# EEE102-02 Lab 2 Report:

# Introduction to VHDL

Kaan Ermertcan - 22202823

02.10.2023

## Purpose:

The purpose of this lab is to introduce us to Vivado environment and to teach us how to implement a combinatorial logic circuit on a BASYS3 using VHDL. We also put into use what we learned in the lectures about combinational logic.

## Design Specifications:

I wanted my design to represent a real-life situation. So, in my design, I decided to represent how the two elements of the house, windows and heating, change according to three natural conditions: Season, weather and time of the day. To digitalize and simplify these variables, I assumed their digital representations and assigned letters to them as written on the table below:

| | **0** | **1** |
|---|---|---|
| Time of the Day (A) | It is daytime. | It is nighttime. |
| Season (B) | It is not winter. | It is winter. |
| Weather (C) | It is not windy. | It is windy. |
| Windows (F) | Windows are closed. | Windows are open. |
| Heating (G) | Heating is turned off. | Heating is turned on. |

*Table 1.1 Digital representations of variables.*

Then, I wrote a truth table of outcomes depending on all possible conditions, with my home in mind. This truth table is what my design is supposed to achieve.

| **Conditions** | | | **Outcomes** | |
|---|---|---|---|---|
| Nighttime (A) | Winter (B) | Windy (C) | Window (F) | Heating (G) |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 |

*Table 1.2 Truth table of my design.*

I chose the leftmost switches and LEDs on the BASYS3 to be my inputs and outputs as in the table below:

| **Inputs** | | | **Outputs** | |
|---|---|---|---|---|
| Nighttime (A) | Winter (B) | Windy (C) | Window (F) | Heating (G) |
| R2 | T1 | U1 | L1 | P1 |

*Table 1.3 Pin codes of the switches and LEDs that will be used.*

# Methodology:

First, I found the SOP (Sum of Products) expression of the truth table:

$$F \equiv (A'B'C') + (A'B'C) + (AB'C')$$

$$G \equiv (A'BC) + (ABC') + (ABC)$$

Then, I simplified these expressions using distributive, complement, and identity properties:

$$F \equiv A'B'(C' + C) + (AB'C') \equiv A'B' + AB'C' \equiv B'(A' + AC')$$

$$G \equiv AB(C' + C) + (A'BC) \equiv AB + A'BC \equiv B(A + A'C)$$

Now, I have an expression that I can implement in VHDL.

After creating a new project in Vivado, I created a design source where I can write my VHDL code that describes FPGA's behavior. At the top of my code (Code 1), I had to define other modules that needed to be imported. In this case, std_logic_1164 package is imported from ieee library. This package includes standard definitions for digital logic values. Using the same syntax, other modules and libraries can also be imported if needed. Then, I had to name my inputs and outputs that will be used in my design in the entity definition. Inputs and outputs are written in my code inside Port(); where each line defines single input and output using this syntax: [name] : in/out STD_LOGIC; . I named my inputs and outputs each with a word as in Table 1.2. After the colon I wrote in, signifying it is an input; or out, signifying it is an output. Lastly, STD_LOGIC is a logic type defined in the imported module. Finally, at the bottom of the code inside the architecture definition, I can define the combinational logic relation between inputs and outputs.

After writing VHDL code for my design, I wrote a testbench code (Code 2) to simulate my design. Testbenches can be used for testing and debugging our designs. It is especially powerful because you can generate any input you want to test with and test any part of your design. After I include the necessary packages in my code as in code 1, I wrote an empty entity definition. It is empty because a testbench has no real inputs or outputs. Then, we define the architecture of the testbench. First, we create a component that we want to test in the testbench and declare its inputs and outputs. Components can also be used for creating a hierarchy or reusing a piece of design in another code. Then, we declare the internal signals of the testbench. After that, we instantiate our component and map its ports to the signals of the testbench using PORT MAP statement. Finally, we can write our test process. Here we set our input signals to be whatever we want and can change them over time. Now, we are finished writing the testbench code.

Now we should map inputs/outputs in the design to physical switches and LEDs (or other input and output features if needed) on the Basys3 by writing a constraints file. In a constraints file, we map a pin of the BASYS3 to an input/output and define the electrical standard to be used for each input/output. Pin codes are printed on the PCB next to each physical component.

## Results:

After I wrote the VHDL code for my design, I ran synthesis, and I could see my FPGA design and RTL schematic.

