Universidad de las Américas Puebla

Escuela de ingeniería

Departamento de computación, electrónica y mecatrónica

UDLAP®

Lab report #2

Course: Digital design LRT2022-7

Equipo: 2

183339	Juan Pablo Lopez Moreno	LMT
185406	Amy Marianee Ramírez Sánchez	LMT

April 1st, 20, San Andrés Cholula, Puebla

1 Abstract

This report will present the results obtained from the second laboratory practice of the Digital Design course. The main objective of this practice was to familiarize ourselves with the functioning of logic gates and boolean functions. Showcasing both a physical and simulation function based on logic gates.

2 Introduction

Logic gates are the fundamental building blocks of digital electronics, as they allow the processing of binary signals through basic logical operations such as AND, OR, and NOT. A logic gate is a device that receives one or more binary input signals (0 or 1) and produces a binary output depending on the logical operation it performs. These operations enable the structured and predictable manipulation of digital information. There are several types of logic gates, among the most common are:

- AND, which outputs 1 only when all its inputs are 1.
- OR, which outputs 1 if at least one of its inputs is 1.
- NOT, which inverts the value of its input, turning 0 into 1 and vice versa.
- Additionally, there are combinations such as NAND, NOR, XOR, and XNOR, which are used to construct more complex functions and optimize circuits.

[1] From these operations, it is possible to build more complex Boolean functions, representing relationships between binary variables using logical connectors, forming the foundation of digital system design. These functions are essential for developing integrated circuits, microprocessors, memory devices, and modern electronic systems. In this context, the main objectives of this lab were to implement and verify the operation of basic logic gates, obtain the truth table and canonical form of a Boolean function, and simulate and program digital circuits using VHDL. In this way, the lab not only aims to experimentally verify the theoretical principles of Boolean algebra but also to develop skills in using simulation software and hardware description languages, which are essential tools in modern digital system design.

3 Objectives

The objectives of this lab are:

- Implement basic circuits to verify the operation of three basic logic gates.
- Verify the proper functioning of the digital circuits by using the Multimeter.
- Program and simulate basic digital circuits using VHDL
- Obtain truth table and canonic form of a Boolean function.
- Program and simulate Boolean functions using VHDL.

For the purpose of this lab, it is important to know about the components used, in the first place, "logic gates are defined as simple digital circuits that take one or more binary inputs and produce a binary output"[2]. On the other hand, boolean funtions consists of a number of Boolean variables joined by the Boolean connectives, like AND and OR. [3] And last, but not least, the software utilized in the experiment is VHDL (Hardware Description Language), which is a language that describes the behavior of electronic circuits, most commonly digital circuits. [4]

4 Methodology

A methodology was established, divided into three main phases: physical implementation in the laboratory, simulation in Multisim, and programming in EDA Playground.

4.1 Physical Laboratory Phase

In the physical laboratory, the following procedure was carried out:

- The power supply was configured with the following parameters:
  - Activated source: Source 3.
  - Output voltage: 5 V.
  - Output current: 300 mA.
- Using the datasheets, the power terminals, inputs, and outputs of each logic gate were identified.
- The basic circuit of each logic gate was built.
- The truth table presented in the datasheet was verified and compared with the one obtained from the real circuit.
- The truth table and the canonical form of the Boolean function were obtained:

$$F(A,B,C,D) = (\bar{B} \cdot D) + (\bar{A} \cdot B \cdot \bar{C}) + (A \cdot \bar{B} \cdot C) + (A \cdot B \cdot \bar{C})$$

- The circuit corresponding to the Boolean function was built.
- The truth table of the Boolean function was verified against the one obtained experimentally from the real circuit.

4.2 Multisim Simulation Phase

In the Multisim environment, the following tasks were performed:

- The circuits corresponding to the basic logic gates (AND, OR, XOR, NOT, NAND, NOR, and XNOR) were constructed, simulated, and their correct operation was verified.
- The Boolean function defined in point 5 of the laboratory phase was constructed and simulated, confirming its proper functioning.

