Efe Tarhan

22002840

EE102-02

13.10.2021

LAB 2: INTRODUCTION TO VHDL

A) PURPOSE

This experiment has been conducted to understand the VHDL coding language, Vivado software, Basys 3 operations, simulating the input and outputs using simulation files. Designing a real-life problem and finding and implementing a solution for that problem is the aim of this lab session. The logic system in the lab experiment is designed for tracking the dental health and hygiene of people. A person should brush his/her teeth at least two times a day and use tooth floss or dental rinse. The dental health tracking system helps people ensure that their dental health is in good condition for each day.

B) METHODOLOGY

Dental health is one of the most important medical problems in Turkey. People don’t have good habits to keep tracking their dental health. This project will help people track their dental wellness and feedback if they have done their dental care appropriately.

The project has been designed in the following way. The X variable stands for brushing teeth in the morning, the Y variable stands for brushing teeth at noon, and the Z variable stands for brushing teeth at night. The Q variable stands for flossing the teeth, and the T variable stands for using the dental rinse.

A person must brush his/her teeth at least two times a day and use tooth floss or dental rinse. Therefore the logical expression is the following :

$$F(Dental\ Health) = [(X.Y + Y.Z + X.Z).(Q + T)] \quad \text{(Eq 1)}$$

The truth table of the logical expression is :

X	*Y*	*Z*	*Q*	*T*	*F*
0	**0**	**0**	**0**	**0**	***0***
0	**0**	**0**	**0**	**1**	***0***
0	**0**	**0**	**1**	**0**	***0***
0	**0**	**0**	**1**	**1**	***0***
0	**0**	**1**	**0**	**0**	***0***
0	**0**	**1**	**0**	**1**	***0***
0	**0**	**1**	**1**	**0**	***0***
0	**0**	**1**	**1**	**1**	***0***
0	**1**	**0**	**0**	**0**	***0***
0	**1**	**0**	**0**	**1**	***0***
0	**1**	**0**	**1**	**0**	***0***
0	**1**	**0**	**1**	**1**	***0***
0	**1**	**1**	**0**	**0**	***0***
0	**1**	**1**	**0**	**1**	***1***
0	**1**	**1**	**1**	**0**	***1***
0	**1**	**1**	**1**	**1**	***1***
1	**0**	**0**	**0**	**0**	***0***
1	**0**	**0**	**0**	**1**	***0***
1	**0**	**0**	**1**	**0**	***0***
1	**0**	**0**	**1**	**1**	***0***
1	**0**	**1**	**0**	**0**	***0***
1	**0**	**1**	**0**	**1**	***0***
1	**0**	**1**	**1**	**0**	***1***
1	**0**	**1**	**1**	**1**	***1***
1	**1**	**0**	**0**	**0**	***0***
1	**1**	**0**	**0**	**1**	***1***
1	**1**	**0**	**1**	**0**	***1***
1	**1**	**0**	**1**	**1**	***1***
1	**1**	**1**	**0**	**0**	***0***
1	**1**	**1**	**0**	**1**	***1***
1	**1**	**1**	**1**	**0**	***1***
1	**1**	**1**	**1**	**1**	***1***

(Table 1: Truth table of expression $F = (X.Y + Y.Z + X.Z).(Q + T)$)

The project will be implemented by using the Basys3 FPGA board. VHDL will be used for coding the device.

Output and input variables in VHDL can be specified by using the following code lines;

