Experiment 2: Introduction to VHDL

Date: 3/10/2022

Name: Sait Sarper Özaslan

Section:102-2

Purpose: The purpose of this experiment was to design a combinational circuit on BASYS3 using VHDL which could be a solution to a possible real-life problem.

Design Specifications: The design of this combinational circuit consists of 3 inputs and 1 output. Firstly, input is inversed and connected to an 'and gate'. Secondly, two inputs are connected to an 'or gate' and their result is connected to same 'and gate'.

This design is supposed to simulate whether an ecosystem will nourish or not. Input 1 can be considered as toxins from a factory that may get released to the river. In such case, input 2 and 3 would be the existence of plants in the water and forest near the river, respectively. I2321

Methodology: For an ecosystem to survive, plants supply oxygen to the animals and animals supply Carbon dioxide to the plants creating a cycle. If this cycle gets disturbed by external interference, which could be a factory that releases toxins to the river in this case, the plants will wither away causing this cycle to break. As a result, the production of oxygen will lessen over time causing animals to die or leave the ecosystem which will decrease the Carbon dioxide production. As a result, the ecosystem surrounding the river may vanish.

As toxins' existence, input1 or X1, would cause destruction to the ecosystem so in any case where toxin is apparent there would not be any ecosystem. In the case of input 2 and input 3, x2 and x3 respectively, any type of plant's existence whether it is in water or not may supply enough oxygen to the ecosystem leading to its' survival.

The truth table of the described case would be,

X1	X2	X3	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

If there are toxins in the river the ecosystem would not survive, which describes the bottom 4 rows of the columns. If there is not any toxins and there is at least one type of plant either in the river or the land near the river the ecosystem can thrive which describes cases through 2 to 4. If there are no toxins and no plants there would not be any ecosystem which describes the first case.

Sum of Products of this truth table would be:

X1’X2’X3+X1’X2X3’+X1’X2X3

When simplified the design would be,

X1’(X2+X3)

After figuring this out, I created a design source and coded in VHDL. Then after that I created a test bench file to simulate the function. Through a constrain file I assigned the values to appropriate gates and LEDs. Lastly, I implemented this design to the BASYS3 after doing bitstream generation.

Questions:

-- How does one specify the inputs and outputs of a module in VHDL?

We can specify the inputs and outputs of a module by naming it as in and out according to appropriate usage for the values. To assigned values we add STD_logic, which carries digital information for the values.

-- How does one use a module inside another code/module? What does PORT MAP do?

We can use a module inside another code/module by writing component [module name] under the main module. Port map lets us connect modules.

-- What is a constraint file? How does it relate your code to the pins on your FPGA?

A constraint file can be considered a straightforward way of assigning placements and timing restriction to the implementation. It lets us assign inputs to pins in the FPGA and outputs to the LEDs.

-- What is the purpose of writing a testbench?

The purpose of writing a testbench is to implement all the possible inputs in the truth table to a simulation and create simulation according to the logic function written beforehand.

Results:

After figuring this out, I created a design source on Vivado and coded, which can be seen in appendices A, this into the lab experiment. After coding, I used Vivado to see an RTL Schematic.

