UNIVERSITY OF GHANA

Name: Evans Acheampong

Student ID: 10987644

Date of Submission: 3rd June, 2023

Lab Index: LAB 1

VHDL Logic Gates Design and Simulation

I. AIM OF THE EXPERIMENT

The aim of this lab is to familiarize with the fundamental concepts of VHDL and its application in designing and simulating logic gates. By completing the lab tasks, the aim is to achieve the following objectives:

1. Design the NAND gate, NOR gate, and Exclusive NOR gate using VHDL.
2. Utilize the simulation capabilities of VHDL to verify the functionality of the designed logic gates.
3. Employ the WAVE feature in ModelSim to force signal values through the input

II. ABSTRACT

This lab report presents the design and simulation of three logic gates (NAND gate, NOR gate, and Exclusive NOR gate) using VHDL. The report provides a comprehensive summary of the lab, including the introduction to VHDL, the importance of the lab session and task, the methodology employed, the results obtained, and a discussion of the outcomes. The report concludes with key takeaways from the lab experience.

III. INTRODUCTION

VHDL, which stands for Very High Speed Integrated Circuits Hardware Description Language, is a language used for describing digital electronic systems. It emerged from the VHSIC (Very High-Speed Integrated Circuits) program initiated by the United States government in 1980. VHDL became a standard language for describing the structure, function, and interconnections of designs and sub-designs. It offers several benefits, such as the ability to simulate designs before manufacturing, facilitating design comparison and correctness testing without hardware prototyping, and enabling design reuse and scalability.

The lab report focuses on the design and simulation of logic gates using VHDL. Logic gates are fundamental building blocks in digital circuit design, and understanding their implementation in VHDL is crucial. The three logic gates covered in this lab are the NAND gate, NOR gate, and Exclusive NOR gate. These gates exhibit different logical behaviors and are widely used in digital circuits.

The lab session and task are important as they provide hands-on experience in implementing VHDL code for logic gates. By designing and simulating these gates, students gain a deeper understanding of VHDL concepts, syntax rules, and data types. Furthermore, the simulation capabilities of VHDL, coupled with tools like ModelSim, allow students to visualize the waveforms of the design ports and verify the correctness of their implementations.

IV. METHODOLOGY

a. Block Diagram of Entity

The block diagram illustrates the structure of the entity, including the input and output port terminals and their respective data types. In this lab, I designed three logic gates: NAND gate, NOR gate, and Exclusive NOR gate. The block diagram for each gate is as follows:

1. **NAND Gate:**

- Inputs:
  - A (data type: std_logic)
  - B (data type: std_logic)

- Output:
  - Y (data type: std_logic)

The entity is declared as nand_gate and A,B are declared as input, Y is declared as Output.

```
entity nand_gate is
   port(A: in std_logic;
        B: in std_logic;
        Y: out std_logic);
end nand_gate;
```

Once the entity is declared, the next step is to define the architecture of the entity (nand_gate). The architecture name is given as “nandLogic”. The architecture of the entity(nand_gate) is defined by “architecture nandLogic of nand_gate is”. Now that we know Y is the output, we know that output is nothing but the complement of AND operation of the two inputs A and B. So, Y is mapped to NOT(A AND B) (Y<=NOT (A AND B)). The Boolean operator ‘AND’ is used to perform the AND operation between A and B. For complementing the AND operation NOT operator is used. The architecture is terminated by “end nandLogic”.

2. **NOR Gate:**

- Inputs:
  - A (data type: std_logic)
  - B (data type: std_logic)
- Output:
  - Y (data type: std_logic)

The entity is declared as nor_gate and A, B are declared as inputs, Y is declared as Output.

```
entity nor_gate is
   port(A: in std_logic;
```

```
        B: in std_logic;
        Y: out std_logic);
end nor_gate;
```

Once the entity is declared, the next step is to define the architecture of the entity (nor_gate). The architecture name is given as “norLogic”. The architecture of the entity(nor_gate) is defined by “architecture norLogic of nor_gate is”. Now that we know Y is the output, we know that output is nothing but the complement of OR operation of the two inputs A and B. So Y is mapped to NOT(A OR B) (Y<=NOT (A AND B)). The Boolean operator ‘OR’ is used to perform the OR operation between A and B. For complementing the OR operation NOT operator is used. The architecture is terminated by “end norLogic”

3. **Exclusive NOR Gate:**

- Inputs:
  - A (data type: std_logic)
  - B (data type: std_logic)
- Output:
  - Y (data type: std_logic)

The entity is declared as xnor_gate and A,B are declared as inputs , Y is declared as Output.

```
entity xnor_gate is
    port(A: in std_logic;
         B: in std_logic;
         Y: out std_logic);
end xnor_gate;
```

