Atalay Üstündağ

22203554

EE102-03

24.10.2023

Lab 04: Arithmetic Logic Unit

1)Purpose of the Experiment

This experiment aims to teach how to construct an Arithmetic Logic Unit with using VHDL. While constructing this ALU, it was crucial to understand how to use the functions “port map, process, port, entity, architecture”.

2) Methodology of Design

The main goal of this lab is to create an ALU with using two 4-bit numbers. . An ALU is a combinational circuit that can perform mathematical operations and is an essential building block for most digital devices. In my ALU there are 8 operations which are summation, substraction, and operation, shift operation, left rotation, taking complement, increment and decrement. And in my ALU, there are 3 inputs which are 4 bus number one (in_num1), 4 bus number two (in_num2) and 3 bus select input (selectt) . As the outputs of my ALU, I have an overflow output (last_overflow) and result output (CONDITION).

Code	Operation
000	Four Bit Adder
001	Four Bit Substractor
010	Four bit AND of number1 and number 2
011	Four bit Logical Shift
100	Left Rotate
101	Four Bit One's Complement
110	Decrement by one
111	Increment by one

Table 1:Code and Operation Table

In the beginning of the experiment, I wrote an 1 Bit Full Adder in order to create an 4 Bit Full Adder later. Then in another design source, I wrote a 4 Bit Full Adder with calling the 1 Bit Full Adder codes which I wrote one step before. And while calling 1 One Bit Full Adder codes, I used port map function. And in the third step, I wrote an design source for ALU and in this design source I wrote the first 4 operations which are in the Table 1 . The reason was that they need an 4 bit full adder. After I was done with the port map for four of the operations the multiplexer will be prepared.

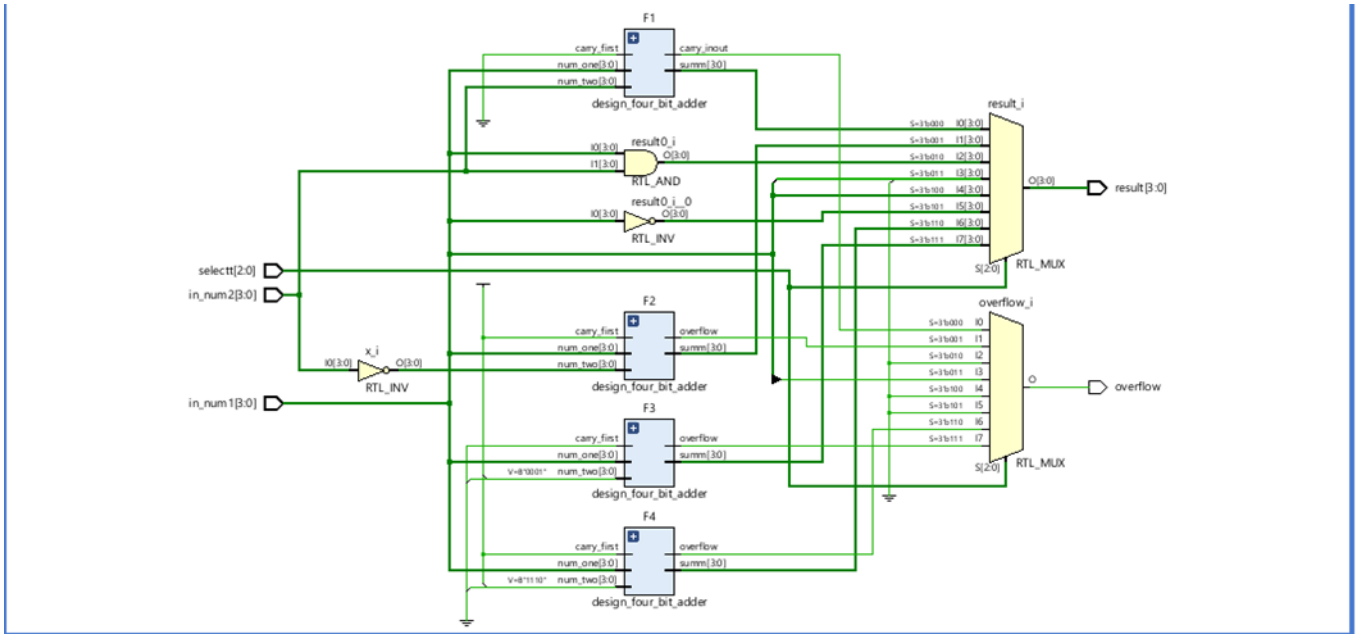

Image 1: RTL for ALU

3.Results:

After I wrote the test bench code for my ALU, I ran the simulation. In the simulation I chose two arbitrary 4 bit numbers and my first number is 1001 and my second number is 1000.