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

entity test1 is
        Port( in1 : in STD_LOGIC;
              in2: in STD_LOGIC;
             out1: out STD_LOGIC);
end test1;
--------------------------------------
```

The port function declares the input and output variables of a logic function. The blue codes are the names of each input or output variable. Red-colored codes indicate if the variable is an input or an output variable. The purple-colored "STD_LOGIC" is a variable type which means that the different signal values (like 1, 0, U, X, etc.) could be assigned to that specific variable. Rather than manually creating the variables, variables can be declared by doing few operations in Vivado Software after clicking the "Add Sources" and "Create Design Sources" buttons.

The "Port map" function has a similar function to the inheritance in coding languages. Usage of the port map is the following. First, the user creates a module with an entity and port function that declares variables and their types; also, a function could be created inside this module. Using the port map function, the user can copy all of those variables and functions in the wanted module of the same project. For example, we created variables and functions in the designing source module by using a port function in this experiment. The port map function is being called at the test bench step, and the variables and functions are copied to the test bench module with the source module itself. So by using the port map function, the design module has been used inside the test bench module. An example port map code is below:

```
UUT: test1 PORT MAP( in1 => in1,
                     in2 => in2,
                    out1 => out1 );
```

As mentioned above, the test1 module is called inside a testbench module. Therefore a module can be used inside another module by using port map mode and wanted variables and functions can be selected by declaring them with the “=>” sign.

