Lab 2: Introduction to VHDL

1) Purpose

In this experiment our aim is to understand how combinational circuits can be implemented using FPGA board BASYS 3 with VHDL language. We are expected to find a real-life problem and design a digital circuit that its output only depends on the combination of inputs to solve the problem found. Then we generate a truth table of the design and find the simplest form of the logic function and write a VHDL code that resemble the design and using Vivado tools generate a schematic representation and simulation of the output signal regarding to input signals. In the experiment e theoretical situation is considered, in which a software company keeps track of its employees' success. Each employee has two tasks for a week, namely writing 2 codes and checking 2 codes written by other employees. If an employee writes two codes, if two code is written whether the person checks written codes by peers or not, or writes 1 code and checks 2 codes, the person considered as successful. As a result, a combinational digital circuit resembling an employee's success is designed.

2) Methodology

A situation where a software company examines its employees performance is considered in this digital design. Each employee is attained to four task 2 coding and 2 code checking written by other employees. In order an employee to be considered as successful s/he must complete 2 codes in one week or complete 1 code and 2 checks in one week. In the logical design there are 4 inputs 2 coding 2 checking and 1 output resembling success of the employee with given conditions. A and B stands for coding tasks and C and D check and F resembles the success of the employee. If A signal is 1, it means 1 coding task is completed. If D signal is 0, then it means that 1 check task is not completed. Truth table with given names is given below (Figure 2.1). The logic expression resulting in the given truth table can be found by

sum of products method and can be expressed as $\sum m(7,11,12,13,14,15)$ and using boolean algebra properties can be simplified to the form:

F = ((C AND D) AND (A OR B)) OR (A AND B)

Code 1 (A)	Code 2 (B)	Check 1 (C)	Check 2 (D)	Success (F)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Figure 2.1 Truth table for the given theoretical situation

Then after finding the logical expression for the solution of the given theoretical situation implementation of the given digital circuit to the BASYS 3 using VHDL is needed. Expression of the inputs and outputs is done via entity declaration.

```
entity lab2 is
   Port ( code1 : in STD_LOGIC;
```

```
        code2 : in STD_LOGIC;
        check1 : in STD_LOGIC;
        check2 : in STD_LOGIC;
        success : out STD_LOGIC);
end lab2;
```

Figure 2.2 entity declaration in VHDL

Port function assigns given variable names, in green, to the given variable type, in or out type of the std_logic variable class,. For example code1 is assigned as a input variable, whereas success is assigned to a output variable. Std_logic means the variable type is taken from the IEEE:STD_LOGIC_1164 library that is declared at the top of the script “library IEEE; use IEEE.STD_LOGIC_1164.ALL;”. The std_logic variable class allows defining signals with most common logic values, such as 1, 0, ‘Z’,.

Then in order to express how these inputs and outputs depends on each other a combinational digital circuit is designed and specified using architecture code

```
architecture Behavioral of lab2 is
begin
success <= ((check1 and check2) and (code1 or code2))or (code1 and code2);
end Behavioral;
```

Figure 2.3 combinational circuit specification

The code between begin and end of the architecture code specifies how the inputs and output is related. Relation between input signals and output signals declared using logic expressions, such as AND, OR, and implemented in FPGA board using logic gates.

Then after the design is completed a simulation of the declared combinational circuit with given inputs is done by adding a simulation source.

In the simulation script we use hierarchical design. We declare the component we use by component declaration code.

```
component lab2
Port ( code1 : in STD_LOGIC;
       code2 : in STD_LOGIC;
       check1 : in STD_LOGIC;
       check2 : in STD_LOGIC;
       success : out STD_LOGIC);
END component;
```

Figure 2.4 Component declaration

Component declaration allows us to hierarchically use the components for example code given in Figure 2.4 specifies the in or out signals in lab2 component and allows us to use lab2 component in our test bench.

```
signal code1 : std_logic:= '0';
```

Figure 2.5 Example of specifying a signal

Then we create signals in our simulation script to use as test signals for the simulation.

```
uut: lab2 PORT MAP(
       code1 => code1,
       code2 => code2,
       check1 => check1,
       check2 => check2,
       success=> success);
```

Figure 2.6 Port map function

Then by using port map function after the begin comment of the architecture we assign signals in our script to the component signals .

```
stim_proc: process
 begin
code1<='0';code2<='0';check1<='0'; check2<='0';
wait for 62.5ns;
end process;
```

Figure 2.7 Process function example

Then we use process function to sequentially assign values for given signals for a given time period to see the output of the signals in the simulation.

After checking the simulation signals and the whether the design works as expected, constraint file created and input and output ports specified on the board regarding to the given combinational circuit. After that, the synthesis, implementation and bitstream generation are completed. Then using hardware manager the FPGA board BASYS 3 programmed and combinational circuit implemented inside the board.

```
set_property PACKAGE_PIN W2 [get_ports {check2}]
  set_property IOSTANDARD LVCMOS33 [get_ports {check2}]
```

Figure 2.8 Example of a constraint file comment. W2 pin in the board assigned to the check 2 signal using set_property code

3) Results

The combinational circuit's schematic description implemented using VHDL can be observed using RTL analysis and it demonstrates the logic gates input signals and output signals connections in the combinational circuit. The design includes 4 input 1 output signals (code1, code2, check1, check2, success), 3 And, 2 Or gates and connections between them can be observed.

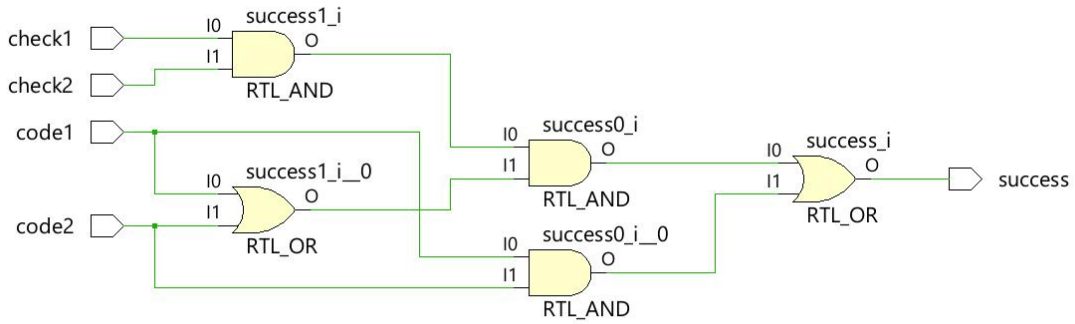

Figure 3.1 Schematic description of the implemented digital circuit.

In the test bench code in order for the each input combinations to be shown ,n 1000ns since there are 4 inputs there are 2^4 combination of the input signals meaning if each signal is executed in 62.5ns then all the cases can be shown in 1000ns and the output of the signal waveform can be observed in the Figure 3.2. The resulting output signal matches the intended design.