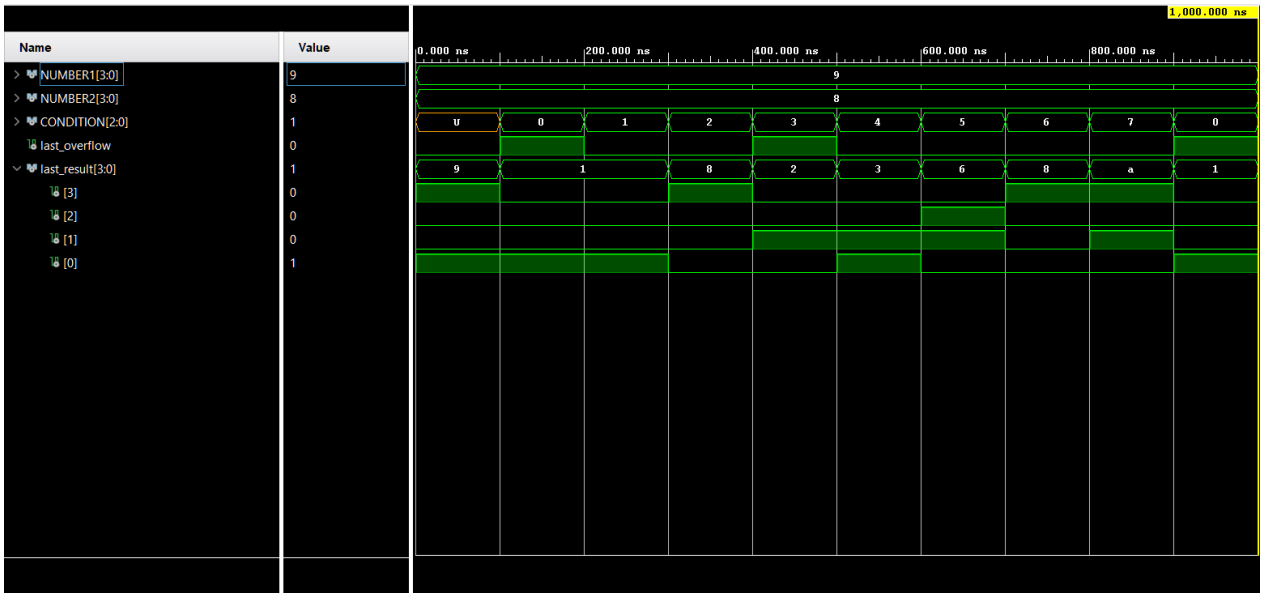

Image 2: Test Bench Simulation for NUMBER1 = "1001" , NUMBER2 = "1000"

Once the simulation procedure went off without an error, the test bench code's bitstream was created for installation on BASYS3. BASYS3 is programed, and various inputs have been tested on it. The experiments' findings are shown with the photos below: (selectt stands for the selection of operation. CONDITION stands for the result of the operation. And V19, U19, E19, U16 leds show the CONDITION. last_overflow means if there is an overflow or not. And L1 led shows last_overflow.)

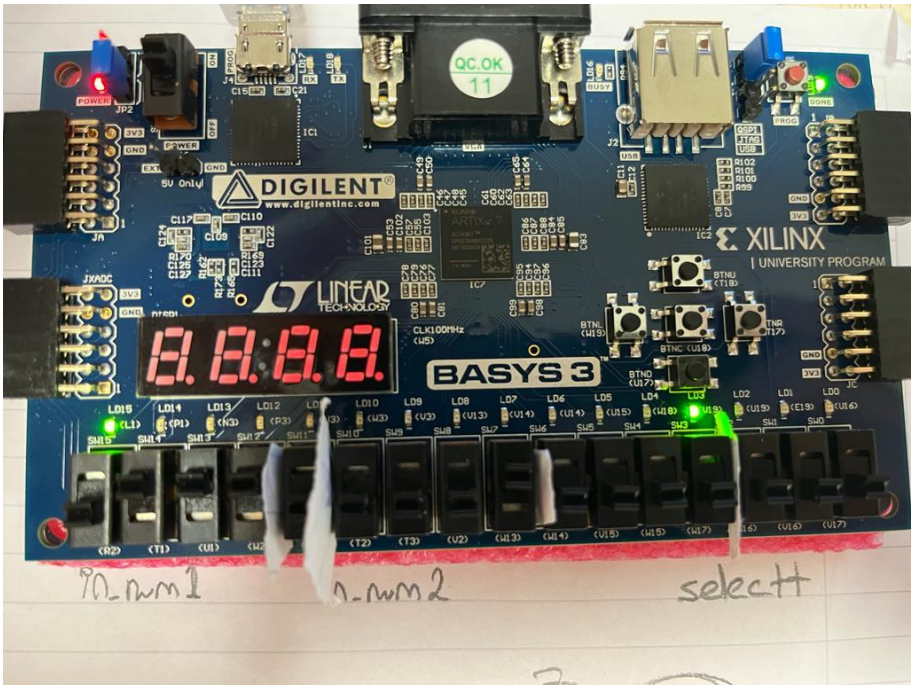

Figure 3: in_num1 = “0111”, in_num2 = “0001”, selectt = “000” then CONDITION = “1000”, last_overflow = ‘1’ : Addition

