Lab 2: Introduction to VHDL

Date: 02.10.2023

Name: Şevval ERBAY

Section: 2

Purpose of the experiment:

The purpose of this experiment is to design and implement a combinational circuit on our BASYS 3 using VHDL, thus learning the basics of FPGA programming.

Methodology:

For this lab, we were asked to design a combinatorial circuit for a real-life problem. For my problem, I chose to check whether a house cat was fed or not. For the circuit I used AND-gates and OR-gates. In Vivado, I set my variables and reconstructed my circuit with behavioral commands. To test the circuit and to be able to simulate the waveform, we wrote a test bench code and simulated every possible combination to show that our code acted parallel to the initial truth table. Test benches do not have any real input or output signals and they cannot work in real life but are very beneficial in simulating the main code based on the given parameters/combinations. The test bench code does not have anything between "entity" and "end", this means that this code cannot connect our set variables to proper inputs and outputs on the simulation. This is where PORT MAP function becomes beneficial. PORT MAP helps us to state which signals the module's inputs and outputs should be connected to in our code. Finally, we uploaded our codes (our constraint files) to our BASYS3 boards and used them to demonstrate our combinatorial circuits. Vivado supports the Xilinx design constraint (XDC) file format in order to restrict time and physical constraints like switches and LEDs. In constraint files, we specify ports according to our top module. We checked which pin corresponds to which switch/LED and specified those pin coordinates in our constraint files. By doing this, we prevented any possible mismatch between our top modules and our BASYS3 boards.

For my problem, I considered whether a cat was fed that morning or not. There are 4 people in a house, mom, dad, son and daughter. Let us call them Mom, Dad, John and Jane respectively. Mom and Dad feed the cat in the morning and John and Jane feed the cat at the evening. The cat has to be fed by at least one of 2 people each morning and evening otherwise it will be hungry. In order to create a circuit for this problem I used 2 OR-gates and 1 AND-gate and needed 4 inputs and 1 output. Names of the inputs and the output are:

- **mom**: Mom
- **dad**: Dad
- **john**: John
- **jane**: Jane
- **cat**: the feeding situation of the cat

The equation to check whether the cat was fed or not is:

$$(mom + dad) \cdot (john + jane) = cat$$

If cat = 1, this means that the cat is properly fed (at least once for morning and evening), and if cat = 0, this means that at least one of the meals the cat supposed to get has been skipped.

mom	dad	john	jane	mom + dad	john + jane	cat
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	0	0
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Table.1 Truth table of the equation ***(mom + dad) . (john + jane) = cat***

Results:

We began by setting our variables as inputs and outputs. In order to assign the inputs and the output we used the Port function and listed those as elements of STD_LOGIC and indicated whether they were inputs or outputs by “in” and “out” statements. Then by using behavioral commands we implemented our equation ***(mom + dad) . (john + jane) = cat*** . These steps are written in “cat1.vhd” (Code.1). After writing our code “cat1.vhd”, we synthesized that code and Vivado created the RTL schematic of our circuit (Figure.1). As shown in this figure, inputs “**mom**” and “**dad**” go into an OR-gate, “**john**” and “**jane**” go into another OR-gate and then the outputs of those OR-gates go into an AND-gate, all in all giving us the output “**cat**”.

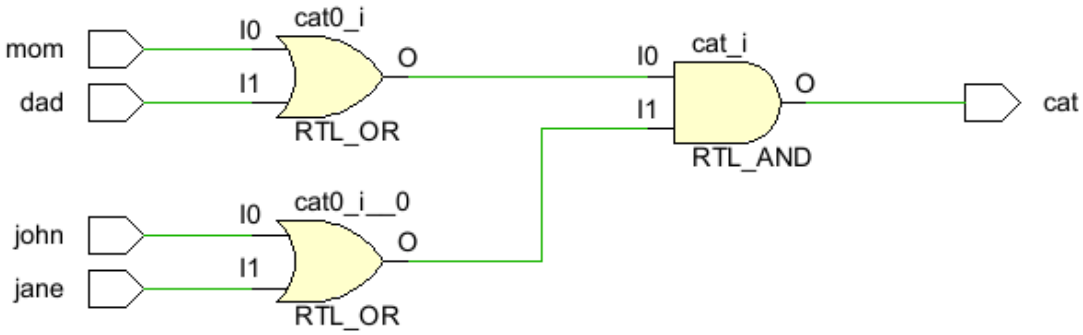


Figure.1 RTL schematic generated by Vivado

Then we were asked to simulate our codes and generate the waveform by using Vivado. In order to do that, we wrote test benches and included all possible scenarios we found in our truth table (Table.1). My test bench code "testbench_cat.vhd" is written in Code.2. In order to fit every scenario in one place, I set the wait time as 10 ns. Then Vivado generated the corresponding waveform (Figure.2).