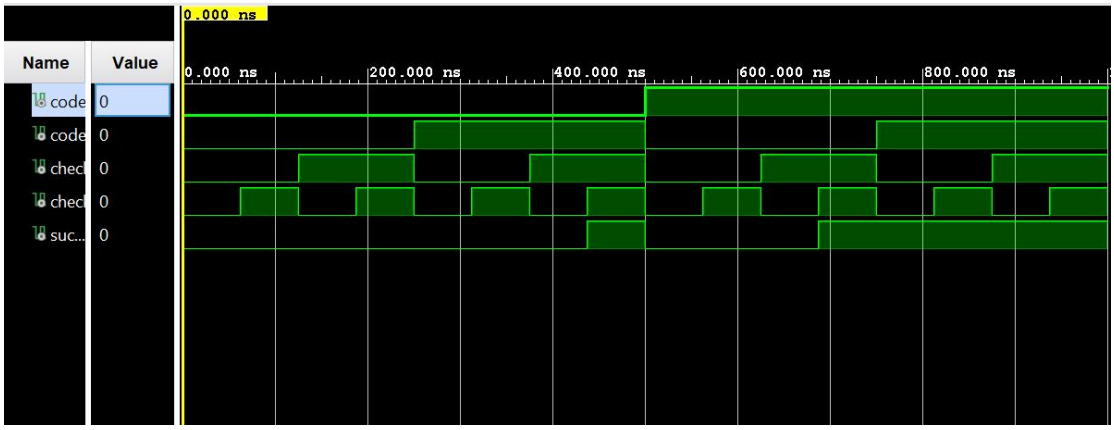

Figure 3.2 Test bench waveform display

After the synthesis, implementation and bitstream code generation are completed the design is ready to be implemented in BASYS 3 board. In the hardware manager section after selecting auto connect the BASYS 3 is connected via the USB connection and then programmed to implement the combinational digital circuit design. Results of given inputs resulted in intended outputs (like in the simulation). Code1 is and code2 inputs specified as R2 and T1 switches and check1 and check2 inputs as U1 and W2 switches and success as L1 led in constraint file. Figures bellow shows the BASYS 3 implementation of the digital design.

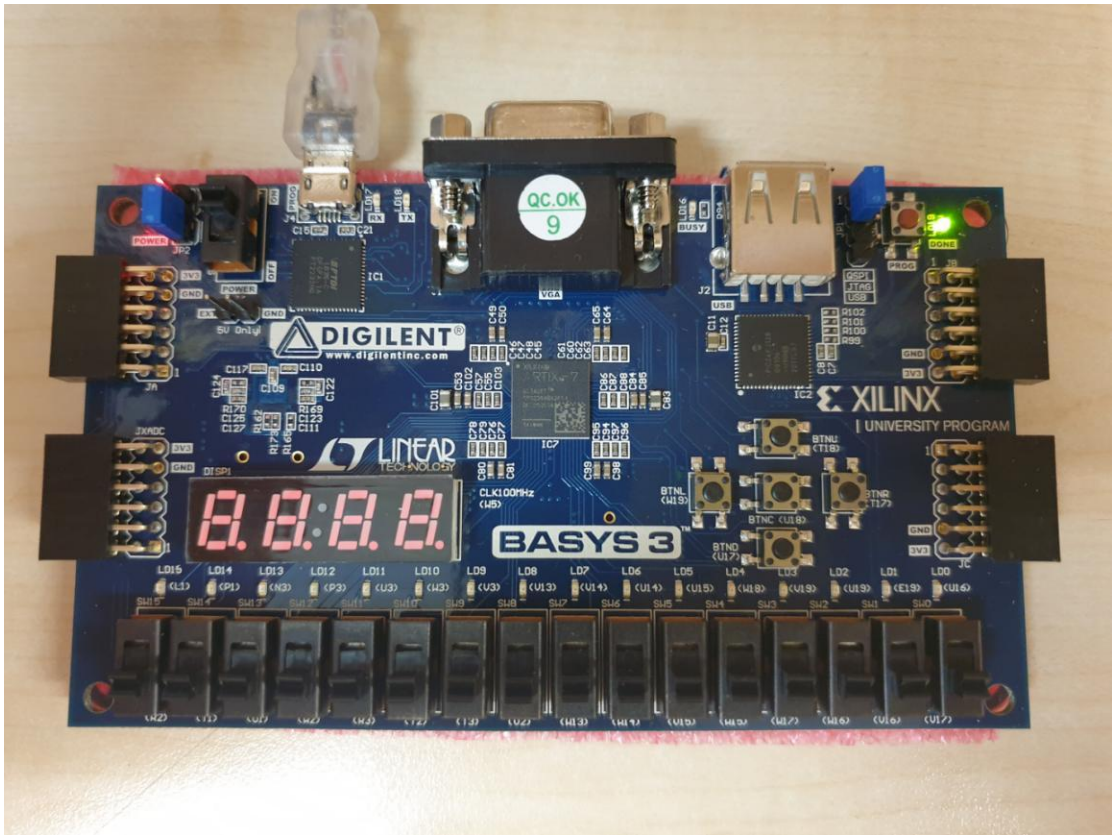

Figure 3.3 showing all inputs 0 (R2 = 0; T1 = 0; U1 = 0; W2 = 0; L1 is off)

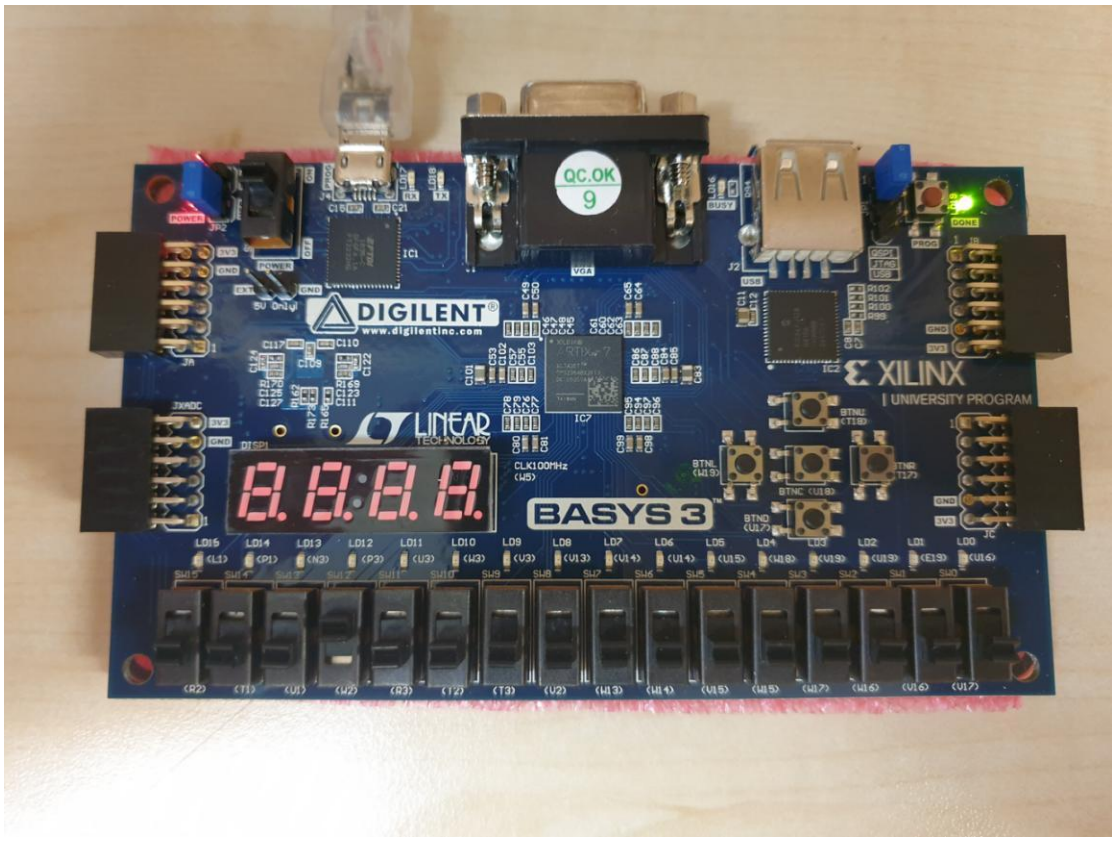

Figure 3.4 showing (R2 = 0; T1 = 0; U1 = 0; W2 = 1; L1 is off)

