

Simulation Exercise 4
Introduction to the Xilinx Vivado Design
Environment
ELEE 2640

Benyamain YACOOB

February 23, 2024

Date Performed: February 16, 2024

Partners: Ara OLADIPO
Andre PRICE

Instructor: Professor PAULIK



Contents

1 Objectives	3
2 Problem Statement	3
3 Materials	3
4 Requirements	3
5 Design Code/Schematic/Simulation of AND, OR, XOR, and NOT Gates	8
5.1 Truth Table Verification	9
5.2 Results and Analysis Discussion	10
6 Functional (Behavioral) Simulation for Generated Design Verification	11
6.1 Results and Analysis Discussion	13
7 Results and Conclusion	13
8 Group Contributions	14

1 Objectives

After completion of this exercise, we will be able to use the Vivado logic design environment to capture, simulate, test, and download a logic circuit to a Basys 3 board. For this class simulation exercise, we are not required to work with an actual Basys 3 board, but one can be made available.

2 Problem Statement

In this Vivado exercise, we will start by implementing the basic logic gates AND, OR, NOT and the compound exclusive OR gate (XOR) and verify their truth tables. Then we will implement a multigate circuit.

3 Materials

The Xilinx's FPGA VIVADO HLx Editions design tools

4 Requirements

Design Flow Diagrams

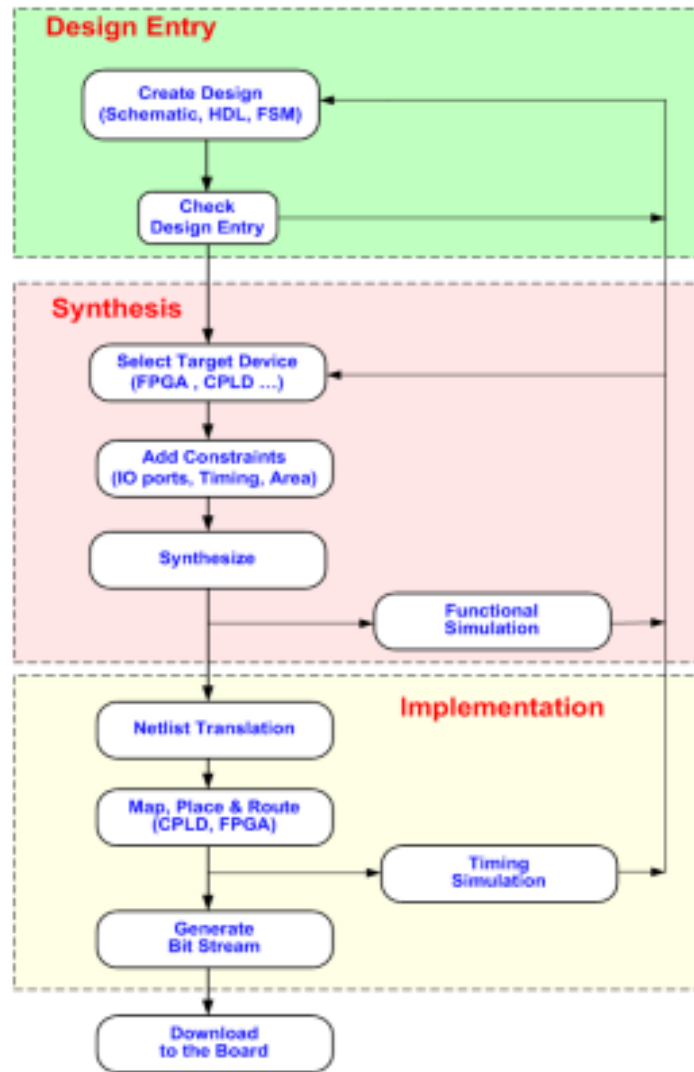


Figure 1: Detailed Design Flow Diagram for FPGA Devices

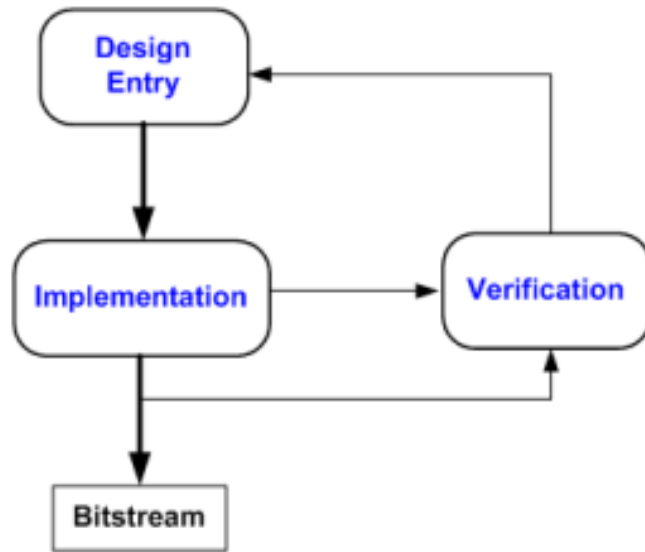


Figure 2: Detailed Design Flow Diagram for FPGA Devices

IO Devices

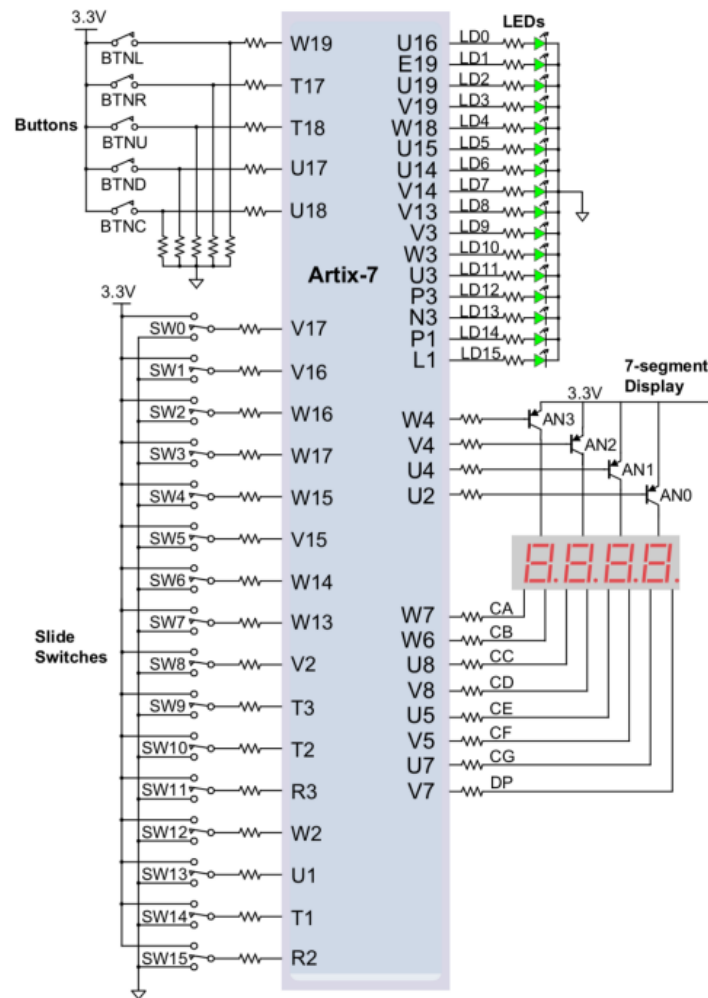


Figure 3: General Purpose IO Devices on the Basys3

Gates

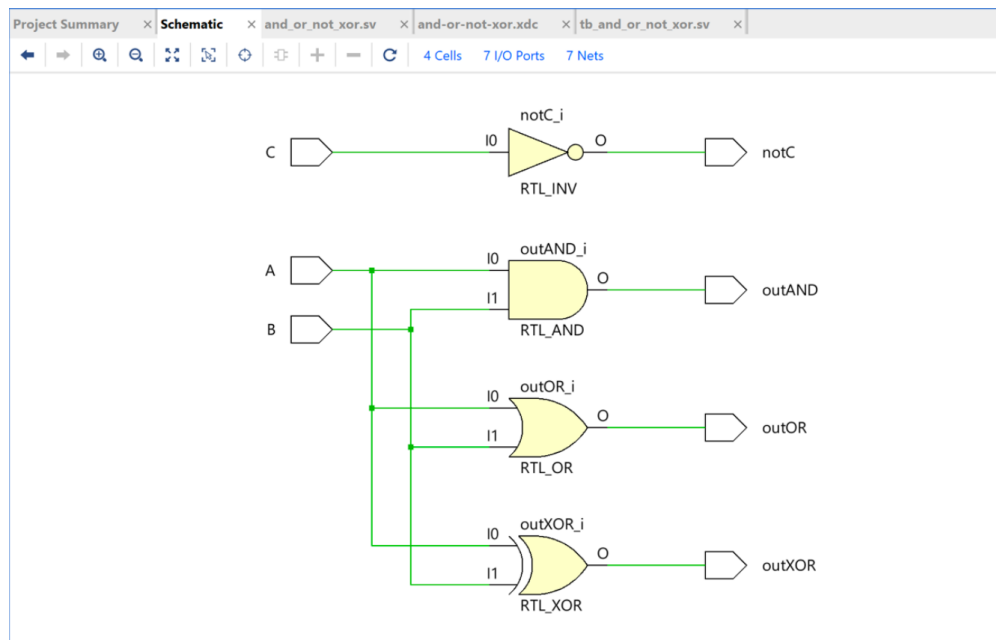


Figure 4: AND, OR, XOR, and NOT Gates

Test Bench

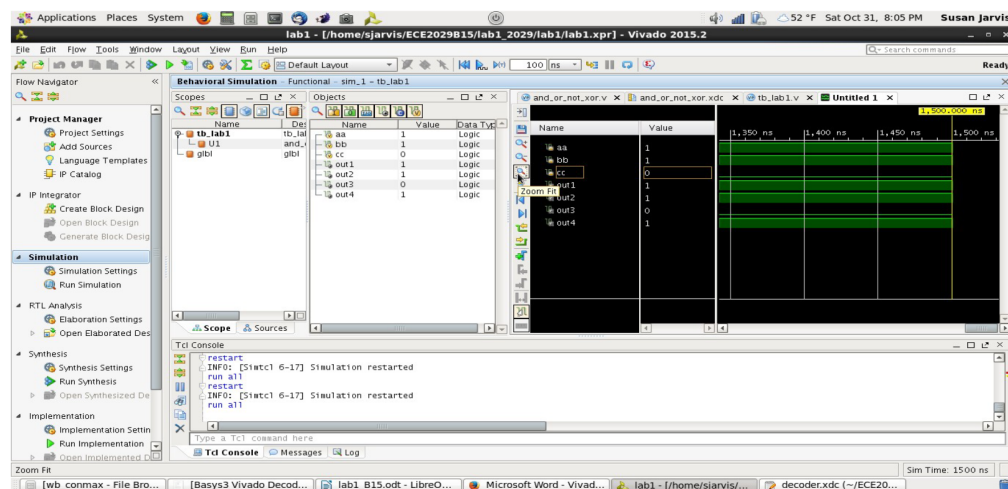
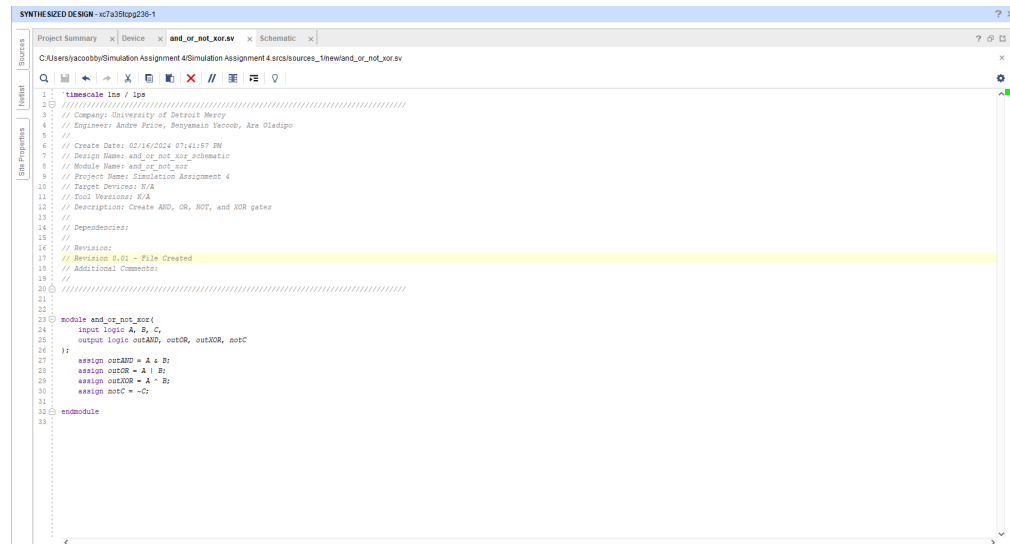