4.3 EDA Playground Programming Phase

Finally, in the EDA Playground platform, the following steps were carried out:

- A single file was created to program and verify the correct operation of the basic logic gates (AND, OR, XOR, NOT, NAND, NOR, and XNOR).
- The Boolean function F(ABCD) was programmed and simulated.

Finally, an analysis was carried out between the results obtained in the simulation and those obtained experimentally in the laboratory.

5 Results

5.1 Physical lab experimentation

For the physical lab, the team developed the basic circuits for each logic gate, obtaining the following tables:

Input		**Output**
A	**B**	**Y**
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth Table for a 2-Input AND Gate, the 74HC08 integrated circuit.

Input		**Output**
A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Truth Table for a 2-Input OR Gate, the 74HC32 integrated circuit.

Input	**Output**
A	**Y**
0	1
1	0

Table 3: Truth Table for a 1-Input NOT Gate, the 74HC04 integrated circuit.

After ensuring that the truth tables were consistent with the datasheets obtained in the pre-lab, the team started analysing the following boolean function:

$$F(A,B,C,D) = (\bar{B} \cdot D) + (\bar{A} \cdot B \cdot \bar{C}) + (A \cdot \bar{B} \cdot C) + (A \cdot B \cdot \bar{C})$$

And after analysing the function, the following truth table was formulated:

Input				**Output**
A	**B**	**C**	**D**	**Y**
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Table 4: Point 5 truth table.

From which, we were able to build the canonic form of the function, based on the maxterms of the truth table, which is represented this way in this way:

$$\begin{aligned} F(A,B,C,D) = {} & (A+B+C+D)\cdot(A+B+\bar{C}+D)\cdot(A+\bar{B}+\bar{C}+D)\cdot(A+\bar{B}+\bar{C}+\bar{D}) \\ & \cdot(\bar{A}+B+C+D)\cdot(\bar{A}+\bar{B}+\bar{C}+D)\cdot(\bar{A}+\bar{B}+\bar{C}+\bar{D}) \end{aligned}$$

Based on this process, the team built a circuit to showcase the functionality of the boolean function:

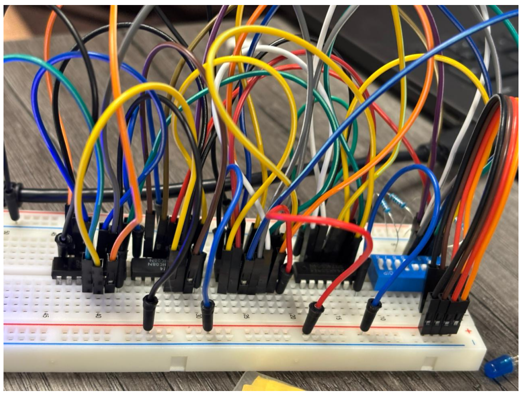
Figure 1: Circuit built in breadboard simulating the Point 5 function.

5.2 Digital lab experimentation

After the development of the physical lab, the team developed in Multisim the same circuits developed in the physical lab, to properly organize and correct if necessary any errors. In the first place, each logic gate was developed and tested:

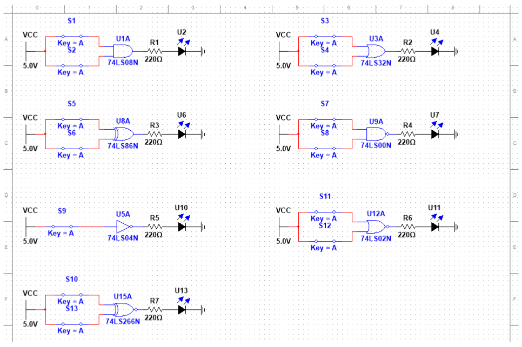

Figure 2: Simulation of the Logic Gates

And then, the point 5 circuit was designed in the plataform:

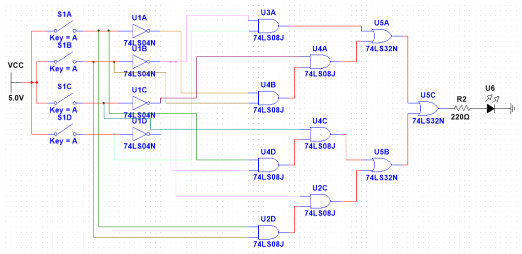

Figure 3: Simulation of the point 5 funciton circuit

5.3 Codes

Finally, the team developed codes in VHDL to prove the functionality of the circuits and function. The first codes where developed in multiple files, one per logic gate:

Código 1: Code 1: AND gate

```
-- Code your design here
library IEEE;
use IEEE.std_logic_1164.all;
entity ent_andGate is

    port(A : in std_logic;          -- AND gate input
         B : in std_logic;          -- AND gate input
         Y : out std_logic);        -- AND gate output

end ent_andGate;
```

```
-- Architecture definition

architecture arq_andGate of ent_andGate is

 begin

    Y <= A and B;

end arq_andGate;
```

Código 2: Code 2: OR gate

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity ent_orGate is

    port(A : in std_logic;        -- AND gate input
         B : in std_logic;        -- AND gate input
         Y : out std_logic);      -- AND gate output

end ent_andGate;

-- Architecture definition

architecture arq_orGate of ent_orGate is

 begin

    Y <= A or B;

end arq_orGate;
```

Código 3: Code 3: NOT gate

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity ent_notGate is

    port(A : in std_logic;        -- NOT gate input
         Y : out std_logic);      -- NOT gate output
```

```
end ent_notGate;

-- Architecture definition

architecture arq_notGate of ent_notGate is

 begin

    Y <= not(A);

end arq_notGate;
```

Código 4: Code 4: NAND gate

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity ent_nandGate is

    port(A : in std_logic;         -- AND gate input
         B : in std_logic;         -- AND gate input
         Y : out std_logic);       -- AND gate output

end ent_nandGate;

-- Architecture definition

architecture arq_nandGate of ent_nandGate is

 begin

    Y <= not(A and B);

end arq_nandGate;
```

Código 5: Code 5: NOR gate

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity ent_norGate is
```

```
    port(A : in std_logic;        -- AND gate input
         B : in std_logic;        -- AND gate input
         Y : out std_logic);      -- AND gate output

end ent_norGate;

-- Architecture definition

architecture arq_norGate of ent_norGate is

 begin

    Y <= not(A or B);

end arq_norGate;
```

Código 6: Code 6: XOR gate

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity ent_xorGate is

    port(A : in std_logic;        -- AND gate input
         B : in std_logic;        -- AND gate input
         Y : out std_logic);      -- AND gate output

end ent_xorGate;

-- Architecture definition

architecture arq_xorGate of ent_xorGate is

 begin

    Y <= A xor B;

end arq_xorGate;
```

Código 7: Code 7: XNOR gate

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
-- Entity declaration

entity ent_xnorGate is

    port(A : in std_logic;          -- AND gate input
         B : in std_logic;          -- AND gate input
         Y : out std_logic);        -- AND gate output

end ent_xnorGate;

-- Architecture definition

architecture arq_xnorGate of ent_xnorGate is

 begin

    Y <= not(A xor B);

end arq_xnorGate;
```

After the development of these codes in EDA PLygrounds, the team created a code that showcases the boolean function of point 5:

Código 8: Code 8: Point 5 Code

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity ent_example_5 is

    port(A : in std_logic;          -- OR gate input
         B : in std_logic;
         C : in std_logic;
         D : in std_logic;
         Y : out std_logic);        -- OR gate output

end ent_example_5;

-- Architecture definition

architecture arq_example_5 of ent_example_5 is

 begin
```

```
    Y <= (((not B)and D)or((not A)and B and (not C))or(A and (not B) and C)o

end arq_example_5;
```

And to test this code, the team created the following:

Código 9: Code 1: Point 5 Testbench

