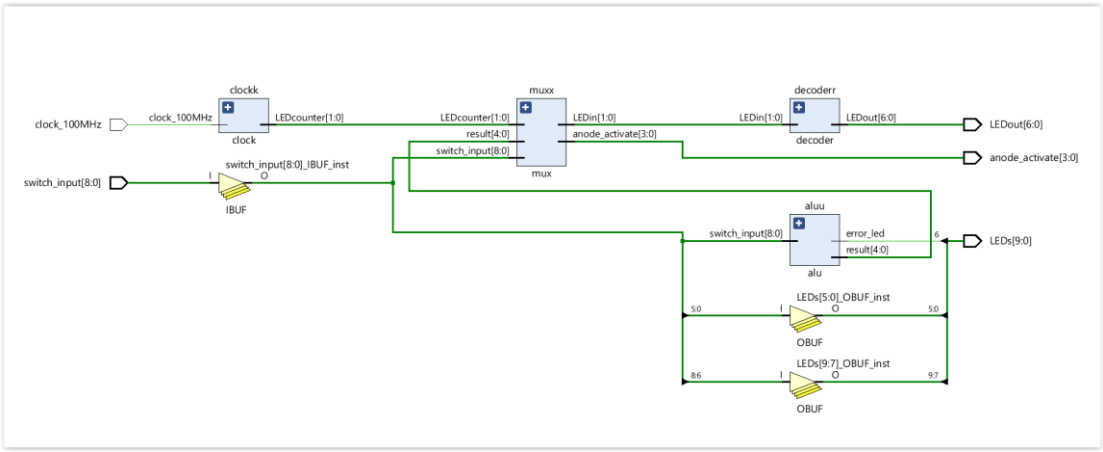

Figure 2: RTL Schematic of Design

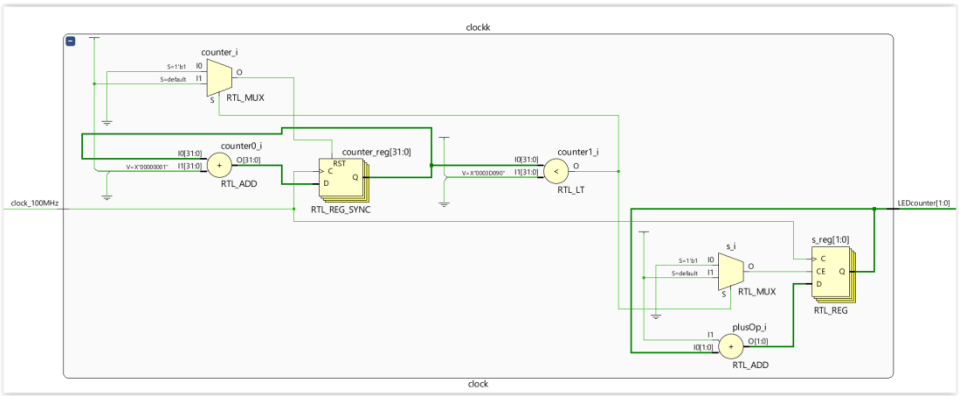

Figure 3: Schematic of Clock Module

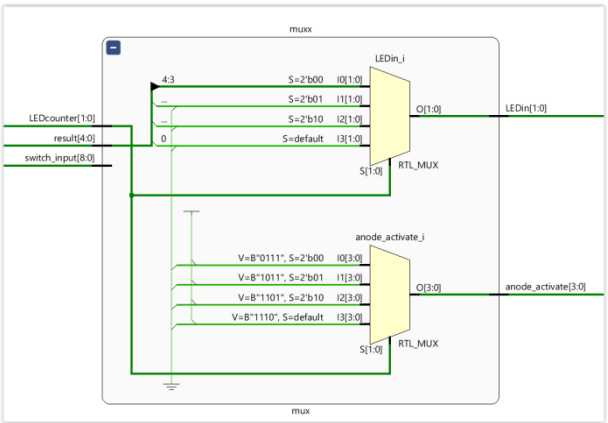

Figure 4: Schematic of Multiplexer Module

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

