Name: Emre Şaş
ID:22102764
Section: 2

Ihsan Dogramaci Bilkent University

EE-102 (Introduction to Digital Design) Lab 4 Report

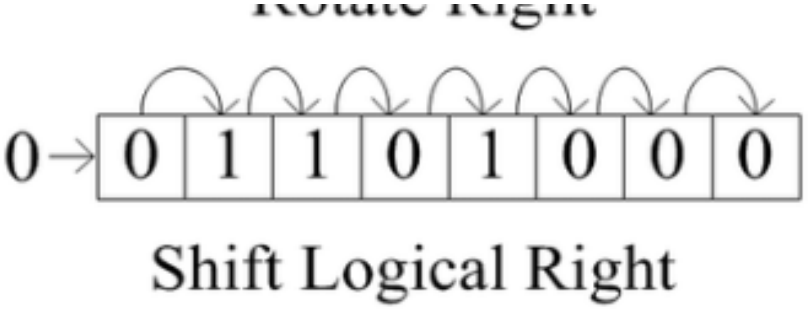
Purpose:

Purpose of this experiment is to designing, creating, simulating a 6-bir arithmetic logic unit (ALU) and implementing it on BASYS3. The ALU designed in this study is eligible to perform eight different functions which are addition, subtraction, logical shift right, logical shift left, arithmetic shift right, One's Complement, bitwise and, rotate right.

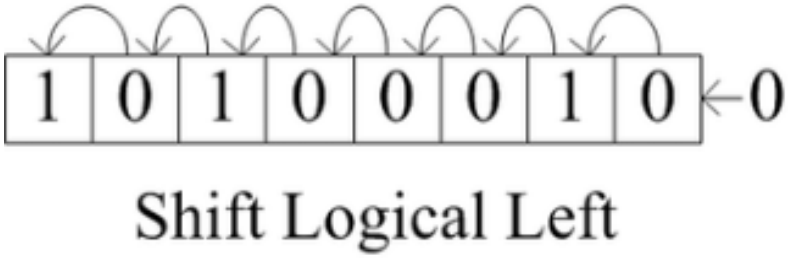
- Addition: A 6-bit addition using full adders is a process of adding two 6-bit binary numbers by employing a series of full adder circuits. Each full adder takes three input bits: two from the binary numbers being added and one from the carry-out of the previous full adder (or as the initial carry-in for the least significant bit). The full adder computes a sum and generates a carry-out, which is then propagated to the next full adder in the series.

- Subtraction: Similar to addition, Using full subtractors, 6-bit subtraction finds the difference between two 6-bit binary numbers. We process each bit from right to left, track borrows, and produce a 6-bit result. No borrow-out means a valid result, while a borrow-out indicates the second number is larger.

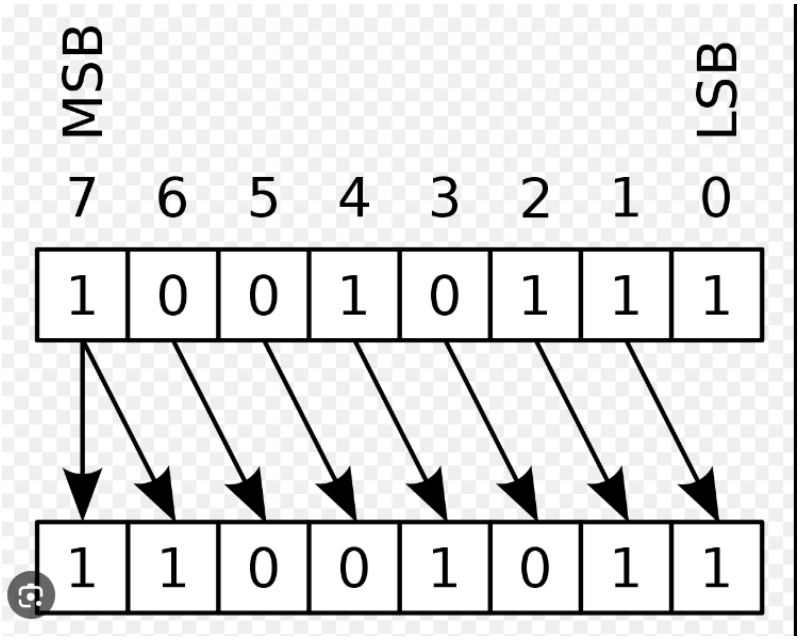
- Logical shift right: Logical shift right is a bitwise operation that shifts the bits of a binary number to the right, filling the vacated bits with zeros. (Figure 1.1)
- Logical shift left : similar to logical shift right, this operation shifts the bits of a binary number to the left, filling the vacated bits with zeros. (Figure 1.2)
- Arithmetic shift right : a bitwise operation that shifts the bits of a binary number to the right while preserving the sign bit (most significant bit), which is shifted into the vacant bits.(Figure 1.3)
- One's Complement : One's complement is a binary number representation in which the negative value of a number is obtained by inverting (changing 0s to 1s and vice versa) all the bits in its positive binary representation. It is a way to represent signed numbers in computer systems, where the leftmost bit typically serves as the sign bit (0 for positive, 1 for negative), and the remaining bits represent the magnitude of the number.
- Bitwise-and: a bitwise binary operation which compares corresponding bits of two binary numbers and sets the result bit to 1 if both input bits are 1.
- Rotate right : bitwise operation that shifts the bits of a binary number to the right while wrapping around any shifted-out bits to the beginning. (Figure 1.4)



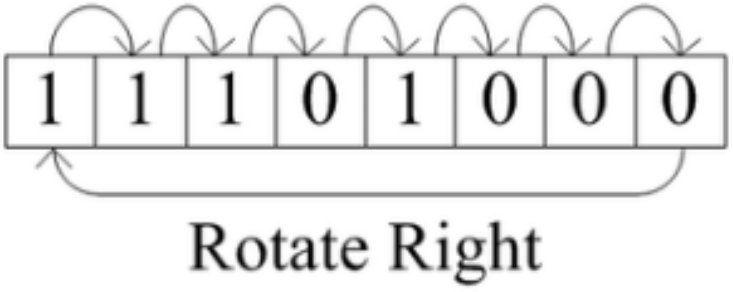
(Figure 1.1)



(Figure 1.2)



(Figure 1.3)



(Figure 1.4)

Methodology:

Although the modules that are defined above are available in Vivado libraries, I constructed these modules without using ready-to-use functions. I used separate modules and integrated them in top module (ALU_2) by using the commands COMPONENT, PORT MAP and the functions created. Therefore, the ALU created is like a multifunctional machine that consist of machines that are created for only one operation. After design phase RTL schematic (Figure 2.1) created, most simplified version can be observed.