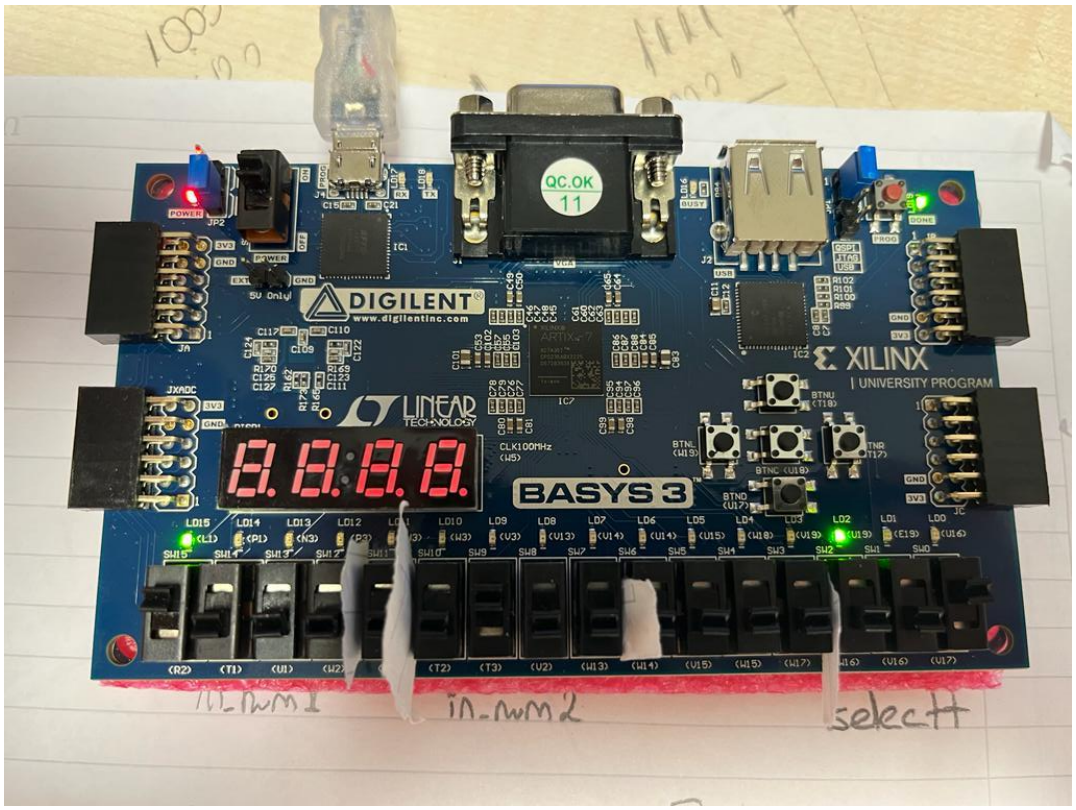

Figure 4: in_num1 = “1000”, in_num2 = “0100”, selectt = “001” then CONDITION = “1101”, last_overflow = ‘1’ : Substraction

