Muhammed Enes İnanç

22001799

05.10.2022

Purpose:

The purpose of this lab work is to see the logic gates and circuit design we learned in the lesson in practice. In addition, it is to teach the Vivado program to code VHDL, which is very important for electrical and electronic engineering, and the use of the Basys3 card.

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

In VHDL, inputs and outputs are encoded in a section defined as the entity section. Entity section defines the interface and specifies the ports to be used in the circuit and the port statements. However, although the entity section does all this work, it does not give information on how to implement the circuit.

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

Design entities are connected to each other via signals. Signals also play a role in transmitting changing values. The port map communicates the inputs with the architecture of the written code. It also plays a leading role in transporting them between modules as much as desired.

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

There are many LEDs and switches on the FPGA. Thanks to the codes written in the Constraint file, we can determine which of these buttons or leds will be used or in which order. In this way, it both completes the circuit task without any confusion and makes it easier for the user to use.

What is the purpose of writing a testbench?

After simplifying and functionalizing the circuit we designed, it applies the numbers in the binary system for each input and creates a simulation. This simulation illustrates what the inputs and outputs are at what time intervals of the circuit. At this stage, we can understand if there is any error in the circuit.

My Combinational Circuit (Fully Working Computer):

Today I wanted to integrate one of the problems I encounter most in real life into my circuit. I have determined 4 basic factors to run a desktop computer in a healthy way and get an image. These factors are computer(in1), monitor(in2), electricity(in3) and electric generator(in4). I have prepared a truth table that several of these factors must be together in order to fully operate a computer. According to this table, I can reach the desired result in only 3 cases out of 16 cases.

In1(computer)	In2 (monitor)	In3 (electricity)	In4 (generator)	Out1
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	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

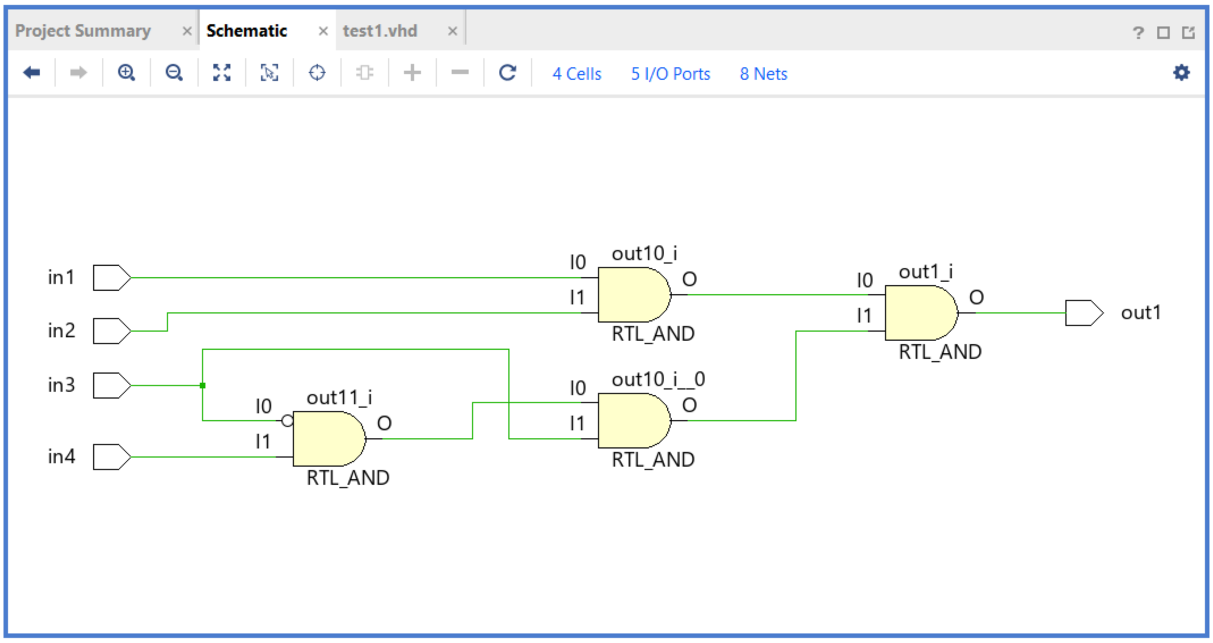
As you can see in the Truth table, the computer is fully functional in only 3 cases. I used the SOP method to create a generic function for these states. As I wrote below, I collected the m14, m15 and m16 values and made some simplifications.