Constraint files help assign variables of a function to specific switches, LEDs or clicks on the FPGA board (Basys3 for the current lab). After assigning variables to constraints, the code can be generated into the bitstream and be uploaded to the FPGA board. A constraint could be declared with the following codes in an XDC file;

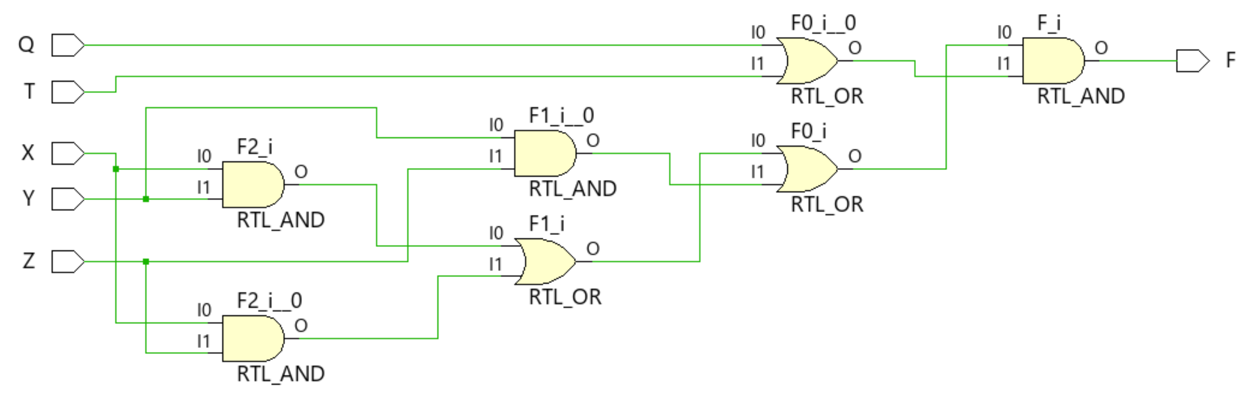
```
set_property PACKAGE_PIN V17 [get_ports {in1}]
        set_property IOSTANDARD LVCMOS33 [get_ports {in1}]
```

The red-colored code means that the blue-colored variable has been assigned to the V17 pin on the board by using the set_property function.

The test bench is an essential feature of Vivado Software. It makes the user available to practice and test possible outputs of the written code by creating a test bench file and simulation. The simulation is created by assigning signal values for specified periods to the wanted variables. The test bench shows the outcomes of the design code without uploading it to the device. Therefore it helps to prevent possible errors and unwanted outcomes of the code rather than uploading and testing the code on the Basys3 each time there is an error.

C) RESULTS

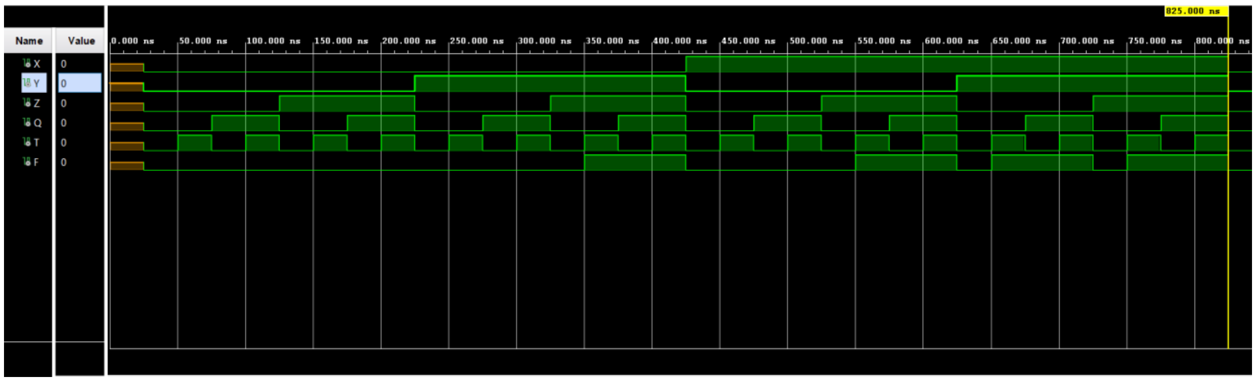
A project was created in Vivado, and a design source with 5 inputs (X, Y, Z, Q, T) and 1 output variable (F) was added. The equation of the logic function has been written in the VHDL language. The synthesis and implementation steps have been done to get the elaborated design of the logic expression, also called the RTL design. RTL design of the function is Image 1.



(Image 1: RTL Schematic of the logic expression)

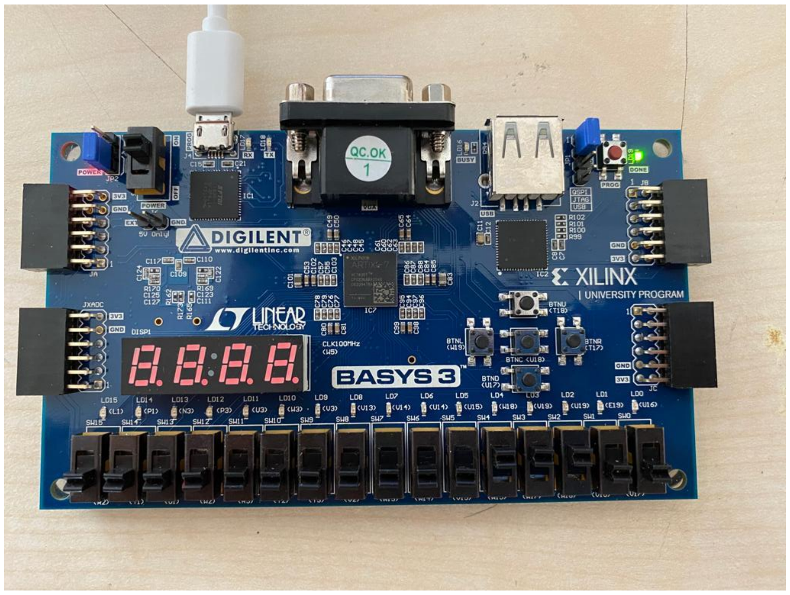
After completing the synthesis and the implementation steps, the test code is ready to be tested. A test bench code has been written to create a signal simulation for the function.

Because there are five variables, there must be $2^5 = 32$ different arrangings. All 32 conditions with different signal combinations were written to code with 25ns durations. The expected result has been achieved without any error. The waveform signal simulation graph is Image 2.