The RTL schematic of the combinational circuit I did on Vivado is,

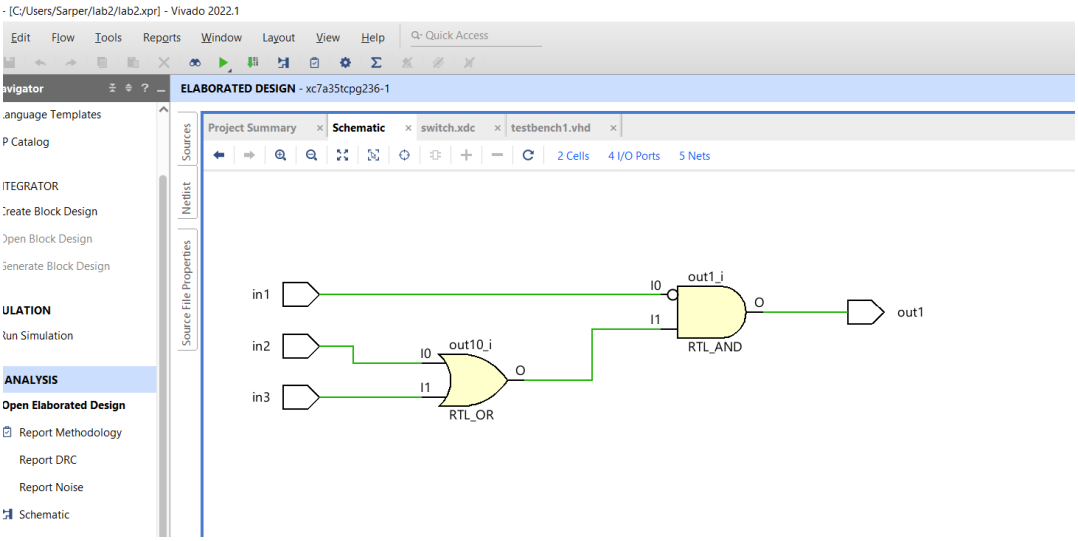

Figure 1: RTL Schematic

Afterwards, I needed to create a simulation so I created a simulation source and wrote a testBench. I wrote all eight possibles which can be seen in appendice B. All eight possibilities in this simulation are:

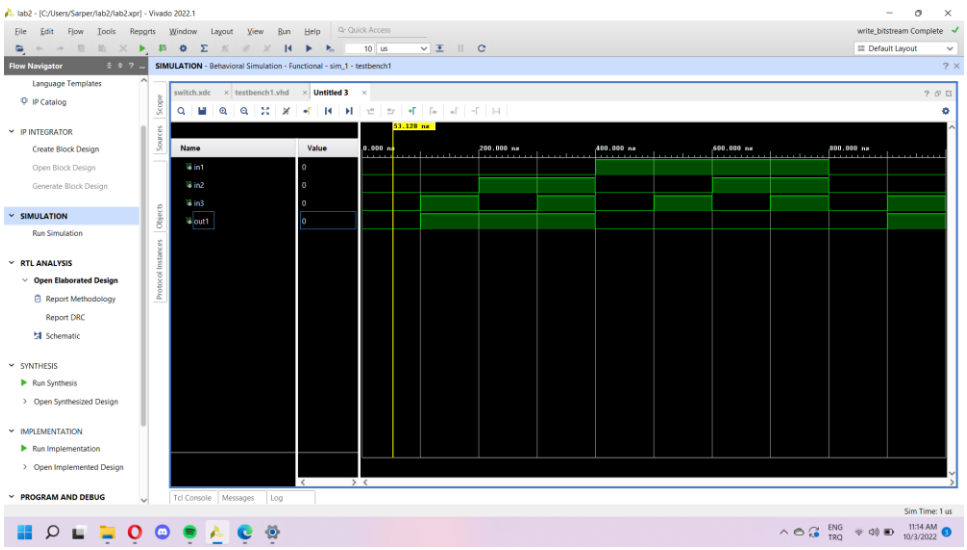

Figure 2: Case 0,0,0

The inputs in Basys3 can be seen in the figure below which are input 1 is switch V17, input 2 is switch V16 and input 3 is W16. All of which are assigned in the constraint file.

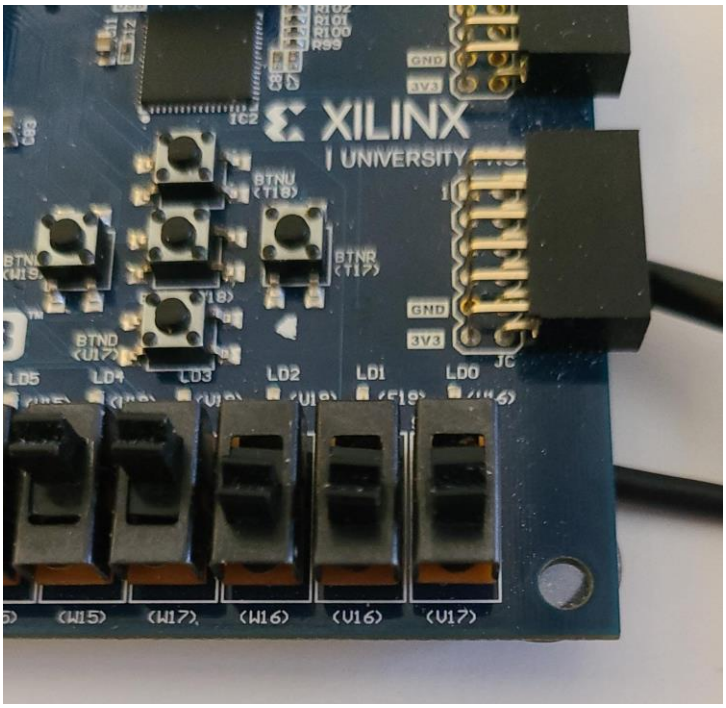

Figure 3: Case 0,0,0 on Vivado

The following results can be seen in the figures below.