In order to be sure whether the design is proper ALU_2_test (Simulation source) is created, and the results can be seen in Figure (2.2). After being sure about design , depending on the test results, constraints were added and each input (every bit of two vectors) were assigned to switches while outputs assigned to LED's.

Design specifications:

Mentioned functions are created under these separate modules and combined in ALU_2:

- six_bit_adder.vhd
- six_bit_substractor.vhd
- logical_shift_right.vhd
- logical_shift_left.vhd
- One_comp.vhd
- bitwise_and.vhd
- rotate_rright.vhd
- arithmetic_shift_right.vhd

In order to select which function is to be used, selection function should be added to ALU_2. Since we will be using 8 functions 3-bit vector would be enough to create options. We can use 2^3=8 different combinations.

- Addition >"000"
- Subtraction >"001"
- Logical shift left>"010"
- Arithmetic shift right>"011"
- Logical shift right>"100"
- One_comp >"101"
- Rotate right >"110"
- Bitwise AND>"111"

Conclusion:

To sum up, design created for ALU was working properly and all of the functions were giving the correct results. Addition (Figure 3.1), Substraction (Figure 3.2), Logical shift left(Figure 3.3), Logical shift right (Figure 3.4), One's Complement (Figure 3.5),rotate right (Figure 3.6), Bitwise AND (Figure 3.7), arithmetic shift right (Figure 3.8) was successfully transferred to BASY3.