```
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
-- empty
end testbench;

architecture tb of testbench is

-- DUT component
component ent_example_5 is
port(
  A: in std_logic;
  B : in std_logic;
  C : in std_logic;
  D : in std_logic;
  Y: out std_logic);
end component;

signal a_in, b_in, c_in, d_in, q_out: std_logic;

begin

  -- Connect UUT
  UUT: ent_example_5 port map(a_in, b_in, c_in, d_in, q_out);

  process
  begin

    a_in <= '0';
    b_in <= '0';
    c_in <= '0';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '0';
    c_in <= '0';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '0';
    c_in <= '1';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '0';
    c_in <= '1';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '1';
    c_in <= '0';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '1';
    c_in <= '0';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '1';
    c_in <= '1';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '0';
    b_in <= '1';
    c_in <= '1';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '0';
    c_in <= '0';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '0';
    c_in <= '0';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '0';
    c_in <= '1';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '0';
    c_in <= '1';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '1';
    c_in <= '0';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '1';
    c_in <= '0';
    d_in <= '1';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '1';
    c_in <= '1';
    d_in <= '0';
    wait for 1 ns;

    a_in <= '1';
    b_in <= '1';
    c_in <= '1';
    d_in <= '1';
    wait for 1 ns;

  end process;
end tb;
```

6 Analysis

Based on the data obtained during the laboratory practice, the observed behavior of the logic gates corresponds correctly to the expected truth tables. The discrepancies that initially appeared in the practical implementation were due to wiring errors and circuit organization, which were later corrected. After completing the laboratory, the circuits were reproduced in Multisim to ensure proper organization and eliminate any possible mistakes. Each logic gate was tested individually, confirming its correct functionality. Finally, the circuits were implemented in EDA Playground using VHDL, where their behavior was simulated and verified. The results in this environment also matched the theoretical expectations, validating the practical and simulated work. In conclusion, both the laboratory implementation and the digital simulations in Multisim and VHDL confirm the correct functionality of the logic gates and the designed circuits.

7 Conclusions

The development of this practice made it possible to implement and verify the operation of three basic logic gates, both in the laboratory and in the simulation environments. The use of the multimeter was essential to confirm the correct functioning of the digital circuits, strengthening the understanding of measurement and verification processes in electronics. Additionally, the Boolean function was analyzed by obtaining its truth table and canonical form, which was then implemented and validated experimentally. This analysis was complemented with simulations in Multisim and programming in VHDL through EDA Playground, confirming the correct logical behavior of the circuits and the function in different environments. In this way, the proposed objectives were satisfactorily achieved, as the laboratory practice not only allowed the experimental validation of theoretical principles but also demonstrated the usefulness of digital simulation tools such as Multisim and VHDL for predicting and verifying circuit behavior.

References

[1] Grupo de Invstigación en Sistemas Inteligentes, Universidad Nacional Autónoma de México, “Compuertas lógicas – sistemas digitales,” https://virtual.cuautitlan.unam.mx/intar/sistdig/compuertas-logicas, 2018, accessed: 2024-10-27.

[2] S. L. Harris and D. Harris, “1 - from zero to one,” in *Digital Design and Computer Architecture*, S. L. Harris and D. Harris, Eds. Morgan Kaufmann, 2022, pp. 1–50. [En línea]. Disponible: https://www.sciencedirect.com/science/article/pii/B9780128200643000015

[3] B. HOLDSWORTH and R. WOODS, “2 - boolean algebra,” in *Digital Logic Design (Fourth Edition)*, fourth edition ed., B. HOLDSWORTH and R. WOODS, Eds. Oxford: Newnes, 2002, pp. 28–42. [En línea]. Disponible: https://www.sciencedirect.com/science/article/pii/B9780750645829500032

[4] Intel Corporation, “Vhdl definition,” https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_vhdl.htm, 2017, accessed: 2024-10-27.