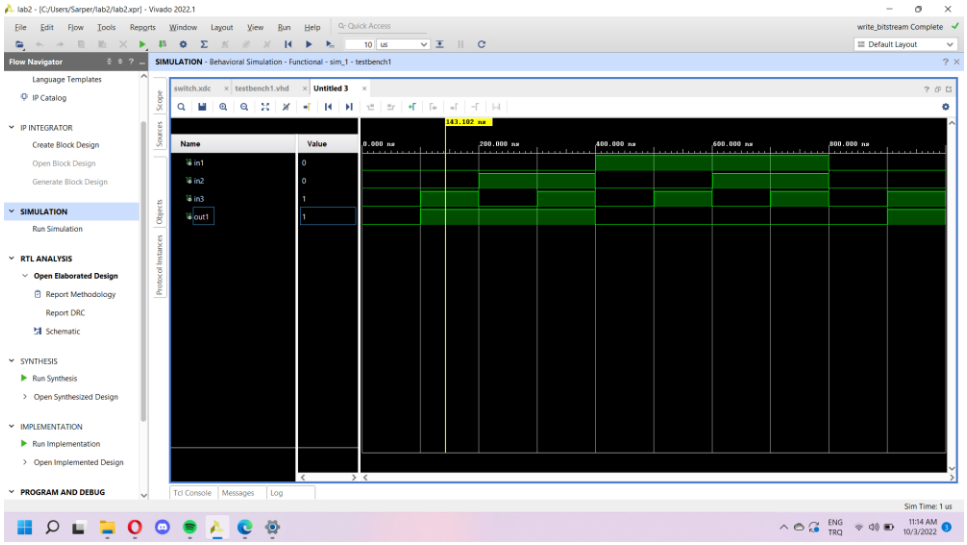

Figure 4: Case 0,0,1

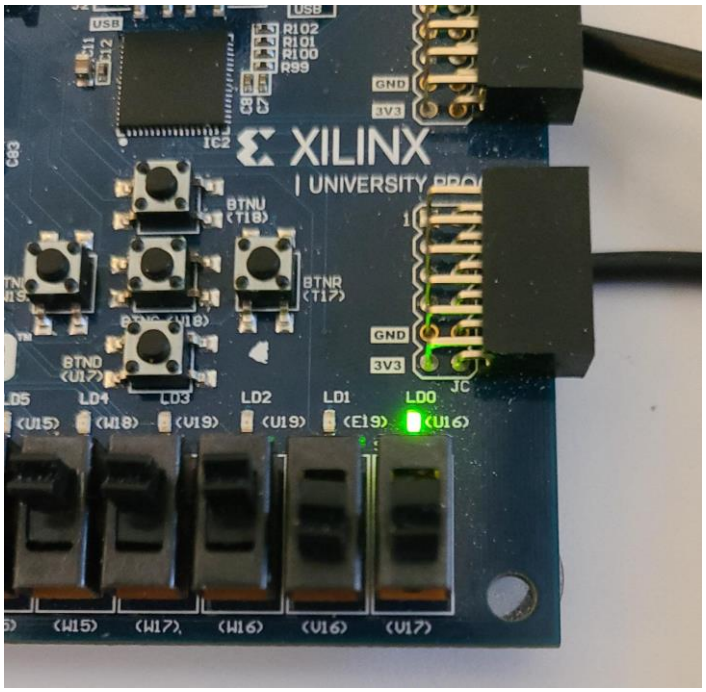

Figure 5: Case 0,0,1 on BASYS3

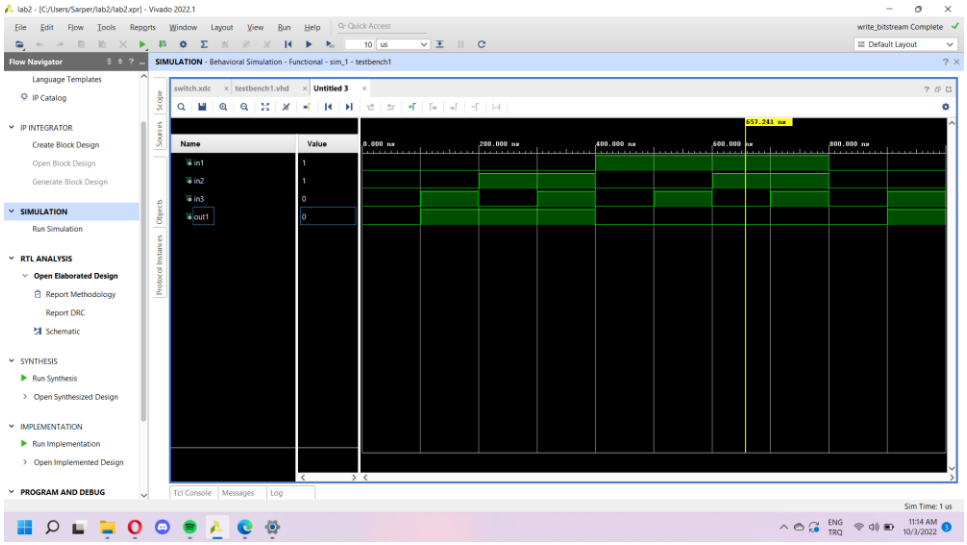

Figure 6: Case 1,1,0

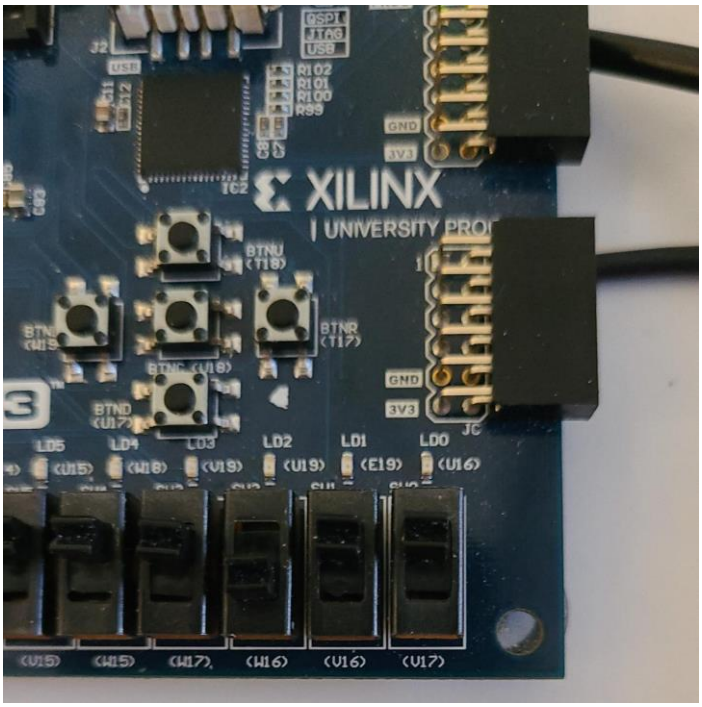

Figure 7: Case 1,1,0 on BASYS3

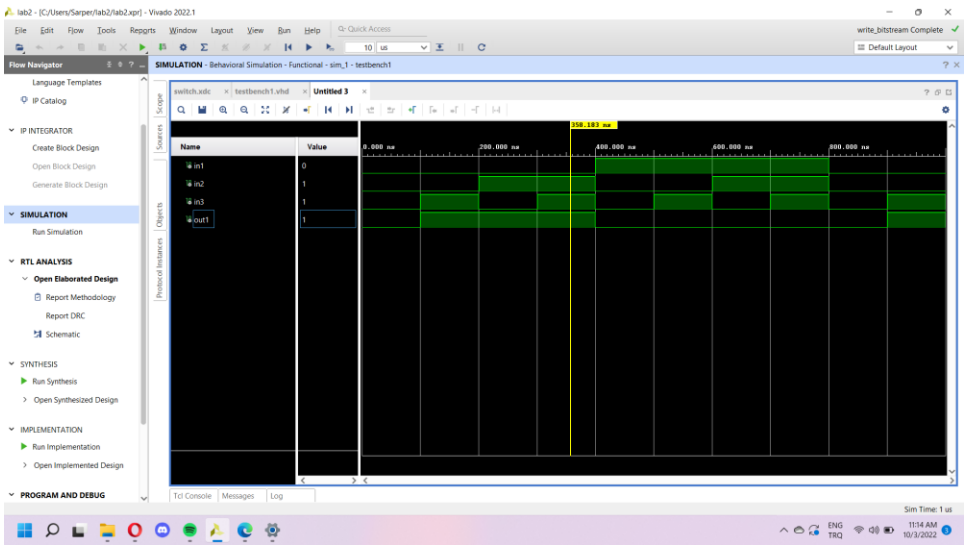

Figure 8: Case 0,1,1

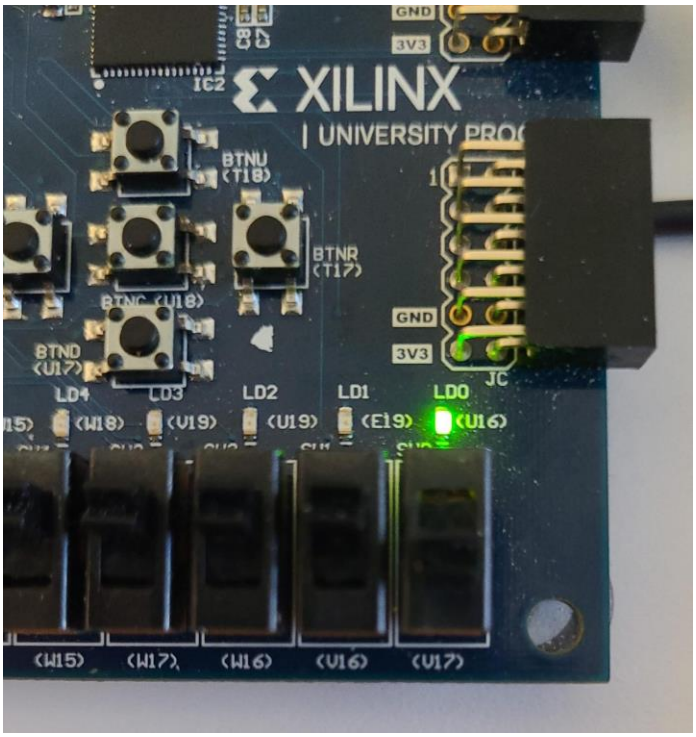

Figure 9: Case 0,1,1 on BASYS3

Other cases can be found in appendices.

After doing the simulation and checking the cases through BASYS3, the results came out as the same as the truth table which would mean the cases where ecosystem would survive are same as the cases described in methodology.

Conclusion: In this experiment, I figured how to solve a possible situation through a logic function I have made through sum of