Figure 5: Test Bench Waveform

5 Design Code/Schematic/Simulation of AND, OR, XOR, and NOT Gates



```

1 //timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company: University of Detroit Mercy
4 // Engineer: Andre Price, Benyemine Yacoub, Are Gladipo
5 //
6 // Create Date: 02/14/2024 07:41:07 PM
7 // Design Name: and_or_not_xor_schematic
8 // Module Name: and_or_not_xor
9 // Project Name: Simulation Assignment 4
10 // Target Device: N/A
11 // Tool Versions: N/A
12 // Description: Create AND, OR, NOT, and XOR gates
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 module and_or_not_xor(
24     input logic A, B, C,
25     output logic outAND, outOR, outXOR, notC
26 )
27     assign outAND = A & B;
28     assign outOR = A | B;
29     assign outXOR = A ^ B;
30     assign notC = ~C;
31
32 endmodule
33

```

Figure 6: Design Code

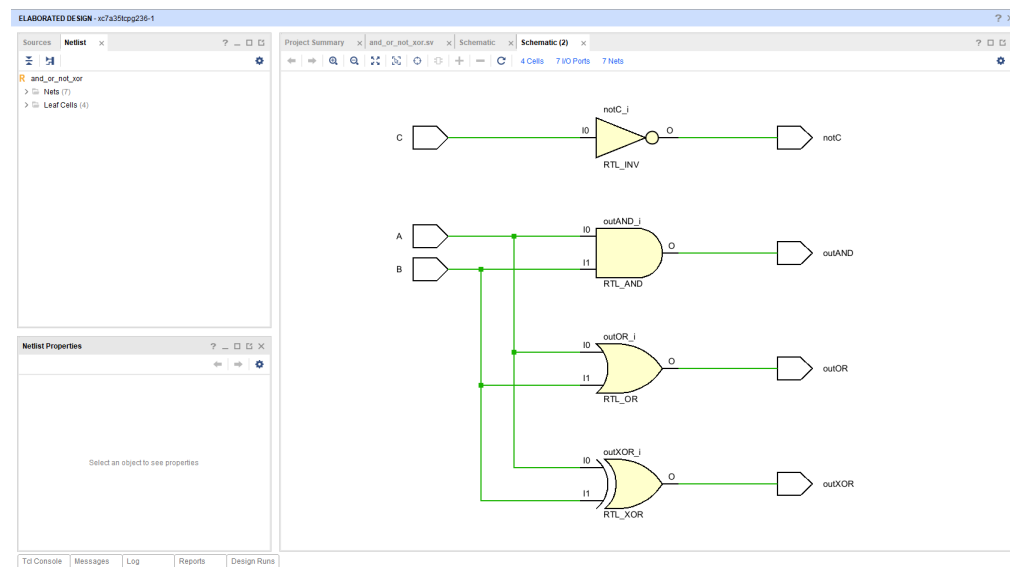


Figure 7: Elaborated Design Schematic

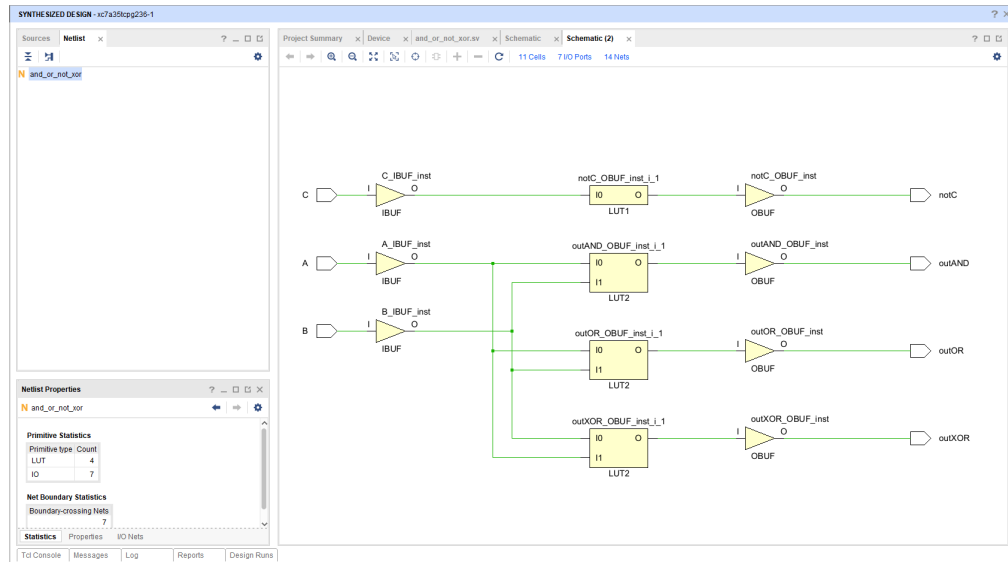


Figure 8: Synthesized Design Schematic

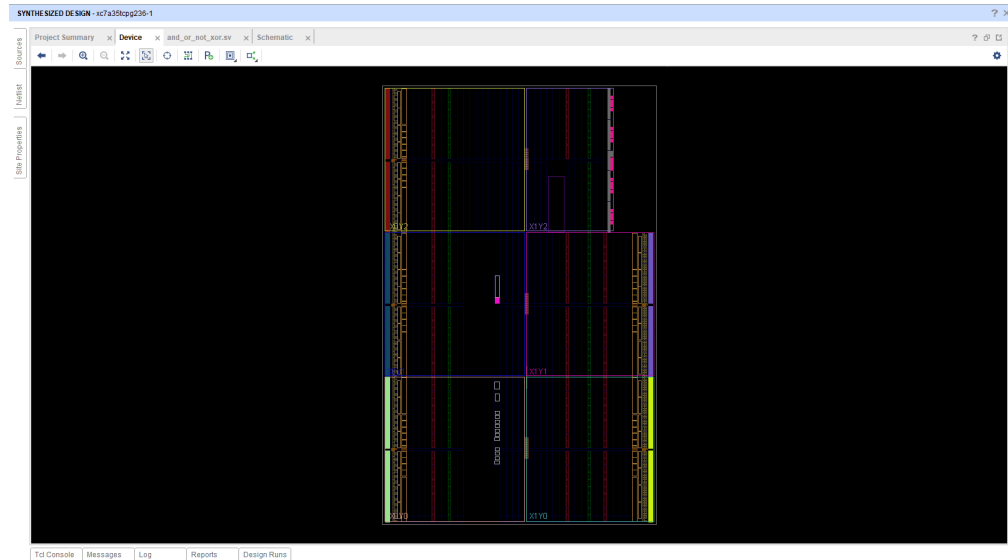


Figure 9: Synthesized Design Simulation

5.1 Truth Table Verification

Using the test bench waveform, we were able to use test variables from the *tb_and_or_not_xor.sv* file. To validate accuracy, we tested various combinations of the three inputs. By tracing the outputs corresponding to each input combination, we were able to es-