*F = (in1 * in2* not in3* in4) + (in1*in2*in3*not in4) + (in1*in2*in3*in4)*

*= (in1 * in2) * ((not in3* in4) + (in3* (not in4 + in4))*

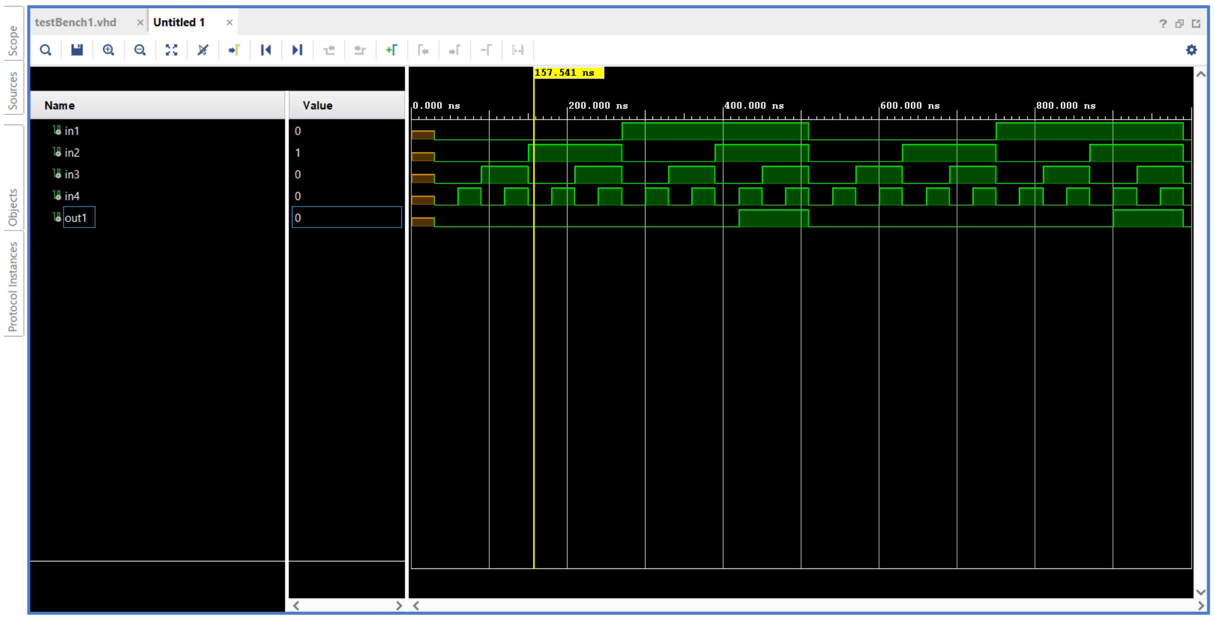
= (in1*in2) * ((not in3* in4)+ in3)

Afterwards, I wrote this function I found in the code section and obtained a schematic through a vivado. This schematic is an illustration made thanks to vivado.