products. I implemented that logic function by writing its' code in VHDL. Through the code I have written in VHDL I observed the behavior in a simulation and realized that the simulation created matched the one in truth table. I also experimented this function in BASYS3 and realized the implemented design also matched the truth table.

Appendices:

(a)VHDL code:

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


entity light1 is
    Port ( in1 : in STD_LOGIC;
           in2 : in STD_LOGIC;
           in3 : in STD_LOGIC;
           out1 : out STD_LOGIC);
end light1;


architecture Behavioral of light1 is


begin
out1 <= (not in1) and (in2 or in3);
```

```
end Behavioral;
```

(b)testBench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

entity testbench1 is

end testbench1;
--assigning inputs
architecture Behavioral of testbench1 is
component light1
    port
    (in1 : IN std_logic;
    in2 : IN std_logic ;
    in3 : IN std_logic ;
    out1 : OUT std_logic );
```

```
end component;
signal in1 : std_logic := '0';
signal in2 : std_logic := '0';
signal in3 : std_logic := '0';
signal out1 : std_logic;

BEGIN
uut: light1 PORT MAP (in1 =>in1,
in2 => in2,
in3 => in3,
out1 => out1);

testbench1 : process
```

```
begin
in1<='0';in2<='0';in3<='0';
wait for 100ns;
in1<='0';in2<='0';in3<='1';
wait for 100ns;
in1<='0';in2<='1';in3<='0';
wait for 100ns;
in1<='0';in2<='1';in3<='1';
wait for 100ns;
in1<='1';in2<='0';in3<='0';
wait for 100ns;
in1<='1';in2<='0';in3<='1';
wait for 100ns;
in1<='1';in2<='1';in3<='0';
wait for 100ns;
in1<='1';in2<='1';in3<='1';
wait for 100ns;
end process;
END;
```