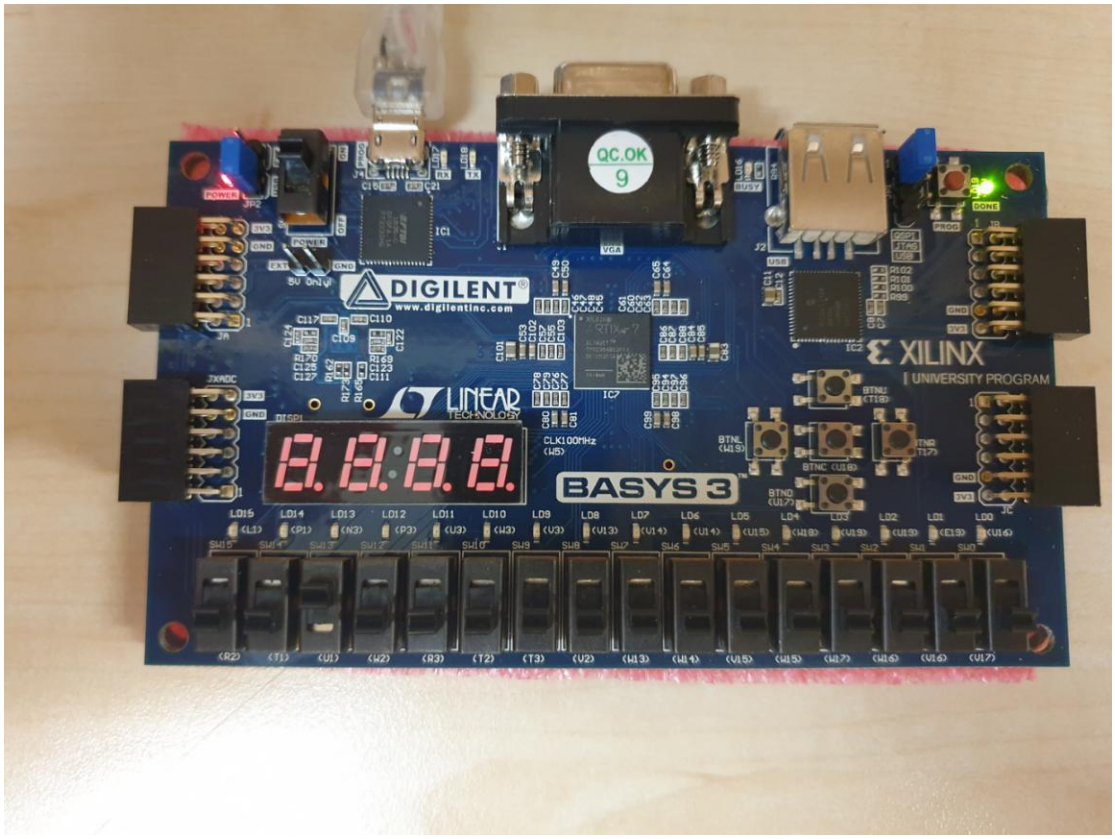

Figure 3.5 showing (R2 = 0; T1 = 0; U1 = 1; W2 = 0, L1 is off)

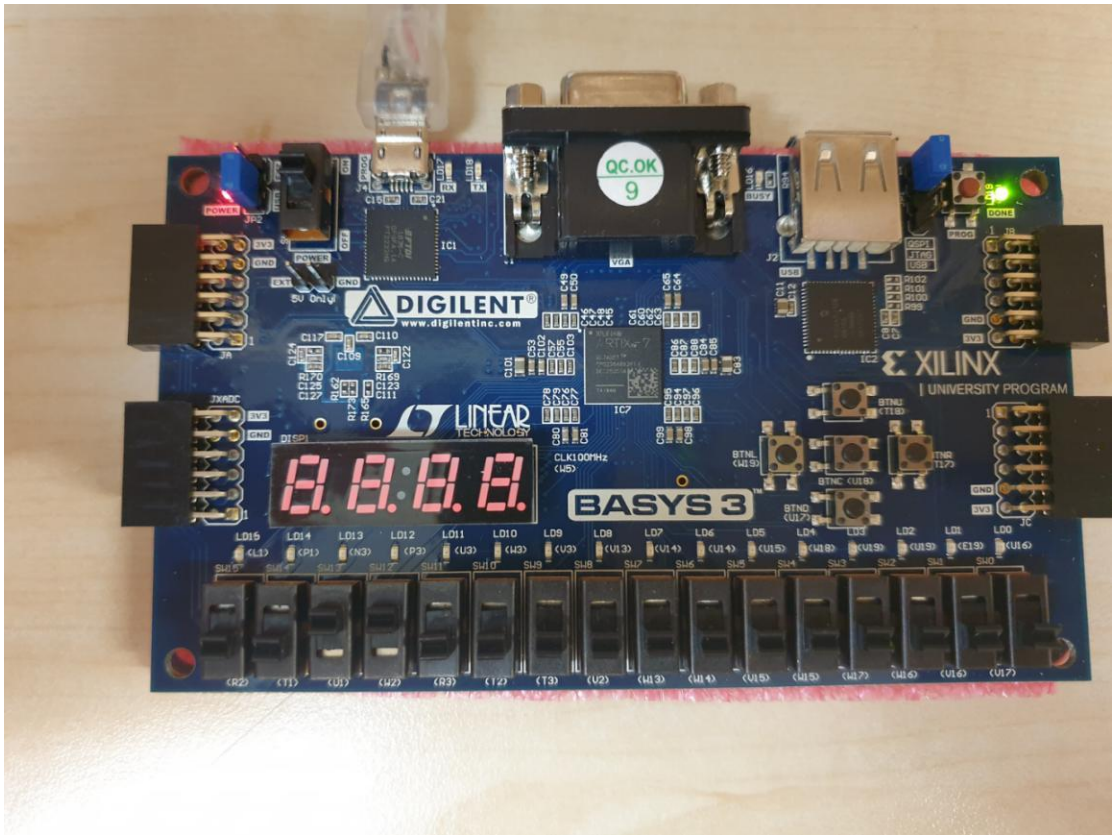

Figure 3.6 showing (R2 = 0; T1 = 0; U1 = 1; W2 = 1, L1 is off)

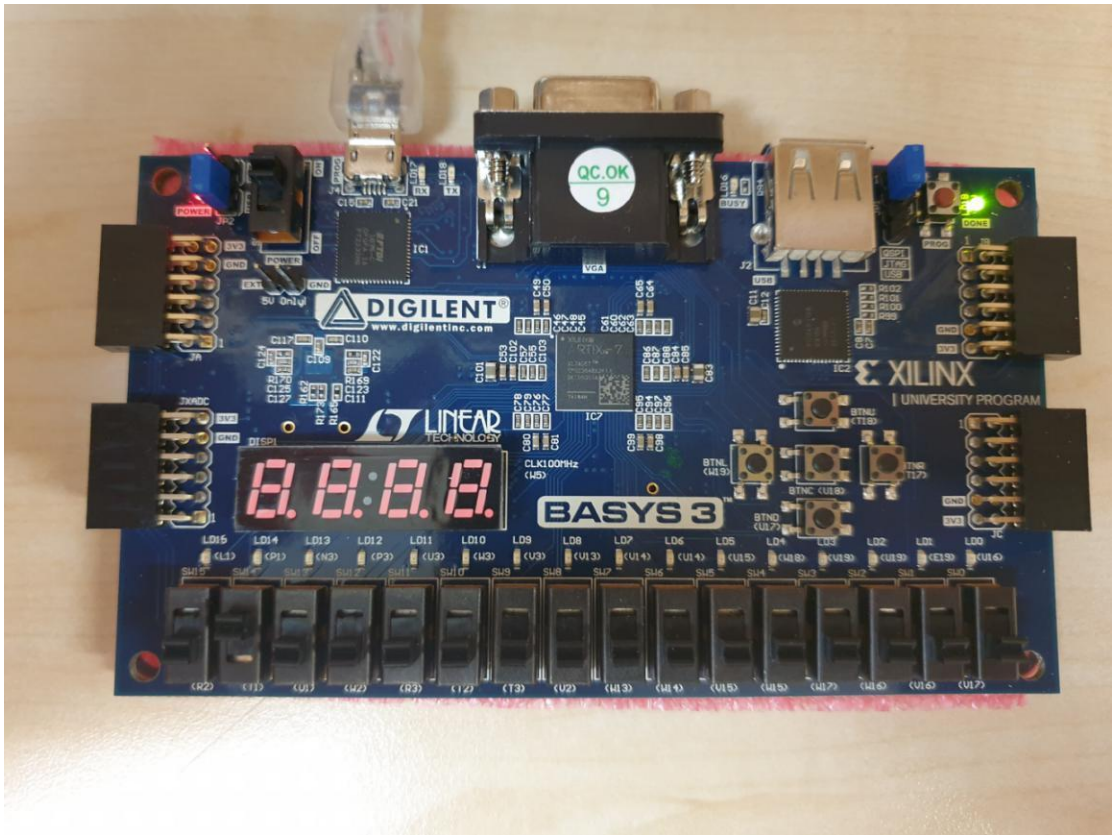

Figure 3.7 showing (R2 = 0; T1 = 1; U1 = 0; W2 = 0; L1 is off)