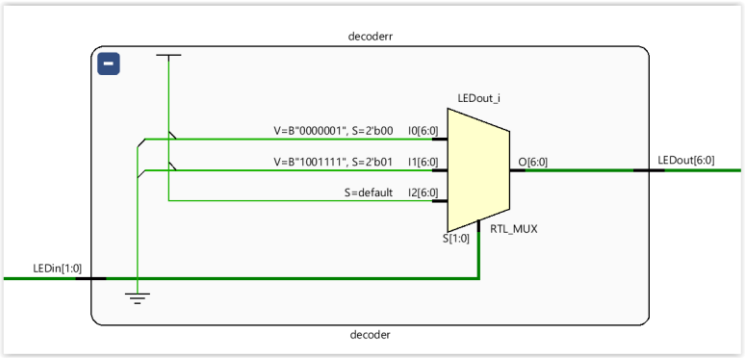

Figure 5: Schematic of Decoder Module

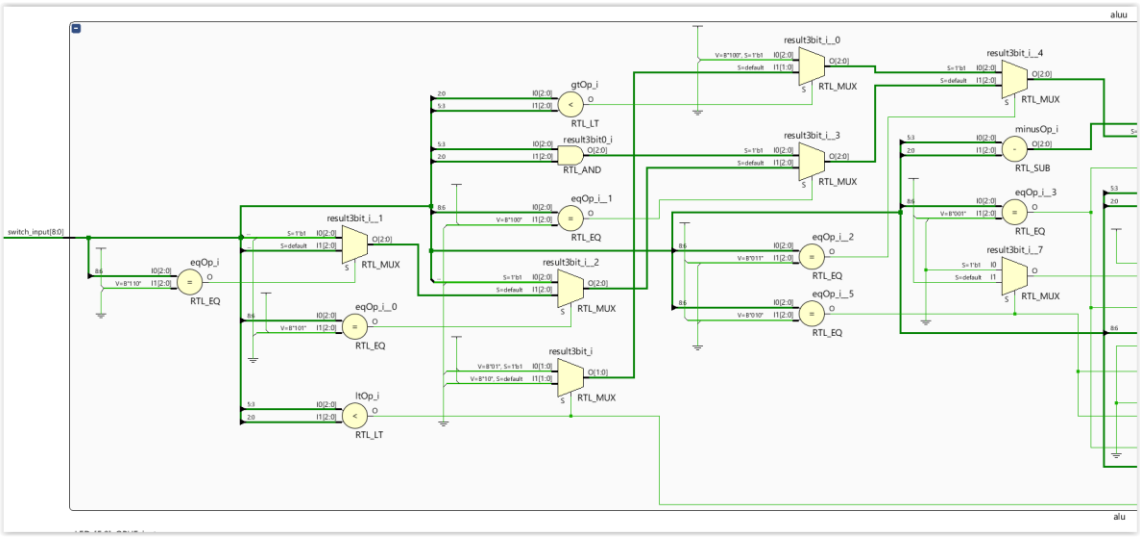

Figure 6: Schematic of First Part of Alu Module

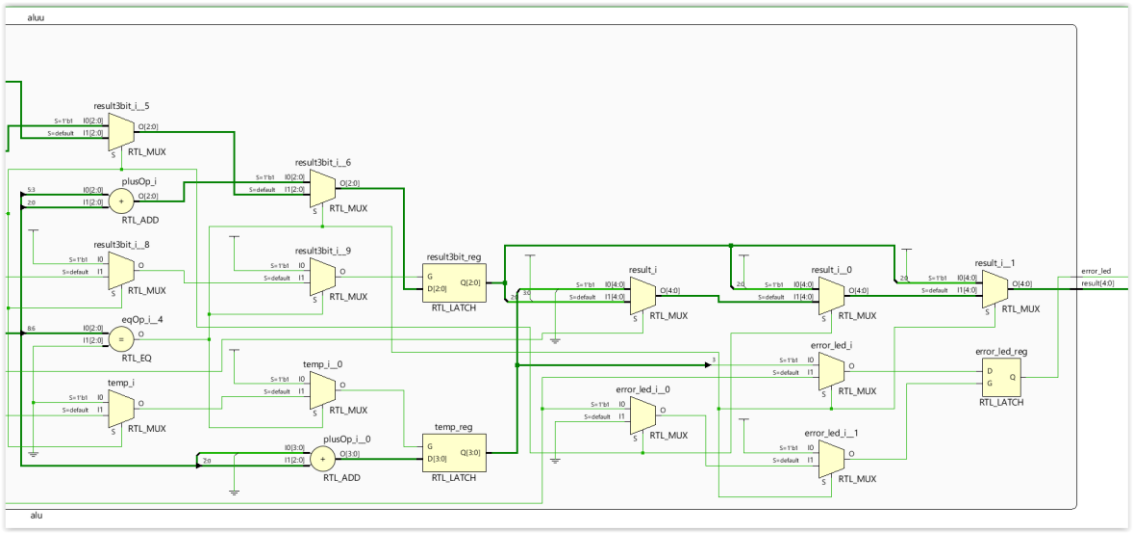

Figure 7: Schematic of Second Part of Alu Module