Schematic

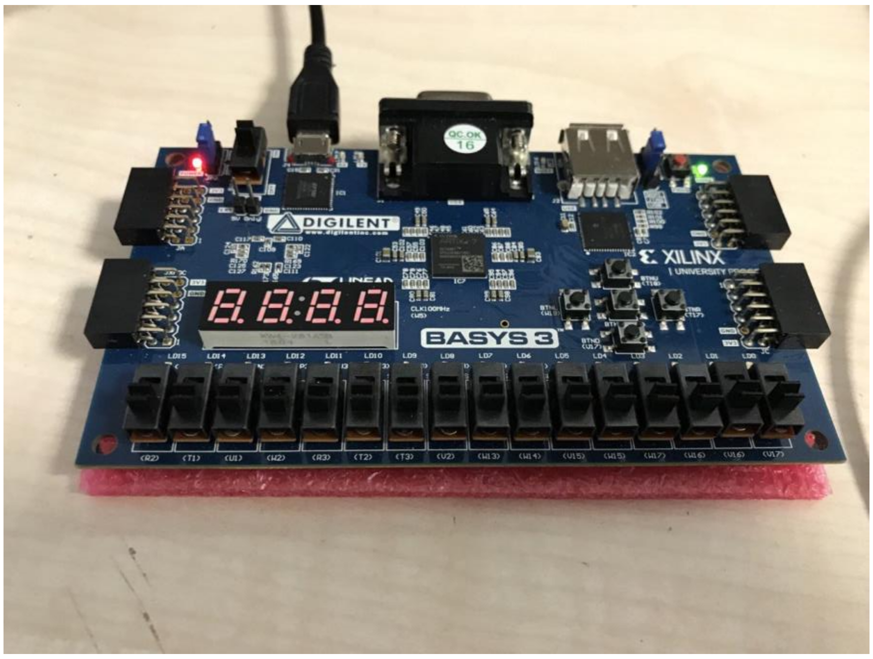
After this section, I created a simulation to try whether the circuit works correctly at different input values, thanks to the feature of vivado.



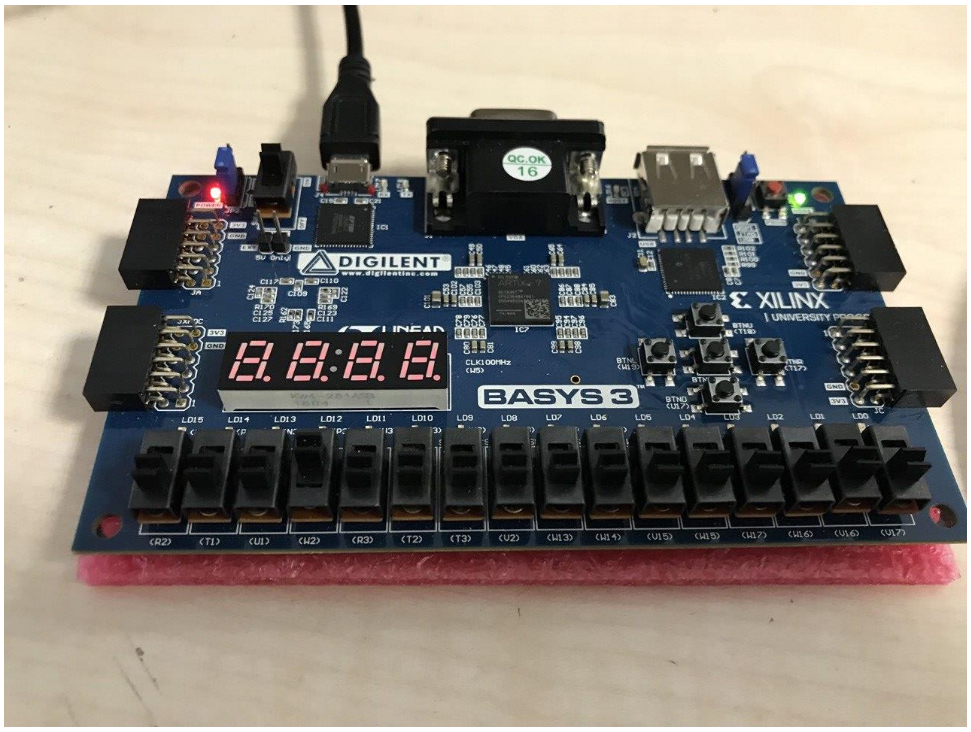
Name	Value
in1	0
in2	1
in3	0
in4	0
out1	0

