Lab 2: Introduction to VHDL

Methodology

Purpose of this lab is coding basic logic circuits, which solve a real-life problem, with using VHDL with the help of Xilinx's Vivado software. To conduct this experiment firstly, a real-life problem is chosen. Then the inputs and outputs of this problem are determined. Outputs generated according to the specified inputs are written to the truth table. I detected the canonical sum of product of function by using minterm. After determined the logical design, I coded the design source in Vivado. It indicated with one line code. Then simulation test part is coded to observe that whether the code gives the true output as I determined before. Finally I copied the constraint part from Moodle basys3.xdc document. This part arranges the input and output pins for the code to work in harmony with the Basys 3 FPGA board.

In my laboratory, the purpose is to code a basic circuit to determine whether I should go to lectures or not according to the lecture timetable. Then I set 3 input variables corresponding to the lectures. These 3 inputs symbolizes by 3 letters (A, B, C).

I. If there is a MATH-241 lesson that day, A is 1. Else, A is 0.

II. If there is an EE-102 lesson that day, B is 1. Else, B is 0.

III. If there is an EE-211 lesson that day, C is 1. Else, C is 0.

The output of the truth table, which I then constructed in Table 1, is based on whether the courses are a 1 or a 0. For the result (output) I symbolized it with the letter R

A	B	C	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 1 – Truth table of function

I prefer to use the method using the sum of product (SOP) because it is simpler to represent this circuit using minterm. Then, I represented R as,

$$R \equiv m_3 + m_6 + m_7$$

which equal to,

$$R \equiv (A'.B.C) + (A.B.C') + (A.B.C)$$

applied the distributive law,

$$R \equiv (B.C).(A+A') + (A.B.C')$$

applied the complement law because A+ A'=1,

$$R \equiv (B.C).(1) + (A.B.C')$$

and again applied the distributive law,

$$R \equiv (B).(C + A.C')$$

applied the absorption law because C+A.C' = A+C,

$$R \equiv (B).(A+C) \equiv B.A + B.C$$

and finally the most simple version is function is obtained as R ≡ B.A + B.C ≡ (B).(A+C)

This function imply the circuit in Figure 1. In the circuit, two AND gates and one OR gates are used. A input symbolizes Math-241, input B symbolizes EE-102 and input C symbolizes EE-211 courses. Then I coded the logical gates array in the design source part. I used the letter R, which is the abbreviation of the result, as the output value.

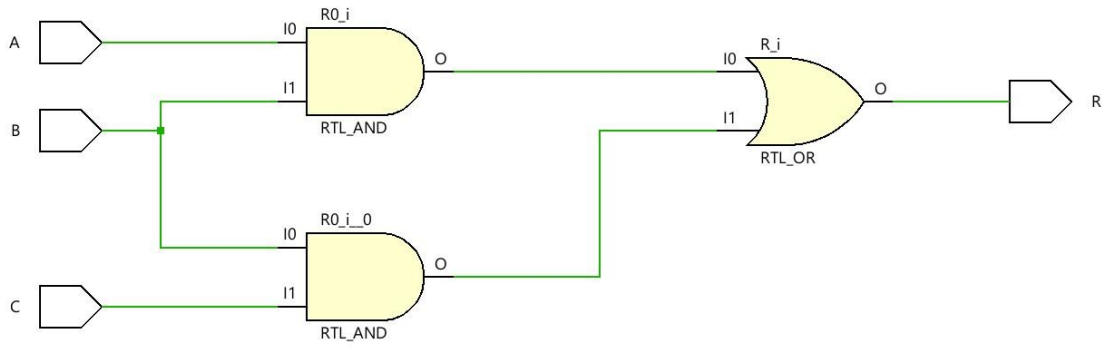

Figure 1 – The circuit design with logic gates

Questions

1. In VHDL, design source code determines the input and output. We defined a port module and we entered the names of inputs and outputs. Then we defined "in" for all inputs and "out" for outputs.
2. To use a module inside another module we defined signal type variable. In my code, it was not necessesary but to make code easier and clear, signal type variables can be defined. When programming, modules are connected via a "port map.". In the "port map" statement, signals are listed in the behavioral phases of that module in the order they appear.
3. When programming a basys 3, we need to state that which physical input and output will be used. So we need to write these as code to work basys3 with true physical switchs and LEDs.
4. Before uploading the code we wrote to the board, testbench code enables us to verify that it functions as intended. It enables us to foresee problematic situations in advance and address them without adding code to the board.

