Bilkent University
EE102-02 Lab 2 Report:
Introduction to VHDL

Mehmet Emre Uncu - 22003884

Purpose:

The purpose of this lab is to design and implement a combinational circuit with VHDL on BASYS3. Our goal is to create input and output, to simulate them in waveform by writing testbench and to run this circuit on FPGA by assigning certain switches and LEDs.

Methodology:

First we thought of a real life problem which was a single lane tunnel situation. Then we identified the inputs and outputs of this problem and put them into an equation with XOR, AND and NOT gates. We turned this equation into a logic circuit with VHDL by specifying the variables. After creating a truthtable to test all input and output combinations, we simulated the circuit as a waveform by

writing testbench. Finally, we uploaded my VHDL code to BASYS3 with generate bitstream and implemented it on FPGA.

Design Specifications:

The design in this lab was to produce a solution for a single lane tunnel. It includes 1 XOR gate, 1 AND gate and 1 NOT gate with 3 inputs and 1 output.

X: the vehicle at the first entrance of the tunnel

Y: the vehicle at the second entrance of the tunnel

Z: tunnel police

F: green light output meaning it's safe to pass

There is a connection between the vehicles X and Y at the two ends of the tunnel with the XOR gate, so the passage is safe only when one of them wants to pass. Meanwhile, when the tunnel police deems it necessary, they can block the passage by pressing the Z button on the front, as long as he does not press the button because of the NOT gate, it does not prevent the passage. These two systems are connected to each other by AND gate.

Results:

- We need to use certain words to specify an input or an output. One of them is 'entity'. The entity defines the inputs and outputs of a module by using ‘port’ function.

```
entity lab2 is
    Port (X: in STD_LOGIC;
          Y: in STD_LOGIC;
          Z: in STD_LOGIC;
```

```
        F: out STD_LOGIC);
end lab2;
```

X, Y and Z are the inputs and F is the outputs in this example.

- If we want to use any another module or package in our code, like any other programming language, we need to import that module or package by calling them in our code.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

- In order to use base design in the testbench code we need to use ‘port map’ function. The port map function allows us to connect our local signals to the module variables, inputs and outputs.

```
UUT: lab2 PORT MAP (
        X => X,
        Y => Y,
        Z => Z,
        F => F
    );
```

- The FPGA consist of physical pins arranged on the board. We cannot relocate or rearrange these pins but we can code the functions by constraint (.xdc) file. All of the FPGA pins that are routed out to physical pins on your board are listed in the master XDC file; for simplicity, they are organized by external component groups. Users only need to uncomment the pins they wish to utilize because all of the pins on the master XDC files are commented out.
- Testbench, helps us to simulate our circuit and understand how it works. It allows us to create a waveform that shows how the function behaves with

the values we enter. It allows us to see whether the outputs on the FPGA match the expectations.

Task 1:

The circuit we designed can be written in terms of variables, such as:

$$F(X, Y, Z) = (X \oplus Y) + Z'$$

To understand how the function works, we created a truthtable with input and output in 2^3=8, since there are 3 inputs, different combinations.

Z	Y	X	X ⊕ Y	Z’	F
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

After we wrote the truthtable, we adapted our circuit to VHDL using Vivado. Then we synthesized the circuit and saw that it was working properly and there was no error. Lastly we obtain the RTL schematic of the circuit with elaborated design (Figure 1.1).