Simulation

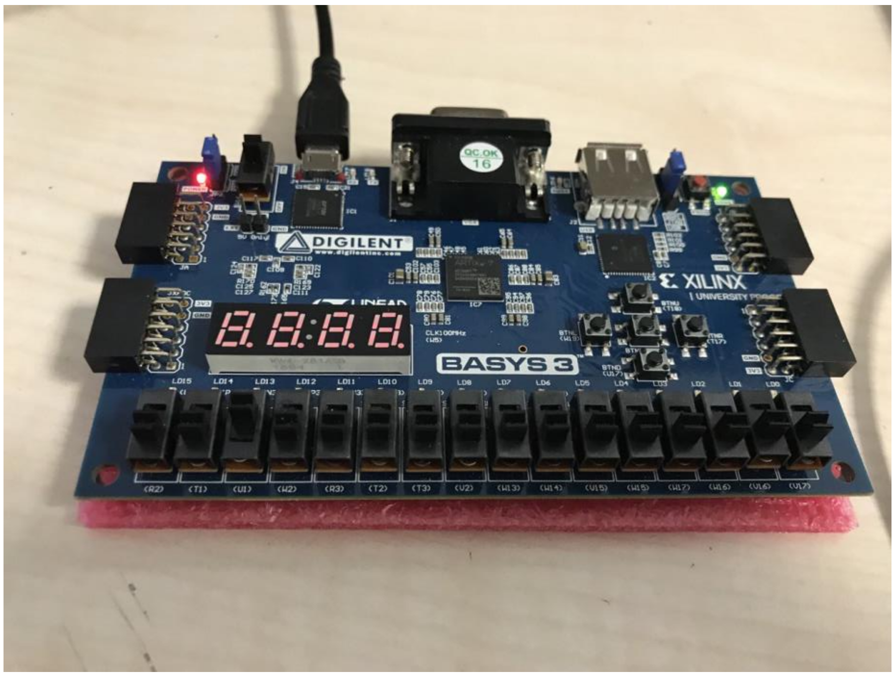
After making sure that everything is working correctly, I created a bitstream and integrated it into my Basys3 board. Thanks to the buttons that I can change the values of the inputs, I got different outputs. And when these outputs are 1, the led light on the card has been successfully lit.