tabish a truth table, and we verified that the outputs generated by our circuits for the corresponding gates were correct. A summary of the mapping of inputs to the output gates can be seen below.

$$aa = A \quad (1)$$

$$bb = B \quad (2)$$

$$cc = C \quad (3)$$

$$out1 = AND \quad (4)$$

$$out2 = OR \quad (5)$$

$$out3 = XOR \quad (6)$$

$$out4 = notC \quad (7)$$

5.2 Results and Analysis Discussion

The process for constructing the AND, OR, XOR, and NOT gates from the assignment's specifications was relatively straightforward. We experienced no inconsistencies between our results and the requirements. Even though the code was given to us, we were able to understand how it worked. Vivado's user interface was initially challenging, but we gradually developed a better grasp and familiarity with the software over time.

6 Functional (Behavioral) Simulation for Generated Design Verification

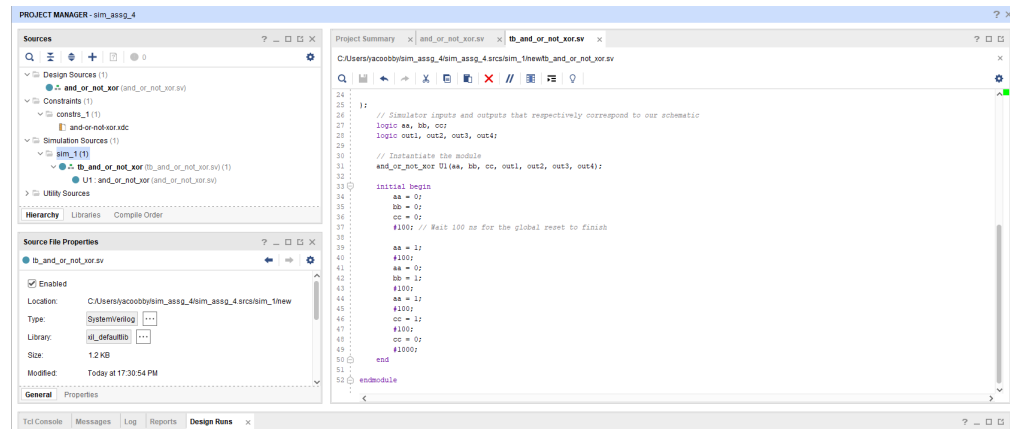


Figure 10: Test Bench Design Code

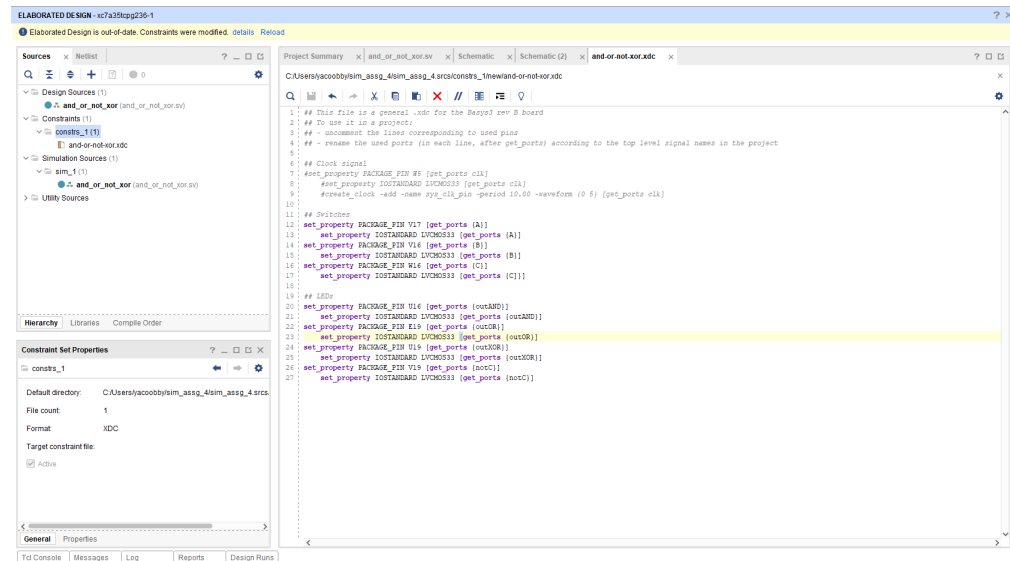


Figure 11: Basys3 Constraints

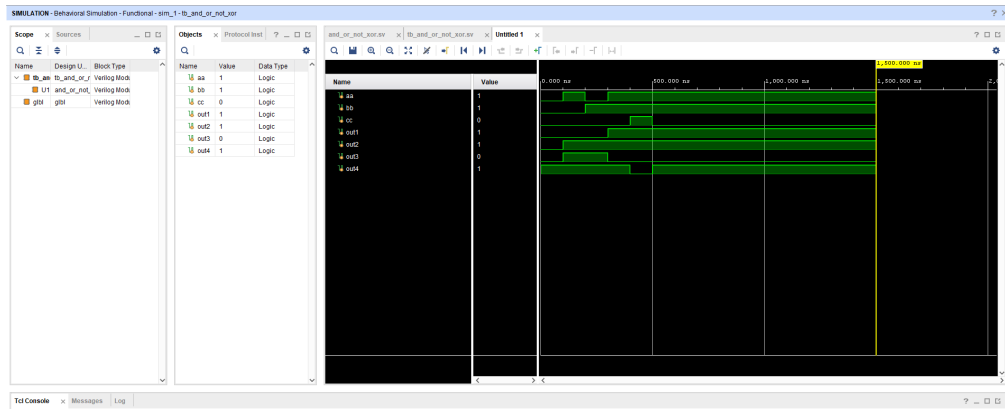


Figure 12: Test Bench Behavioral Simulation

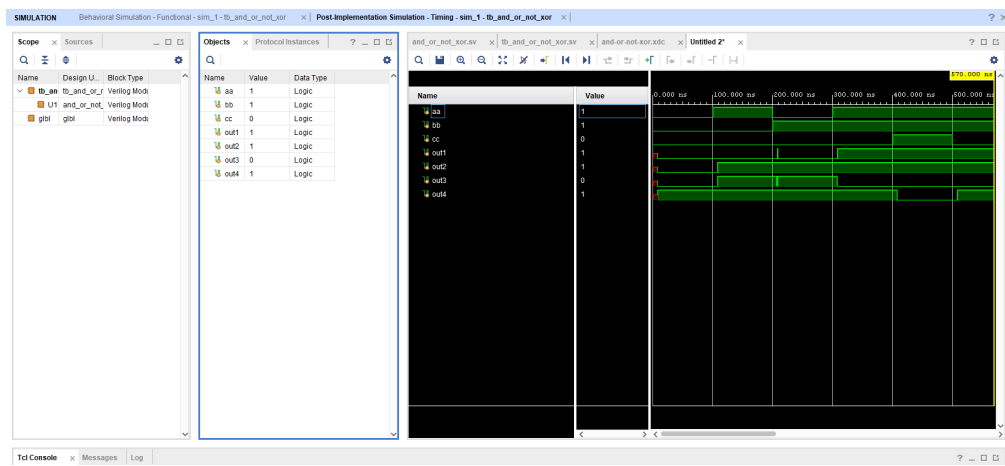


Figure 13: Test Bench Post Implementation Timing Simulation

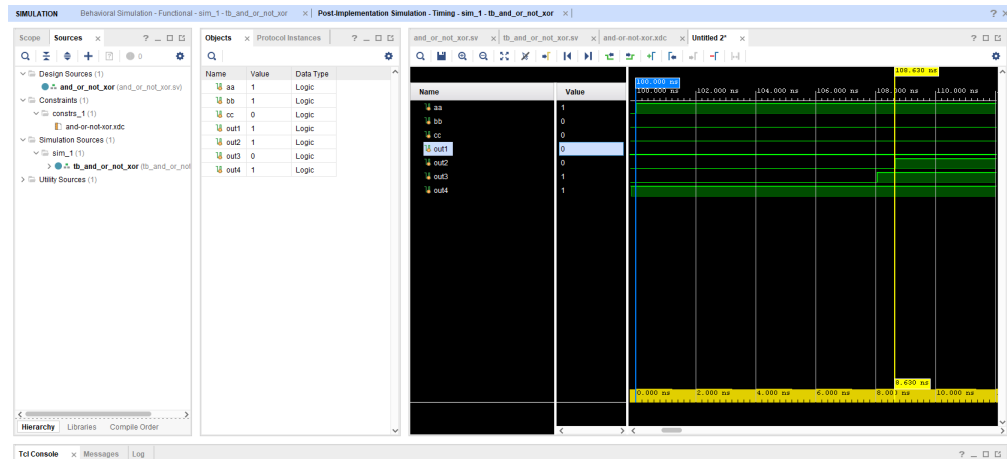


Figure 14: Test Bench Post Implementation Timing Simulation Propagation Delay

6.1 Results and Analysis Discussion