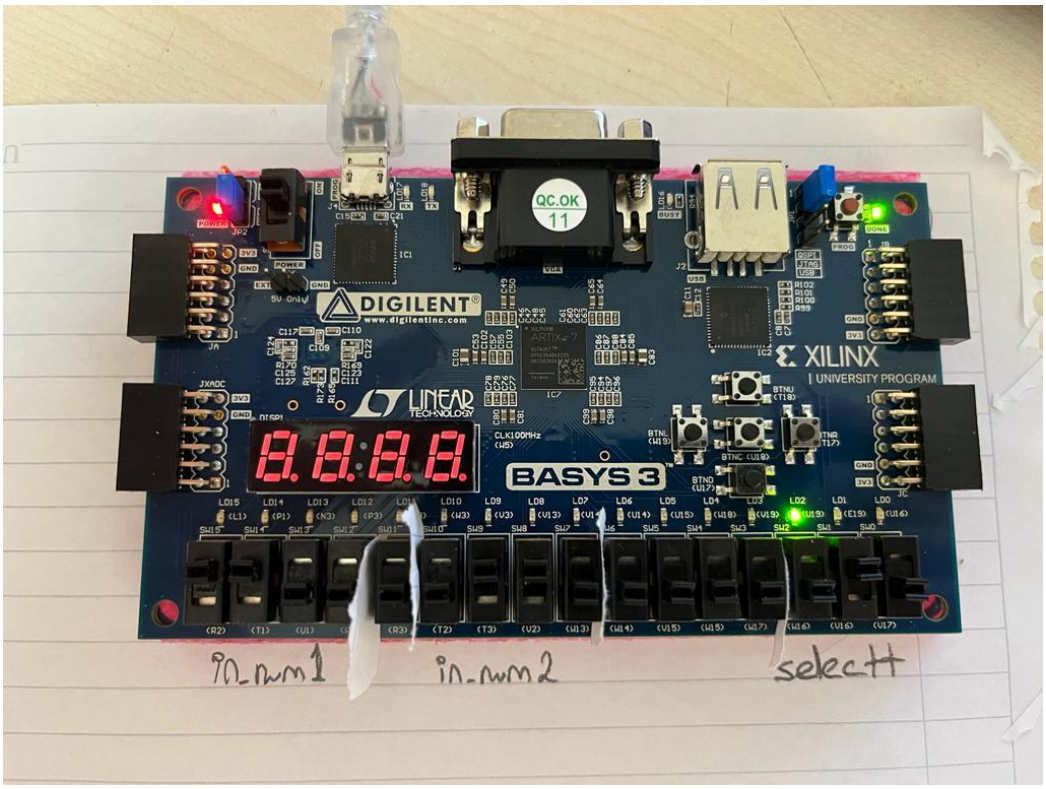

Figure 5 : in_num1 = “1100”, in_num2 = “0110”, selectt = “010” then CONDITION = “0100”, last_overflow = ‘0’ : Four bit AND

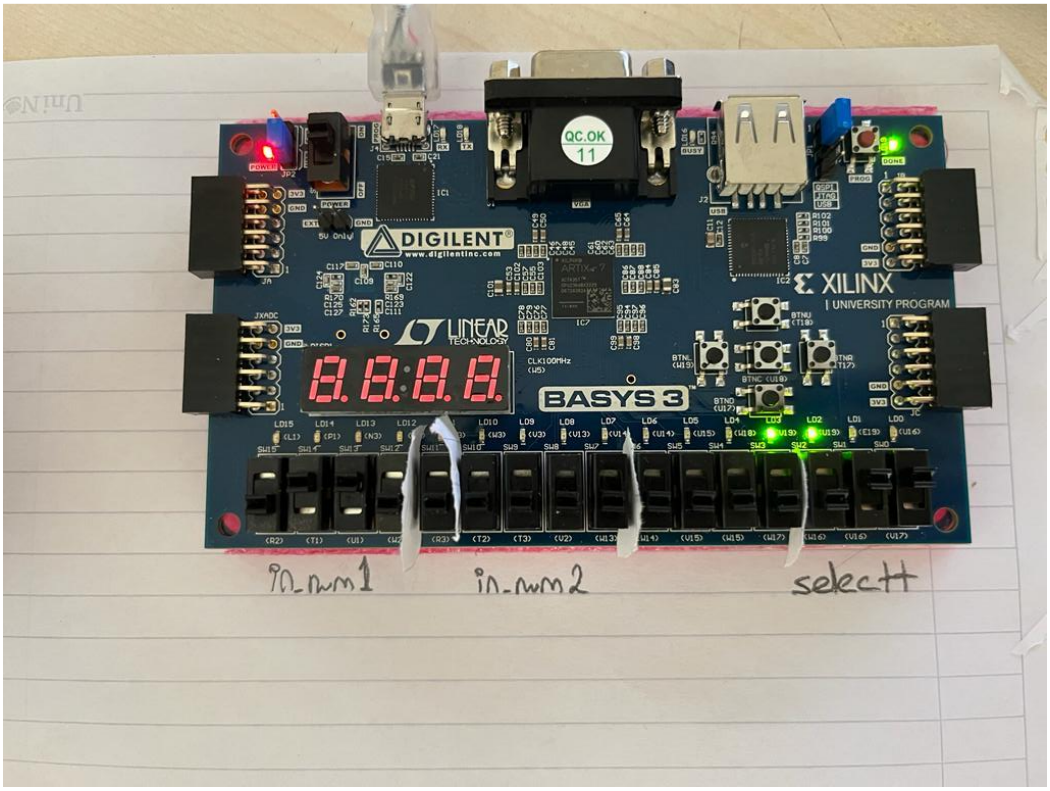

Figure 6: in_num1 = “0110”, in_num2 = “0000”, selectt = “011” then CONDITION = “1100”, last_overflow = ‘0’ : Logical Shift