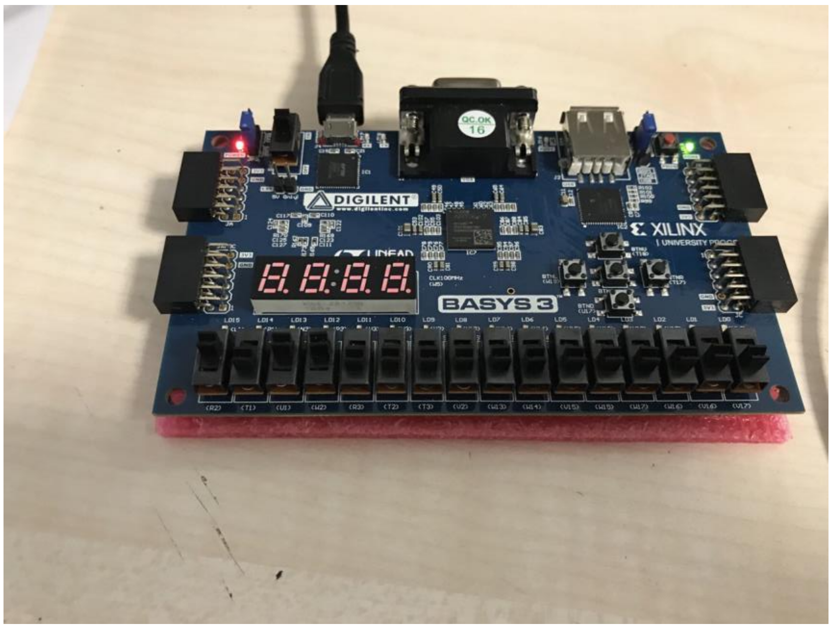
Inputs: 0000   Output:0



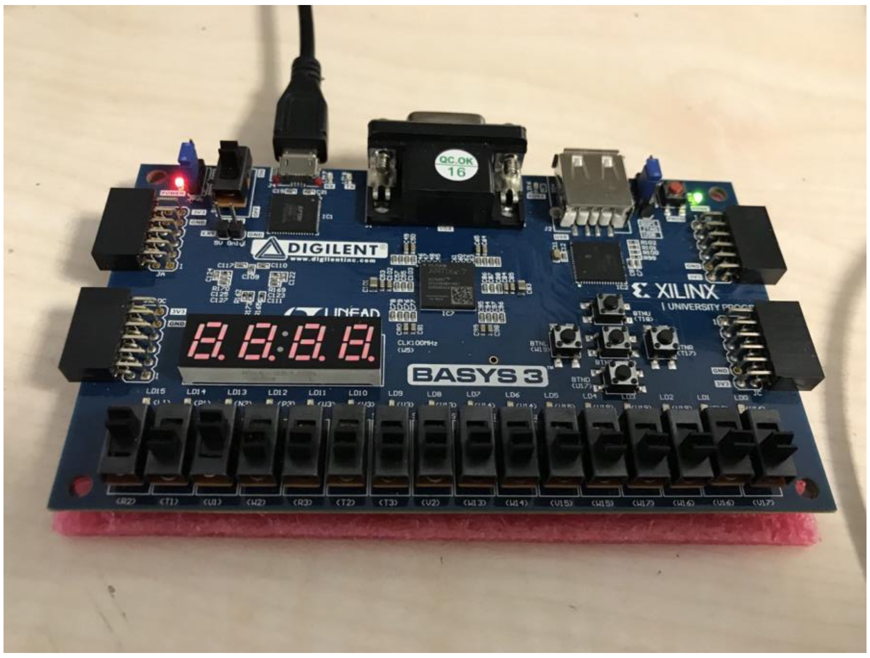
Inputs: 0001   Output:0



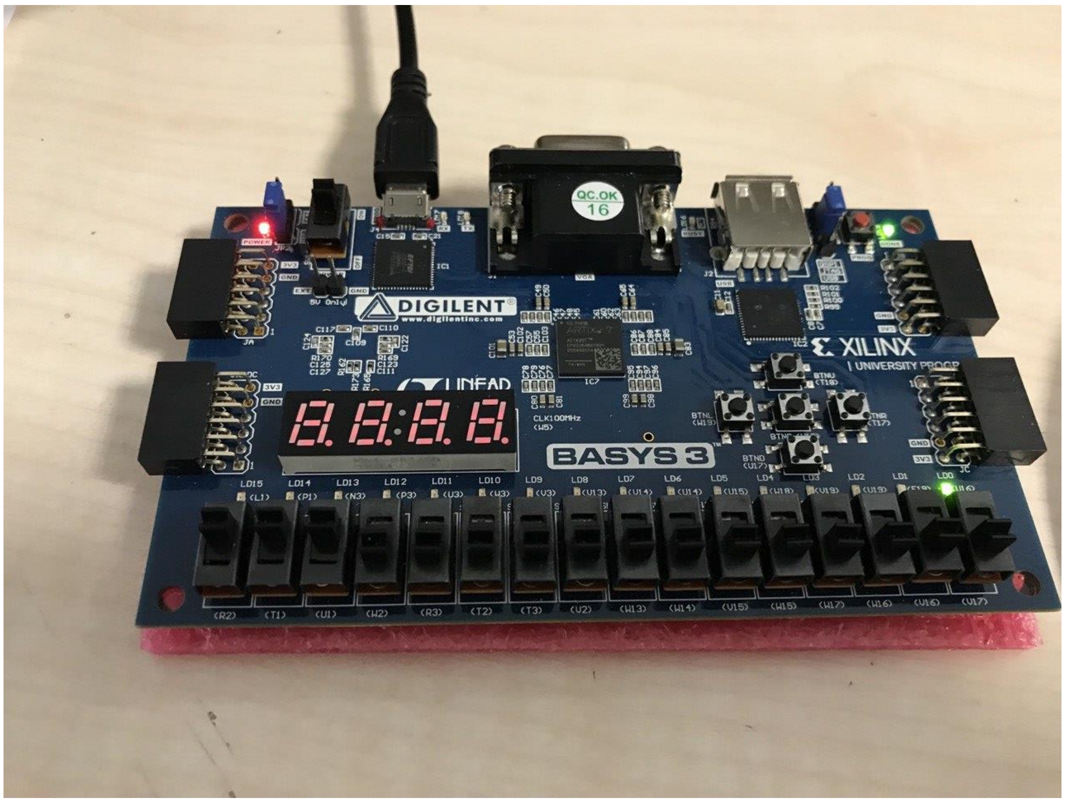
Inputs: 0010   Output:0



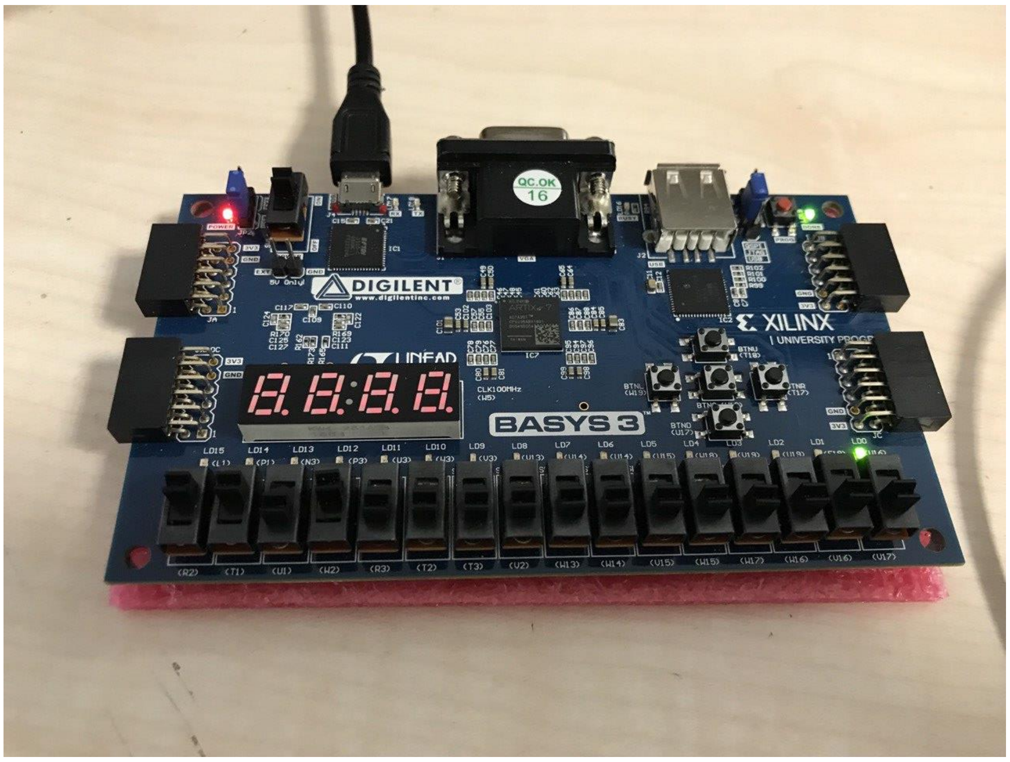
Inputs: 1011   Output:0



Inputs: 1010   Output: 0



Inputs: 1110   Output: 1


Inputs: 1101   Output: 1

Conclusion:

The logic gates we learned in the lesson were used in circuits. Also learned how to use Basys3 card. Learned how to write code, create simulations, create schematics, and most importantly, establish a connection between basys3 card and vivado in Vivado. What VHDL is is well understood.