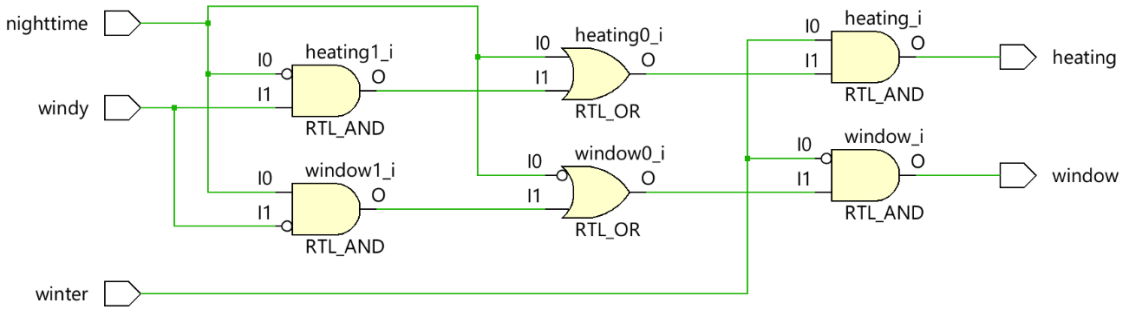


*Figure 2.1 RTL Schematic.*

Here, I can see that 4 and gates and 2 or gates with 4 inverters will be generated on the FPGA.

Now I will test it with a testbench. I wrote the testbench to cycle through every combination and change to the next one every 50 ns. I ran the simulation and saw how the outputs change depending on different inputs in a graph.

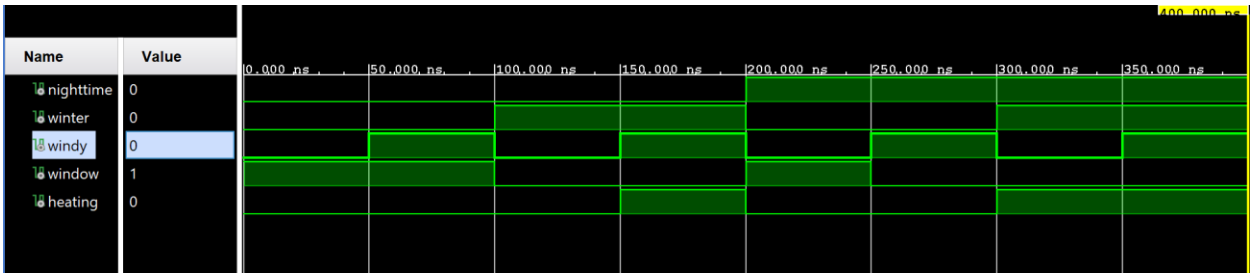


*Figure 2.2 Simulation Results.*

The simulation results match our truth table. So, I continued implementing our design by writing a constraints file. After successfully completing constraints, I generated a bitstream file to be uploaded to and run on the BASYS3. Then, from the Hardware Manager in Vivado, I connected to my BASYS3 and successfully ran the bitstream on it. I tested all possible switch combinations and found that they matched with what is on my truth table as can be seen from the pictures:

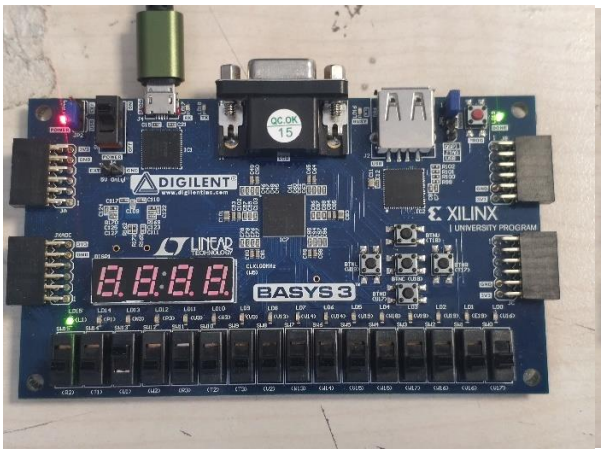


*Figure 3.1 Results: Case 001*

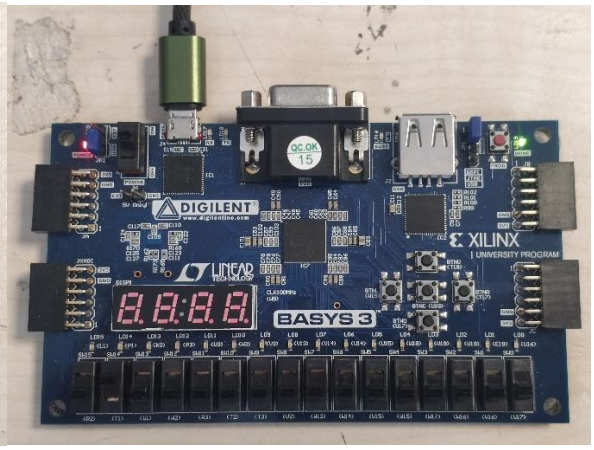


*Figure 3.2 Results: Case 010*

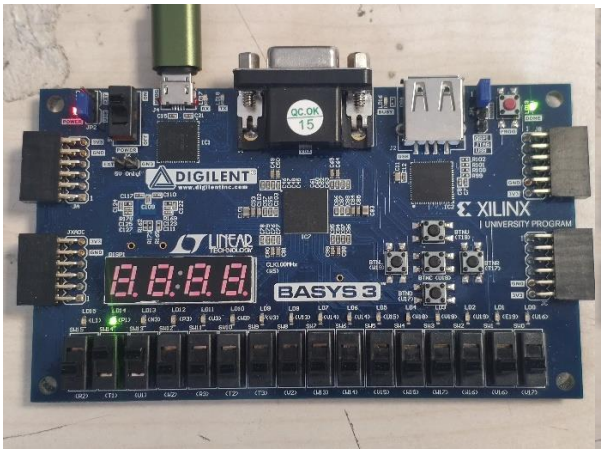


*Figure 3.3 Results: Case 011*

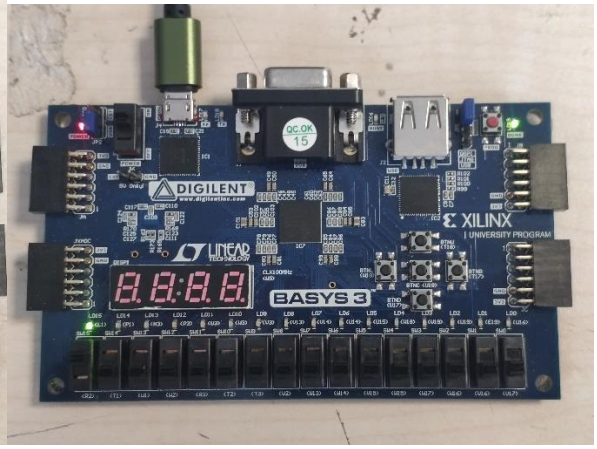


*Figure 3.4 Results: Case 100*

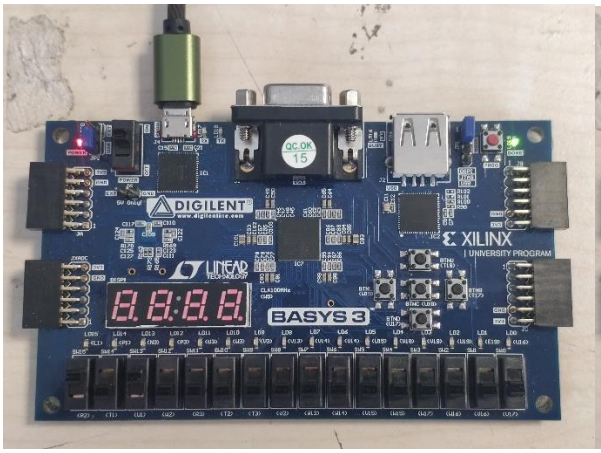


*Figure 3.5 Results: Case 101*

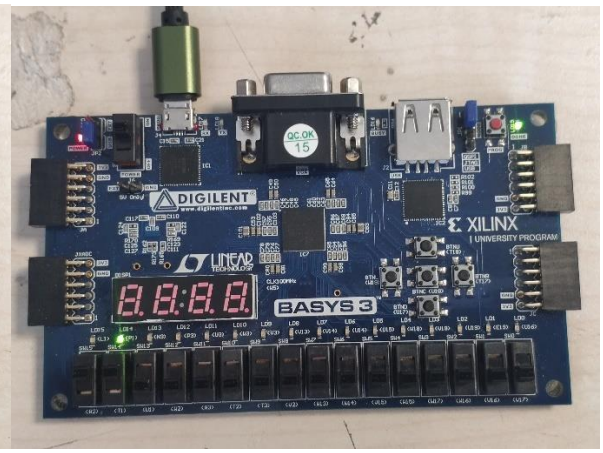


*Figure 3.6 Results: Case 110*

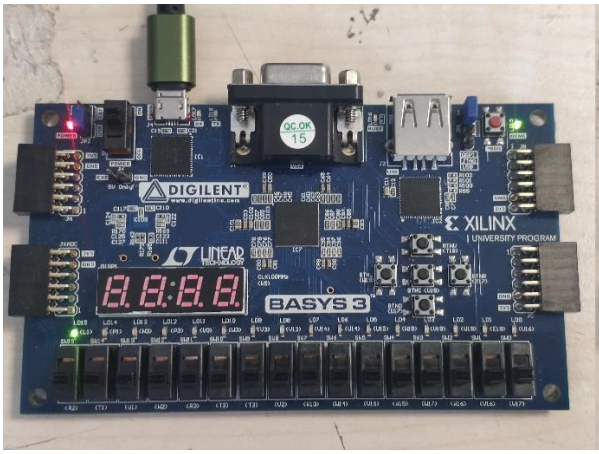


*Figure 3.7 Results: Case 000*

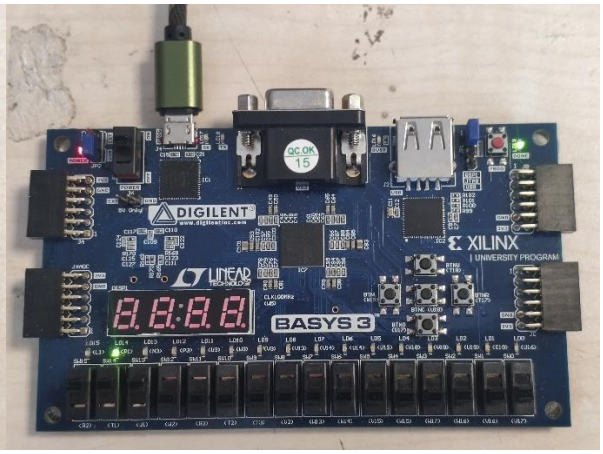


*Figure 3.8 Results: Case 111*

## Conclusion:

In this lab, I learned how to design a basic combinational logic circuit, simulate it with a testbench and implement it on my BASYS3 board. I also had a chance to represent a real-life situation in a truth table and find an expression for it using what we learned in the lectures. I also did some research to understand what each line of code does in my VHDL scripts and gained more insight about VHDL. My implementation was successful as both my simulation and BASYS3 results were the same as what my design ought to achieve.

# Appendices:

**Code 1: main.vhd**

```
----------------------------------------------------------------------------------
-- EEE102 Lab 2
-- Kaan Ermertcan
-- Create Date: 09/30/2023 12:40:10 PM
-- Module Name: main - Behavioral
----------------------------------------------------------------------------------

-- Definition of the modules that will be used
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Specifying inputs and outputs
entity main is
   Port ( nighttime : in STD_LOGIC;
          winter : in STD_LOGIC;
          windy : in STD_LOGIC;
          window : out STD_LOGIC;
          heating : out STD_LOGIC);
end main;

architecture Behavioral of main is
begin
-- Defining the relation between inputs and outputs.