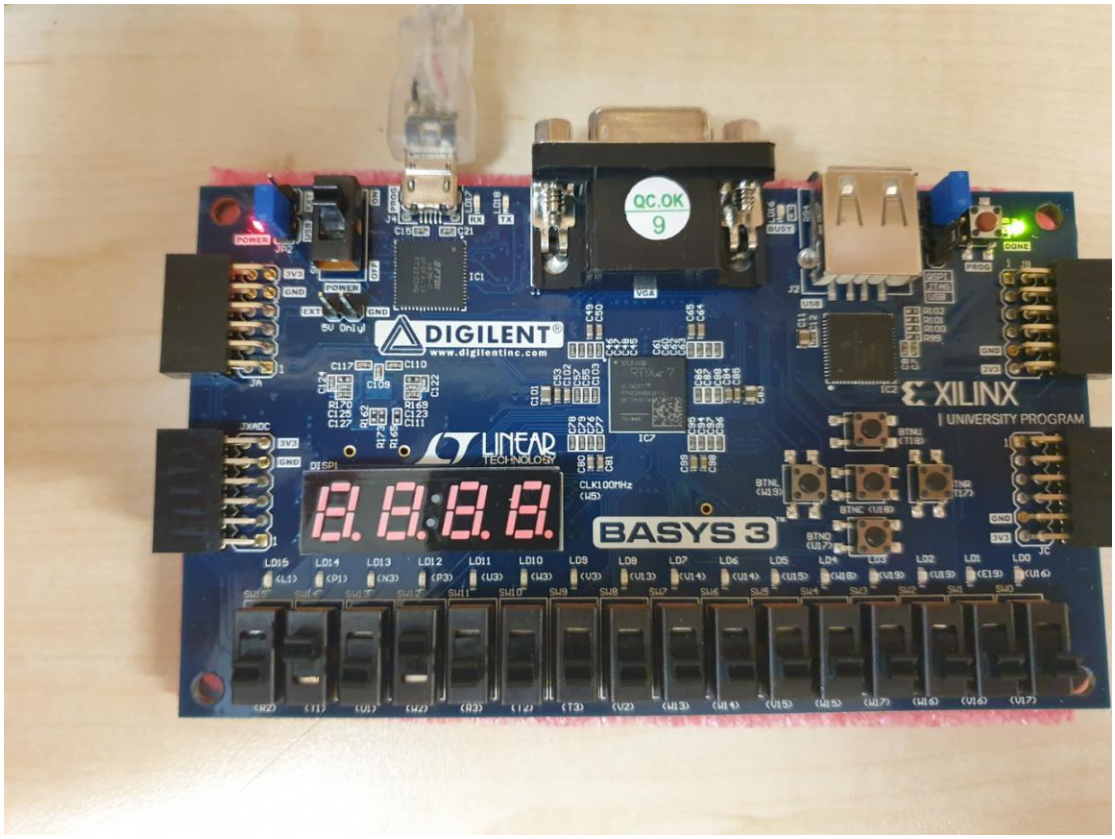

Figure 3.8 showing (R2 = 0; T1 = 1; U1 = 0; W2 = 1; L1 is off)

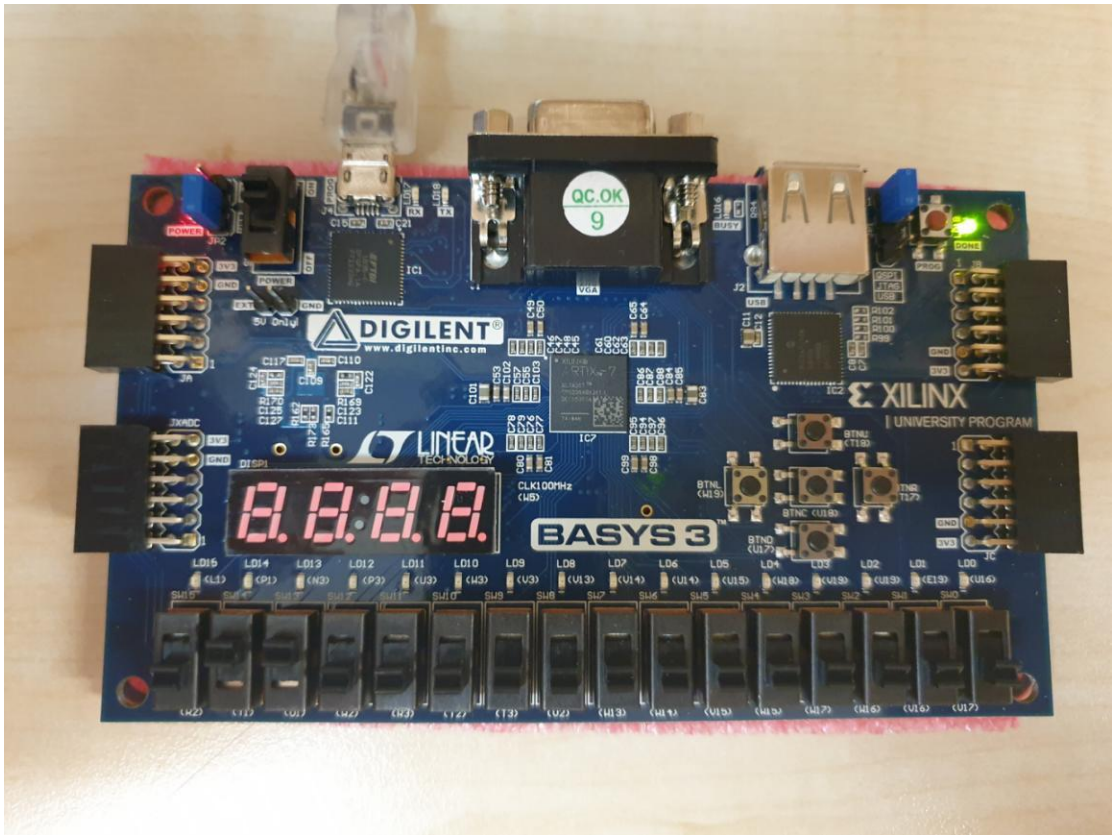

Figure 3.9 showing (R2 = 0; T1 = 1; U1 = 1; W2 = 0; L1 is off)