(c) The other 4 simulations and their BASYS3 photos

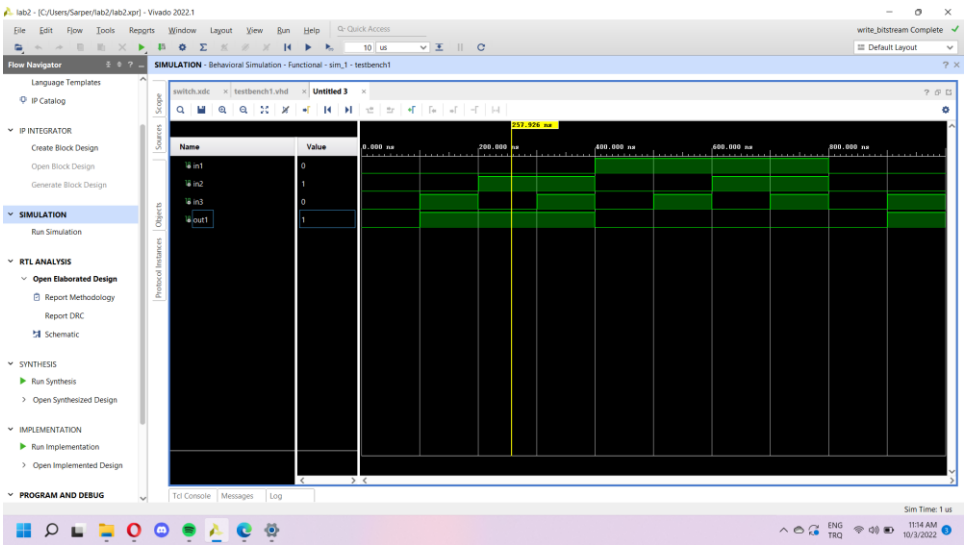

Figure 10: Case 0,1,0

Figure 11: Case 0,1,0 on BASYS3

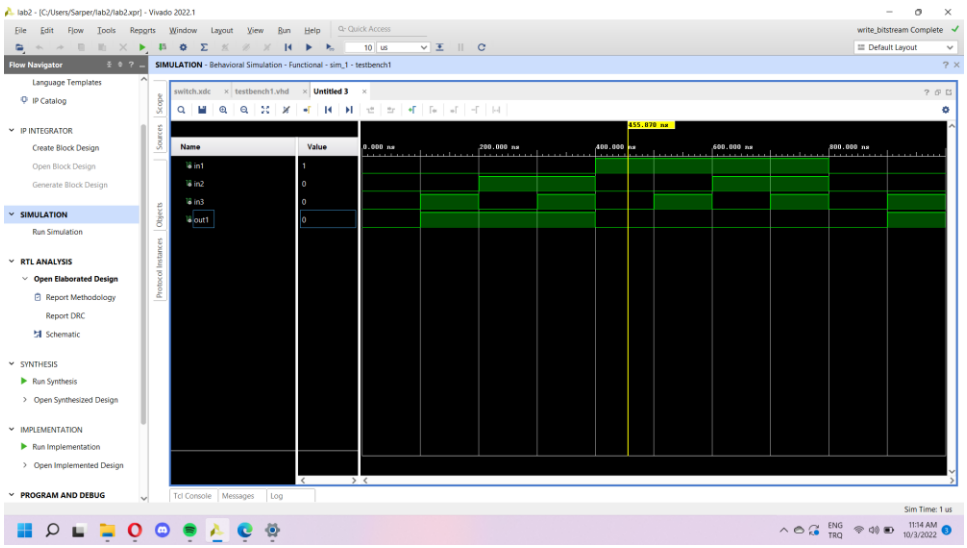

Figure 12: Case 1,0,0

Figure 13: Case 1,0,0 on BASYS3

