EEE 102 Lab 2 Report

Questions to Answer:

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

- While designing the circuit via VHDL, the boolean variables are assigned as input or output in the “Entity Declaration” part. We can specify the variables as input or output.

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

- In order to use our codes in different modules we can simply assign the input and output signals from the first module in another.
- PORT MAPs shows which inputs are “mapped” onto the outputs.

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

- A constraint file decides which switches or LEDs etc. on the BASYS3 board will be equivalent to the values in our code. Without it the system would not know which switches and LEDs to assign to different inputs in our circuit.

4) What is the purpose of writing a testbench?

- A testbench helps us to simulate many different scenarios on our circuit that would take too much time or would be difficult to test by hand on a BASYS3. A testbench allows us to assign different values to our inputs and see how the output changes accordingly.

Experimental Work:

I designed a circuit consisting of two and-gates and a xor-gate visible in Figure 1.1. The circuit is Out1 = (X and Y) xor (Z and T). To implement this circuit in Vivado I used VHDL to both write the design code and the simulation testbench. I have provided my VHDL codes for both of these files at the end of this document. In order to simulate the outcome of the circuit in different scenarios with different truth values we used a simulation testbench. I cycled through every possible binary combination of truth values for ‘X’, ‘Y’, ‘Z’, ‘T’ and recorded the outcome for ‘Out1’. The waveform of ‘Out1’ from the simulation can be seen in Figure 1.2. Along with the truth table in Figure 1.3.

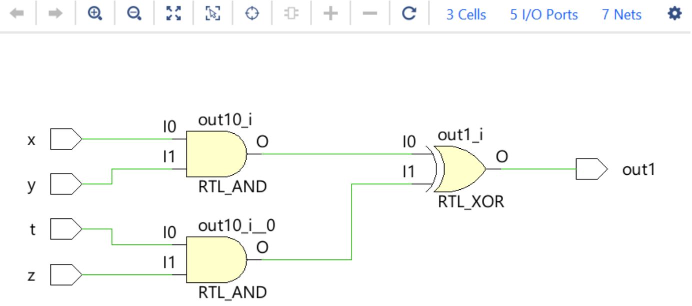


Figure 1.1: RTL Schematic

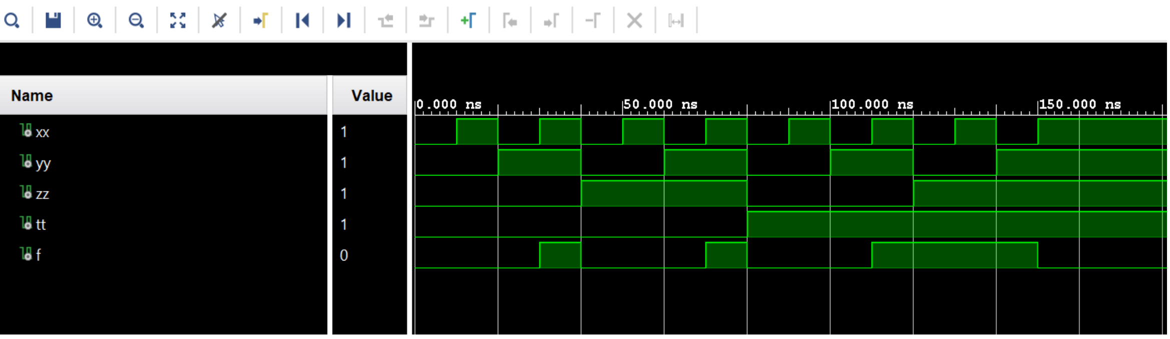
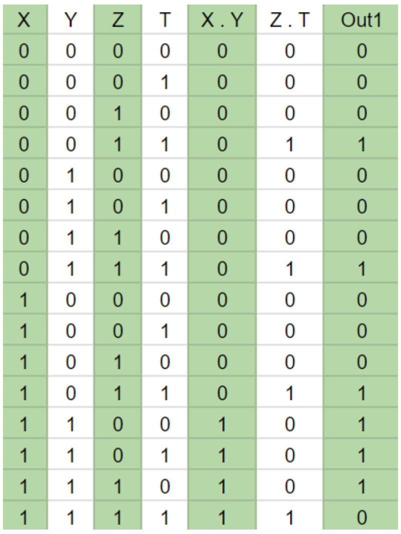