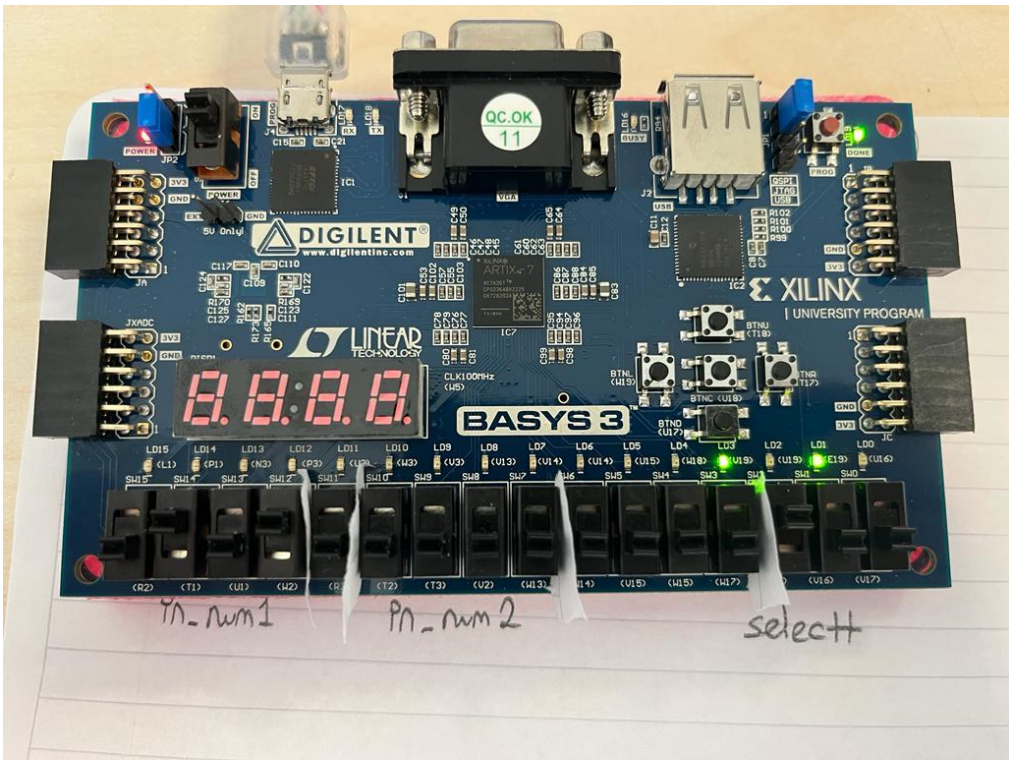

Figure 7 : in_num1 = “0101”, in_num2 = “0000”, selectt = “100” then CONDITION = “1010”, last_overflow = ‘0’ : Left Rotate

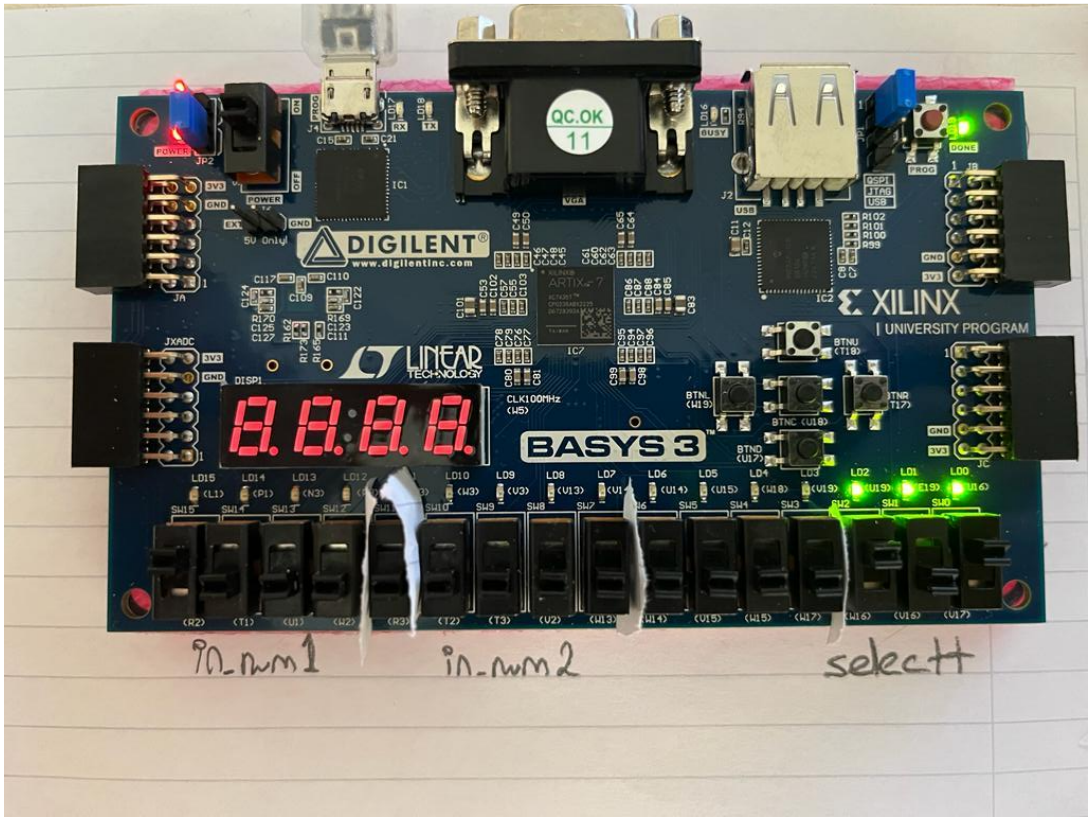

Figure 8 : in_num1 = “1000”, in_num2 = “0000”, selectt = “101” then CONDITION = “0111”, last_overflow = ‘0’ : Four Bit One's Complement

