**Combinational Circuit Design**

**CENG 3151**

Submitted by

Michael Lanford

1816337

Computer Engineering

University of Houston-Clear Lake

Houston, Texas 77058

January 30, 2023

# Table of Contents

**1** [**Abstract 3**](#_Toc126060766)

[**2 Introduction 3**](#_Toc126060767)

**3 Requirements……………………………………………………………………….……......3**

**4 Prelab………………………………………………………………………………..……….4**

**5 Implementation……………………………………………………….…………………......5**

5.1 Design Code / Design Diagrams**……………………………………………………………5**

5.2 Schematics**……………………………………………………………………………….....6**

5.3 Testbench**…………………………………………………………………………………....7**

5.4 Waveform / Results**…………………………………………………………………………9**

[6 Conclusion **10**](#_Toc126060768)

**Abstract**

# A combinational circuit is a circuit where the output depends solely on its inputs. In this lab, we will be using Xilinx Vivado to build a combinational circuit that will accept an input of 3-bits and output a 5-bit value while satisfying the equation y = 3x + 9. The major results of this experiment will be a waveform that shows the correct output for each input, which will reflect our truth table in the prelab.

# Introduction

For this project, we will be using Xilinx Vivado to build and test a combinational circuit that will accept some input and produce some output.

1. **Requirements**

Design a combinational network that can perform arithmetic and logic operations. The circuit has 3 1-bit inputs labeled X0, X1, and X2 along with 5 1-bit outputs labeled Y0, Y1, Y2, Y3, and Y4. It must be able to perform the arithmetic operation 3x+9 using these inputs and produce the correct output. The figure of this circuit can be seen below:

Diagram, schematic

Description automatically generated

**Figure 1:** Diagram for the circuit to be designed.

1. **Prelab**

For this prelab, we were required to draw the truth table, develop K-maps, and derive the equation for the output of the above circuit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X0 | X1 | X2 | Y0 | Y1 | Y2 | Y3 | Y4 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Table 1:** Truth Table for the combinational circuit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AB C | 00 | 01 | 11 | 10 |  |
| 0 | 0 | 0 | 1 | 1 | Y0 = X0 + X1X2 | |
| 1 | 0 | 1 | 1 | 1 |  |

**Table 2:** K-Map for Y0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AB C | 00 | 01 | 11 | 10 |  |
| 0 | 1 | 1 | 1 | 0 | Y1 = X0’X1’ + X1X2’ + X0X2 |
| 1 | 1 | 0 | 1 | 1 |  |

**Table 3:** K-Map for Y1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AB C | 00 | 01 | 11 | 10 |  |
| 0 | 0 | 1 | 0 | 1 | Y2 = X0’X1’X2 + X0’X1X2’ + X0X1X2 + X0X1’X2’ |
| 1 | 1 | 0 | 1 | 0 |  |

**Table 4:** K-Map for Y2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AB C | 00 | 01 | 11 | 10 |  |
| 0 | 0 | 1 | 1 | 0 | Y3 = X1 |
| 1 | 0 | 1 | 1 | 0 |  |

**Table 5:** K-Map for Y3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AB C | 00 | 01 | 11 | 10 |  |
| 0 | 1 | 1 | 1 | 1 | Y4 = X2’ |
| 1 | 0 | 0 | 0 | 0 |  |

**Table 6:** K-Map for Y4

1. **Implementation**

The first step of implementation was to create a new project in Xilinx Vivado. After setting up the project with the correct options, we added a design source file and added the necessary inputs and outputs to it. We then coded the equations for Y0, Y1, Y2, Y3, and Y4 into the file and then created a simulation file. We then added in the component instantiation, interface signal declarations, instance port map, and the test cases to the simulation file and tested the waveform for it.