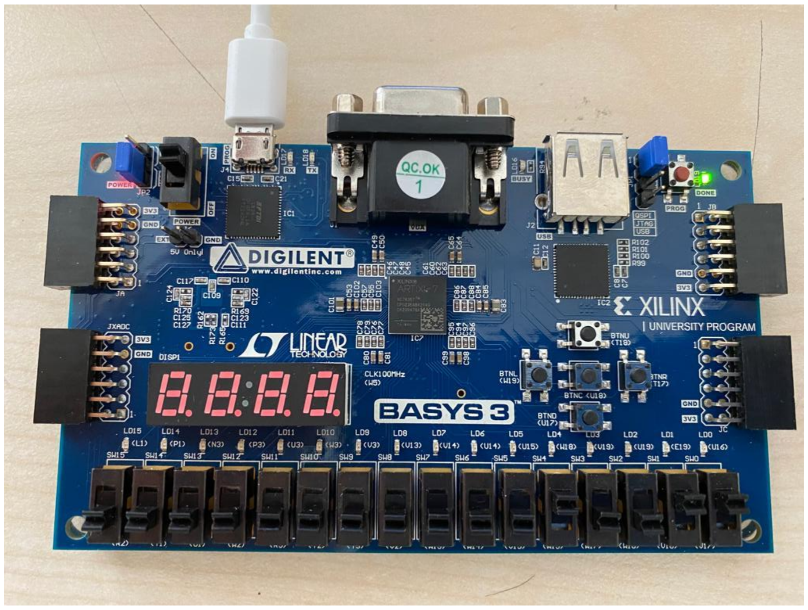


(Image 2: Output Waveform Signal Simulation of the logic expression)

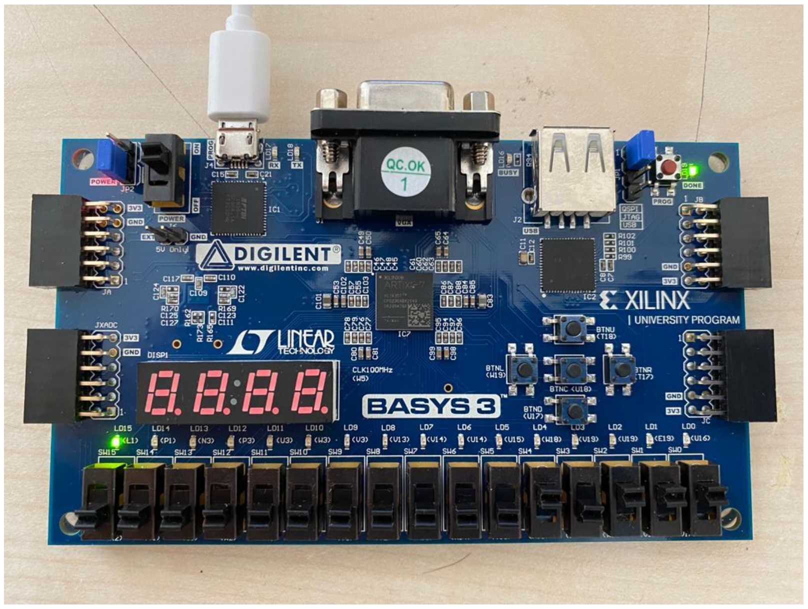
No errors appeared after completing synthesis, implementation, and simulation steps; therefore, the code was uploaded to Basys3 FPGA. By adding constraints, inputs of the logic expression have been assigned to the first five switches from the right (W15, W17, W16, V16, V17), and the output of the expression has been assigned to the green led at the far left (L1). After creating the constraint file, the code was ready for bitstream generation. After creating bitstream, the device has auto-connected to the Basys3, and the device has been programmed to implement the logic function. Examples of the implementation of the code can be seen from the images below.



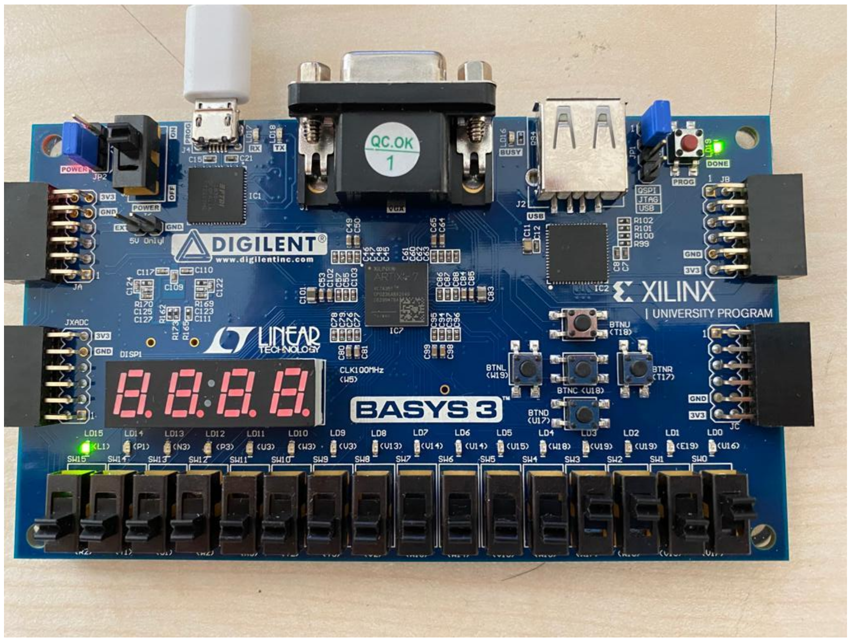
(Image 3: $X \ <= \ 1, Y \ <= \ 1, Z \ <= \ 1, Q \ <= \ 0, T \ <= \ 0$ hence $F \equiv 0$ which means dental health is bad)



(Image 4: $X <= 1, Y <= 0, Z <= 0, Q <= 1, T <= 1$ hence $F \equiv 0$ which means dental health is bad)



(Image 5: $X <= 1, Y <= 0, Z <= 1, Q <= 1, T <= 1$ hence $F \equiv 1$ which means dental health is well)



(Image 6: $X <= 0, Y <= 1, Z <= 1, Q <= 0, T <= 1$ hence $F \equiv 1$ which means dental health is well)

D) CONCLUSION

This experiment was essential for understanding the structure of Basys3, the FPGA board. Learning how to code in VHDL, creating a simulation, synthesis, and creating bitstreams are other aims of this experiment. Lastly, it was essential to learn how to implement logic gates and functions in real life.

E) APPENDICES

1-VHDL Code

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

entity Sources is
    Port ( X : in STD_LOGIC;
           Y : in STD_LOGIC;
           Z : in STD_LOGIC;
           Q : in STD_LOGIC;
           T : in STD_LOGIC;
           F : out STD_LOGIC);
end Sources;

architecture Behavioral of Sources is

begin

F <= ((X and Y) or (X and Z)or (Y and Z))and (Q or T);
end Behavioral;
```