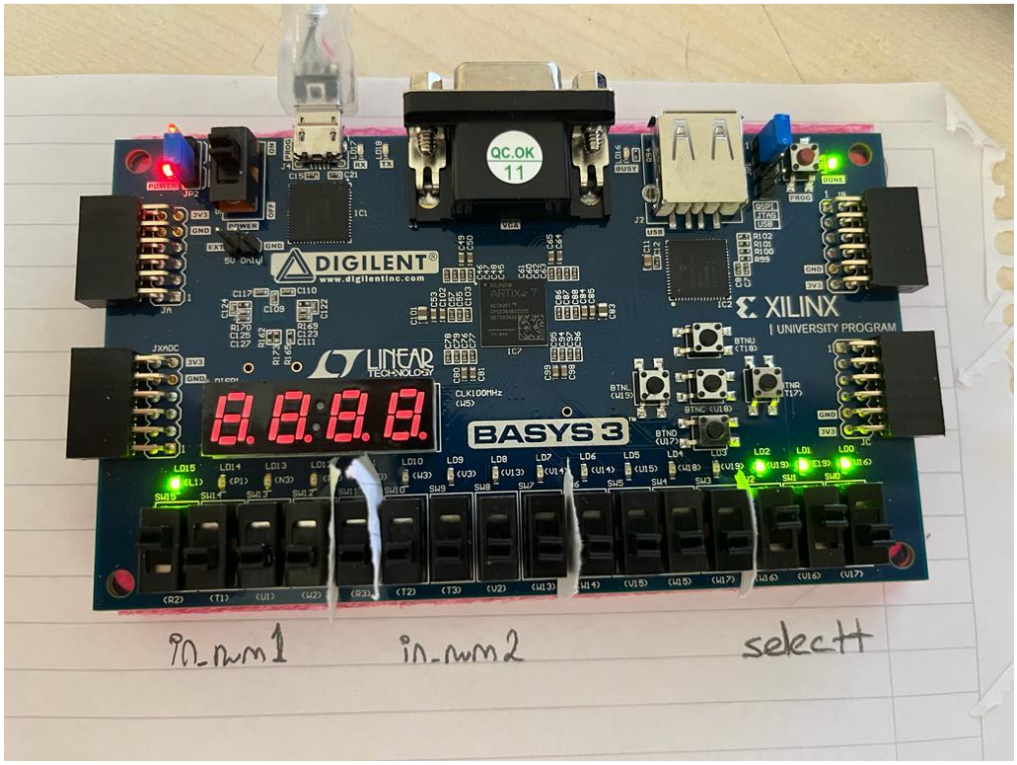

Figure 9 : in_num1 = “1000”, in_num2 = “0000”, selectt = “110” then CONDITION = “0111”, last_overflow = ‘1’ : Decrement by one