Design Source Code:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values
```

```
--use IEEE.NUMERIC_STD.ALL;


-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;


entity test1 is

   Port ( in1 : in STD_LOGIC;

          in2 : in STD_LOGIC;

          in3 : in STD_LOGIC;

          in4 : in STD_LOGIC;

          out1 : out STD_LOGIC);

end test1;


architecture Behavioral of test1 is


begin

out1 <= (in1 and in2) and (((not in3) and in4) or in3);


end Behavioral;
```

Testbench Codes:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
```

```
-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;


-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;


entity testBench1 is

end testBench1;

architecture Behavioral of testBench1 is

COMPONENT test1

PORT( in1 : IN STD_LOGIC;

in2 : IN STD_LOGIC;

in3 : IN STD_LOGIC;

in4 : IN STD_LOGIC;

out1 : OUT STD_LOGIC);

END COMPONENT;

SIGNAL in1 : STD_LOGIC;

SIGNAL in2 : STD_LOGIC;

SIGNAL in3 : STD_LOGIC;

SIGNAL in4 : STD_LOGIC;

SIGNAL out1 : STD_LOGIC;

BEGIN
```

```
UUT: test1 PORT MAP(
        in1 => in1,
        in2 => in2,
        in3 => in3,
        in4 => in4,
        out1 => out1
);
testBench1 : PROCESS
BEGIN
wait for 30 ns;
in1<='0';
in2<='0';
in3<='0';
in4<='0';
wait for 30 ns;
in1<='0';
in2<='0';
in3<='0';
in4<='1';
wait for 30 ns;
in1<='0';
in2<='0';
in3<='1';
in4<='0';
wait for 30 ns;
in1<='0';
```

```
in2<='0';

in3<='1';

in4<='1';

wait for 30 ns;

in1<='0';

in2<='1';

in3<='0';

in4<='0';

wait for 30 ns;

in1<='0';

in2<='1';

in3<='0';

in4<='1';

wait for 30 ns;

in1<='0';

in2<='1';

in3<='1';

in4<='0';

wait for 30 ns;

in1<='0';

in2<='1';

in3<='1';

in4<='1';

wait for 30 ns;

in1<='1';

in2<='0';
```

```
in3<='0';

in4<='0';

wait for 30 ns;

in1<='1';

in2<='0';

in3<='0';

in4<='1';

wait for 30 ns;

in1<='1';

in2<='0';

in3<='1';

in4<='0';

wait for 30 ns;

in1<='1';

in2<='0';

in3<='1';

in4<='1';

wait for 30 ns;

in1<='1';

in2<='1';

in3<='0';

in4<='0';

wait for 30 ns;

in1<='1';

in2<='1';

in3<='0';
```

```
in4<='1';
wait for 30 ns;
in1<='1';
in2<='1';
in3<='1';
in4<='0';
wait for 30 ns;
in1<='1';
in2<='1';
in3<='1';
in4<='1';
END PROCESS;

end Behavioral;
```

Constraint File Codes:

```
set_property PACKAGE_PIN W2 [get_ports {in4}]
   set_property IOSTANDARD LVCMOS33 [get_ports {in4}]
set_property PACKAGE_PIN U1 [get_ports {in3}]
   set_property IOSTANDARD LVCMOS33 [get_ports {in3}]
set_property PACKAGE_PIN T1 [get_ports {in2}]
   set_property IOSTANDARD LVCMOS33 [get_ports {in2}]
set_property PACKAGE_PIN R2 [get_ports {in1}]
   set_property IOSTANDARD LVCMOS33 [get_ports {in1}]
set_property PACKAGE_PIN U16 [get_ports {out1}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1}]
```