Results

I created the truth table and I determined the simplest version of my function by using sum of product (SOP) technique. I chose the m_3, m_6 and m_7. I simplified the canonical form and obtained $R \equiv B.(A+C)$. After I obtained the logic function I wrote it by using VHDL in Vivado. When all coding process are applied, I started the Generate Bitstream to transfer the codes to Basys3 board.

In the simulation part, I changed all variables one by one and hold for all of them 100 nano second. It can be seen in Figure 2 that A, B and C changed one by one then I observed that how output value changed. According to my function, the output values matched exactly so I can infer that my code was accurate.

After uploading my codes to basys 3, I tested the conditions like $A \equiv 0$, $B \equiv 1$ and $C \equiv 1$. From the third figure through the ninth figure, I presented each of these scenarios in turn in each photograph. The light turned on three times, as I had anticipated.

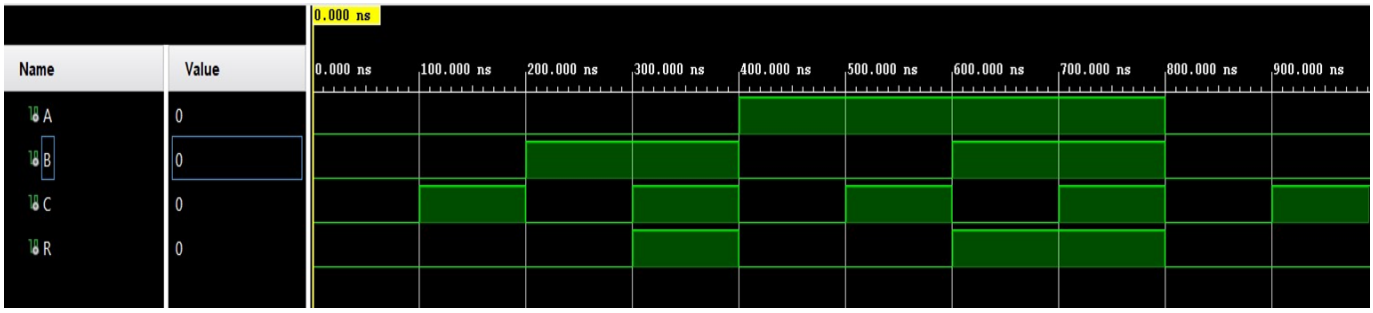

Figure 2 - TestBench simulation holding for 100ns

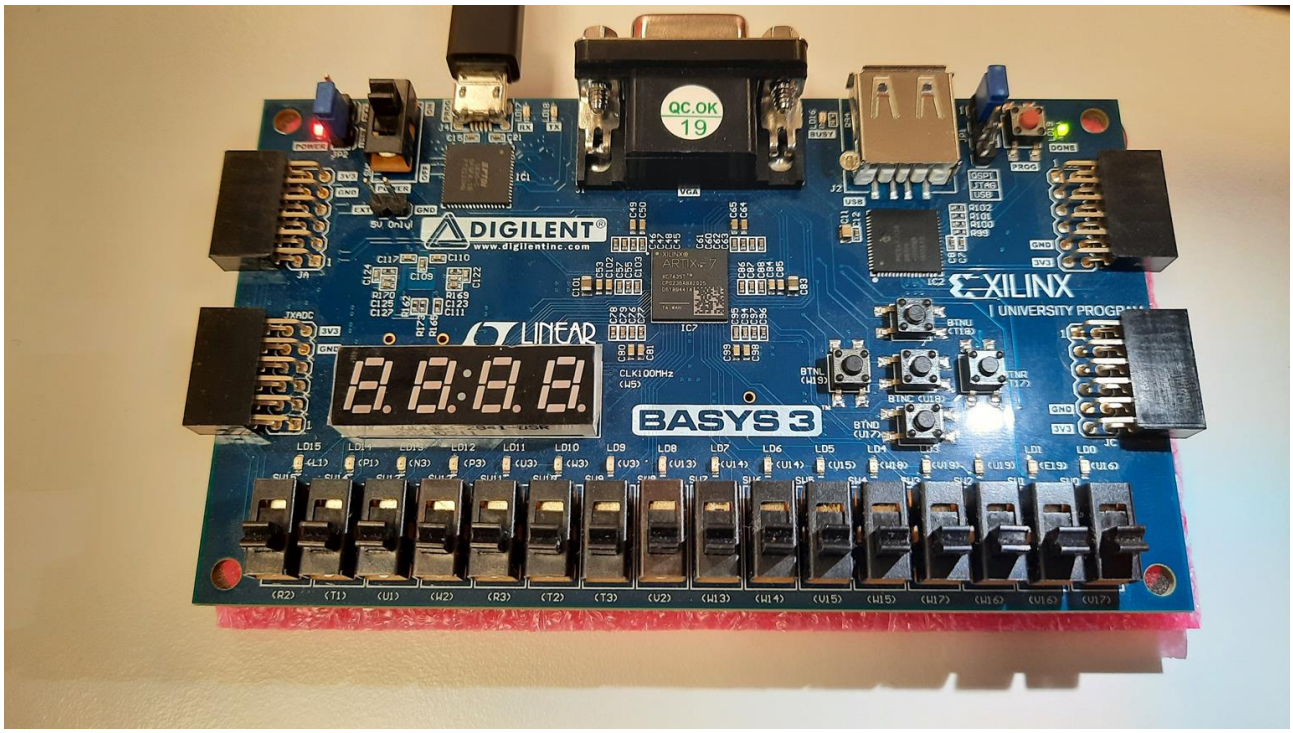

Figure 3 - R ≡ 0 (LED is off), A ≡ 0, B ≡ 0, C ≡ 0