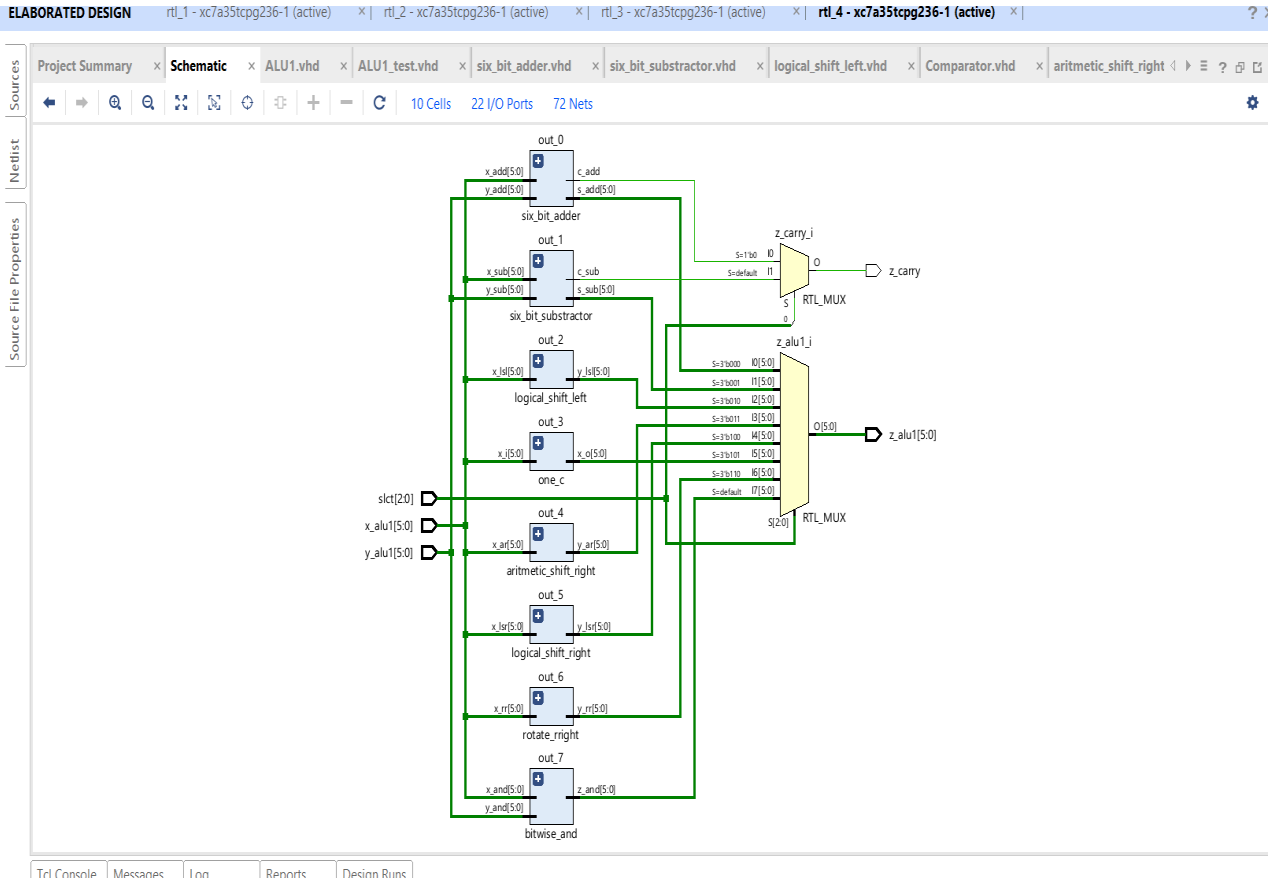


Figure 2.1

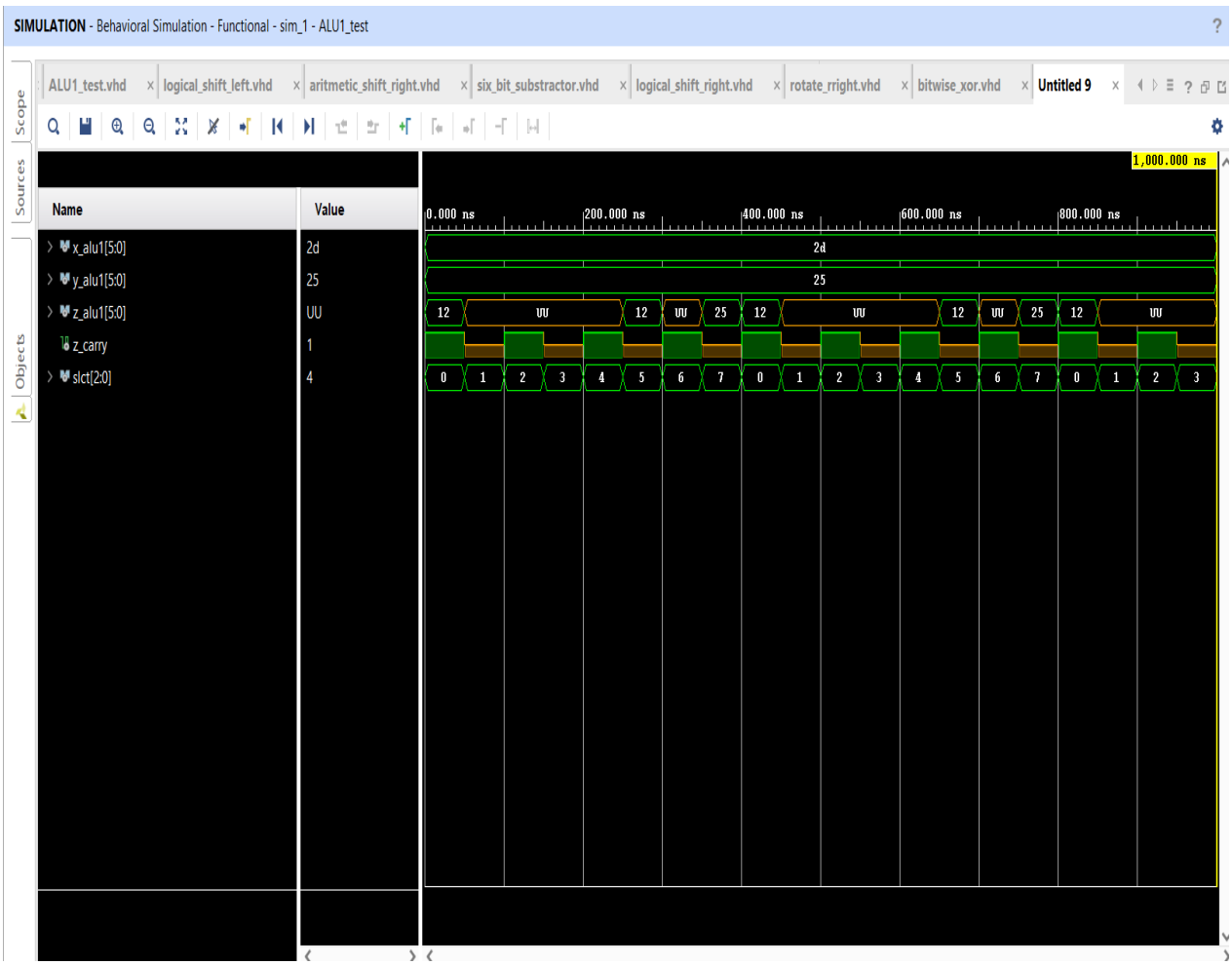


Figure 2.2

Appendices:

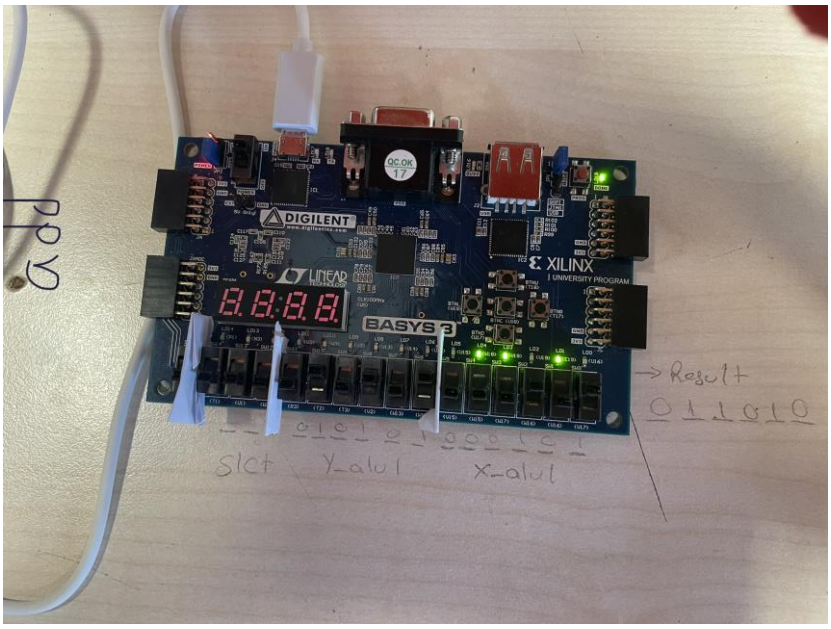


Figure 3.1

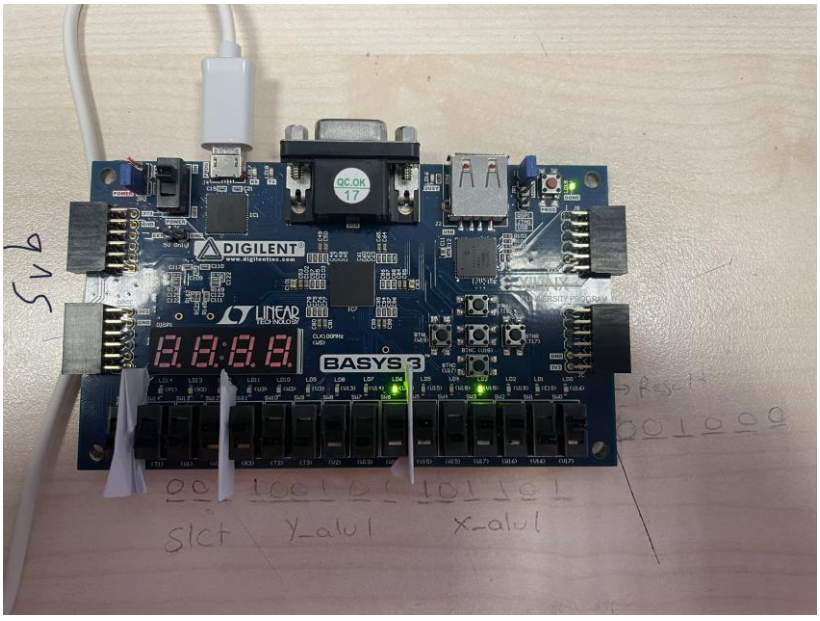


Figure 3.2

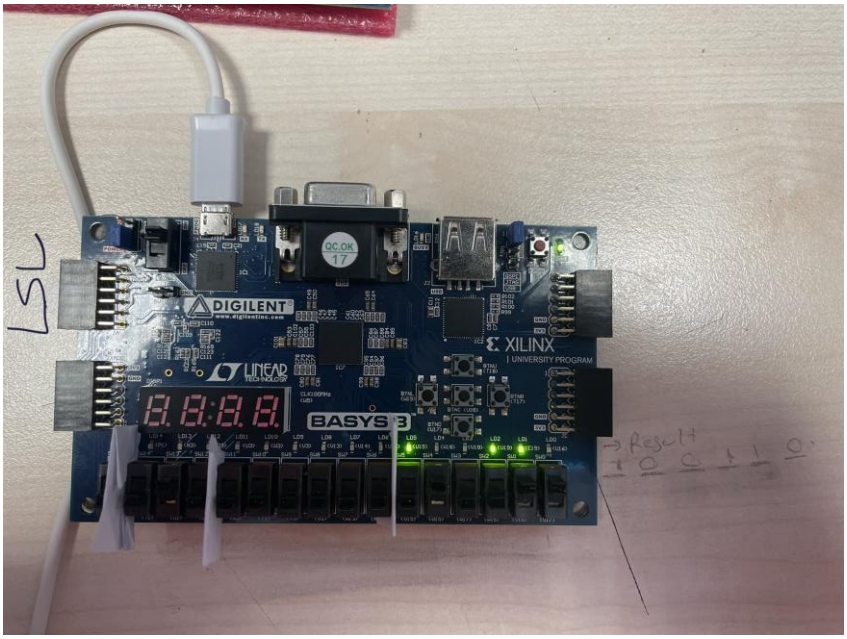


Figure 3.3

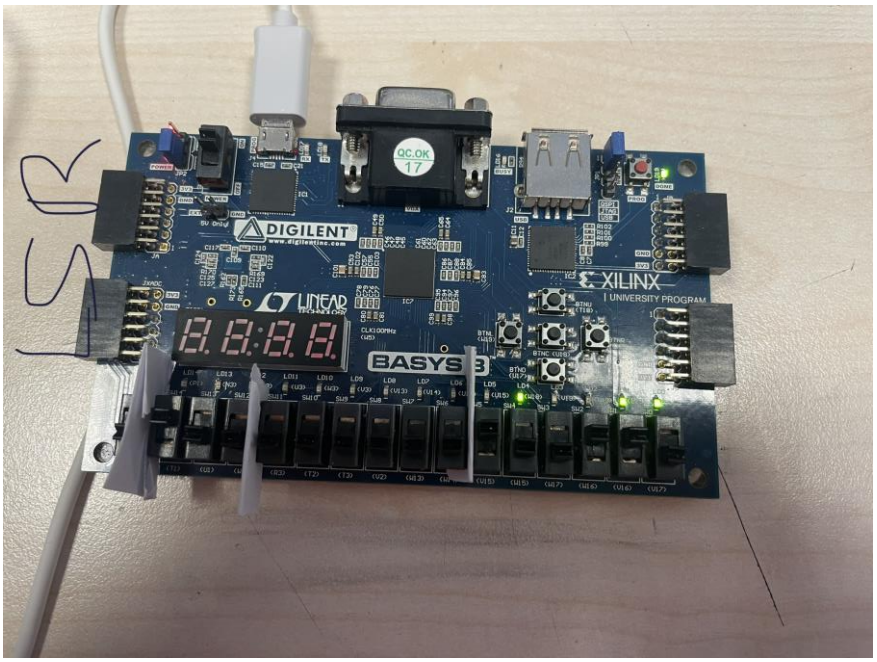


Figure 3.4

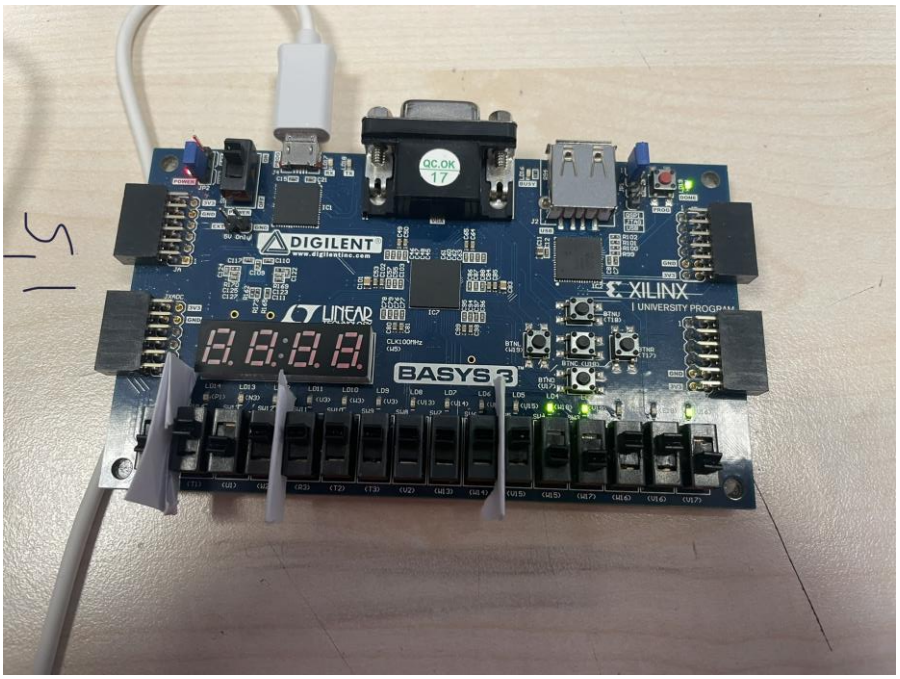


Figure 3.5

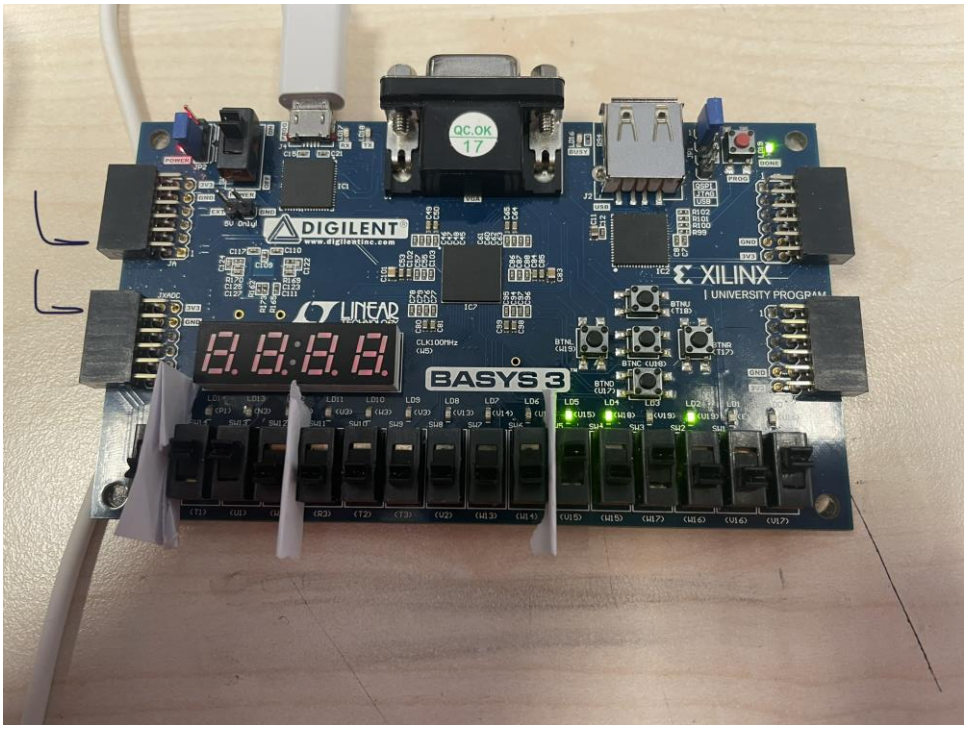


Figure 3.6

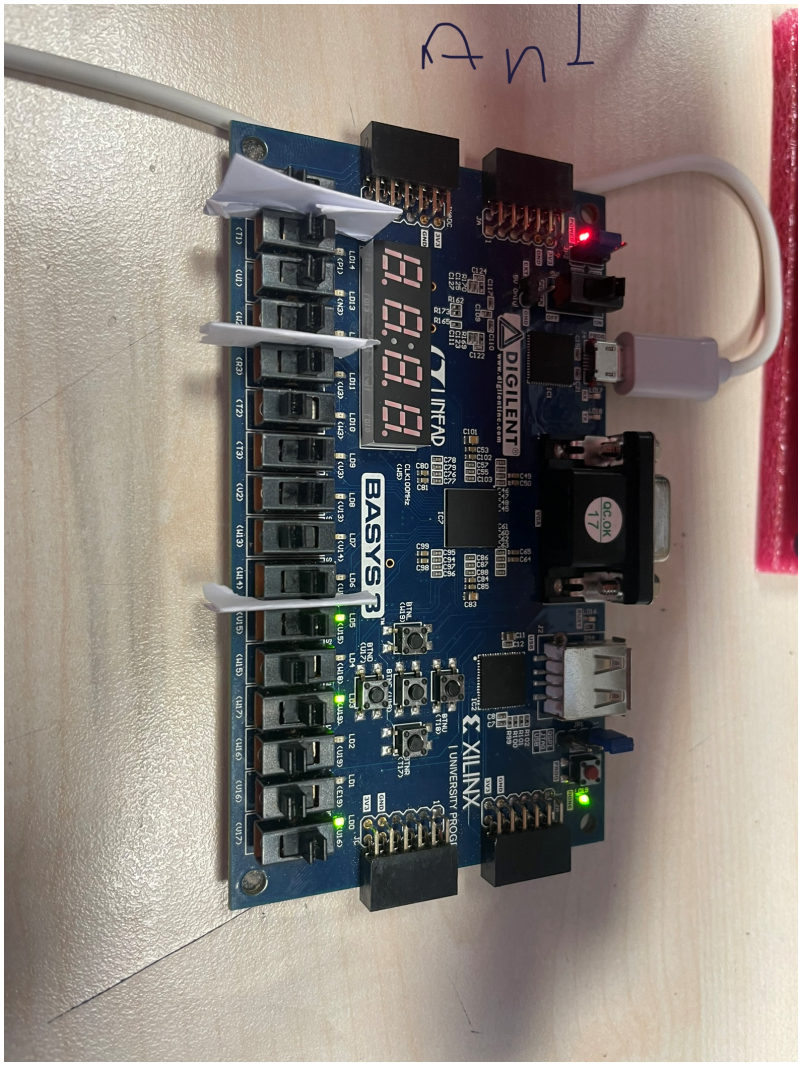


Figure 3.7

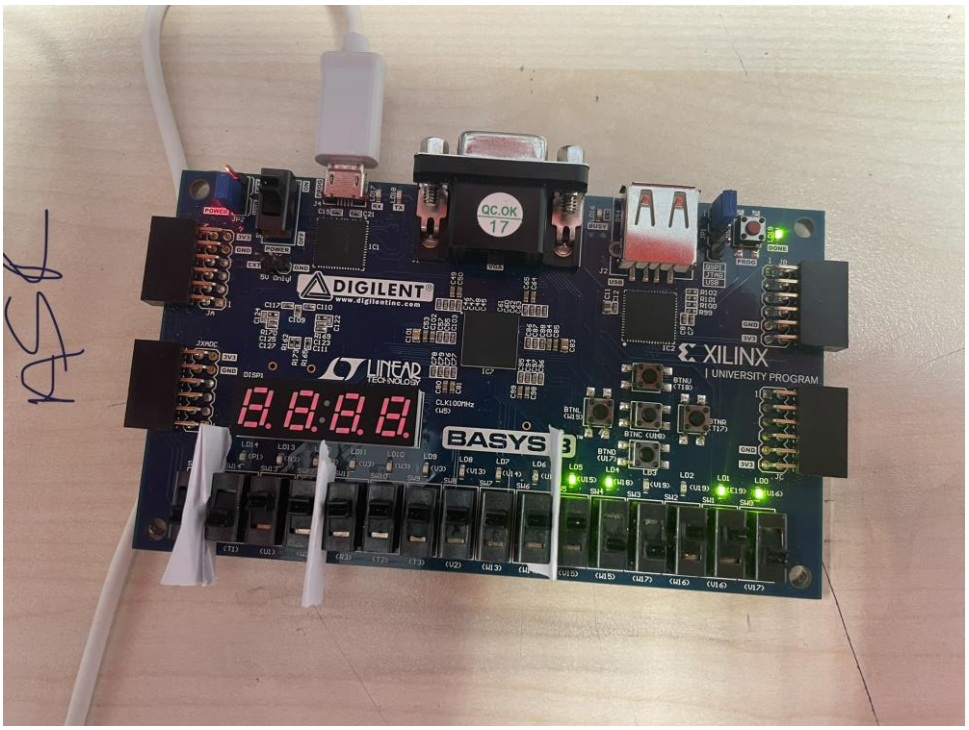


Figure 3.8

TOP MODULE Codes:

-- Company:

-- Engineer:Emre Şa$

--

-- Create Date: 21.10.2023 15:48:21

-- Design Name:

-- Module Name: ALU_2 - Behavioral

-- Project Name:

```
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ALU1 is
    Port (x_alu1: in STD_LOGIC_VECTOR (5 DOWNTO 0);
```

```
        y_alu1: in STD_LOGIC_VECTOR (5 DOWNTO 0);
        z_alu1: out STD_LOGIC_VECTOR (5 DOWNTO 0);
        z_carry: out STD_LOGIC;
-- slct is a 3 bit vector to choose the mode
        slct: in STD_LOGIC_VECTOR (2 DOWNTO 0));
end ALU1;

architecture Behavioral of ALU1 is
-- add components six_bit_adder, six_bit_substracter,logical_shift_left, comparator, aritmetic_shift_right, logical_shift_right, rotate_right, bitwise_xor

--create component six_bit_adder

COMPONENT six_bit_adder
PORT (x_add :in STD_LOGIC_VECTOR (5 DOWNTO 0);
    y_add :in STD_LOGIC_VECTOR (5 DOWNTO 0);
    c_add: out std_logic;