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

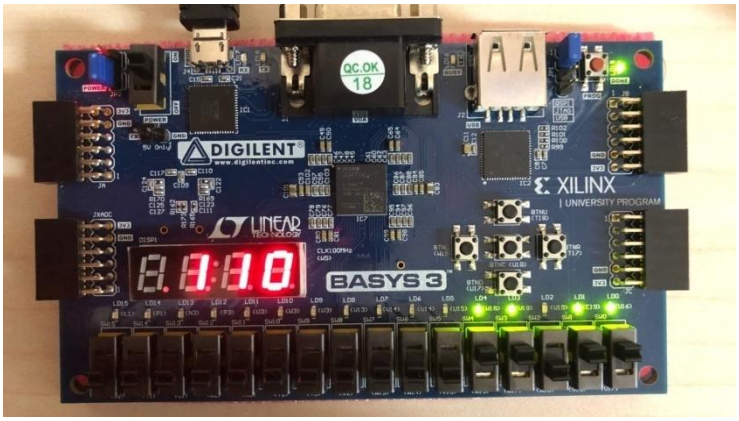

Figure 8: Demo 1 => Addition (000) => 011 + 011 = 110

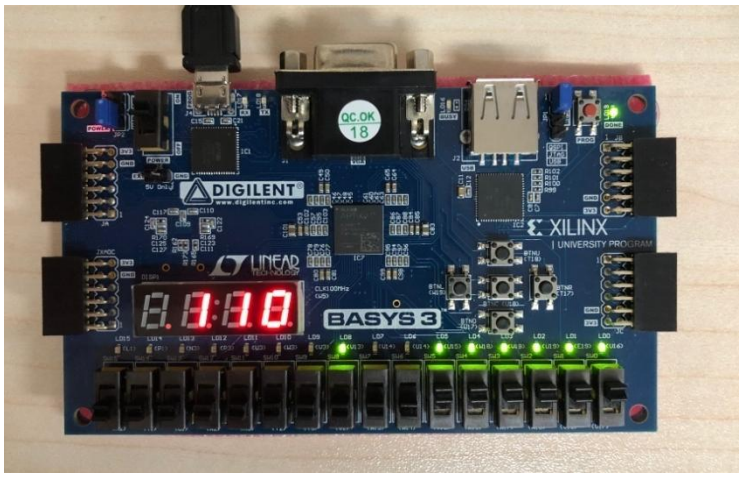

Figure 9: Demo 2 => Addition (000) => 111 + 111 = 110 (Error LED)