Once the entity is declared, the next step is to define the architecture of the entity (xnor_gate). The architecture name is given as “xnorLogic”. The architecture of the entity(xnor_gate) is defined by “architecture xnorLogic of xnor_gate is”. Now that we know Y is the output, we know that output is nothing but the complement of the XOR operation of the two inputs A and B. So Y is mapped to NOT(A XOR B) (Y<=NOT (A XOR B)). The Boolean operator ‘XOR’ is used to perform the AND operation between A and B. For complementing the XOR operation NOT operator is used. The architecture is terminated by “end xorLogic”.

```
architecture xnorLogic of xnor_gate i
 begin
    Y <= not(A xor B);
end xnorLogic;
```

b. **Explanation of Architectures**

Each logic gate has its corresponding architecture, which defines its behavior based on the input values. The architectures are implemented using concurrent statements in VHDL. Here is an explanation of the architectures for the three logic gates:

1. **NAND Gate Architecture:**

- The NAND gate architecture utilizes the "nand" built-in operator in VHDL.
- The output (Y) is assigned the result of the logical NAND operation between inputs A and B.

2. **NOR Gate Architecture:**

- The NOR gate architecture utilizes the "nor" built-in operator in VHDL.
- The output (Y) is assigned the result of the logical NOR operation between inputs A and B.

3. **Exclusive NOR Gate Architecture:**

- The Exclusive NOR gate architecture combines the "xor" and "not" built-in operators in VHDL.
- The output (Y) is assigned the result of the logical XOR operation between inputs A and B, followed by a logical NOT operation.

By implementing these architectures in VHDL, the desired behavior of the logic gates is achieved, enabling their simulation and validation.

Note: The data type used for the inputs and output is `std_logic`, which represents a single bit in VHDL and is commonly used for digital signals in digital circuit designs.

V. RESULTS AND DISCUSSION

1. Truth Table of Logic Gates:
    - NAND Gate:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

    - NOR Gate:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

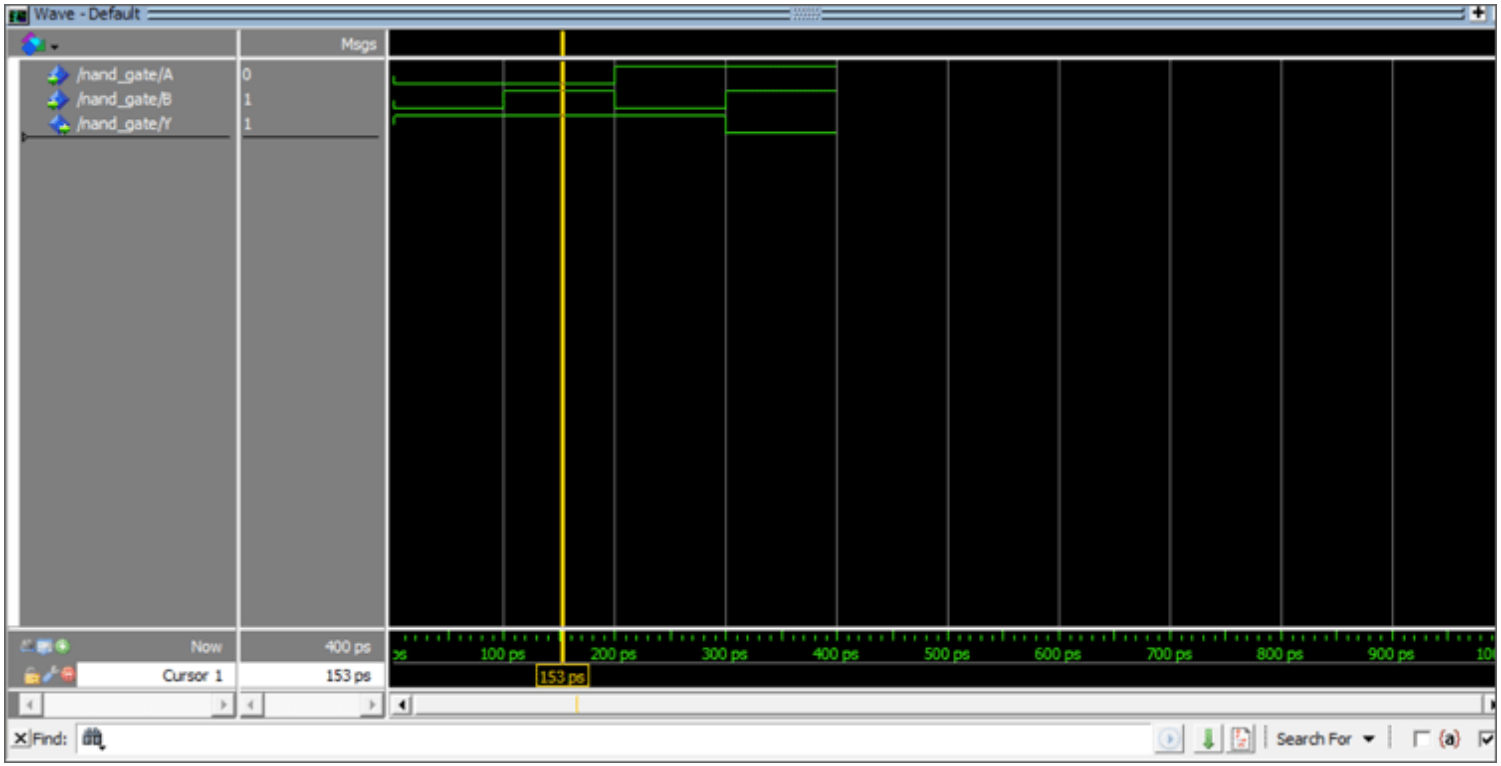
- Exclusive NOR Gate:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