    s_add: out STD_LOGIC_VECTOR (5 DOWNTO 0));
END COMPONENT;

Component one_c
port (x_i: in  STD_LOGIC_VECTOR(5 downto 0);
        x_o : out STD_LOGIC_VECTOR(5 downto 0)
        );
end component;

COMPONENT six_bit_substractor
```

```
PORT (x_sub :in STD_LOGIC_VECTOR (5 DOWNTO 0);
     y_sub :in STD_LOGIC_VECTOR (5 DOWNTO 0);
     c_sub: out std_logic;
     s_sub: out STD_LOGIC_VECTOR (5 DOWNTO 0));
END COMPONENT;

COMPONENT logical_shift_left
Port ( x_lsl : in STD_LOGIC_VECTOR (5 downto 0);
      y_lsl : out STD_LOGIC_VECTOR (5 downto 0));
END COMPONENT;

COMPONENT aritmetic_shift_right
Port ( x_ar : in STD_LOGIC_VECTOR (5 downto 0);
      y_ar : out STD_LOGIC_VECTOR (5 downto 0));
END COMPONENT;

COMPONENT logical_shift_right
Port ( x_lsr : in STD_LOGIC_VECTOR (5 downto 0);
       y_lsr : out STD_LOGIC_VECTOR (5 downto 0));
END COMPONENT;

COMPONENT rotate_rright
Port ( x_rr : in STD_LOGIC_VECTOR (5 downto 0);
       y_rr : out STD_LOGIC_VECTOR (5 downto 0));
END COMPONENT;

COMPONENT bitwise_and
```

```
Port ( x_and : in STD_LOGIC_VECTOR (5 downto 0);
       y_and : in STD_LOGIC_VECTOR (5 downto 0);
       z_and : out STD_LOGIC_VECTOR (5 downto 0));
END COMPONENT;