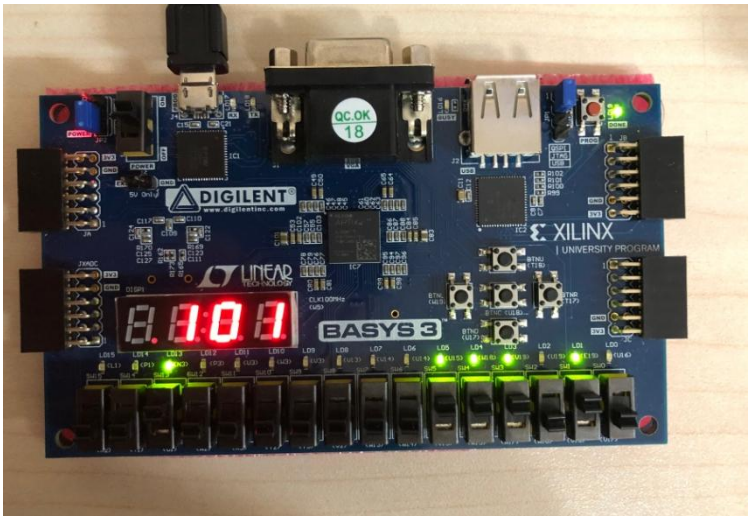

Figure 10: Demo 3 => Subtraction (001) => 111 - 010 = 101

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

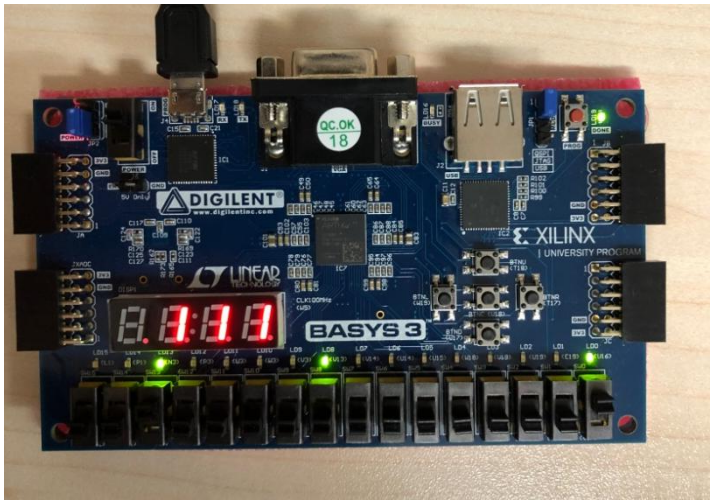

Figure 11: Demo 4 => Subtraction (001) => 000 - 001 = 111 (Error LED)

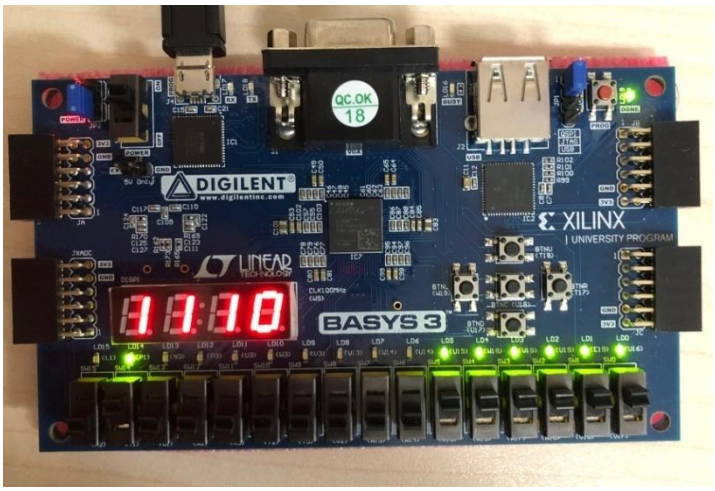

Figure 12: Demo 5 => Add with Carry (010) => 111 + 111 = 1110

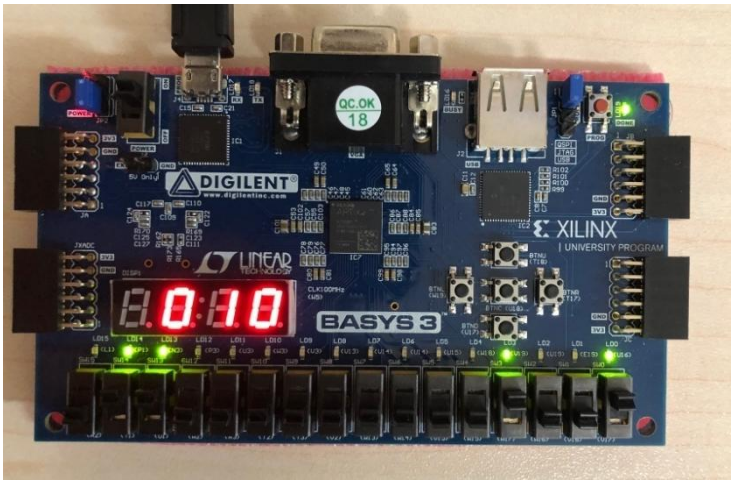

Figure 13: Demo 6 => Comparison (011) => 001 = 001 (Equal means 010 in displays)