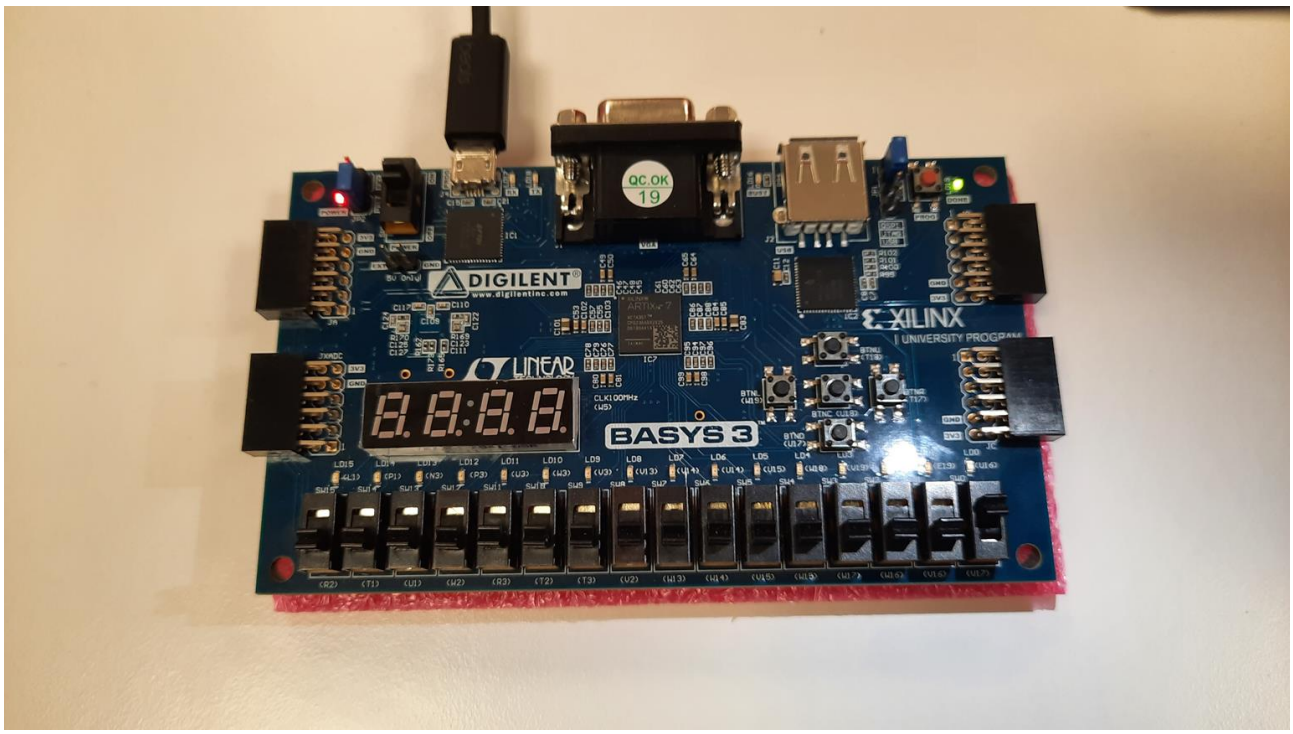

Figure 4 - R ≡ 0 (LED is off), A ≡ 0, B ≡ 0, C ≡ 1

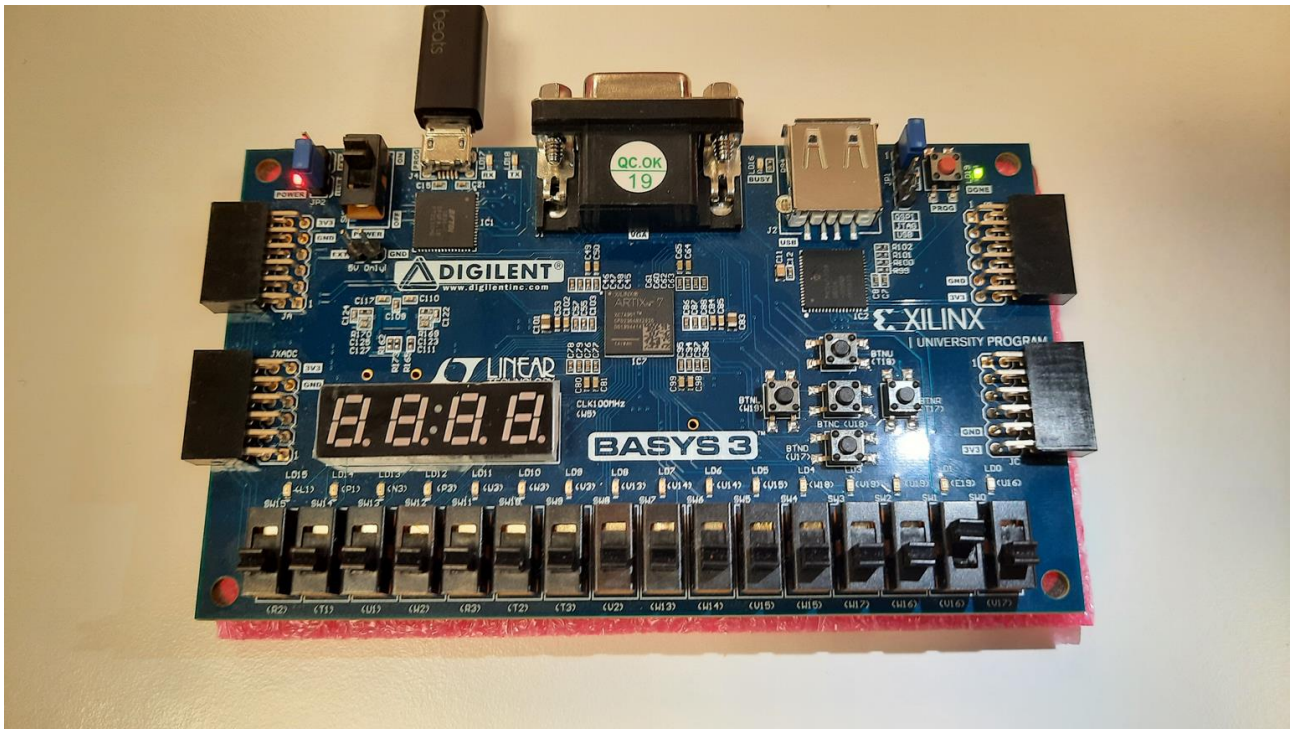

Figure 5 - R ≡ 0 (LED is off), A ≡ 0, B ≡ 1, C ≡ 0

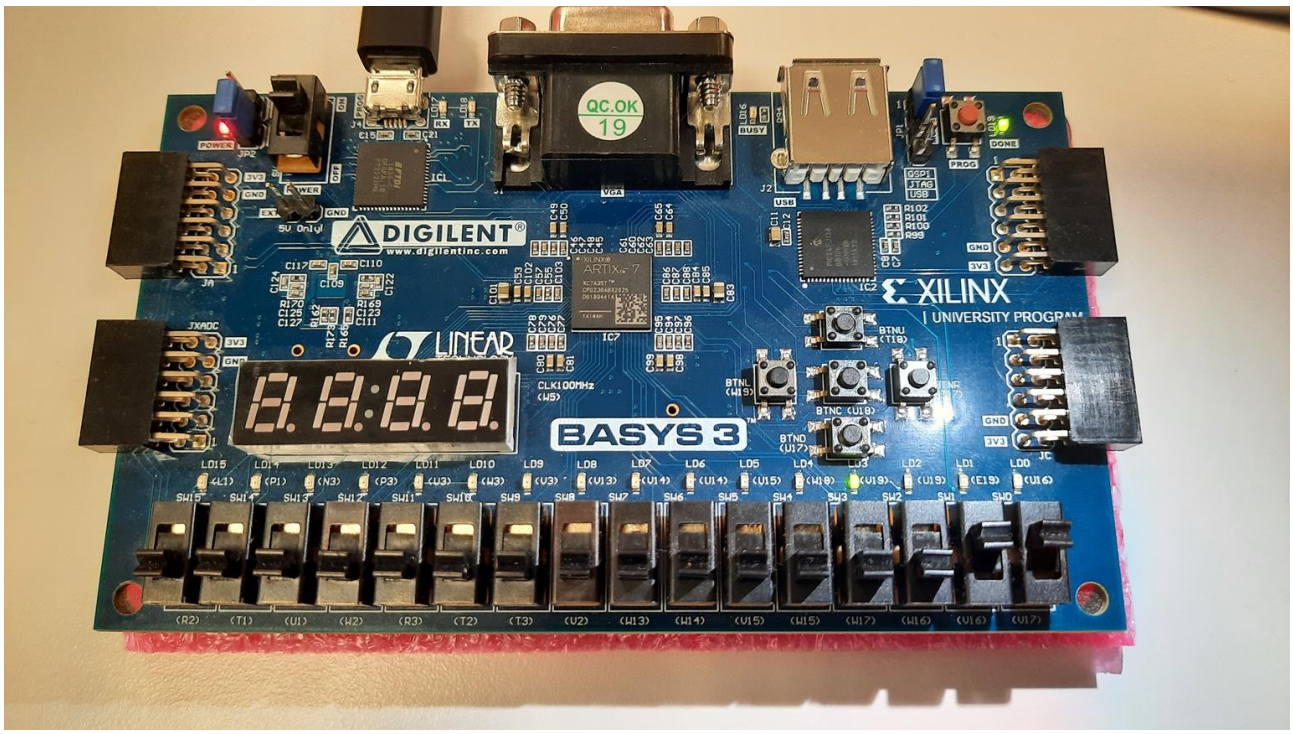

Figure 6 - R ≡ 1 (LED is on), A ≡ 0, B ≡ 1, C ≡ 1

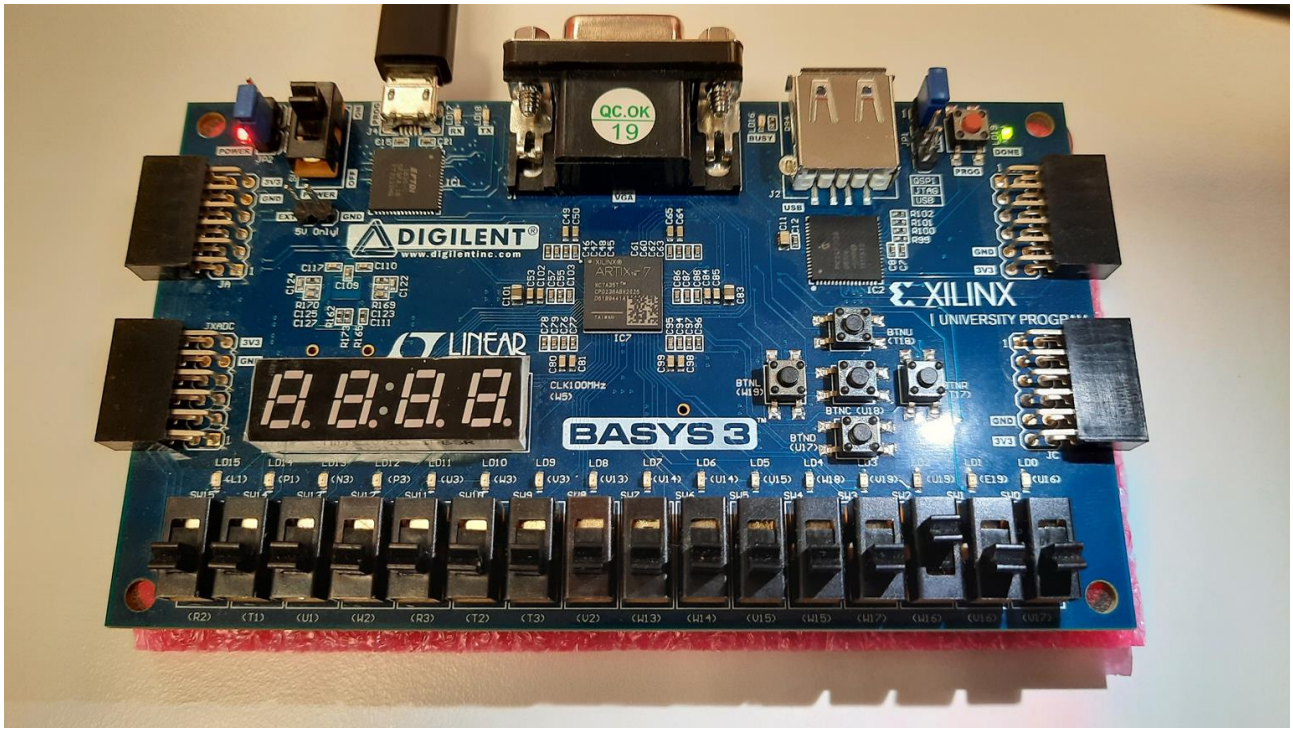

Figure 7 - R ≡ 0 (LED is off), A ≡ 1, B ≡ 0, C ≡ 0

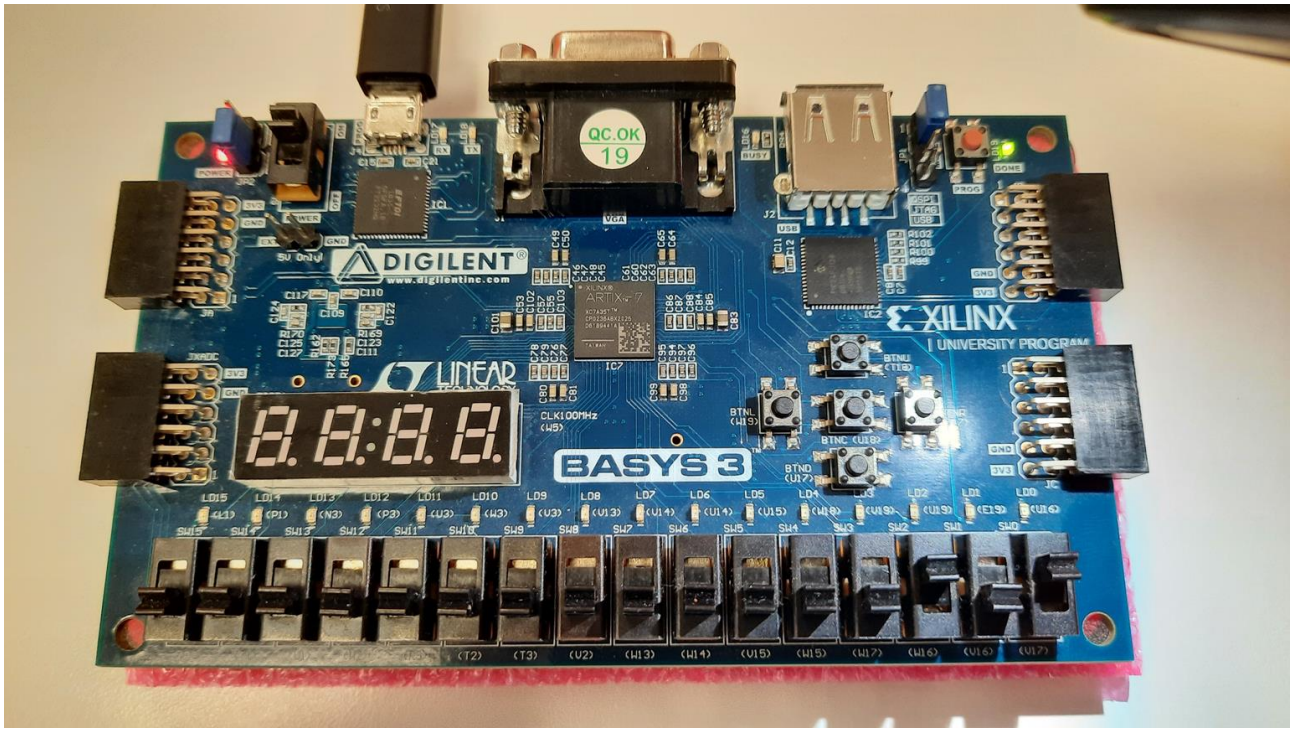

Figure 8 - R ≡ 0 (LED is off), A ≡ 1, B ≡ 0, C ≡ 1

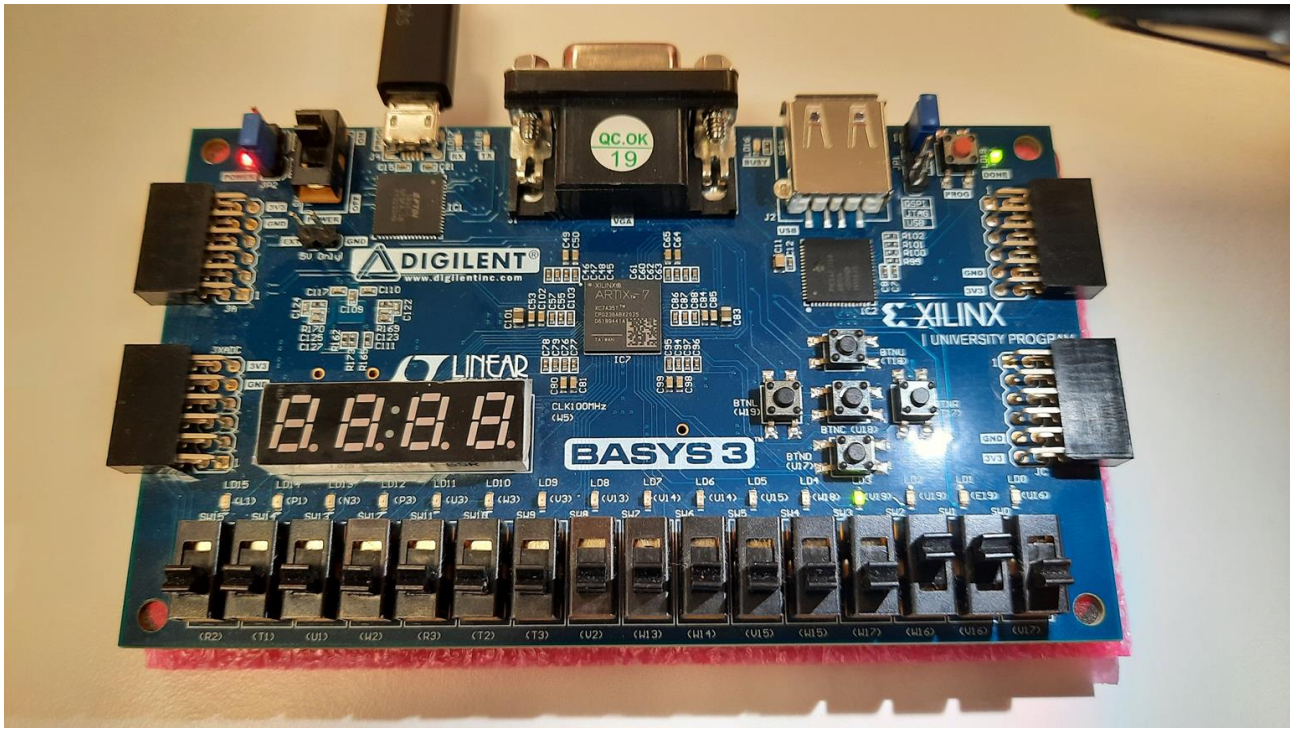

Figure 9 - R ≡ 1 (LED is on), A ≡ 1, B ≡ 1, C ≡ 0

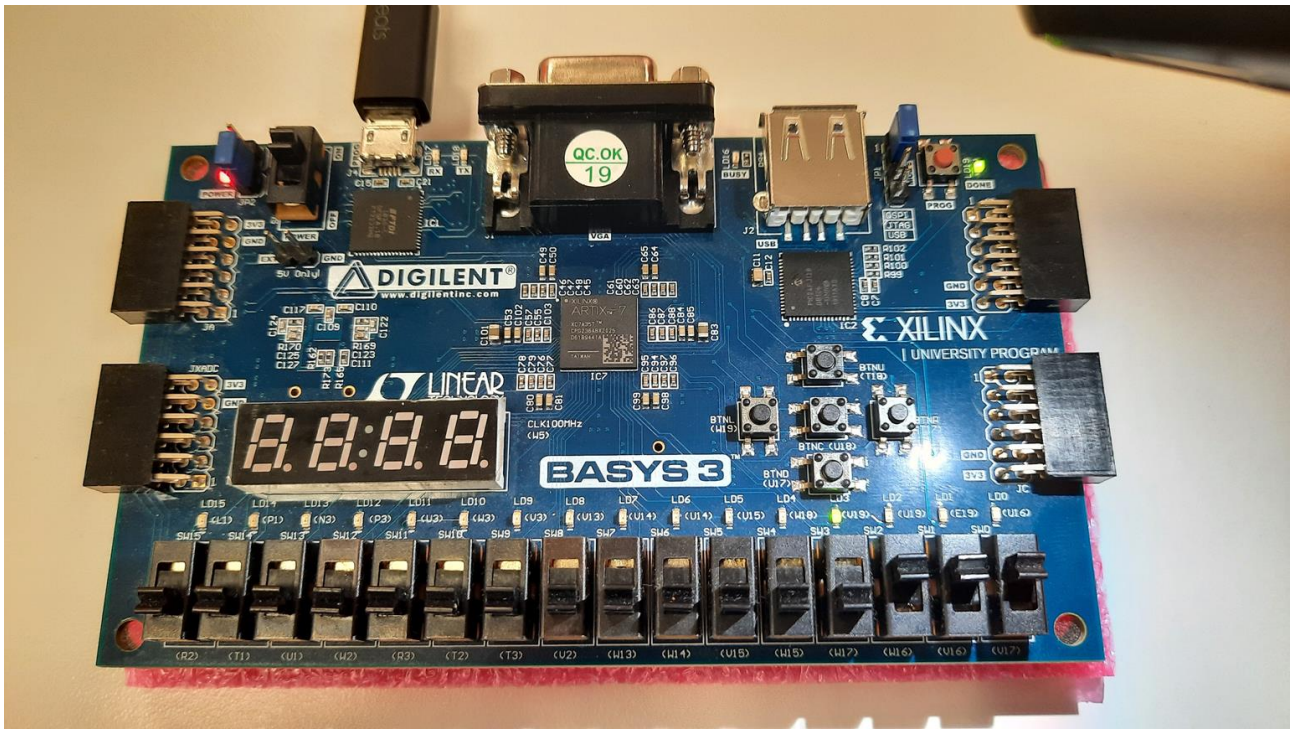