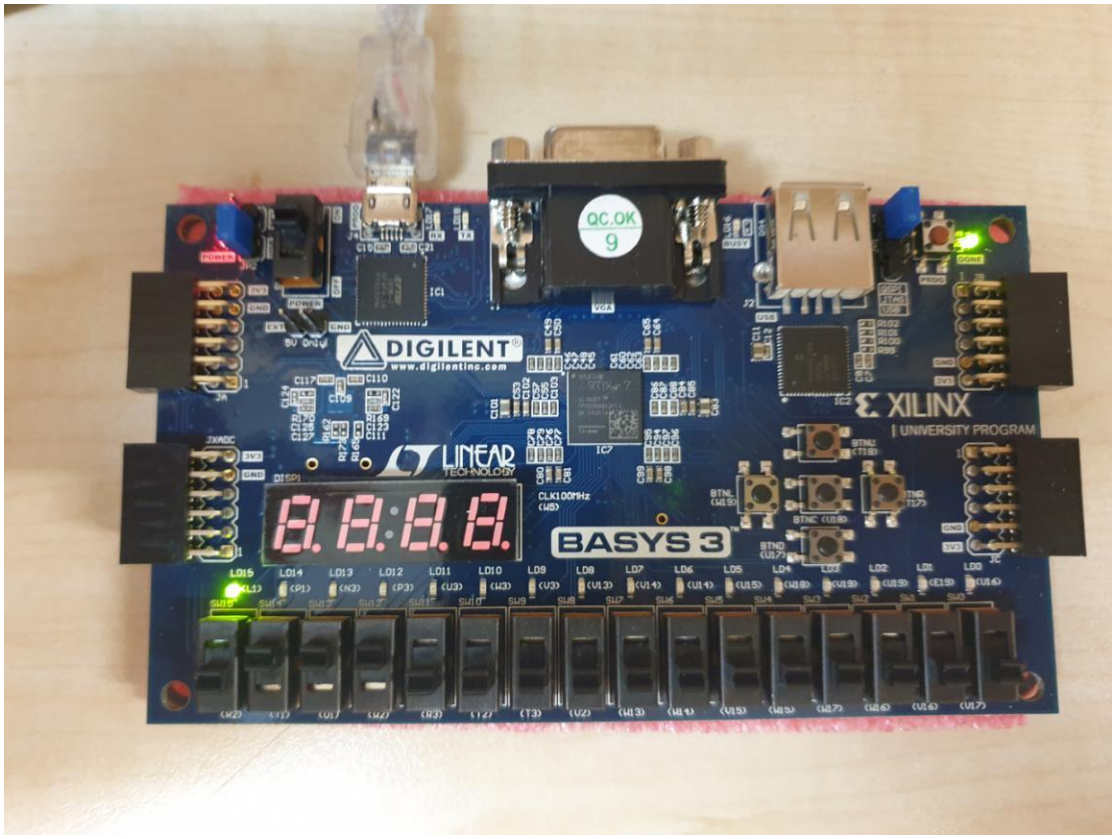

Figure 3.10 showing (R2 = 0; T1 = 1; U1 = 1; W2 = 1; L1 is on)

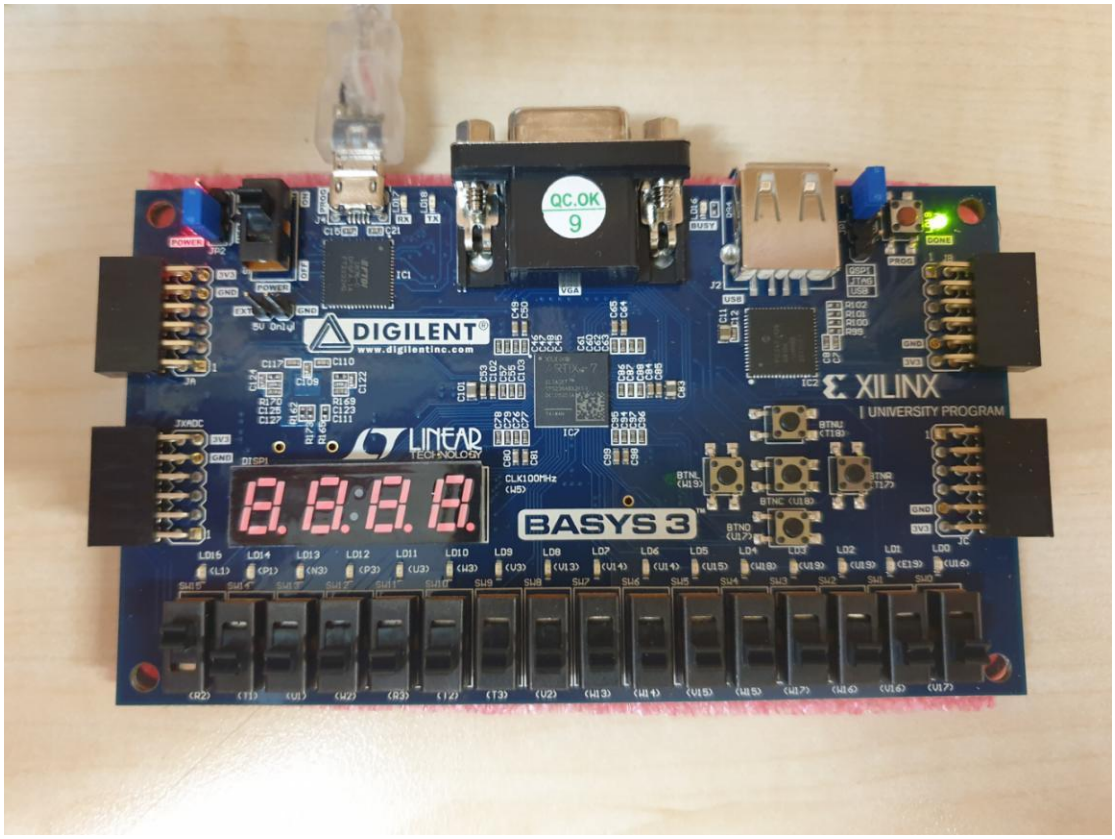

Figure 3.11 showing (R2 = 1; T1 = 0; U1 = 0; W2 = 0; L1 is off)

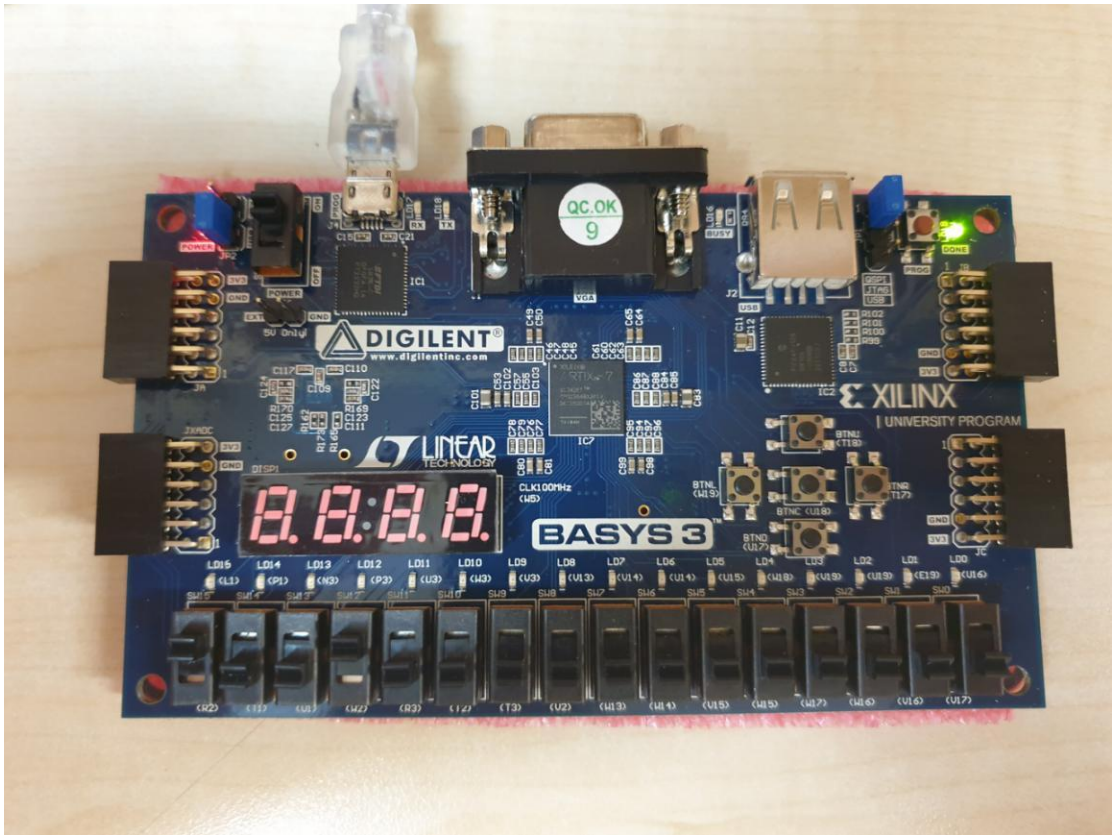

Figure 3.12 showing (R2 = 1; T1 = 0; U1 = 0; W2 = 1; L1 is off)