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

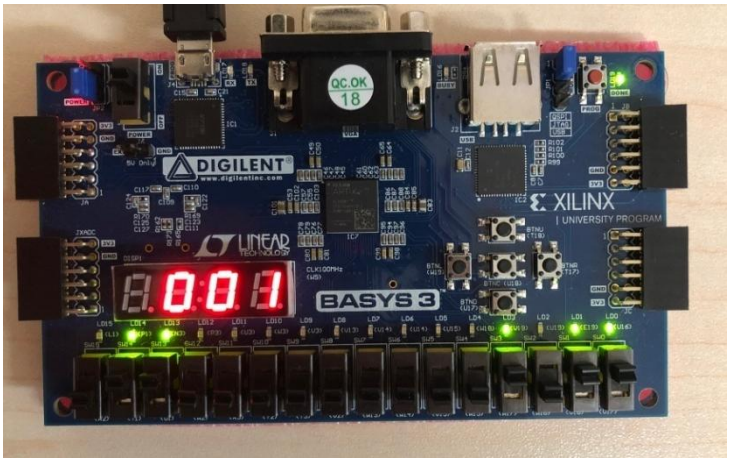

Figure 14: Demo 7 => Comparison (011) => 011 > 001 (A > B means 001)

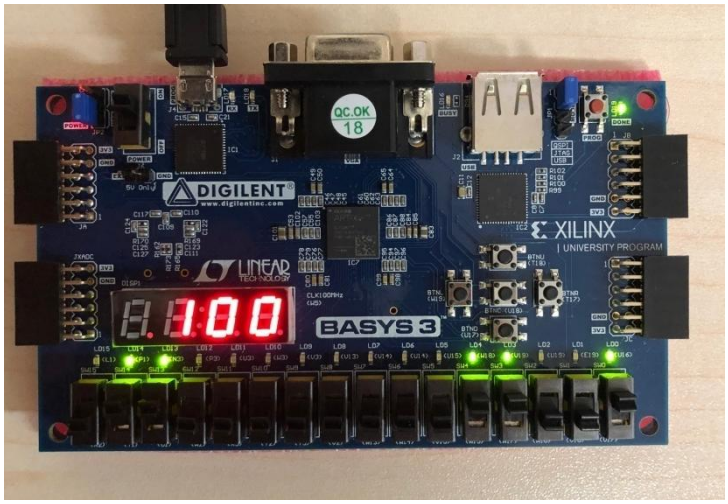

Figure 15: Demo 8 => Comparison (011) => 001 < 011 (A < B means 100)