Creating the test workbench file and running the simulation was straightforward. Initially, we encountered an issue with unmatched brackets in the test bench file, but we quickly resolved it. We also observed the minor "glitch" mentioned in the rubric, which resulted from the absence of a delay when changing *aa* to 0 and *bb* to 1, as seen in Fig 13. Following the rubric's suggestion, we added the delay, and the source files reflect these changes.

Furthermore, we were unsure if the test bench waveform was expected to match precisely with what was presented in the rubric. Nonetheless, we obtained a remarkably similar output, with a slight discrepancy of approximately 0.002 ns. We were unable to set the marker exactly at 108.632 ns, but we placed it as close as possible at 108.630 ns. Ultimately, we concluded that this minor difference was not significant.

7 Results and Conclusion

This section discusses the results of the Vivado simulation experiments. The discussion provides in-depth explanations of the results and any discrepancies between the results and theoretical expectations. It also explores potential sources of error and discusses the accuracy of the tests, the debugging process, and what was learned from these experiences.

As per the rubric, the objectives we were expected to achieve were:

- a. Be able to describe a design flow for digital systems.

- b. How to use Vivado Design Studio tools to enter a design and synthesize our implementation for a particular FPGA.
- c. How to use ISim simulator to create stimulus signals and verify the functional and timing behavior of a digital system.

Our team successfully achieved all three objectives by working through the design flow. We were able to use Vivado and SystemVerilog. Additionally, to validate the various implemented gates, we created and ran a test bench simulation, confirming the accuracy of our wired circuits. The straightforward process of using Vivado and SystemVerilog resulted in minimal issues. As we worked on the project, we gained familiarity with the user interface, and just like the process of learning MultiSIM, we hope to gain a more comprehensive understanding of the software the more we use it.

8 Group Contributions

The work for this simulation assignment was spread among three individuals, with Andre Price and Benyamain Yacoob working on using Vivado software and writing with SystemVerilog to wire the circuits. Ara Oladipo helped with debugging and worked on completing the report, collaborating with Andre and Benyamain to utilize the screenshots gathered from Vivado. Ara also wrote the discussions and analysis, with Benyamain and Andre proofreading the report before submitting it. Any questions that arose during any part of this report process were answered through team collaboration and research.