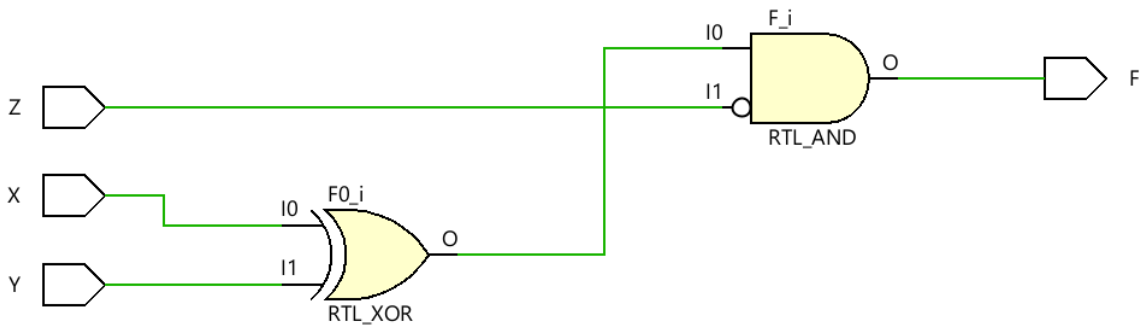

Figure 1.1 Schematics of the Designed Circuit

After all these steps we created a testbench to validate and test my circuit (Code-2). We ran the simulation to use testbench, this process gave us a waveform at 10ns intervals according to the inputs and outputs of our circuit (Figure 1.2).

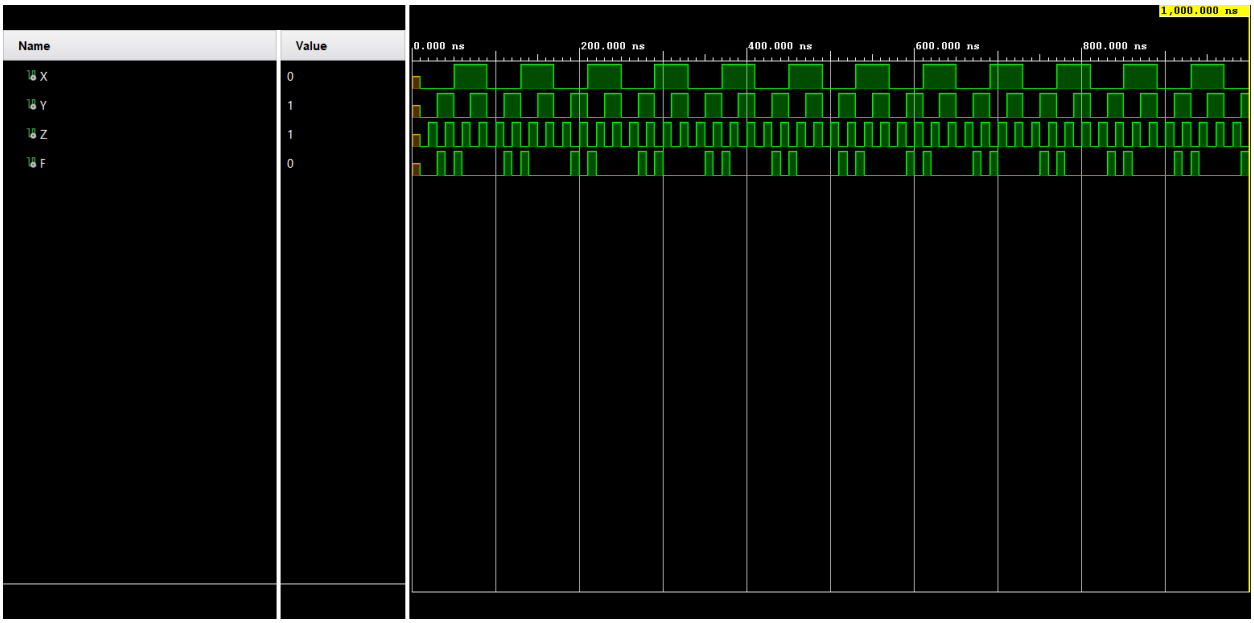

Figure 1.2 Waveform of the Simulation

Task 2:

To demonstrate our circuit on BASYS3 FPGA, we first created a constraint file. Afterwards, we entered the switches we will use as inputs and the LED as outputs in this file.