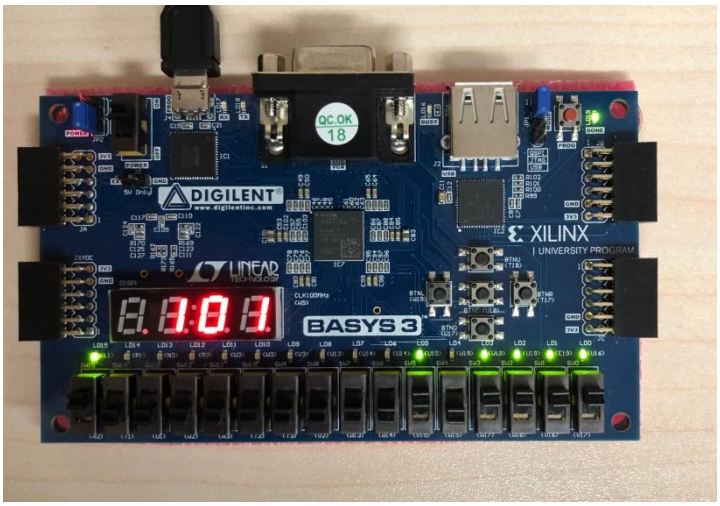

Figure 16: Demo 9 => AND (100) => 111 AND 101 = 101

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

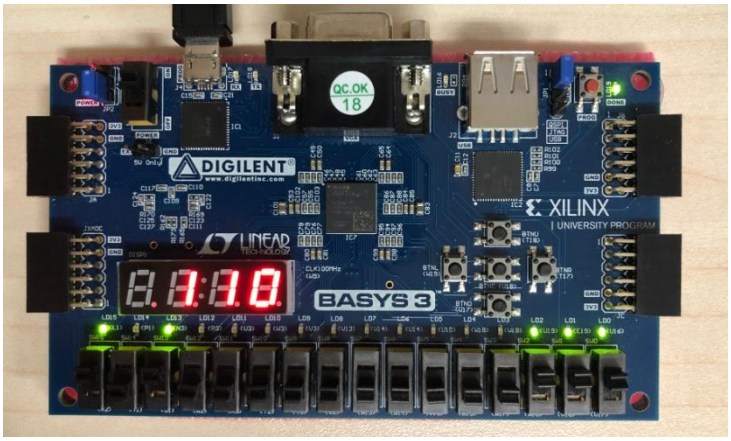

Figure 17: Demo 10 => Shift Left (101) => Shift Left (111) = 110

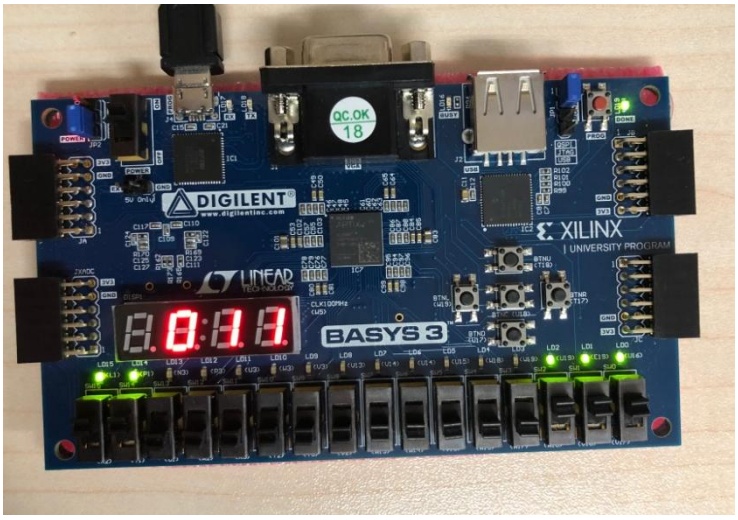

Figure 18: Demo 11 => Shift Right (110) => Shift Right(111) = 011

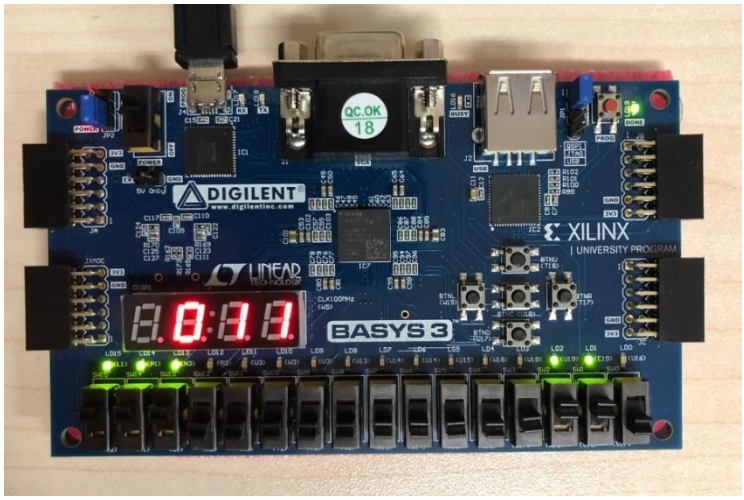

Figure 19: Demo 12 => Rotate Right (111) => Rotate Right(110) = 011

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

Conclusion:

In this lab work, I learned to do arithmetic and logical operations in VHDL. It was easy due to unsigned and numeric libraries. With only usage of mathematical signs of operations, I could reach my intended result. When I writing addition operation, carry out was the problem. But I defined this operation as:

```
result3bit <= switch_input(5 downto 3) + switch_input(2 downto 0);   -- add
temp <= ('0' & switch_input(5 downto 3)) + switch_input(2 downto 0);
error_led <= temp(3);
result <= "11" & result3bit;
```

and I could able to show my 3 digit result and light error LED if there is carry out digit.

Also, I concatenated some two digit vector (LEDin) with result3bit and I tried to open first digit of seven segment display only when we select “add with carry” operation. For other cases, I wanted to hide first digit. For this case, in the decoder module, I used this two digit vector as:

```
case LEDin is
      when "00" => LEDout <= "0000001"; --0
      when "01" => LEDout <= "1001111"; --1
      when others => LEDout <= "1111111"; --no LED activated
end case;
```

and I achieved to hide first digit when operation is not “add with carry”. I did this with concatenated ‘0’ at the beginning of each last three digits’ outputs in the multiplexer, such as:

```
case LEDcounter is
      when "00" =>
         anode_activate <= "0111"; --LED1 is activated
         LEDin <= result(4 downto 3);
      when "01" =>
         anode_activate <= "1011"; --LED2 is activated
         LEDin <= '0' & result(2);
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

So, if LEDin(for first digit of seven segment displays) is “11” or “10”, first LED will not be activated. For all operations except “add with carry”, I used this line of code:

