Ryerson University

Department of Electrical, Computer & Biomedical Engineering
Faculty of Engineering & Architectural Science

Programs: Computer Engineering

Course Number	**COE608**
Course Title	**Computer Organization and Architecture**
Semester/Year	**Spring 2020**
Instructor	**Patrick Siddavaatam**

Lab Report No.	**1**

Lab Title	**Introduction to Quartus II**

Section No.	**011**
Group No.	
Submission Date	**30 June 2020**
Due Date	**2 July 2020**

Name	Student ID	Signature*
Duanwei Zhang	*500824903*	DZ
Xinyu Hadrian Hu	500194233	XHH

**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:*
www.ryerson.ca/senate/current/pol60.pdf.

Names: Xinyu Hadrian Hu, and Duanwei Zhan

Student Number: 500194233, and 500824903

TA: Jasminder Singh

Date: June 25, 2020

COE 608: Computer Architecture and Design

COE 608: Lab 1 Report

Objective

The purpose of this lab is to re-acquaint ourselves with the VHDL language, and to re-familiarize ourselves with the Quartus 13.1 IDE. The goal is to generate an XOR gate.

Design and Implementation

The design and implementation of the XOR gate was utilized using the following Truth Table:

See Figure 1: XOR Gate Truth Table below.

XOR Gate		
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1: XOR Gate Truth Table

This logic-gate diagram from All About Circuits website is an excellent graphical display of the xor gate. See Figure 2: XOR Gate Logic Diagram on the next page.

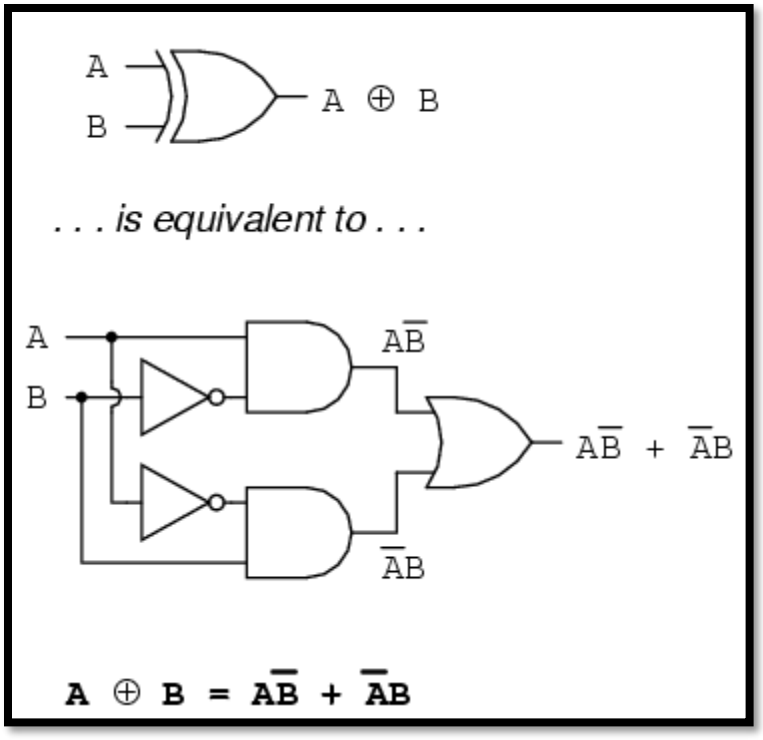


Figure 2: XOR Gate Logic Diagram

The function of the XOR gate is therefore: $F = (A\ and\ (not\ B))\ or\ ((not\ A)\ and\ B)$.

Observations and Results

The following figures: Figure 3: Functional Waveform for 2to1 multiplexer and Figure 4: Timing Waveform for 2to1 multiplexer are the waveforms generated from Quartus II.

Waveforms: Functional and Timing

Below are the waveforms, both functional and timing:

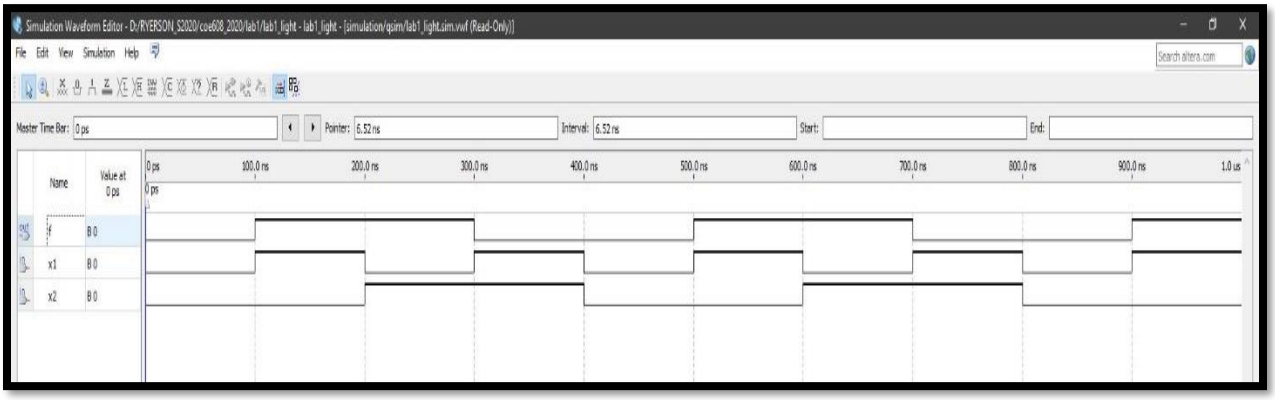


Figure 3: Functional Waveform for 2to1 multiplexer

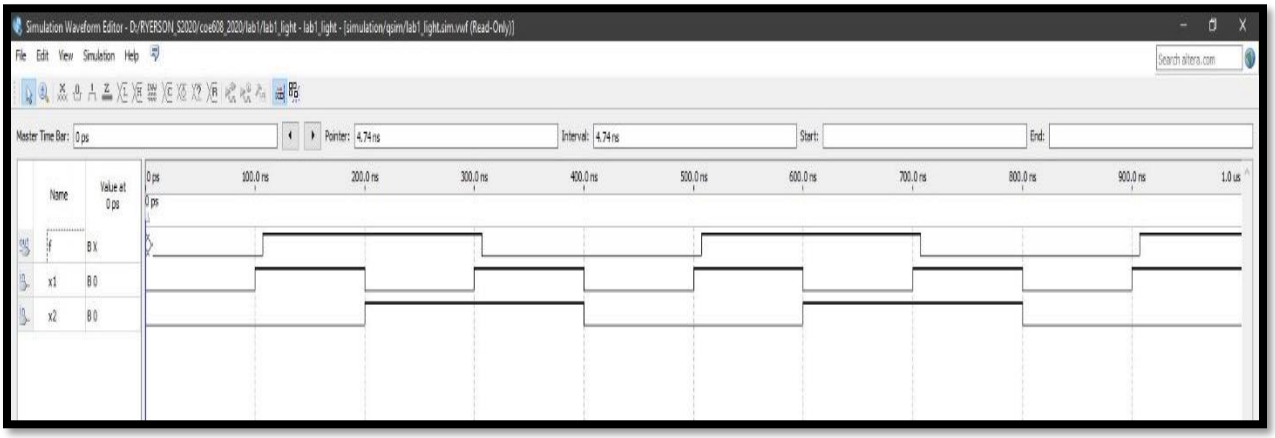


Figure 4: Timing Waveform for 2to1 multiplexer

Discussions and Conclusion

As can be observed, there is only a slight delay of less than 1 ns with the implementation of the code for the 2-to-1 multiplexer.

This XOR function works as intended. For example, when there is either a one or a zero, then there would be one. If the inputs have both zeroes and ones, then the output is zero.

Appendix: VHDL Codes

I have included the VHDL codes and the screenshots of the waveforms for better viewing.

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   entity lab1_light is
4      port
5      (x1, x2 : in std_logic;
6      f : out std_logic);
7   end lab1_light;
8
9   architecture logicfunction of lab1_light is
10     begin
11        f <= ((x1 and (not x2)) or ((not x1) and x2));
12     end logicfunction;
13
14
15
```

Simulation Waveform Editor - D:/RYERSON_S2020/coe608_2020/lab1/lab1_light - lab1_light - [simulation/qsim/lab1_light.sim.vwf (Read-Only)]

File Edit View Simulation Help

Search altera.com

Master Time Bar: 0 ps Pointer: 6.52 ns Interval: 6.52 ns Start: End:

Name	Value at 0 ps
f	B 0
x1	B 0
x2	B 0

0 ps 100.0 ns 200.0 ns 300.0 ns 400.0 ns 500.0 ns 600.0 ns 700.0 ns 800.0 ns 900.0 ns 1.0 us

0 ps

0% 00:00:00

Simulation Waveform Editor - D:/RYERSON_S2020/coe608_2020/lab1/lab1_light - lab1_light - [simulation/qsim/lab1_light.sim.vwf (Read-Only)]

File Edit View Simulation Help

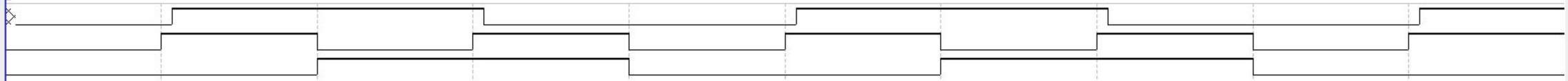
Search altera.com

Master Time Bar: 0 ps Pointer: 4.74 ns Interval: 4.74 ns Start: End:

Name	Value at 0 ps
f	B X
x1	B 0
x2	B 0

0 ps 100.0 ns 200.0 ns 300.0 ns 400.0 ns 500.0 ns 600.0 ns 700.0 ns 800.0 ns 900.0 ns 1.0 us

0 ps



0% 00:00:00