- Waveform Simulation:

- The waveform simulations of the logic gates were conducted using ModelSim.
- Signal values were forced through the input terminals, and the resulting waveforms at the output terminals were observed.
- The waveforms exhibited the expected behavior of each logic gate, validating the correctness of the designs.

NAND Gate Waveform Simulation

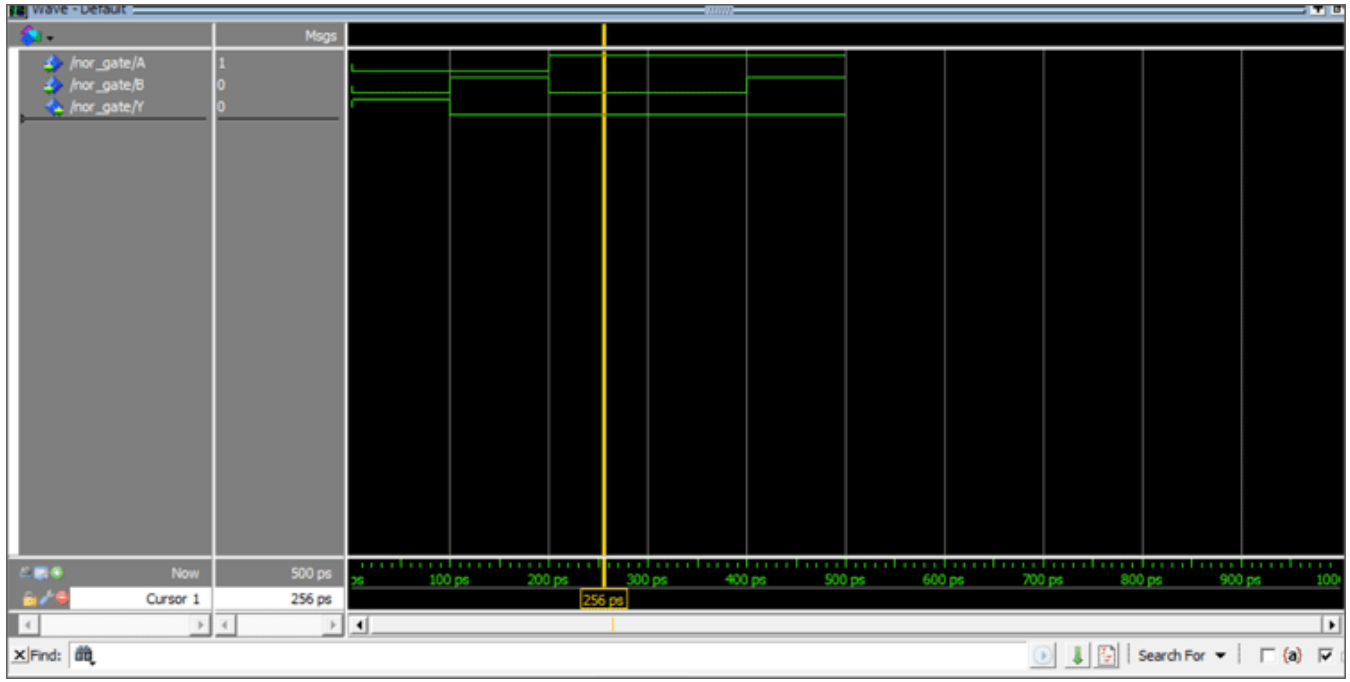


Now that all the input combinations for A and B(00,01,10,11) are forced to the input signals, move the cursor throughout the waveform graph from 0ps to 400ps. The values of A, B, Y at corresponding time intervals is given below.