```
result <= "11" & result3bit;
```

In the add with carry operation, I used this lines of code:

```
temp <= ('0' & switch_input(5 downto 3)) + switch_input(2 downto 0); --add with carry
result <= '0' & temp;
```

So, the output of “add with carry” operation will be 4-bit binary and first digit of seven segment displays will work for this operation. Because LEDin for first(for first digit of seven segment displays) will be “01” or “00” for this operation.

Appendices:

Top Module:

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


entity main_lab5 is
  Port(
    switch_input: in std_logic_vector(8 downto 0);
    clock_100MHz: in std_logic;
    LEDout: out std_logic_vector (6 downto 0);
    anode_activate: out std_logic_vector (3 downto 0);
    LEDs: out std_logic_vector (9 downto 0)
    );
end main_lab5;


architecture Behavioral of main_lab5 is
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

```
component clock is
Port (
   clock_100MHz: in std_logic;
   LEDcounter: out std_logic_vector (1 downto 0)
   );
end component;

component mux is -- 4-1 multiplexer ile switchleri ledlere göre 4'lü grupluyoruz
Port (
   LEDcounter: in std_logic_vector (1 downto 0);
   switch_input: in std_logic_vector (8 downto 0);
   LEDin: out std_logic_vector (1 downto 0);
   anode_activate: out std_logic_vector (3 downto 0);
   result: in std_logic_vector (4 downto 0)
   );
end component;

component decoder is -- decoder kullanarak 0001 ile 0000001'i eşleştiriyoruz
Port (
   LEDin: in std_logic_vector(1 downto 0);
   LEDout: out std_logic_vector (6 downto 0)
   );
end component;

component alu is -- decoder kullanarak 0001 ile 0000001'i eşleştiriyoruz
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

```
Port (
   switch_input: in std_logic_vector(8 downto 0);
   result: out std_logic_vector (4 downto 0);
   error_led: out std_logic
   );
end component;

signal LEDcounter: std_logic_vector (1 downto 0);
signal LEDin: std_logic_vector (1 downto 0);
signal result: std_logic_vector (4 downto 0);
signal error_led: std_logic;

begin
LEDs(9 downto 7) <= switch_input(8 downto 6);
LEDs(5 downto 3) <= switch_input(5 downto 3);
LEDs(2 downto 0) <= switch_input(2 downto 0);
LEDS(6) <= error_led;

clockk: clock port map ( clock_100MHz => clock_100MHz, LEDcounter => LEDcounter);
decoderr: decoder port map ( LEDin => LEDin, LEDout => LEDout);
muxx: mux port map ( LEDcounter => LEDcounter, switch_input => switch_input, LEDin => LEDin,
anode_activate => anode_activate, result => result);
aluu: alu port map ( switch_input => switch_input, result => result, error_led => error_led);
end Behavioral;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

ALU module:

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

entity alu is
   Port(
   switch_input: in std_logic_vector(8 downto 0);
   result: out std_logic_vector (4 downto 0);
   error_led: out std_logic
      );
end alu;

architecture Behavioral of alu is
signal temp: std_logic_vector (3 downto 0);
signal result3bit: std_logic_vector (2 downto 0);
begin
process(switch_input)
begin
   if switch_input(8 downto 6) = "000" then
      result3bit <= switch_input(5 downto 3) + switch_input(2 downto 0);   -- add
      temp <= ('0' & switch_input(5 downto 3)) + switch_input(2 downto 0);
      error_led <= temp(3);
      result <= "11" & result3bit;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

```
elsif switch_input(8 downto 6) = "001" then
   result3bit <= switch_input(5 downto 3) - switch_input(2 downto 0); --subtract
   result <= "11" & result3bit;
   if switch_input(5 downto 3) < switch_input(2 downto 0) then
      error_led <= '1';
   end if;
elsif switch_input(8 downto 6) = "010" then
   temp <= ('0' & switch_input(5 downto 3)) + switch_input(2 downto 0); --add with carry
   result <= '0' & temp;
elsif switch_input(8 downto 6) = "011" then
   if switch_input(5 downto 3) > switch_input(2 downto 0) then --comparison
      result3bit <= "100";
      result <= "11" & result3bit;
   elsif switch_input(5 downto 3) < switch_input(2 downto 0) then
      result3bit <= "001";
      result <= "11" & result3bit;
   else
      result3bit <= "010";
      result <= "11" & result3bit;
   end if;
elsif switch_input(8 downto 6) = "100" then
   result3bit <= switch_input(5 downto 3) AND switch_input(2 downto 0); --AND
   result <= "11" & result3bit;
elsif switch_input(8 downto 6) = "101" then
   result3bit <= switch_input(1 downto 0) & '0'; --shift left
   result <= "11" & result3bit;
```