2- Test Bench Code

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
entity testBench1 is
end testBench1;
architecture Behavioral of testBench1 is
COMPONENT Sources
 PORT( X : in STD_LOGIC;
            Y : in STD_LOGIC;
            Z : in STD_LOGIC;
            Q : in STD_LOGIC;
            T : in STD_LOGIC;
            F : out STD_LOGIC);
 END COMPONENT;

 SIGNAL X :  STD_LOGIC;
 SIGNAL Y :  STD_LOGIC;
 SIGNAL Z :  STD_LOGIC;
 SIGNAL Q :  STD_LOGIC;
 SIGNAL T :  STD_LOGIC;
 SIGNAL F :  STD_LOGIC;
BEGIN
 UUT: Sources PORT MAP(
X => X,
Y => Y,
Z => Z,
Q => Q,
T => T,
F => F
 );
 testBench1 : PROCESS
 BEGIN
 wait for 25 ns;
 X <='0';
 Y <='0';
 Z <='0';
 Q <='0';
 T <='0';
 wait for 25 ns;
 X <='0';
 Y <='0';
 Z <='0';
 Q <='0';
 T <='1';
 wait for 25 ns;
 X <='0';
 Y <='0';
 Z <='0';
 Q <='1';
 T <='0';
 wait for 25 ns;
 X <='0';
 Y <='0';
 Z <='0';
 Q <='1';
 T <='1';
```