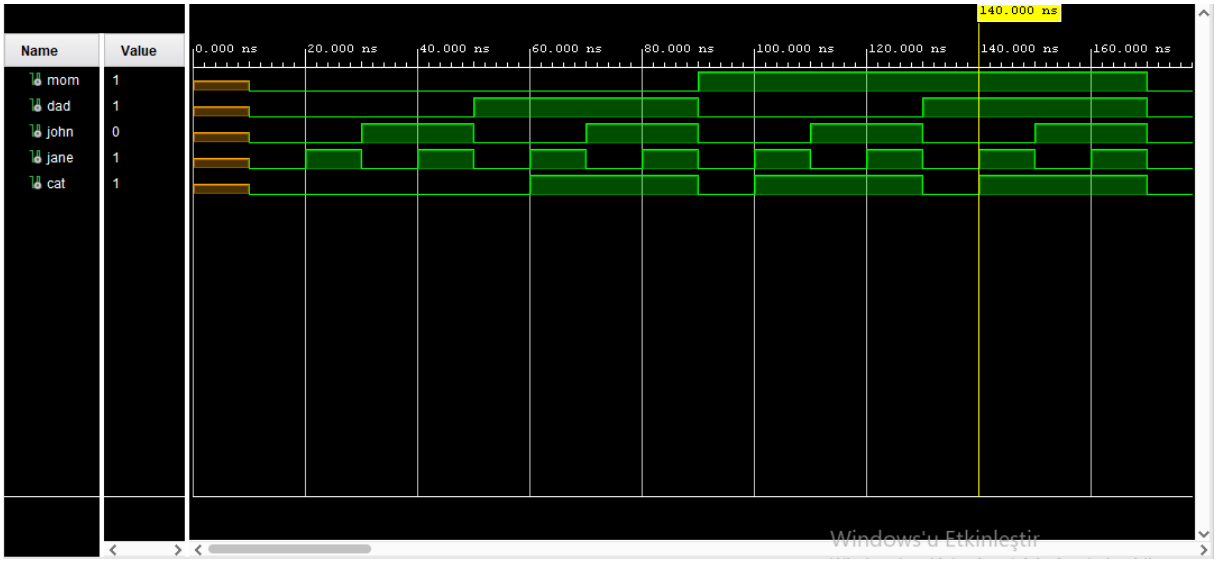


Figure.2 Waveform of my circuit generated by Vivado

By using a constraint file, we assigned switches to inputs and a LED for the output. The assigned switches and the LED are as follows:

mom : V17

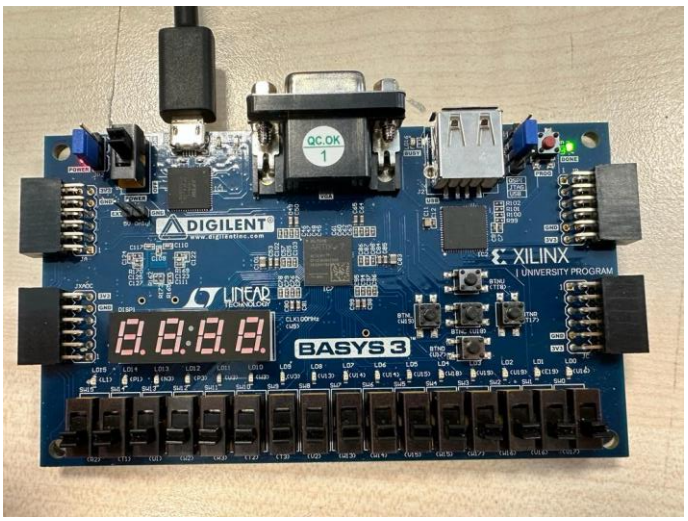
dad : V16

john : W16

jane : W17

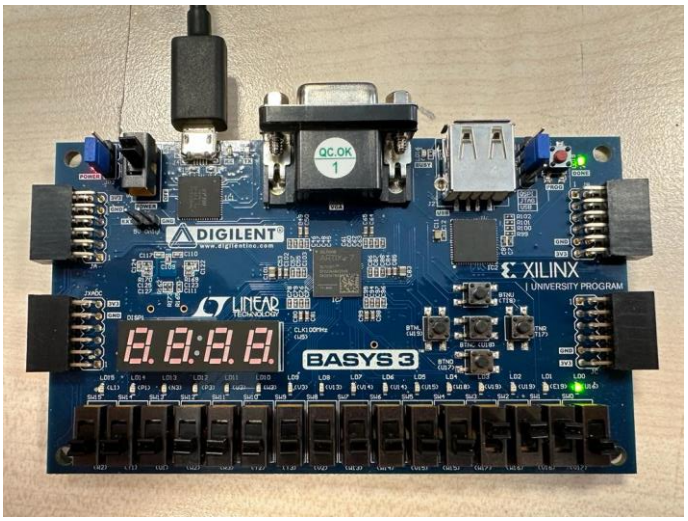
cat : U16

Finally, we generated the ".bit" file and uploaded it to our BASYS3 boards. Then by opening and closing the assigned switches we observed the LED to check whether our truth table and equation was correct. Some of these different switch combinations are given in the following figures:



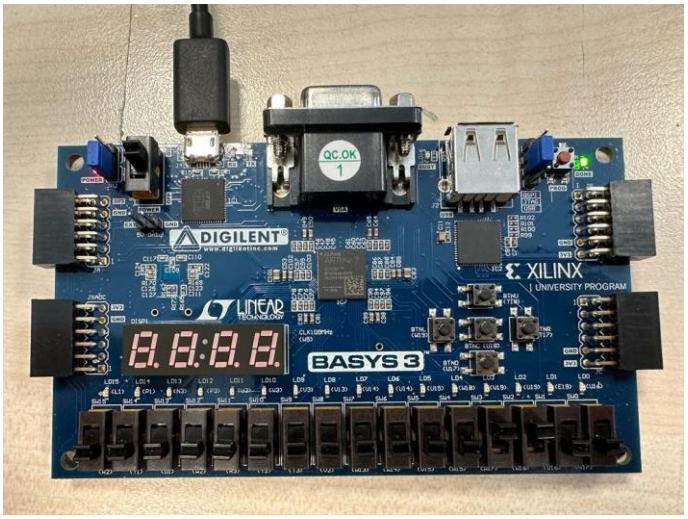
mom : 0
dad : 0
john : 0
jane : 0
cat : 0

Figure. 3 Combination 0-0-0-0



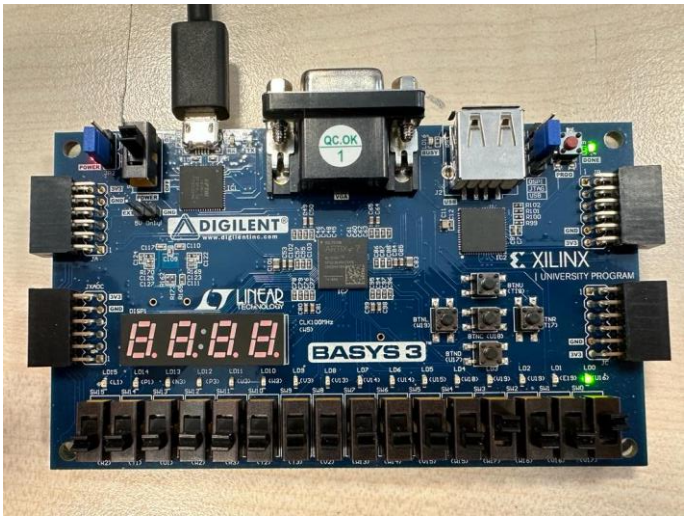
mom : 0
dad : 1
john : 1
jane : 1
cat : 1

Figure.4 Combination 0-1-1-1



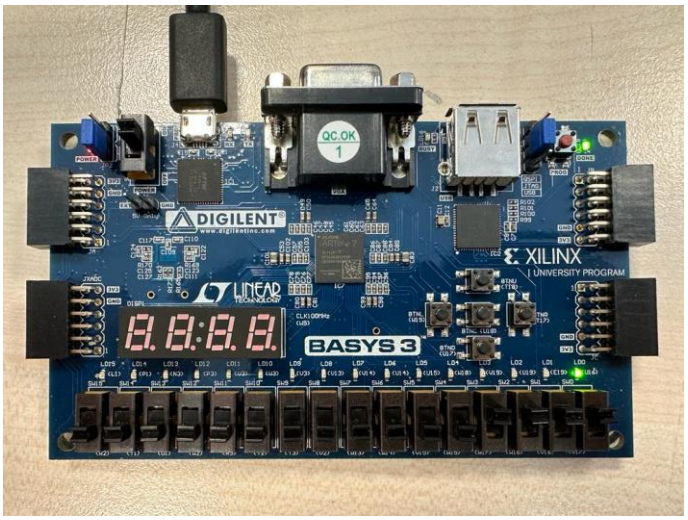
mom : 0
dad : 0
john : 1
jane : 1
cat : 0