X: V17 switch

Y: V16 switch

Z: W16 switch

F: E19 LED

Lastly we generate a .bit file and we imported our VHDL design to BASYS3. Thus, we animated the circuit we created using FPGA (Figure 2.1 to Figure 2.8).

Conclusion:

In this lab, we introduced the basics of digital design using VHDL and FPGA. We turned a real life case into a circuit with logic gates. Using Vivado, we learned how to determine input and output, turn them into a circuit with logic gates, synthesize our circuit to reveal possible errors, create a testbench to simulate our circuit as a waveform and experience how it works. Finally, we learned to demonstrate the circuit we created on FPGA. Possible errors in this experiment would be due to technical and implementation errors, but we did not encounter such a situation. Because the results we obtained on truthtable, waveform and FPGA matched each other exactly.

Appendix:

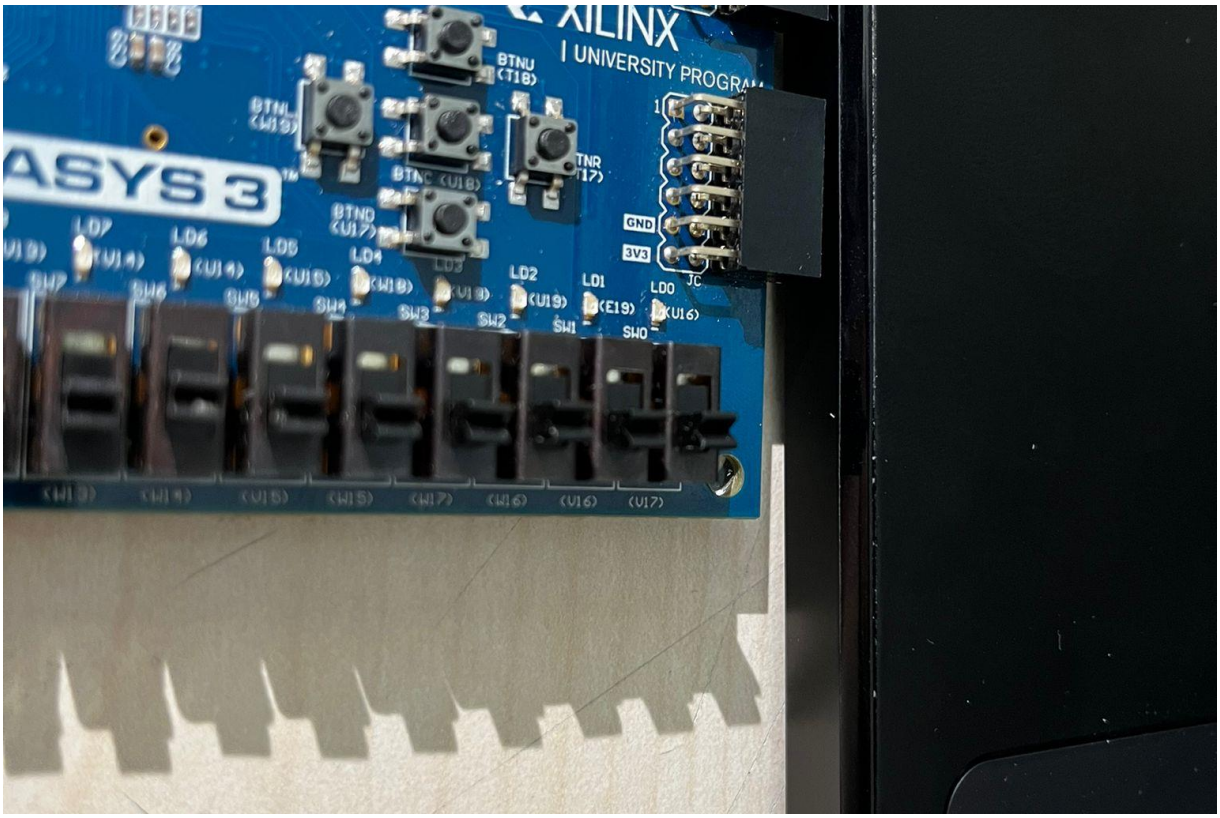

Figure 2.1 Example of Working FPGA

Figure 2.2 Example of Working FPGA

Figure 2.3 Example of Working FPGA

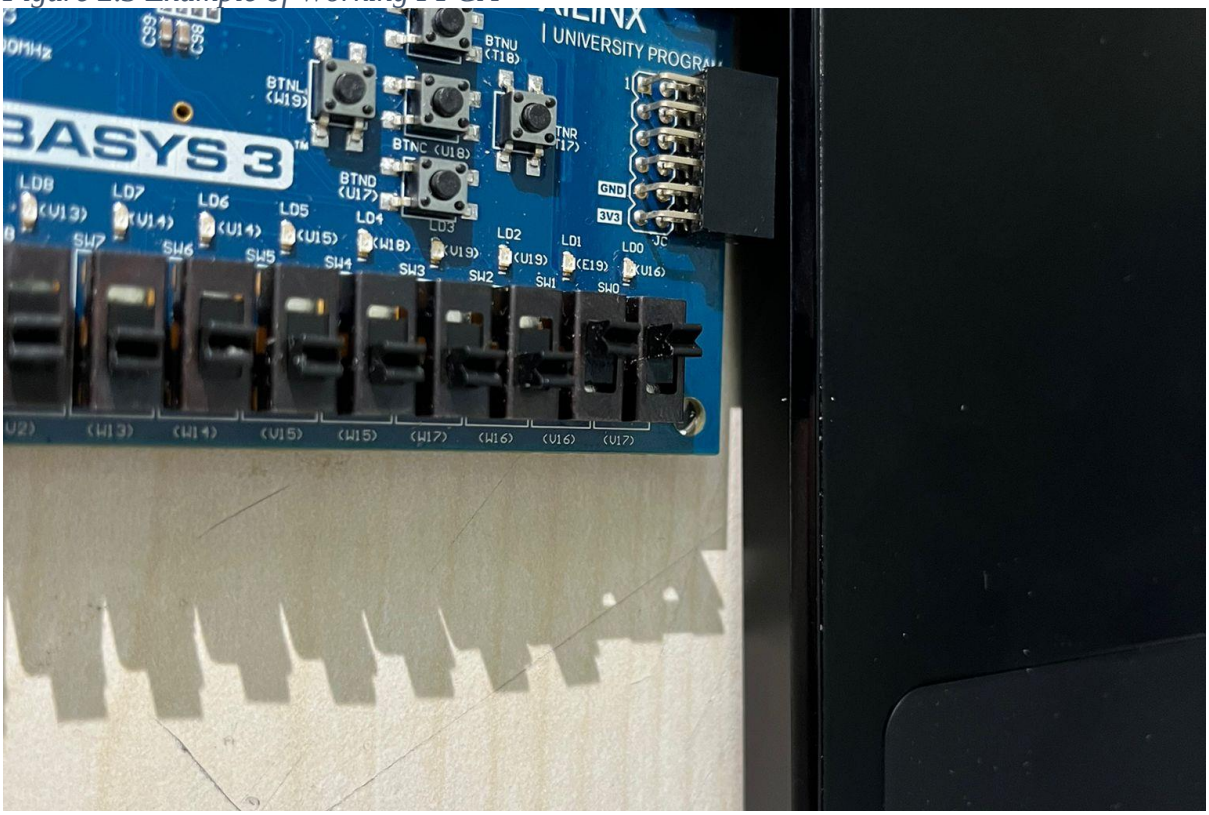

Figure 2.4 Example of Working FPGA

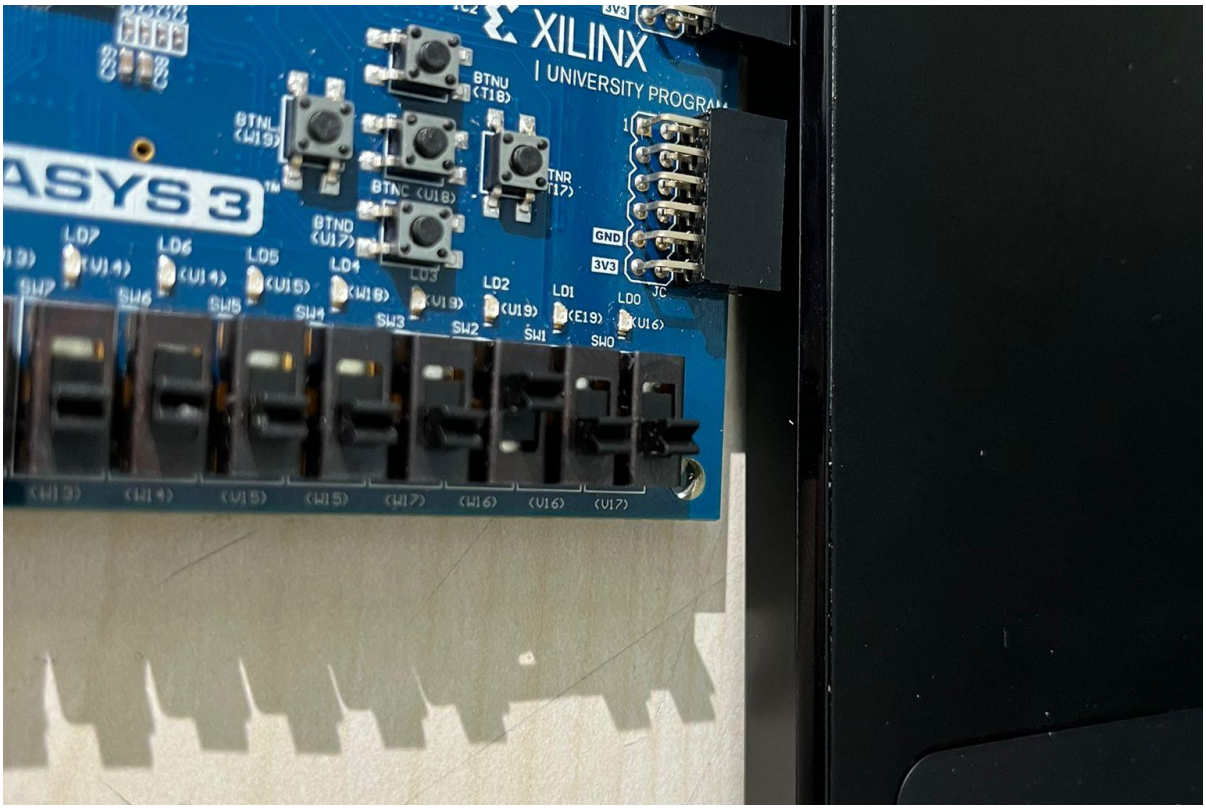