**4.1 Design Code / Design Diagrams**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab1Design is

Port ( X0 : in STD\_LOGIC;

X1 : in STD\_LOGIC;

X2 : in STD\_LOGIC;

Y0 : out STD\_LOGIC;

Y1 : out STD\_LOGIC;

Y2 : out STD\_LOGIC;

Y3 : out STD\_LOGIC;

Y4 : out STD\_LOGIC);

end Lab1Design;

architecture Behavioral of Lab1Design is

begin

--Taken via Kmap from prelab

<= X0 OR (X1 AND X2);

Y1 <= ((NOT X0) AND (NOT X1)) OR (X1 AND (NOT X2)) OR (X0 AND X2);

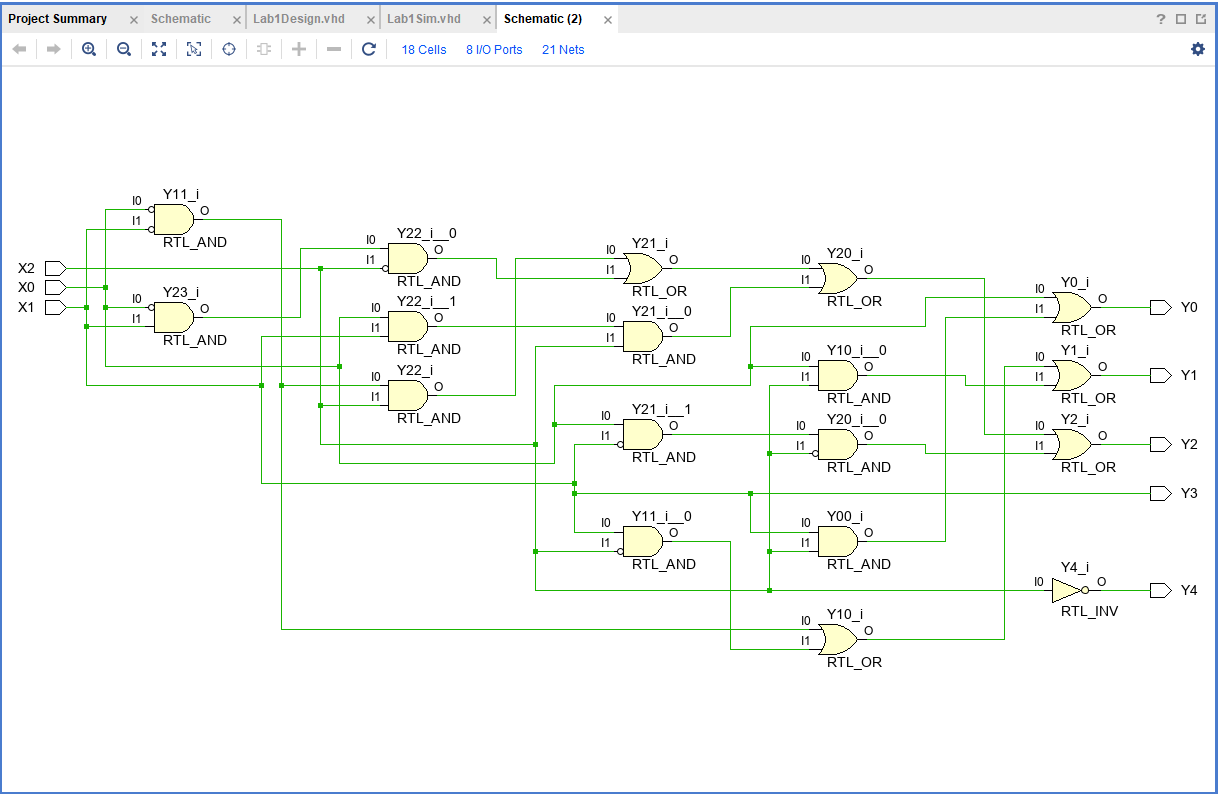
Y2 <= ((NOT X0) AND (NOT X1) AND X2) OR ((NOT X0) AND X1 AND (NOT X2)) OR (X0 AND X1 AND X2) OR (X0 AND (NOT X1) AND (NOT X2));