```
wait for 25 ns;

X <='0';

Y <='0';

Z <='1';

Q <='0';

T <='0';

wait for 25 ns;

X <='0';

Y <='0';

Z <='1';

Q <='0';

T <='1';

wait for 25 ns;

X <='0';

Y <='0';

Z <='1';

Q <='1';

T <='0';

wait for 25 ns;

X <='0';

Y <='0';

Z <='1';

Q <='1';

T <='1';

wait for 25 ns;

X <='0';

Y <='1';
```

```
Z <='0';
Q <='0';
T <='0';
wait for 25 ns;
X <='0';
Y <='1';
Z <='0';
Q <='0';
T <='1';
wait for 25 ns;
X <='0';
Y <='1';
Z <='0';
Q <='1';
T <='0';
wait for 25 ns;
X <='0';
Y <='1';
Z <='0';
Q <='1';
T <='1';
wait for 25 ns;
X <='0';
Y <='1';
Z <='1';
Q <='0';
T <='0';
```

```
wait for 25 ns;

X <='0';

Y <='1';

Z <='1';

Q <='0';

T <='1';

wait for 25 ns;

X <='0';

Y <='1';

Z <='1';

Q <='1';

T <='0';

wait for 25 ns;

X <='0';

Y <='1';

Z <='1';

Q <='1';

T <='1';

wait for 25 ns;

X <='1';

Y <='0';

Z <='0';

Q <='0';

T <='0';

wait for 25 ns;

X <='1';

Y <='0';
```

```
Z <='0';
Q <='0';
T <='1';
wait for 25 ns;
X <='1';
Y <='0';
Z <='0';
Q <='1';
T <='0';
wait for 25 ns;
X <='1';
Y <='0';
Z <='0';
Q <='1';
T <='1';
wait for 25 ns;
X <='1';
Y <='0';
Z <='1';
Q <='0';
T <='0';
wait for 25 ns;
X <='1';
Y <='0';
Z <='1';
Q <='0';
T <='1';
```

```
wait for 25 ns;

X <='1';

Y <='0';

Z <='1';

Q <='1';

T <='0';

wait for 25 ns;

X <='1';

Y <='0';

Z <='1';

Q <='1';

T <='1';

wait for 25 ns;

X <='1';

Y <='1';

Z <='0';

Q <='0';

T <='0';

wait for 25 ns;

X <='1';

Y <='1';

Z <='0';

Q <='0';

T <='1';

wait for 25 ns;

X <='1';

Y <='1';
```

```
Z <='0';

Q <='1';

T <='0';

wait for 25 ns;

X <='1';

Y <='1';

Z <='0';

Q <='1';

T <='1';

wait for 25 ns;

X <='1';

Y <='1';

Z <='1';

Q <='0';

T <='0';

wait for 25 ns;

X <='1';

Y <='1';

Z <='1';

Q <='0';

T <='1';

wait for 25 ns;

X <='1';

Y <='1';

Z <='1';

Q <='1';

T <='0';
```

```
 wait for 25 ns;
 X <='1';
 Y <='1';
 Z <='1';
 Q <='1';
 T <='1';

 END PROCESS;
end Behavioral;
```

3-Constraints

```
#PINS
set_property PACKAGE_PIN V17 [get_ports {T}]
   set_property IOSTANDARD LVCMOS33 [get_ports {T}]
set_property PACKAGE_PIN V16 [get_ports {Q}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Q}]
set_property PACKAGE_PIN W16 [get_ports {Z}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Z}]
set_property PACKAGE_PIN W17 [get_ports {Y}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Y}]
set_property PACKAGE_PIN W15 [get_ports {X}]
   set_property IOSTANDARD LVCMOS33 [get_ports {X}]

#LEDS
set_property PACKAGE_PIN L1 [get_ports {F}]
   set_property IOSTANDARD LVCMOS33 [get_ports {F}]
```