SIGNAL signal_0: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_1: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_2: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_3: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_4: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_5: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_6: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_7: STD_LOGIC_VECTOR (5 downto 0);
SIGNAL signal_carry_0 : STD_LOGIC;
SIGNAL signal_carry_1 : STD_LOGIC;


begin

out_0: six_bit_adder port map (x_alu1,y_alu1, signal_carry_0, signal_0);
out_1: six_bit_substractor port map (x_alu1,y_alu1, signal_carry_1, signal_1);
out_2: logical_shift_left port map (x_alu1, signal_2);
out_3: one_c Port Map( x_alu1,signal_5);
out_4: aritmetic_shift_right port map(x_alu1 ,signal_3);
out_5:logical_shift_right port map( x_alu1, signal_4);
out_6: rotate_rright  port map  ( x_alu1, signal_6);
out_7: bitwise_and port map ( x_alu1,y_alu1, signal_7);
```

with slct select

z_alu1 <= signal_0 when "000",

signal_1 when "001",

signal_2 when "010",

signal_3 when "011",

signal_4 when "100",

signal_5 when "101",

signal_6 when "110",

signal_7 when others ;

with slct(0) select

z_carry <= signal_carry_0 when '0',

signal_carry_1 when others ;

end Behavioral;

TEST BENCH:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 16:57:04
-- Design Name:
-- Module Name: ALU2_test - Behavioral
-- Project Name:
-- Target Devices:
```