Figure.5 Combination 0-0-1-1



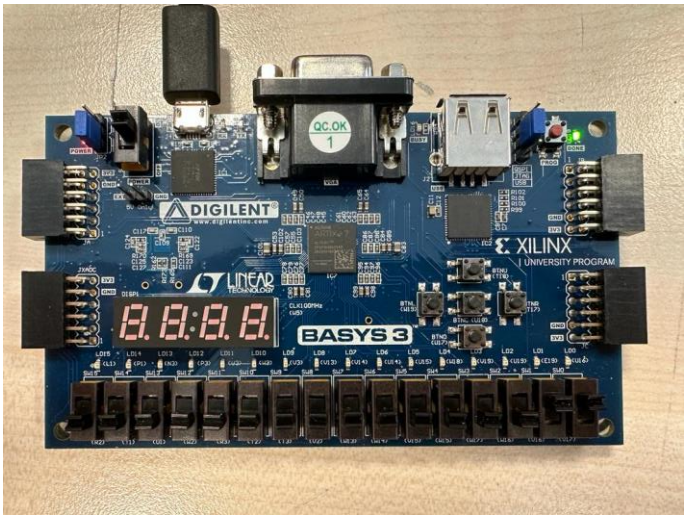
mom : 1
dad : 0
john : 0
jane : 1
cat : 1

Figure.6 Combination 1-0-0-1



mom : 1
dad : 1
john : 1
jane : 1
cat : 1

Figure.7 Combination 1-1-1-1



mom : 1
dad : 1
john : 0
jane : 0
cat : 0

Figure.8 Combination 1-1-0-0

Conclusion:

In this lab, we learned how to use BASYS3 boards and Vivado effectively, and code in VHDL. In other words, we were basically introduced to digital design. We solved a real life problem to familiarize with the theoretical concepts and learn the function of the logic gates found in everyday life. We used schemes to visualize circuits, did truth tables for our equations and analysed waveforms. Thus, one can say that we implemented what we learned in the classroom and better understood theories we discussed in class. All in all, the experiment was successful considering that what we found in theory matched the real life implementation and simulations.

Appendix:

Code.1 Implementation of my circuit (cat1.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity cat1 is
   Port ( mom : in STD_LOGIC;
          dad : in STD_LOGIC;
          john : in STD_LOGIC;
          jane : in STD_LOGIC;
          cat : out STD_LOGIC);
end cat1;
architecture Behavioral of cat1 is
begin
cat <= (mom or dad) and (john or jane);
end Behavioral;
```

Code.2 Test Bench (testbench_cat.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity testbench_cat is
end testbench_cat;
architecture Behavioral of testbench_cat is
COMPONENT cat1
 PORT(  mom : IN std_logic;
        dad : IN std_logic;
        john : IN std_logic;
        jane : IN std_logic;
        cat : OUT std_logic);
END COMPONENT;
signal mom : std_logic;
signal dad : std_logic;
signal john : std_logic;
signal jane : std_logic;
signal cat : std_logic;
begin
UUT: cat1 PORT MAP(  mom => mom,
                     dad => dad,
                     john => john,
                     jane => jane,
                     cat => cat);
testbench_cat: PROCESS
BEGIN
wait for 10 ns;
 mom <='0';
 dad <='0';
 john <='0';
```

```
 jane <='0';
wait for 10 ns;
 mom <='0';
 dad <='0';
 john <='0';
 jane <='1';
wait for 10 ns;
 mom <='0';
 dad <='0';
 john <='1';
 jane <='0';
wait for 10 ns;
 mom <='0';
 dad <='0';
 john <='1';
 jane <='1';
wait for 10 ns;
 mom <='0';
 dad <='1';
 john <='0';
 jane <='0';
wait for 10 ns;
 mom <='0';
 dad <='1';
 john <='0';
 jane <='1';
wait for 10 ns;
 mom <='0';
 dad <='1';
 john <='1';
 jane <='0';
```

```
wait for 10 ns;
 mom <='0';
 dad <='1';
 john <='1';
 jane <='1';
wait for 10 ns;
 mom <='1';
 dad <='0';
 john <='0';
 jane <='0';
wait for 10 ns;
 mom <='1';
 dad <='0';
 john <='0';
 jane <='1';
wait for 10 ns;
 mom <='1';
 dad <='0';
 john <='1';
 jane <='0';
wait for 10 ns;
 mom <='1';
 dad <='0';
 john <='1';
 jane <='1';
wait for 10 ns;
 mom <='1';
 dad <='1';
 john <='0';
 jane <='0';
wait for 10 ns;
```

```
 mom <='1';
 dad <='1';
 john <='0';
 jane <='1';
wait for 10 ns;
 mom <='1';
 dad <='1';
 john <='1';
 jane <='0';
wait for 10 ns;
 mom <='1';
 dad <='1';
 john <='1';
 jane <='1';
 END PROCESS;
end Behavioral;
```