Y3 <= X1;

Y4 <= (NOT X2);

end Behavioral;

**4.2 Schematics**



**Figure 2:** Combinational Circuit Diagram

**4.3 Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab1Sim is

end Lab1Sim;

architecture Behavioral of Lab1Sim is

component Lab1Design is

--Instantiate test component

Port ( X0 : in STD\_LOGIC;

X1 : in STD\_LOGIC;

X2 : in STD\_LOGIC;

Y0 : out STD\_LOGIC;

Y1 : out STD\_LOGIC;

Y2 : out STD\_LOGIC;

Y3 : out STD\_LOGIC;

Y4 : out STD\_LOGIC);

end component;

--End component instantiation

--Declare Interface Signals

signal X0, X1, X2: std\_logic;

signal Y0, Y1, Y2, Y3, Y4: std\_logic;

--End Signal Declaration

begin

--Mapping port for instance

uut: Lab1Design PORT MAP (X0, X1, X2, Y0, Y1, Y2, Y3, Y4);

--End port mapping

process

begin

--Begin Truth Table Declaration

--Case 1 expect output 9

X0 <= '0';

X1 <= '0';

X2 <= '0';

wait for 10 ns;

--Case 2 expect output 12

X0 <= '0';

X1 <= '0';

X2 <= '1';

wait for 10 ns;

--Case 3 expect output 15

X0 <= '0';

X1 <= '1';

X2 <= '0';

wait for 10 ns;

--Case 4 expect output 18

X0 <= '0';

X1 <= '1';

X2 <= '1';

wait for 10 ns;

--Case 5 expect output 21

X0 <= '1';

X1 <= '0';

X2 <= '0';

wait for 10 ns;

--Case 6 expect output 24

X0 <= '1';

X1 <= '0';

X2 <= '1';

wait for 10 ns;

--Case 7 expect output 27

X0 <= '1';

X1 <= '1';

X2 <= '0';

wait for 10 ns;

--Case 8 expect output 30

X0 <= '1';

X1 <= '1';

X2 <= '1';

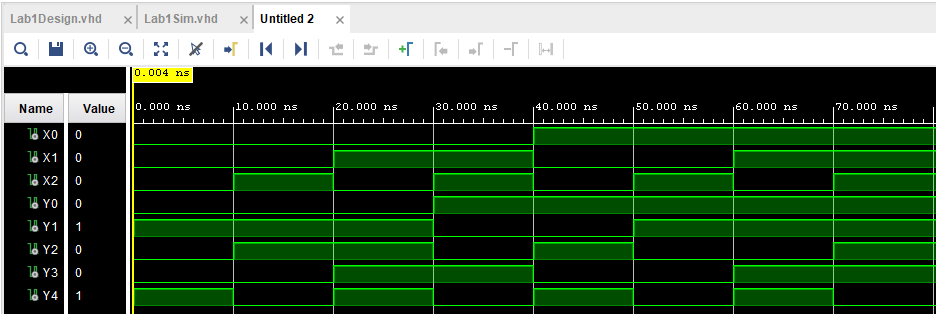
wait;

end process;

end Behavioral;

**4.4 Waveform / Results**

The waveform below shows that the code we made above in the Testbench and Design Code / Design Diagrams sections was able to produce correct results for each of our inputs that we found in the truth table. The input 000 produced 01001, 001 produced 01100, 010 produced 01111, 011 produced 10010, 100 produced 10101, 101 produced 11000, 110 produced 11011, and 111 produced 11110.



**Figure 3:** Combinational Circuit Waveform

# Conclusion

This experiment was a good introduction to the computer architecture lab and was a great refresher on how to code in Xilinx Vivado. Through our code, we were able to successfully make a combinational circuit that took a 3-bit input and produced a 5-bit output that mirrored our truth table values and was further supported in the waveform from our simulation.

# 