Figure 2.5 Example of Working FPGA

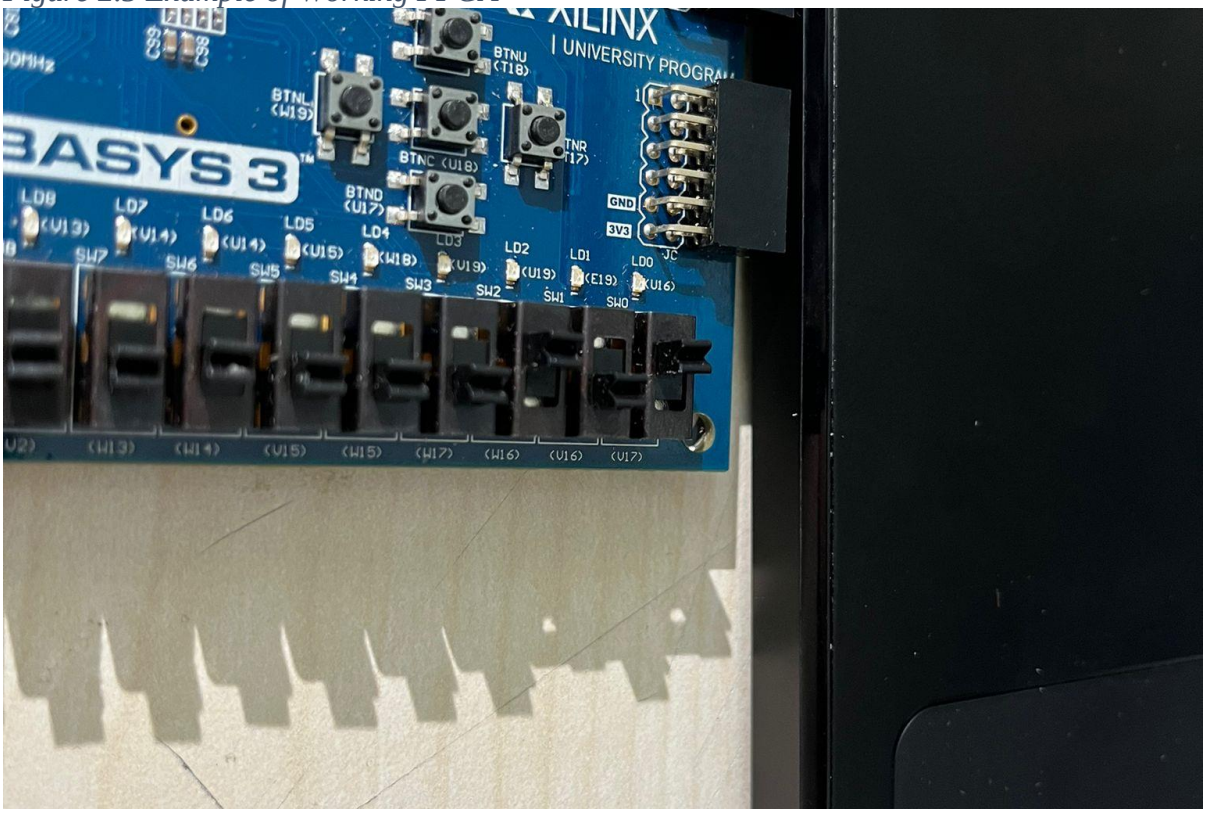

Figure 2.6 Example of Working FPGA

Figure 2.7 Example of Working FPGA

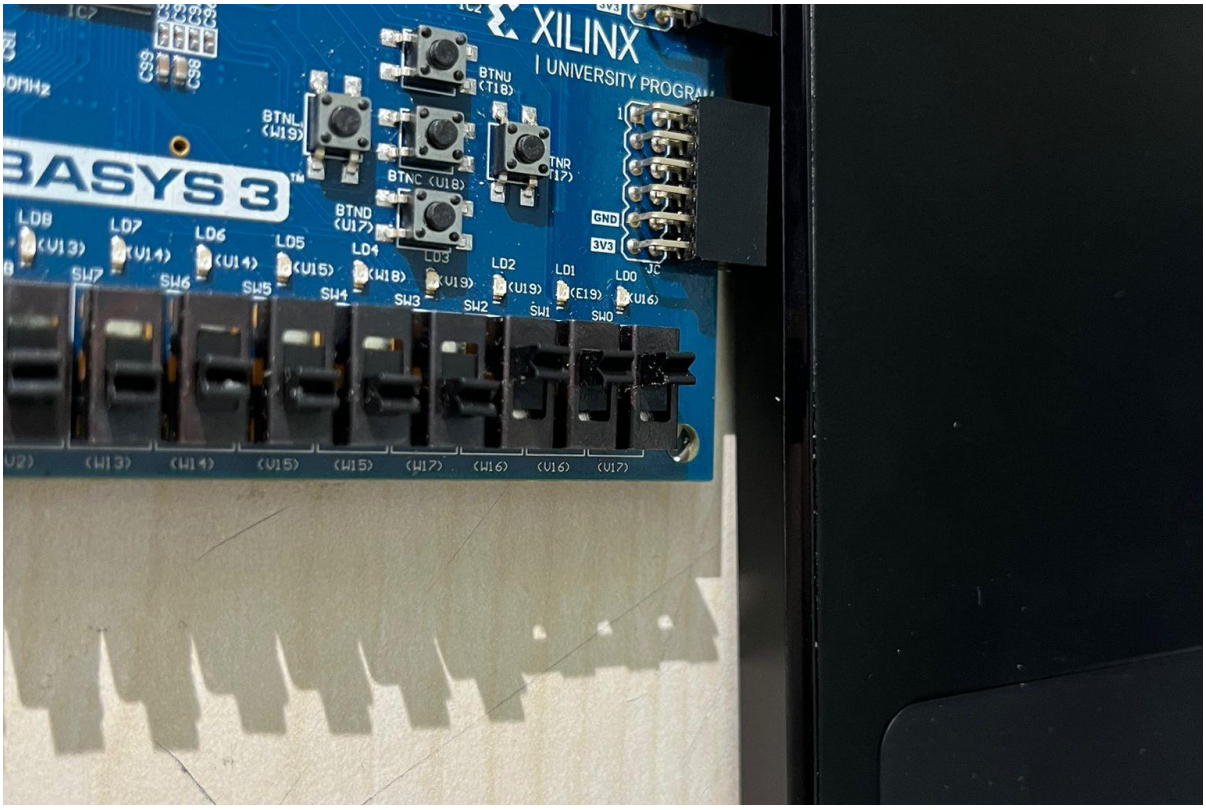

Figure 2.8 Example of Working FPGA

Code-1 Implantation of Designed Circuit

```
entity lab2 is
    Port ( X : in STD_LOGIC;
           Y : in STD_LOGIC;
           Z : in STD_LOGIC;
           F : out STD_LOGIC);
end lab2;

architecture emre of lab2 is

begin
F <= (X xor Y) and not Z;

end emre;
```

Code-2 Testbench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity test_lab2 is