```
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ALU1_test is
end ALU1_test;

architecture Behavioral of ALU1_test is
component ALU1
Port (x_alu1: in STD_LOGIC_VECTOR (5 DOWNTO 0);
      y_alu1: in STD_LOGIC_VECTOR (5 DOWNTO 0);
```

```
        z_alu1: out STD_LOGIC_VECTOR (5 DOWNTO 0);
        z_carry: out STD_LOGIC;
-- slct is a 3 bit vector to choose the mode
        slct: in STD_LOGIC_VECTOR (2 DOWNTO 0));
        end component ;

signal x_alu1: STD_LOGIC_VECTOR (5 downto 0);
signal y_alu1: STD_LOGIC_VECTOR (5 downto 0);
signal z_alu1: STD_LOGIC_VECTOR (5 downto 0);
signal z_carry: STD_LOGIC ;
signal slct: STD_LOGIC_VECTOR (2 downto 0);

begin
UUT: ALU1 PORT MAP(
x_alu1 => x_alu1,
y_alu1 => y_alu1,
z_alu1 => z_alu1,
z_carry => z_carry,
slct => slct
);
Test_Bench: PROCESS
begin
slct <= "000";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "001";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "010";
```

```
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "011";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "100";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "101";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "110";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
slct <= "111";
x_alu1 <= "101101";
y_alu1 <= "100101";
wait for 50ns;
end process ;
end Behavioral;
```

CODES FOR six_bit_adder:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer: Emre Şaş
--
```

```
-- Create Date: 21.10.2023 19:29:38
-- Design Name:
-- Module Name: six_bit_adder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 six_bit_adder is
--  Port ( );
```

```
    Port ( x_add : in STD_LOGIC_VECTOR (5 downto 0);
           y_add : in STD_LOGIC_VECTOR (5 downto 0);
           c_add : out STD_LOGIC;
           s_add : out STD_LOGIC_VECTOR (5 downto 0));

  end six_bit_adder;

architecture Behavioral of six_bit_adder is
SIGNAL carry_signal : STD_LOGIC_VECTOR (4 downto 0);
COMPONENT full_adder
PORT (x_1 : in STD_LOGIC;
y_1 : in STD_LOGIC;
c_1 : in STD_LOGIC;
s_1 : out STD_LOGIC;
c_2 : out STD_LOGIC);
end component;


begin
SBA1: full_adder PORT MAP (x_add(0),y_add(0),'0',s_add(0),carry_signal(0));
SBA2: full_adder PORT MAP (x_add(1),y_add(1),carry_signal(0),s_add(1),carry_signal(1));
SBA3: full_adder PORT MAP (x_add(2),y_add(2),carry_signal(1),s_add(2),carry_signal(2));
SBA4: full_adder PORT MAP (x_add(3),y_add(3),carry_signal(2),s_add(3),carry_signal(3));
SBA5: full_adder PORT MAP (x_add(4),y_add(4),carry_signal(3),s_add(4),carry_signal(4));
SBA6: full_adder PORT MAP (x_add(5),y_add(5),carry_signal(4),s_add(5),c_add);
end Behavioral;
```

CODES FOR six_bit_substractor:

```
-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 19:43:21
-- Design Name:
-- Module Name: six_bit_substractor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 six_bit_substractor is
Port ( x_sub : in STD_LOGIC_VECTOR (5 downto 0);
y_sub : in STD_LOGIC_VECTOR (5 downto 0);
c_sub : out STD_LOGIC;
s_sub : out STD_LOGIC_VECTOR (5 downto 0));

--  Port ( );
end six_bit_substractor;

architecture Behavioral of six_bit_substractor is
SIGNAL carry_signal_f : STD_LOGIC_VECTOR (4 downto 0);
SIGNAL carry_signal_g : STD_LOGIC_VECTOR (5 downto 0);

component full_adder
PORT (
x_1 : in STD_LOGIC;
y_1 : in STD_LOGIC;
c_1 : in STD_LOGIC;
s_1 : out STD_LOGIC;
c_2 : out STD_LOGIC);
END COMPONENT;
---One's complement should be generated to be able to perform x_sub + (-y_sub) after that full_adder module can be used. and instead of 0 1 should be added
begin
process is
begin
for m in 0 TO 5 loop
carry_signal_g(m) <= not y_sub(m);
end loop ;
wait;
```

```
end process ;
SBS1: full_adder PORT MAP
(x_sub(0),carry_signal_g(0),'1',s_sub(0),carry_signal_f(0));
SBS2: full_adder PORT MAP
(x_sub(1),carry_signal_g(1),carry_signal_f(0),s_sub(1),carry_signal_f(1));
SBS3: full_adder PORT MAP
(x_sub(2),carry_signal_g(2),carry_signal_f(1),s_sub(2),carry_signal_f(2));
SBS4: full_adder PORT MAP
(x_sub(3),carry_signal_g(3),carry_signal_f(2),s_sub(3),carry_signal_f(3));
SBS5: full_adder PORT MAP
(x_sub(4),carry_signal_g(4),carry_signal_f(3),s_sub(4),carry_signal_f(4));
SBS6: full_adder PORT MAP
(x_sub(5),carry_signal_g(5),carry_signal_f(4),s_sub(5),c_sub);
end Behavioral;
```

CODES FOR LOGICAL SHIFT LEFT:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 19:54:34
-- Design Name:
-- Module Name: logical_shift_left - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
```

```
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 logical_shift_left is
PORT( x_lsl: in std_logic_vector (5 downto 0);
      y_lsl: out std_logic_vector (5 downto 0));


--  Port ( );
end logical_shift_left;

architecture Behavioral of logical_shift_left is
```

```
begin
y_lsl(0)<= '0';
process is
begin
for i in 0 to 4 loop
y_lsl(i+1)<=x_lsl(i);
end loop;
wait;
end process;

end Behavioral;
```

CODES FOR ARITHMETIC SHIFT RIGHT:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 23:33:01
-- Design Name:
-- Module Name: aritmetic_shift_right - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

```
--
----------------------------------------------------------------------------------


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 aritmetic_shift_right is
port( x_ar: in std_logic_vector (5 downto 0);
      y_ar: out std_logic_vector (5 downto 0));