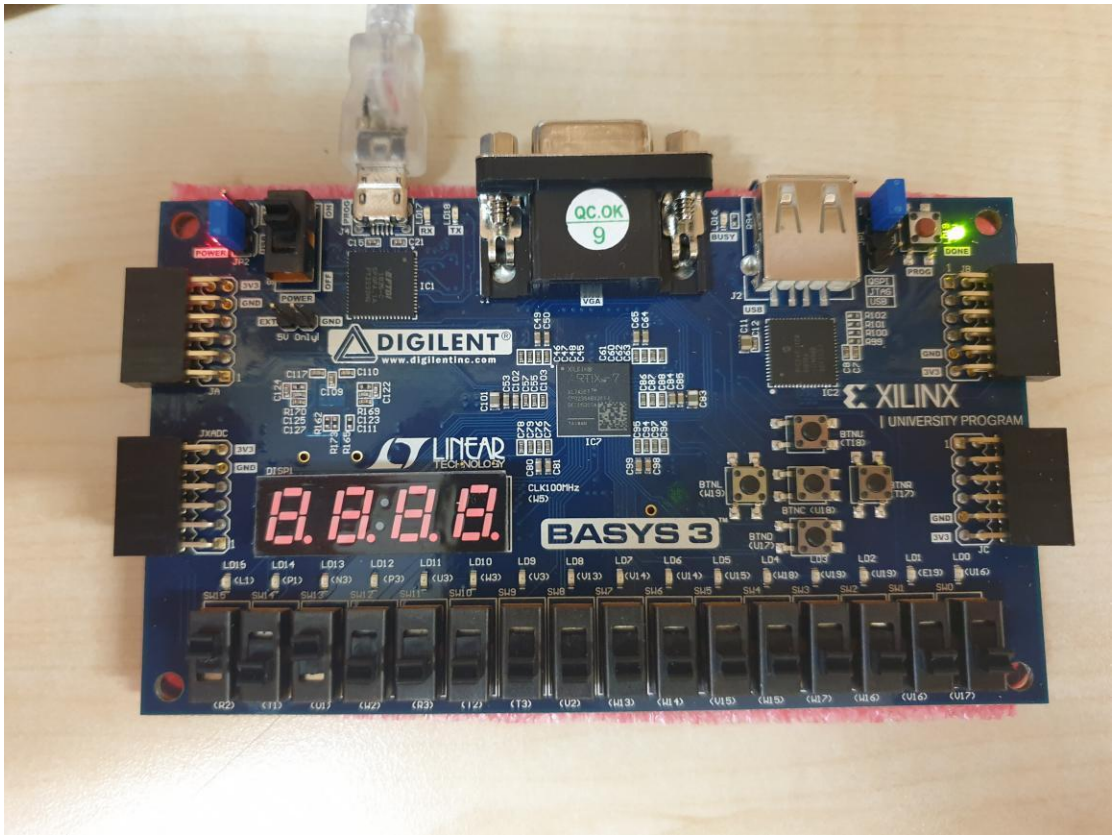

Figure 3.13 showing (R2 = 1; T1 = 0; U1 = 1; W2 = 0; L1 is off)

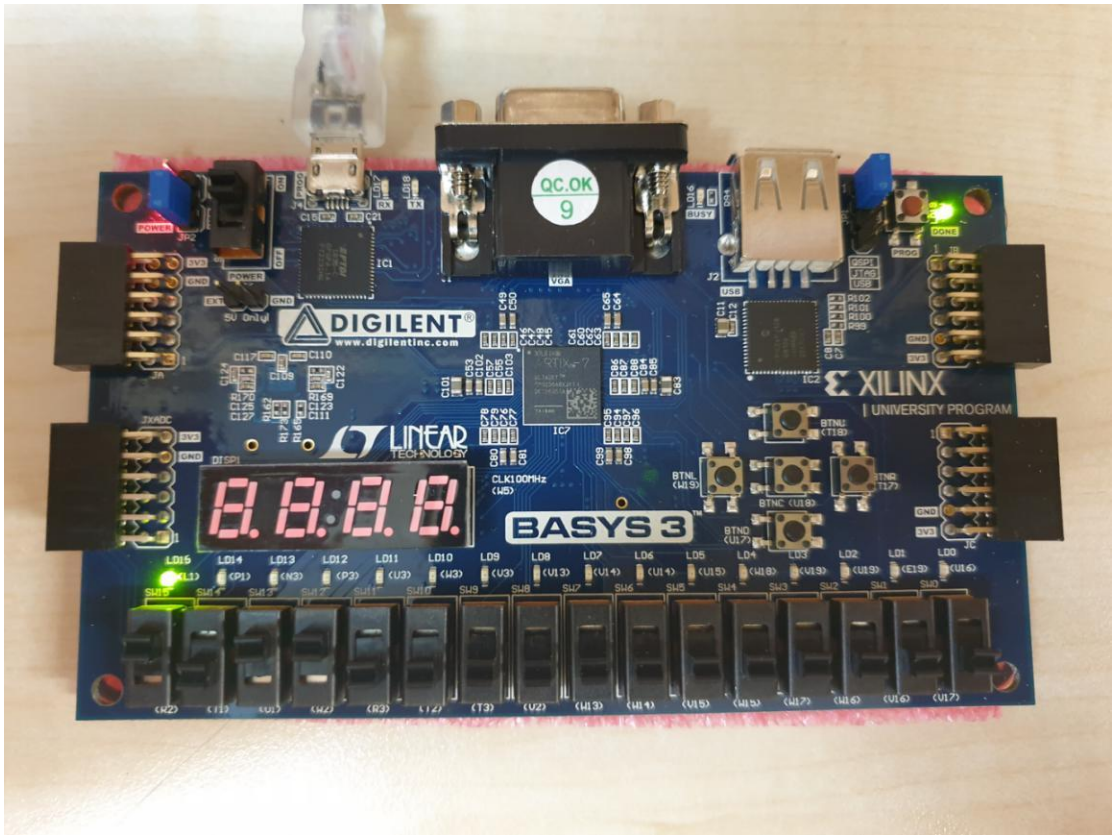

Figure 3.14 showing (R2 = 1; T1 = 0; U1 = 1; W2 = 1; L1 is on)