end test_lab2;

architecture emre of test_lab2 is
    COMPONENT lab2
        PORT( X : IN STD_LOGIC;
              Y : IN STD_LOGIC;
              Z : IN STD_LOGIC;
              F : OUT STD_LOGIC);
    END COMPONENT;
    SIGNAL X : STD_LOGIC;
    SIGNAL Y : STD_LOGIC;
    SIGNAL Z : STD_LOGIC;
    SIGNAL F : STD_LOGIC;
begin
    UUT: lab2 PORT MAP(
           X => X,
           Y => Y,
           Z => Z,
           F => F
        );
    test_lab2 : PROCESS
    BEGIN
```

```
    wait for 10 ns;
    X<='0';
    Y<='0';
    Z<='0';
    wait for 10 ns;
    X<='0';
    Y<='0';
    Z<='1';
    wait for 10 ns;
    X<='0';
    Y<='1';
    Z<='0';
    wait for 10 ns;
    X<='0';
    Y<='1';
    Z<='1';
    wait for 10 ns;
    X<='1';
    Y<='0';
    Z<='0';
    wait for 10 ns;
    X<='1';
    Y<='0';
    Z<='1';
    wait for 10 ns;
    X<='1';
    Y<='1';
    Z<='0';
    wait for 10 ns;
    X<='1';
    Y<='1';
    Z<='1';
  END PROCESS;


end emre;
```

Reference:

https://digilent.com/reference/programmable-logic/guides/vivado-xdc-file#:~:text=This%20file%20contains%20the%20constraints,File%20to%20a%20Vivado%20Project