Figure 1.2: Simulation Waveform



X	Y	Z	T	X . Y	Z . T	Out1
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Figure 1.3: Truth Table

Demonstrating the circuit on BASYS3:

In order to run this project on an FPGA board we need a constraint file. I used the switches V17, V16, W16, W17 and they correspond to X, Y, Z, T respectively. I assigned the output led ‘Out1’ to the led U16. The code of my constraint file can again be found at the end of this document. I have provided some example scenarios I tested on my BASYS3.

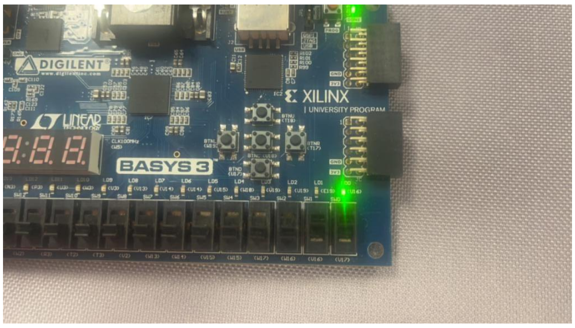


Figure 2.1: X=Y=1 , Z=T=0

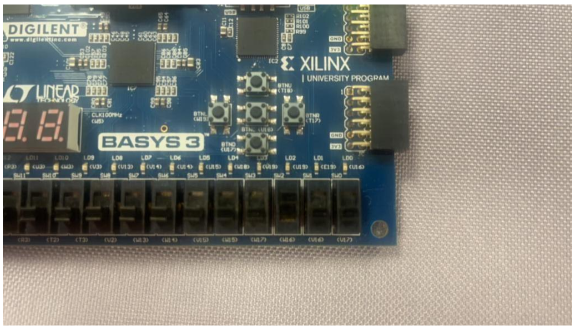


Figure 2.2: X=Y=T=0, Z=1

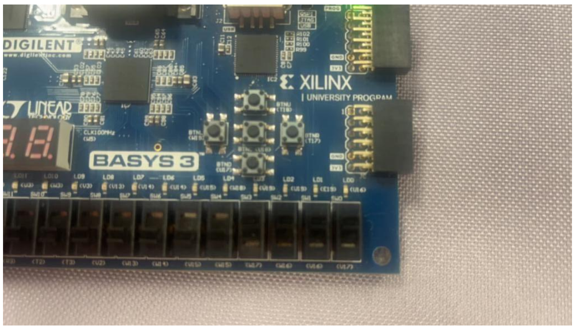


Figure 2.3: X=Y=Z=T=1

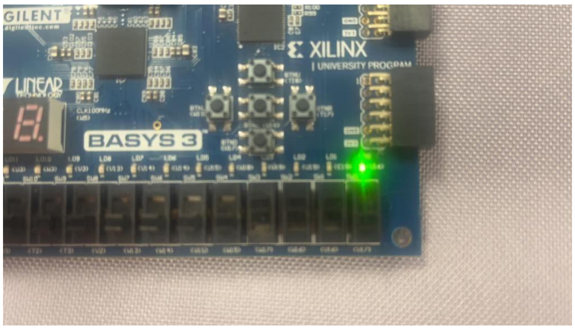


Figure 2.4: X=0, Y=Z=T=1

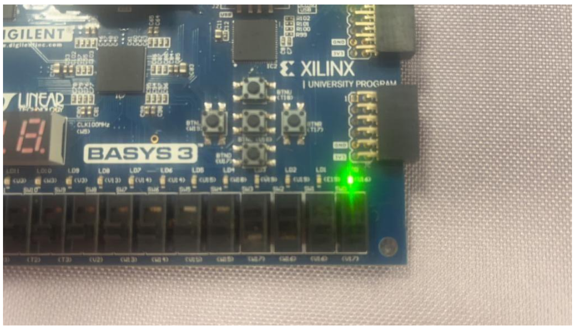


Figure 2.5: X=Y=0, Z=T=1

Conclusion:

By designing, developing, and implementing a circuit on a FPGA board (BASYS3 in this case) using Vivado Design Suite as a software and VHDL as the coding language I designed a circuit consisting of two and-gates and a xor-gate. I then simulated the circuits behaviour under different truth values using a simulation testbench again coded in VHDL. Comparing the results from the simulation and the truth table gave us consistent results. The circuit was

Out1 = (X and Y) xor (Z and T)

and the cases where Out1 = 1 were (1,1,1,0),(1,1,0,1),(1,0,1,1)(0,1,1,1),(1,1,0,0),(0,0,1,1).
And in the remaining 10 cases Out1 = 0.

Design Code:

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

entity and_gate is
   Port ( x : in STD_LOGIC;
        y : in STD_LOGIC;
        z : in STD_LOGIC;
        t : in STD_LOGIC;
        out1 : out STD_LOGIC);
end and_gate;

architecture Behavioral of and_gate is

begin
   out1 <= (x and y) xor (t and z);

end Behavioral;
```

Testbench Code:

```
library IEEE;
use IEEE.STD_LO
use IEEE.STD_LO
use IEEE.STD_LO

entity and_gate
end entity and_

architecture Be
  signal xx, yy
  signal f : st
begin
  U1: entity wo
    port map (
      x => xx,
      y => yy,
      z => zz,
      t => tt,
      out1 => f
    );

  process
  begin
    xx <= '0';
    yy <= '0';
    zz <= '0';
    tt <= '0';
    wait for 10
    xx <= '1';
    yy <= '0';
    zz <= '0';
    tt <= '0';
    wait for 10
    xx <= '0';
    yy <= '1';
    zz <= '0';
    tt <= '0';
    wait for 10
    xx <= '1';
    yy <= '1';
    zz <= '0';
    tt <= '0';
    wait for 10
```

```
xx <= '0';
yy <= '0';
zz <= '1';
tt <= '0';
wait for 10
xx <= '1';
yy <= '0';
zz <= '1';
tt <= '0';
wait for 10
xx <= '0';
yy <= '1';
zz <= '1';
tt <= '0';
wait for 10
xx <= '1';
yy <= '1';
zz <= '1';
tt <= '0';
wait for 10
xx <= '0';
yy <= '0';
zz <= '0';
tt <= '1';
wait for 10
xx <= '1';
yy <= '0';
zz <= '0';
tt <= '1';
wait for 10
xx <= '0';
yy <= '1';
zz <= '0';
tt <= '1';
wait for 10
xx <= '1';
yy <= '1';
zz <= '0';
tt <= '1';
wait for 10
xx <= '0';
yy <= '0';
zz <= '1';
tt <= '1';
wait for 10
```

```
    xx <= '1';
    yy <= '0';
    zz <= '1';
    tt <= '1';
    wait for 10
    xx <= '0';
    yy <= '1';
    zz <= '1';
    tt <= '1';
    wait for 10
    xx <= '1';
    yy <= '1';
    zz <= '1';
    tt <= '1';
    wait for 10
    wait;
  end process;
end Behavioral;
```

Constraint Code:

```
set_property PACKAGE_PIN V17 [get_ports {x}]
        set_property IOSTANDARD LVCMOS33 [get_ports {x}]
set_property PACKAGE_PIN V16 [get_ports {y}]
        set_property IOSTANDARD LVCMOS33 [get_ports {y}]
set_property PACKAGE_PIN W16 [get_ports {z}]
        set_property IOSTANDARD LVCMOS33 [get_ports {z}]
set_property PACKAGE_PIN W17 [get_ports {t}]
        set_property IOSTANDARD LVCMOS33 [get_ports {t}]
set_property PACKAGE_PIN U16 [get_ports {out1}]
        set_property IOSTANDARD LVCMOS33 [get_ports {out1}]
```