window <= not winter and (not nighttime or (nighttime and not windy));
heating <= winter and (nighttime or (not nighttime and windy));
end Behavioral;
```

---

**Code 2: testbench.vhd**

```
----------------------------------------------------------------------------------
-- EEE102 Lab 2
-- Kaan Ermertcan
-- Create Date: 09/30/2023 01:00:42 PM
-- Module Name: testbench - Behavioral
----------------------------------------------------------------------------------

-- Definition of the modules that will be used
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity testbench is
end testbench;
```

```
-- Specifying inputs and outputs of the component
architecture Behavioral of testbench is
component main
 PORT(
 nighttime: in STD_LOGIC;
 winter: in STD_LOGIC;
 windy: in STD_LOGIC;
 window: out STD_LOGIC;
 heating: out STD_LOGIC);
end component;

-- Defining signals of the testbench
 signal nighttime: STD_LOGIC;
 signal winter: STD_LOGIC;
 signal windy: STD_LOGIC;
 signal window: STD_LOGIC;
 signal heating: STD_LOGIC;
begin

-- Mapping ports of the component to ports of main
UUT: main PORT MAP(
nighttime => nighttime,
winter => winter,
windy => windy,
window => window,
heating => heating
);

-- Defining the process of the simulation
testbench: PROCESS
 begin
 -- 000
 nighttime<='0';
 winter<='0';
 windy<='0';
 -- 001
 wait for 50 ns;
 nighttime<='0';
 winter<='0';
 windy<='1';
 -- 010
 wait for 50 ns;
 nighttime<='0';
 winter<='1';
 windy<='0';
 -- 011
 wait for 50 ns;
```

```
 nighttime<='0';
 winter<='1';
 windy<='1';
 -- 100
 wait for 50 ns;
 nighttime<='1';
 winter<='0';
 windy<='0';
 -- 101
 wait for 50 ns;
 nighttime<='1';
 winter<='0';
 windy<='1';
 -- 110
 wait for 50 ns;
 nighttime<='1';
 winter<='1';
 windy<='0';
 -- 111
 wait for 50 ns;
 nighttime<='1';
 winter<='1';
 windy<='1';
 wait for 50ns;
end PROCESS;
end Behavioral;
```