```
    elsif switch_input(8 downto 6) = "110" then
        result3bit <= '0' & switch_input(2 downto 1) ; --shift right
        result <= "11" & result3bit;
    else
        result3bit <= switch_input(0) & switch_input(2 downto 1); --rotate right
        result <= "11" & result3bit;
    end if;
end process;
end Behavioral;
```

Clock module:

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

entity clock is
    Port (
    clock_100MHz: in std_logic;
    LEDcounter: out std_logic_vector (1 downto 0)
    );
end clock;

architecture Behavioral of clock is
    signal counter: integer:= 0;
    signal s    : std_logic_vector (1 downto 0);
begin
```

```
    process(clock_100MHz)
    begin
        if rising_edge(clock_100MHz) then
            if counter < 250000 then
                counter <= counter + 1;
            else
                counter <= 0;
                s <= s+1;
            end if;
        end if;
    end process;

LEDcounter <= std_logic_vector(s);
end Behavioral;
```

Multiplexer Module:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.std_logic_unsigned.all;
entity mux is
    Port (
    LEDcounter: in std_logic_vector (1 downto 0);
    switch_input: in std_logic_vector (8 downto 0);
    LEDin: out std_logic_vector(1 downto 0);
    anode_activate: out std_logic_vector (3 downto 0);
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

```
    result: in std_logic_vector (4 downto 0)
    );
end mux;

architecture Behavioral of mux is
begin
    process(LEDcounter)
    begin
        case LEDcounter is
        when "00" =>
            anode_activate <= "0111"; --LED1 is activated
            LEDin <= result(4 downto 3);
        when "01" =>
            anode_activate <= "1011"; --LED2 is activated
            LEDin <= '0' & result(2);
        when "10" =>
            anode_activate <= "1101"; --LED3 is activated
            LEDin <= '0' & result(1);
        when others =>
            anode_activate <= "1110"; --LED2 is activated
            LEDin <= '0' & result(0);
        end case;
    end process;

end Behavioral;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

Decoder Module:

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


entity decoder is
    Port (
        LEDin: in std_logic_vector(1 downto 0);
        LEDout: out std_logic_vector (6 downto 0)
    );
end decoder;


architecture Behavioral of decoder is
begin
    process(LEDin)
    begin
        case LEDin is
        when "00" => LEDout <= "0000001"; --0
        when "01" => LEDout <= "1001111"; --1


        when others => LEDout <= "1111111"; --no LED activated
        end case;
    end process;


end Behavioral;
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

Constraints:

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clock_100MHz]
   set_property IOSTANDARD LVCMOS33 [get_ports clock_100MHz]

# Switches
set_property PACKAGE_PIN V17 [get_ports {switch_input[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[0]}]
set_property PACKAGE_PIN V16 [get_ports {switch_input[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[1]}]
set_property PACKAGE_PIN W16 [get_ports {switch_input[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[2]}]
set_property PACKAGE_PIN W17 [get_ports {switch_input[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[3]}]
set_property PACKAGE_PIN W15 [get_ports {switch_input[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[4]}]
set_property PACKAGE_PIN V15 [get_ports {switch_input[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[5]}]

set_property PACKAGE_PIN U1 [get_ports {switch_input[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[6]}]
set_property PACKAGE_PIN T1 [get_ports {switch_input[7]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[7]}]
set_property PACKAGE_PIN R2 [get_ports {switch_input[8]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {switch_input[8]}]
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

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

set_property PACKAGE_PIN V13 [get_ports {LEDs[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[6]}]

set_property PACKAGE_PIN N3 [get_ports {LEDs[7]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[7]}]
set_property PACKAGE_PIN P1 [get_ports {LEDs[8]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[8]}]
set_property PACKAGE_PIN L1 [get_ports {LEDs[9]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[9]}]
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 5
02.04.2021

#7 segment display

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

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