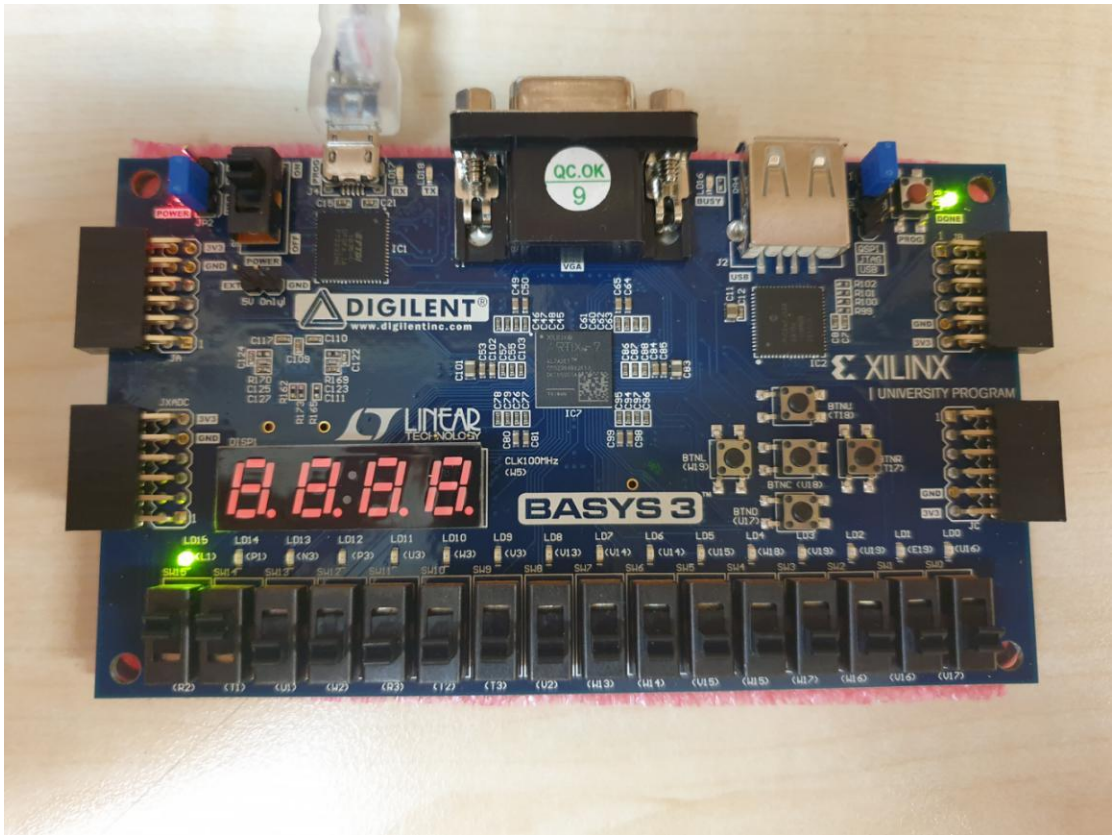

Figure 3.15 showing (R2 = 1; T1 = 1; U1 = 0; W2 = 0; L1 is on)

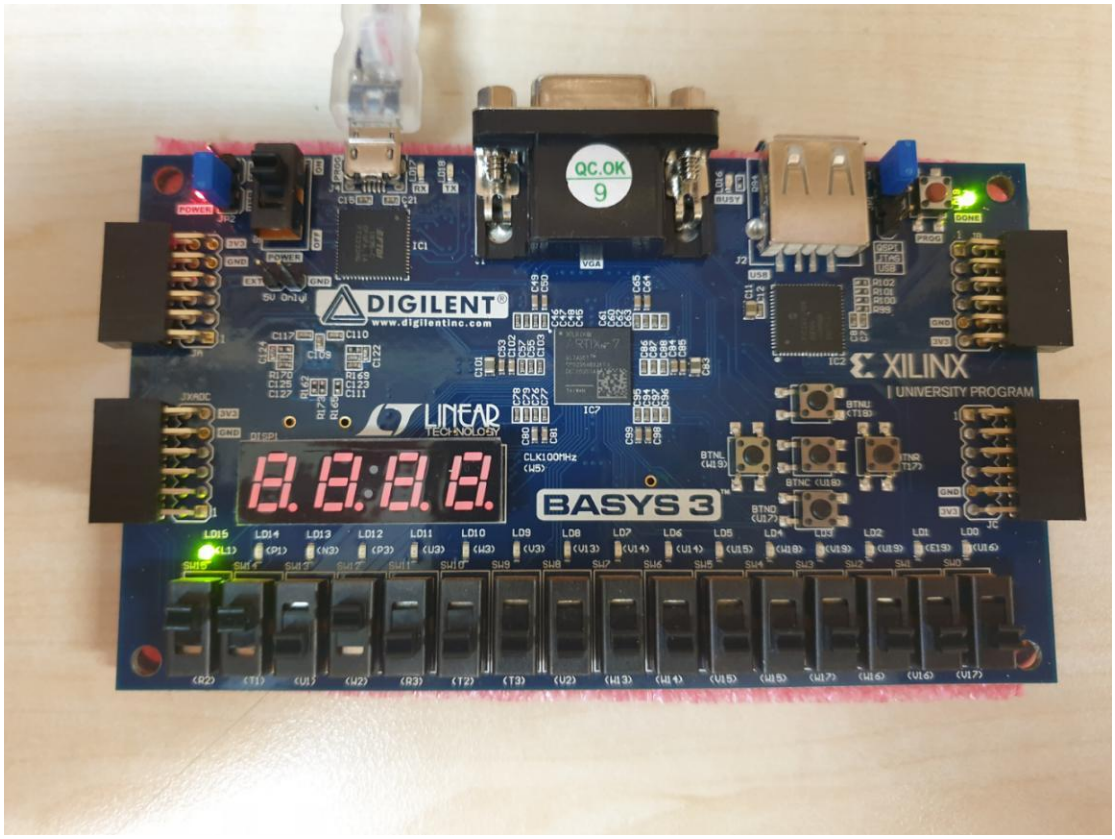

Figure 3.16 showing (R2 = 1; T1 = 1; U1 = 0; W2 = 1; L1 is on)

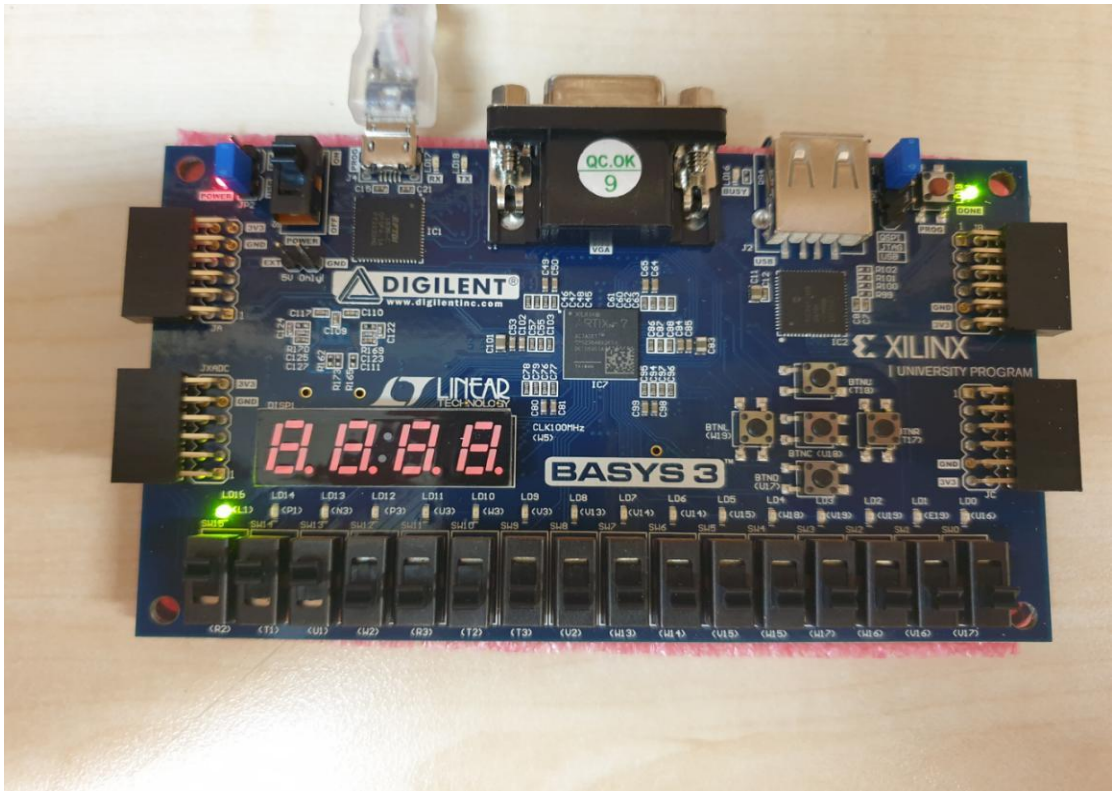

Figure 3.17 showing (R2 = 1; T1 = 1; U1 = 1; W2 = 0; L1 is on)