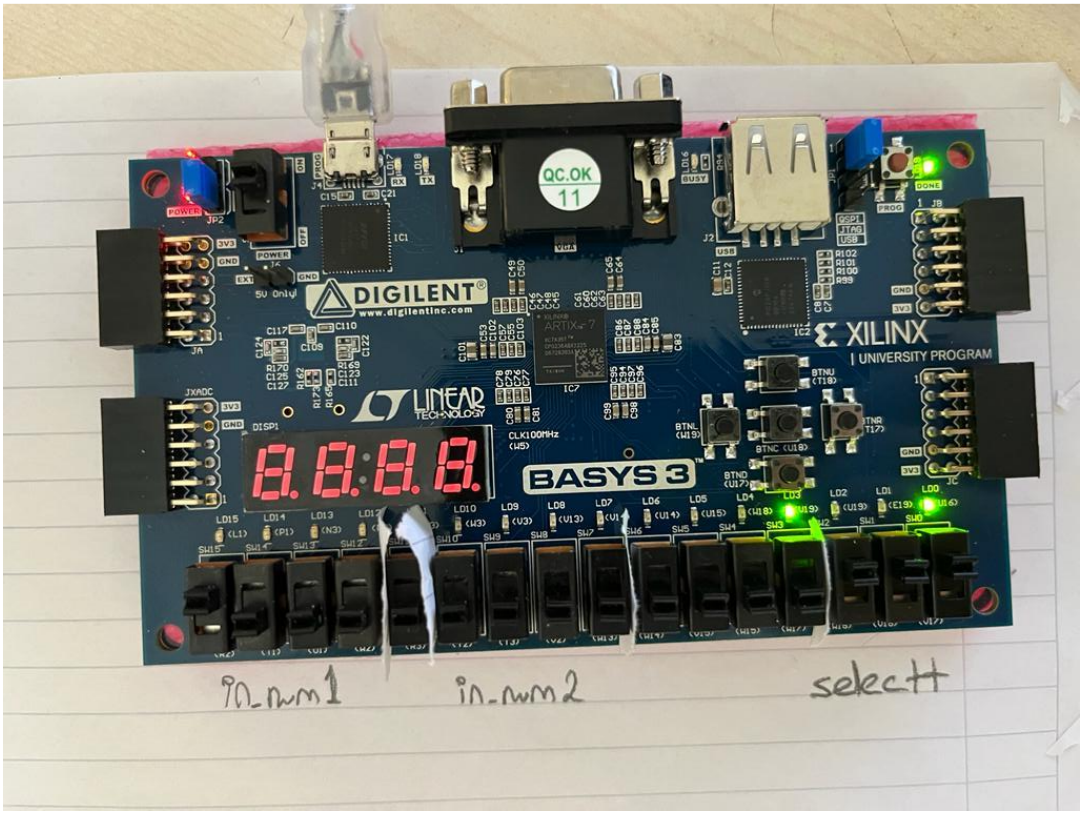

Figure 10 : in_num1 = “1000”, in_num2 = “0000”, selectt = “111” then CONDITION = “1001”, last_overflow = ‘0’ : Increment by one

4.Conclusion:

In this lab, the implementation of the component instantiation process has been thoroughly tested. Several faults and issues arose when coding, however I managed to debug successfuly thanks to the error information. The intended ALU's RTL Schemtic has increased our ability to visualize the theoretic expressions that we have been utilizing in the lectures. This lab was essential before beginning the term project since it was critical for the VHDL's usability. Because before this lab session, I could not visualize how the full adder is constructed and how the multiplexer works but now I know.

One Bit Adder Codes:

```
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 design_one_bit_adder is
    Port ( carry_zero : in STD_LOGIC;
           in1 : in STD_LOGIC;
           in2 : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry_one : out STD_LOGIC);
end design_one_bit_adder;

architecture Behavioral of design_one_bit_adder is

begin
```

```
sum <= in1 XOR in2 XOR carry_zero ;

carry_one <= ((in1 XOR in2) and carry_zero) or (in1 and in2);



end Behavioral;
```

4 Bit Adder Codes:

```
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;

library design_one_bit_adder;
use design_one_bit_adder.all;



entity design_four_bit_adder is
    Port ( num_one : in STD_LOGIC_VECTOR ( 3 downto 0);
           num_two : in STD_LOGIC_VECTOR ( 3 downto 0);
           carry_first : in STD_LOGIC;
           carry_inout : inout STD_LOGIC;
           overflow: out STD_LOGIC;
           summ : out STD_LOGIC_VECTOR ( 3 downto 0));
end design_four_bit_adder;
```

```
architecture Behavioral of design_four_bit_adder is

signal carry : STD_LOGIC_VECTOR (3 downto 1);

COMPONENT design_one_bit_adder port ( carry_zero, in1, in2: in STD_LOGIC ; sum,
carry_one: out STD_LOGIC );

end COMPONENT;

begin

A0: design_one_bit_adder port map (carry_zero => carry_first, in1=>num_one(0), in2=>
num_two(0), sum => summ(0),

carry_one => carry(1));

A1: design_one_bit_adder port map (carry_zero => carry(1), in1=>num_one(1), in2=>
num_two(1), sum => summ(1),

carry_one => carry(2));

A2: design_one_bit_adder port map (carry_zero => carry(2), in1=>num_one(2), in2=>
num_two(2), sum => summ(2),

carry_one => carry(3));

A3: design_one_bit_adder port map (carry_zero => carry(3), in1=>num_one(3), in2=>
num_two(3), sum => summ(3),
```

```
carry_one => carry_inout);




overflow <= carry_inout xor carry(3);
end Behavioral;
```