Timing(ps)	A	B	Y
0-100	0	0	1
100-200	0	1	1
200-300	1	0	1
300-400	1	1	0

The above values of A, B, Y corresponds to the Truth Table of the NAND Gate.

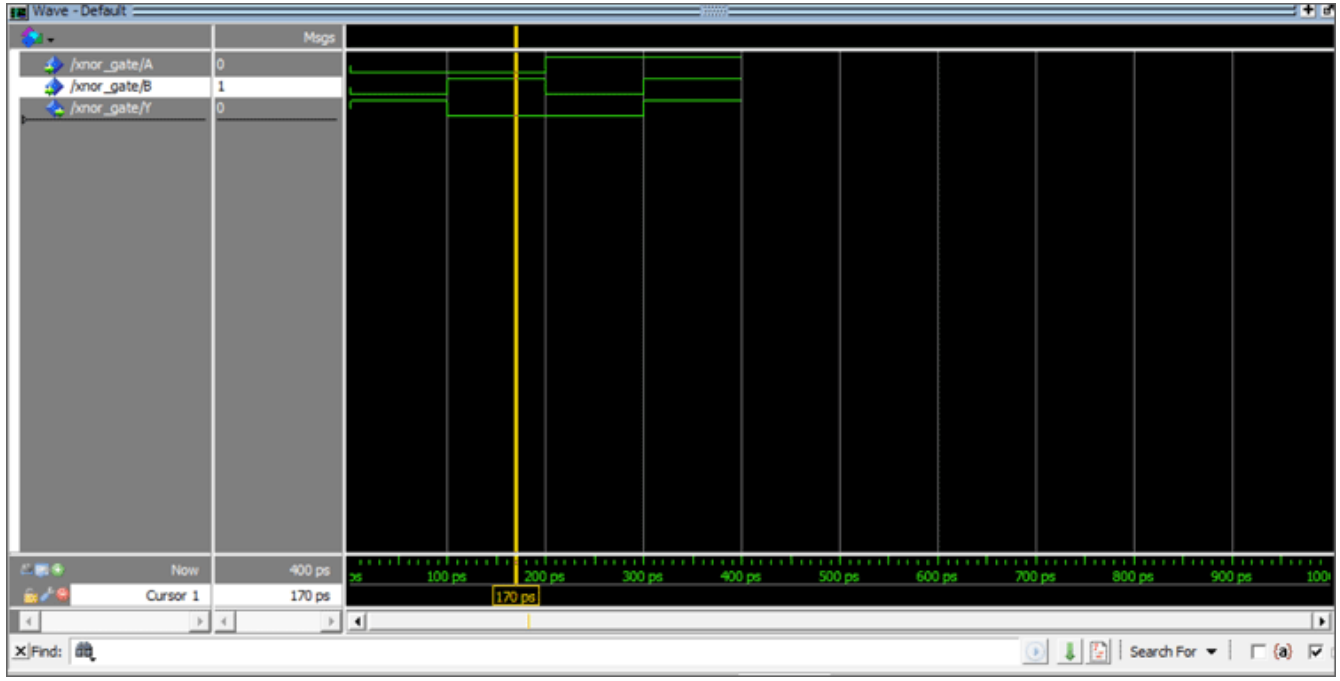
NOR Gate Waveform Simulation



Timing(ps)	A	B	Y
0-100	0	0	1
100-200	0	1	0
200-300	1	0	0
300-400	1	1	0

The above values of A, B, Y corresponds to the Truth Table of the NOR Gate.

XNOR Gate Waveform Simulation



Now that all the input combinations for A and B(00,01,10,11) are forced to the input signals, move the cursor throughout the waveform graph from 0ps to 400ps. The values of A, B, Y at corresponding time intervals are given below.

Timing(ps)	A	B	Y
0-100	0	0	1
100-200	0	1	0
200-300	1	0	0
300-400	1	0	1

The above values of A, B, Y corresponds to the Truth Table of the XNOR Gate.

- Comparison of Truth Tables and Simulation Results:

- The simulation results matched the expected truth tables for each logic gate.
- The outputs of the logic gates corresponded to the specified truth table values for all input combinations.
- The comparison confirmed the accuracy of the VHDL implementations.

VI. CONCLUSION

In conclusion, this lab provided valuable insights into VHDL and its application in designing and simulating logic gates. By implementing the NAND gate, NOR gate, and Exclusive NOR gate using VHDL, I gained hands-on experience in writing VHDL code, understanding the architecture of digital circuits, and visualizing waveform simulations. The lab enhanced my understanding of VHDL's role in digital circuit design and its importance in modern electronic fabrications. Additionally, the lab reinforced the concepts of libraries, data types, syntax rules, and the concurrent execution nature of VHDL. Overall, the lab experience improved my knowledge and skills in VHDL and its practical applications in digital circuit design.