--  Port ( );
end aritmetic_shift_right;

architecture Behavioral of aritmetic_shift_right is

begin
y_ar(5)<= x_ar(5);
process is
begin
for i in 0 to 4 loop
y_ar(4-i)<=x_ar(5-i);
```

```
end loop;
wait;
end process;


end Behavioral;
```

CODES FOR LOGICAL SHIFT RIGHT:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 23:56:51
-- Design Name:
-- Module Name: logical_shift_right - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 logical_shift_right is
port( x_lsr: in std_logic_vector (5 downto 0);
      y_lsr: out std_logic_vector (5 downto 0));
--  Port ( );
end logical_shift_right;

architecture Behavioral of logical_shift_right is
begin
y_lsr(5)<= '0';
process is
begin
for i in 0 to 4loop
y_lsr(4-i)<=x_lsr(5-i);
end loop;
wait;
end process;
```

```
end Behavioral;
```

CODES FOR ONE'S COMPLEMENT:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer: Emre Şaş
--
-- Create Date: 21.10.2023 19:29:38
-- Design Name:
-- Module Name: one_c
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity one_c is
    Port ( x_i : in  STD_LOGIC_VECTOR(5 downto 0);
           x_o : out STD_LOGIC_VECTOR(5 downto 0));
end one_c;

architecture Behavioral of one_c is
begin
    process(x_i)
    begin
        for i in 0 to 5 loop
            -- Invert each bit in the one's complement
            x_o(i) <= not x_i(i);
        end loop;
    end process;
end Behavioral;
```

CODES FOR ROTATE RİGHT

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer: E
--
-- Create Date: 22.10.2023 00:57:31
```

```
-- Design Name:
-- Module Name: rotate_rright - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 rotate_rright is
--  Port ( );
PORT ( x_rr: in std_logic_vector (5 downto 0);
```

```
    y_rr: out std_logic_vector (5 downto 0));
end rotate_rright;



architecture Behavioral of rotate_rright is
begin
y_rr(5)<= x_rr(0);
process is
begin
for i in 1 to 5 loop
y_rr(i-1)<=x_rr(i);
end loop;
wait;
end process;



end Behavioral;
```

CODES FOR BITWISE AND:

```
----------------------------------------------------------------------------------
-- Company: zıpçıktı aş
-- Engineer: Emre şaş
--
-- Create Date: 22.10.2023 01:01:27
-- Design Name:
-- Module Name: bitwise_and - Behavioral
```

```
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 bitwise_and is
--  Port ( );
Port ( x_and : in STD_LOGIC_VECTOR (5 downto 0);
y_and : in STD_LOGIC_VECTOR (5 downto 0);
z_and : out STD_LOGIC_VECTOR (5 downto 0));
```

```
end bitwise_and;

architecture Behavioral of bitwise_and is
begin
z_and(0) <= x_and(0) and y_and(0);
z_and(1) <= x_and(1) and y_and(1);
z_and(2) <= x_and(2) and y_and(2);
z_and(3) <= x_and(3) and y_and(3);
z_and(4) <= x_and(4) and y_and(4);
z_and(5) <= x_and(5) and y_and(5);



end Behavioral;
```

CODES FOR FULL ADDER

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 19:37:13
-- Design Name:
-- Module Name: full_adder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 full_adder is
Port ( x_1 : in STD_LOGIC;
y_1 : in STD_LOGIC;
c_1 : in STD_LOGIC;
s_1 : out STD_LOGIC;
c_2 : out STD_LOGIC);

--  Port ( );
end full_adder;

architecture Behavioral of full_adder is
begin
s_1<= (x_1 XOR y_1 XOR c_1);
c_2 <= (x_1 AND y_1) OR (x_1 AND c_1) OR (y_1 AND c_1);




end Behavioral;
```

CODE FOR CONSTRAINTS:

```
# CONSTRAINTS WERE TAKEN FROM ADDITIONAL MATERIALS FROM MOODLE
set_property PACKAGE_PIN V17 [get_ports {x_alu1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x_alu1[0]}]
set_property PACKAGE_PIN V16 [get_ports {x_alu1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x_alu1[1]}]
set_property PACKAGE_PIN W16 [get_ports {x_alu1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x_alu1[2]}]
set_property PACKAGE_PIN W17 [get_ports {x_alu1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x_alu1[3]}]
set_property PACKAGE_PIN W15 [get_ports {x_alu1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x_alu1[4]}]
set_property PACKAGE_PIN V15 [get_ports {x_alu1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x_alu1[5]}]
set_property PACKAGE_PIN W14 [get_ports {y_alu1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {y_alu1[0]}]
set_property PACKAGE_PIN W13 [get_ports {y_alu1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {y_alu1[1]}]
set_property PACKAGE_PIN V2 [get_ports {y_alu1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {y_alu1[2]}]
set_property PACKAGE_PIN T3 [get_ports {y_alu1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {y_alu1[3]}]
set_property PACKAGE_PIN T2 [get_ports {y_alu1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {y_alu1[4]}]
set_property PACKAGE_PIN R3 [get_ports {y_alu1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {y_alu1[5]}]
set_property PACKAGE_PIN W2 [get_ports {slct[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {slct[0]}]
set_property PACKAGE_PIN U1 [get_ports {slct[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {slct[1]}]
```

```
set_property PACKAGE_PIN T1 [get_ports {slct[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {slct[2]}]
#set_property PACKAGE_PIN R2 [get_ports {slct[15]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {slct[15]}]


# LEDs
set_property PACKAGE_PIN U16 [get_ports {z_alu1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_alu1[0]}]
set_property PACKAGE_PIN E19 [get_ports {z_alu1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_alu1[1]}]
set_property PACKAGE_PIN U19 [get_ports {z_alu1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_alu1[2]}]
set_property PACKAGE_PIN V19 [get_ports {z_alu1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_alu1[3]}]
set_property PACKAGE_PIN W18 [get_ports {z_alu1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_alu1[4]}]
set_property PACKAGE_PIN U15 [get_ports {z_alu1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_alu1[5]}]
set_property PACKAGE_PIN U14 [get_ports {z_carry}]
set_property IOSTANDARD LVCMOS33 [get_ports {z_carry}]
```