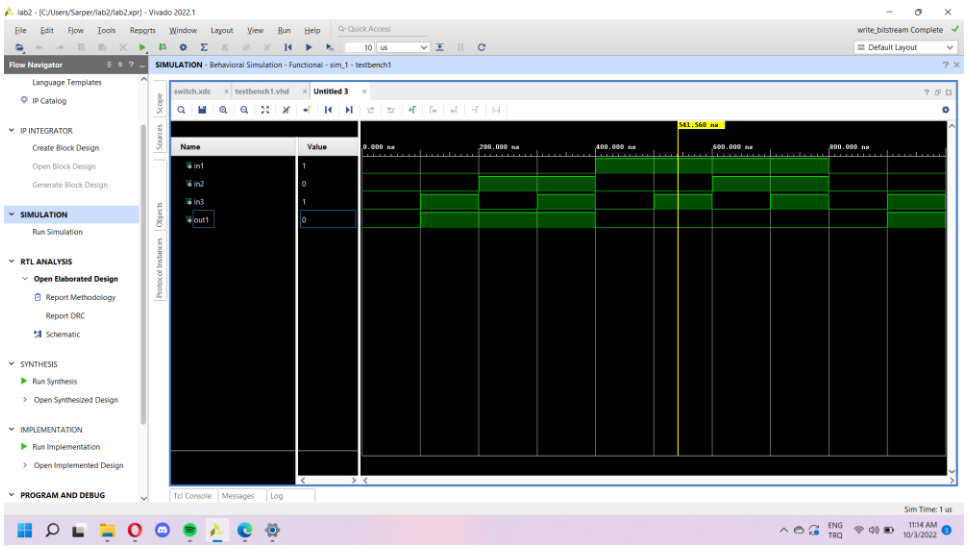

Figure 14: Case 1,0,1

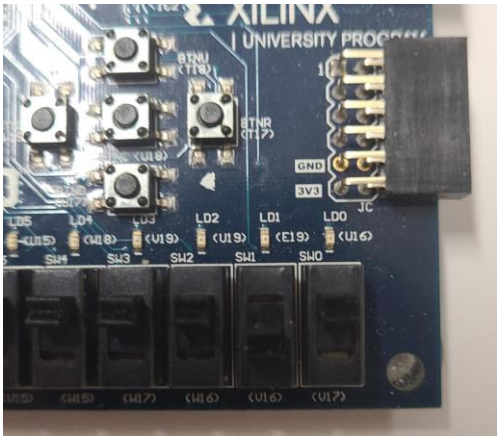

Figure 15: Case 1,0,1 on BASYS3

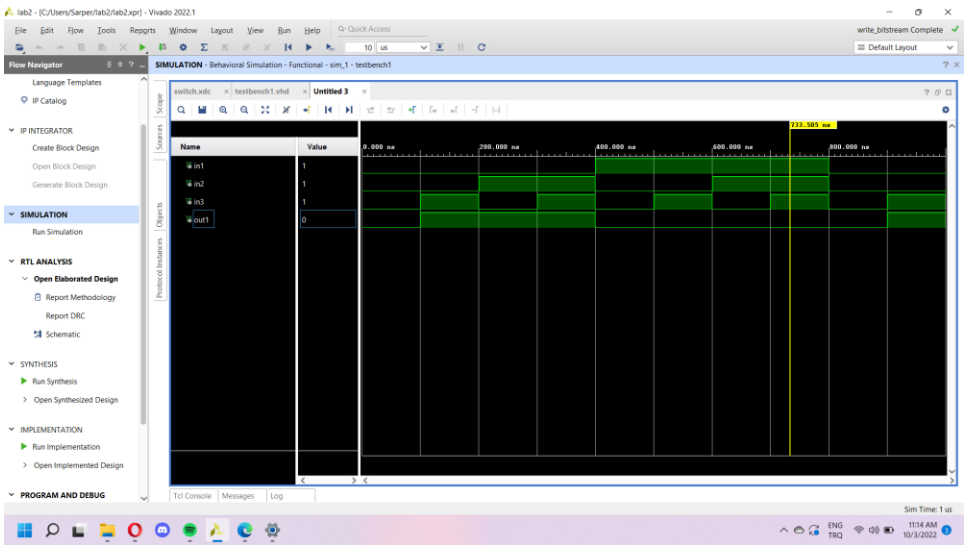

Figure 16: Case 1,1,1

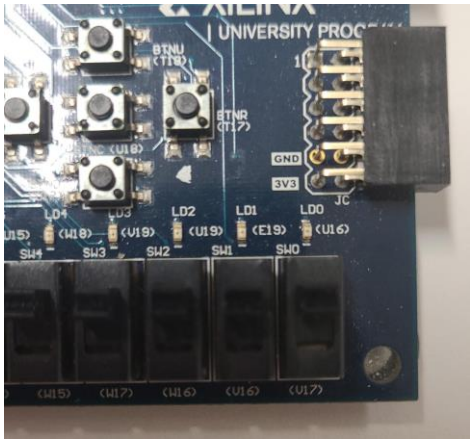

Figure 17: Case 1,1,1 on BASYS3