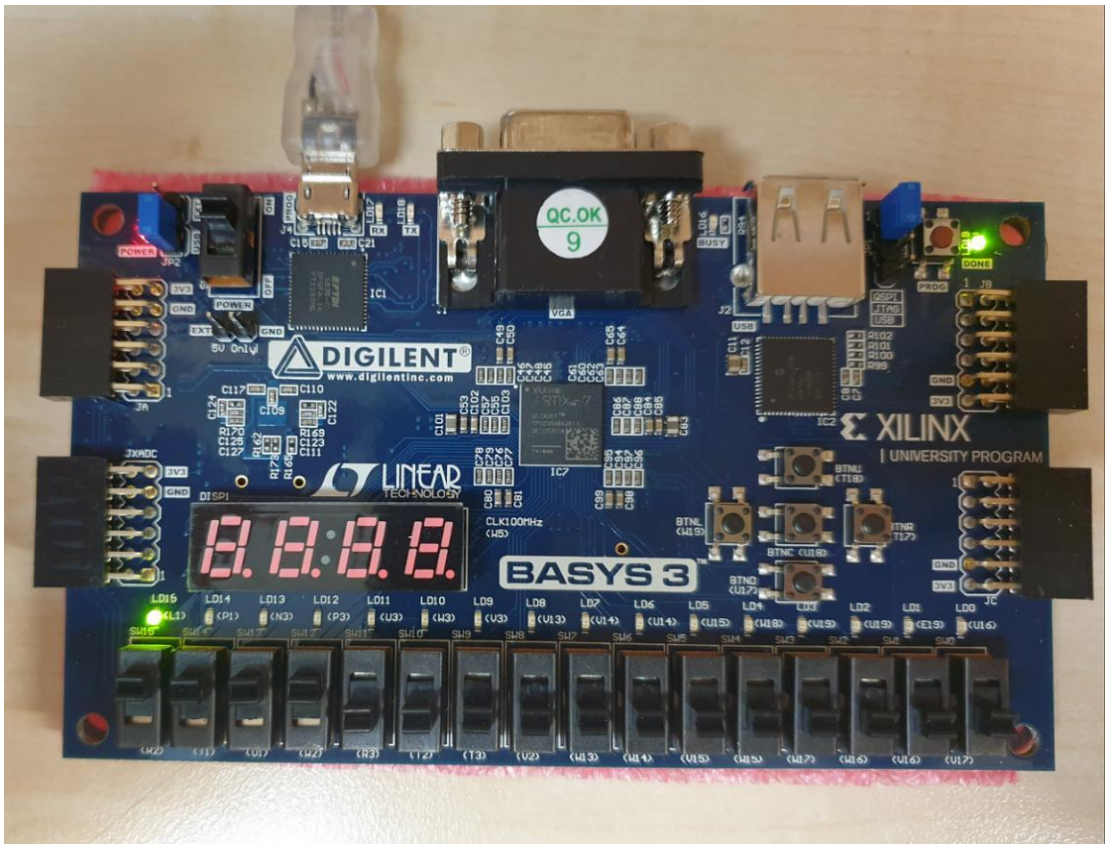

Figure 3.18 showing (R2 = 1; T1 = 1; U1 = 1; W2 = 1; L1 is on)

4) Conclusion

In this experiment we understand combinational digital circuit design and its implementation in FPGA board BASYS 3 with VHDL. We learned how to describe a combinational digital circuit in VHDL language and we also learned how to simulate it using hierarchical programming and how to examine the schematic representation of a given design. In order to implement the design to a FPGA board we also learned how to specify input and output signals on the board using a constraint file. Consequently, we understand the structure of both VHDL and BASYS 3 board and how to create designs simulations and implementations of the designs.

5) Appendices

1- VHDL design code

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

entity lab2 is
    Port ( code1 : in STD_LOGIC;
           code2 : in STD_LOGIC;
           check1 : in STD_LOGIC;
           check2 : in STD_LOGIC;
           success : out STD_LOGIC);
end lab2;

architecture Behavioral of lab2 is

begin
success <= ((check1 and check2)and (code1 or code2))or (code1 and code2);
```

```
end Behavioral;
```

2- Test Bench Code

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

entity lab2_testbench is
end lab2_testbench;

architecture Behavioral of lab2_testbench is
component lab2
Port ( code1 : in STD_LOGIC;
       code2 : in STD_LOGIC;
       check1 : in STD_LOGIC;
       check2 : in STD_LOGIC;
       success : out STD_LOGIC);
END component;
signal code1 : std_logic:= '0';
signal code2 : std_logic:='0';
signal check1 : std_logic:='0';
signal check2 : std_logic:='0';
signal success : std_logic;
begin
uut: lab2 PORT MAP(
       code1 => code1,
       code2 => code2,
       check1 => check1,
```

```
        check2 => check2,
        success=> success);
stim_proc: process
 begin
code1<='0';code2<='0';check1<='0'; check2<='0';
wait for 62.5ns;
code1<='0';code2<='0';check1<='0'; check2<='1';
wait for 62.5ns;
code1<='0';code2<='0';check1<='1'; check2<='0';
wait for 62.5ns;
code1<='0';code2<='0';check1<='1'; check2<='1';
wait for 62.5ns;
code1<='0';code2<='1';check1<='0'; check2<='0';
wait for 62.5ns;
code1<='0';code2<='1';check1<='0'; check2<='1';
wait for 62.5ns;
code1<='0';code2<='1';check1<='1'; check2<='0';
wait for 62.5ns;
code1<='0';code2<='1';check1<='1'; check2<='1';
wait for 62.5ns;
code1<='1';code2<='0';check1<='0'; check2<='0';
wait for 62.5ns;
code1<='1';code2<='0';check1<='0'; check2<='1';
wait for 62.5ns;
code1<='1';code2<='0';check1<='1'; check2<='0';
wait for 62.5ns;
code1<='1';code2<='0';check1<='1'; check2<='1';
wait for 62.5ns;
code1<='1';code2<='1';check1<='0'; check2<='0';
```

```
wait for 62.5ns;
code1<='1';code2<='1';check1<='0'; check2<='1';
wait for 62.5ns;
code1<='1';code2<='1';check1<='1'; check2<='0';
wait for 62.5ns;
code1<='1';code2<='1';check1<='1'; check2<='1';
wait for 62.5ns;
 end process;
end Behavioral;
```

3- Constraints File

```
set_property PACKAGE_PIN W2 [get_ports {check2}]
   set_property IOSTANDARD LVCMOS33 [get_ports {check2}]
set_property PACKAGE_PIN U1 [get_ports {check1}]
   set_property IOSTANDARD LVCMOS33 [get_ports {check1}]
set_property PACKAGE_PIN T1 [get_ports {code2}]
   set_property IOSTANDARD LVCMOS33 [get_ports {code2}]
set_property PACKAGE_PIN R2 [get_ports {code1}]
   set_property IOSTANDARD LVCMOS33 [get_ports {code1}]
set_property PACKAGE_PIN L1 [get_ports {success}]
   set_property IOSTANDARD LVCMOS33 [get_ports {success}]
```