Figure 9 - R ≡ 1 (LED is on), A ≡ 1, B ≡ 1, C ≡ 1

Conclusion

In this lab, I learned how logic gates can be used to build a circuit for a function I want to design and how this circuit can be designed by coding on Vivado with VHDL. I did my circuit design with VHDL so I learn how to code in VHDL. Also I experienced with Vivado atmosphere. Even though it is a simple lab, it is crucial since the subsequent labs will directly relate to what we performed in this lab. By creating such fundamental combinational circuits, we may understand better and advance our understanding of VHDL's fundamental concepts. I also learned how to simulate my codes without basys 3 board. I was able to run and test my codes without transferring them to the board thanks to simulation.

Appendices

I.Design Code

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

entity furkan2022 is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           R : out STD_LOGIC);
```

```
end furkan2022;

architecture Behavioral of furkan2022 is

begin
R <= (A and B) or (B and C);
end Behavioral;
```

II.Test Bench Code

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.Numeric_Std.all;


entity Design1_tb is
end;

architecture Behavioral of Design1_tb is
component furkan2022
Port ( A : in STD_LOGIC;
       B : in STD_LOGIC;
       C : in STD_LOGIC;
       R : out STD_LOGIC);
end component;

signal A: STD_LOGIC;
signal B: STD_LOGIC;
signal C: STD_LOGIC;
signal R: STD_LOGIC;
begin
 uut: furkan2022 port map ( A => A,
    B => B,
    C => C,
    R => R );

Design1_tb: process
begin
A <= '0';
B <= '0';
C <= '0';
wait for 100ns;
A <= '0';
```

```
B <= '0';
C <= '1';
wait for 100ns;
A <= '0';
B <= '1';
C <= '0';
wait for 100ns;
A <= '0';
B <= '1';
C <= '1';
wait for 100ns;
A <= '1';
B <= '0';
C <= '0';
wait for 100ns;
A <= '1';
B <= '0';
C <= '1';
wait for 100ns;
A <= '1';
B <= '1';
C <= '0';
wait for 100ns;
A <= '1';
B <= '1';
C <= '1';
wait for 100ns;

end process;
end;
```

III. Constraints Code

```
set_property PACKAGE_PIN V17 [get_ports {A}]
   set_property IOSTANDARD LVCMOS33 [get_ports {A}]
set_property PACKAGE_PIN V16 [get_ports {B}]
   set_property IOSTANDARD LVCMOS33 [get_ports {B}]
set_property PACKAGE_PIN W16 [get_ports {C}]
   set_property IOSTANDARD LVCMOS33 [get_ports {C}]
set_property PACKAGE_PIN V19 [get_ports {R}]
   set_property IOSTANDARD LVCMOS33 [get_ports {R}]
```