ALU Codes:

```
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 design_ALU is
    Port ( in_num1 : in STD_LOGIC_VECTOR (3 downto 0);
           in_num2 : in STD_LOGIC_VECTOR (3 downto 0);
           selectt : in STD_LOGIC_VECTOR (2 downto 0);
           overflow : out STD_LOGIC ;
           result : out STD_LOGIC_VECTOR (3 downto 0));

end design_ALU ;

architecture Behavioral of design_ALU is
```

```
COMPONENT design_four_bit_adder port ( num_one, num_two: in STD_LOGIC_VECTOR ( 3 downto 0);

carry_first: in STD_LOGIC ;carry_inout: inout STD_LOGIC;

overflow: out STD_LOGIC; summ: out STD_LOGIC_VECTOR( 3 downto 0));

end component;



signal first_result, second_result, third_result, fourth_result : STD_LOGIC_VECTOR (3 downto 0);

signal first_overflow, second_overflow, third_overflow, fourth_overflow : STD_LOGIC;

signal first_null, second_null,third_null,fourth_null : STD_LOGIC;

signal x : STD_LOGIC_VECTOR (3 downto 0 );


begin

x <= not(in_num2);


F1: design_four_bit_adder port map ( num_one => in_num1, num_two => in_num2, carry_first => '0',

summ => first_result, carry_inout => first_null , overflow => first_overflow );


F2: design_four_bit_adder port map ( num_one => in_num1, num_two => x , carry_first => '1',

summ => second_result, carry_inout => second_null, overflow => second_overflow);


F3: design_four_bit_adder port map ( num_one => in_num1, num_two => "0001", carry_first => '0',

summ => third_result, carry_inout => third_null, overflow => third_overflow);
```

```
F4: design_four_bit_adder port map ( num_one => in_num1, num_two => "1110", carry_first =>
'1',

summ => fourth_result, carry_inout => fourth_null, overflow => fourth_overflow);


 Allu :process(in_num1, in_num2, selectt) is

 begin

    case selectt is


        when "000" => -- Adder--

        result <= first_result;

        overflow <= first_overflow;


        when "001" => -- Substractor--

        result <= second_result;

        overflow <= second_overflow;


        when "010" => --AND--

        result <= in_num1 and in_num2;

        overflow <= '0';


        when "011" => --Logical Shift--


        result(3 downto 0) <= in_num1(2 downto 0) & '0';

        overflow <= in_num1(3);
```

```
when "100" => -- Left Rotate--
result(3) <= in_num1(2);
result(2) <= in_num1(1);
result(1) <= in_num1(0);
result(0) <= in_num1(3);

overflow <= '0';

when "101" => -- Four Bit One's Complement--

result <= not(in_num1);
overflow <= '0';

when "110" => --Decrement by one--
result <= fourth_result;
overflow <= fourth_overflow;

when "111" =>  --Increment by one--

result <= third_result;
overflow <= third_overflow;

when others =>
result <= in_num1 ;
overflow <= '0';
```

```
    end case;
  end process;


end Behavioral;
```

Test Bench Codes:

```
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 testbench_ALU is
--  Port ( );
end testbench_ALU;

architecture Behavioral of testbench_ALU is
COMPONENT design_ALU PORT(in_num1, in_num2: in STD_LOGIC_VECTOR (3 downto
0);
 selectt : in STD_LOGIC_VECTOR (2 downto 0);
 result : out STD_LOGIC_VECTOR (3 downto 0);
 overflow : out STD_LOGIC);
END COMPONENT;
```

```
signal NUMBER1 : STD_LOGIC_VECTOR (3 downto 0);
signal NUMBER2 : STD_LOGIC_VECTOR (3 downto 0);
signal CONDITION : STD_LOGIC_VECTOR (2 downto 0);
signal last_overflow : STD_LOGIC;
signal last_result : STD_LOGIC_VECTOR (3 downto 0);


begin
UUT: design_ALU port map(in_num1 => NUMBER1, in_num2 => NUMBER2,
selectt => CONDITION, result => last_result,
overflow => last_overflow );

start: process
begin
NUMBER1 <= "1001";
NUMBER2 <= "1000";
wait for 100ns;
CONDITION <= "000";
wait for 100ns;
CONDITION <= "001";
wait for 100ns;
CONDITION <= "010";
wait for 100ns;
CONDITION <= "011";
wait for 100ns;
```

```
CONDITION <= "100";
wait for 100ns;
CONDITION <= "101";
wait for 100ns;
CONDITION <= "110";
wait for 100ns;
CONDITION <= "111";
end process;



end Behavioral;
```

Constraint Codes:

```
set_property IOSTANDARD LVCMOS33 [get_ports {in_num1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num2[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in_num2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {result[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {result[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {result[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {result[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {selectt[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {selectt[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {selectt[0]}]
set_property PACKAGE_PIN R2 [get_ports {in_num1[3]}]
set_property PACKAGE_PIN T1 [get_ports {in_num1[2]}]
set_property PACKAGE_PIN U1 [get_ports {in_num1[1]}]
set_property PACKAGE_PIN W2 [get_ports {in_num1[0]}]
set_property PACKAGE_PIN T2 [get_ports {in_num2[3]}]
set_property PACKAGE_PIN T3 [get_ports {in_num2[2]}]
set_property PACKAGE_PIN V2 [get_ports {in_num2[1]}]
set_property PACKAGE_PIN W13 [get_ports {in_num2[0]}]
set_property PACKAGE_PIN W16 [get_ports {selectt[2]}]
```

```
set_property PACKAGE_PIN V16 [get_ports {selectt[1]}]
set_property PACKAGE_PIN V17 [get_ports {selectt[0]}]
set_property PACKAGE_PIN U16 [get_ports {result[0]}]
set_property PACKAGE_PIN E19 [get_ports {result[1]}]
set_property PACKAGE_PIN U19 [get_ports {result[2]}]
set_property PACKAGE_PIN V19 [get_ports {result[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports overflow]
set_property PACKAGE_PIN L1 [get_ports overflow]
```