

COMP ENG 2DI4

Digital Logic

Lab 1: Logic Design

Due date: Friday, October 1st, 2021

By:

Jil Shah

Shiv Thakar

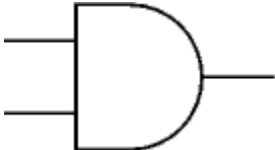
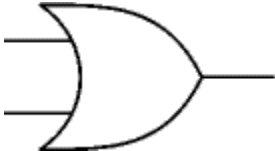
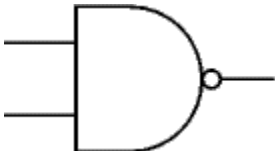
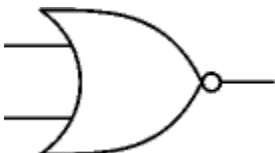
As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

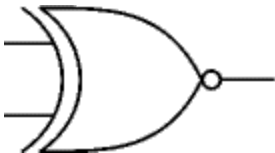
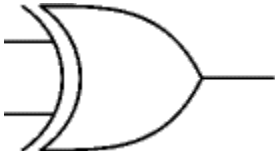
Submitted by: **Jil Shah, Shiv Thakar**

Part 1.4.1: Pre-Laboratory Preparation

1.

Table 1: Logic Operation Table

Logic Operation	Truth Table	Logic Symbol	Integrated Circuit Name																
AND	<table><tr><th>x</th><th>y</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	z	0	0	0	0	1	0	1	0	0	1	1	1		TTL	CMOS
	x	y	z																
	0	0	0																
	0	1	0																
	1	0	0																
1	1	1																	
	7408	4081																	
OR	<table><tr><th>x</th><th>y</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	z	0	0	0	0	1	1	1	0	1	1	1	1		7432	4071
	x	y	z																
	0	0	0																
	0	1	1																
	1	0	1																
1	1	1																	
NAND	<table><tr><th>x</th><th>y</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	z	0	0	1	0	1	1	1	0	1	1	1	0		7400	4011
	x	y	z																
	0	0	1																
	0	1	1																
	1	0	1																
1	1	0																	
NOR	<table><tr><th>x</th><th>y</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	z	0	0	1	0	1	0	1	0	0	1	1	0		7402	4001
	x	y	z																
	0	0	1																
	0	1	0																
	1	0	0																
1	1	0																	

XNOR	<table><tr><th>x</th><th>y</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	z	0	0	1	0	1	0	1	0	0	1	1	1		747266 (chip 74LS86)	4077
x	y	z																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	
XOR	<table><tr><th>x</th><th>y</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	z	0	0	0	0	1	1	1	0	1	1	1	0		7486	4030
x	y	z																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	

2. In the context of digital logic, logical equivalence means when two or more statements/circuits can be substituted for each other. This is only possible when they have identical truth tables. For example, the two statements below are proved to be equivalent using Postulates and Theorems.

$$F(x, y, z) = (x + 0)x$$

$$F(x, y, z) = x \cdot x, \text{ using Postulate 2 (a)}$$

$$F(x, y, z) = x, \text{ using Theorem 1 (b)}$$

$$G(x, y, z) = xyz' + xyz + x$$

$$G(x, y, z) = x(yz' + yz + 1), \text{ using Postulate 4 (a)}$$

$$G(x, y, z) = x(y(z' + z) + 1), \text{ using Postulate 4 (a)}$$

$$G(x, y, z) = x(y + 1), \text{ using Postulate 5 (a)}$$

$$G(x, y, z) = x, \text{ using Theorem 2 (a)}$$

Also, can be seen by the following boolean examples:

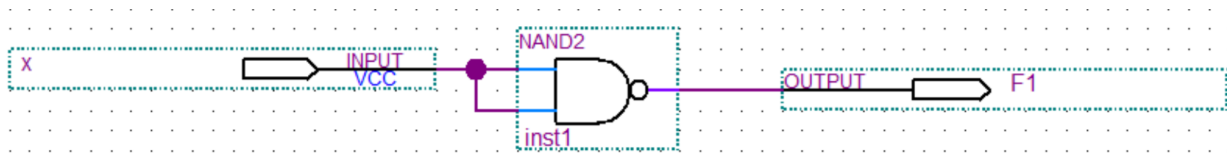
$F1 = (A' * B' + AB)'$			$F2 = (A' * B')' * B$		
A	B	F1	A	B	F2
0	0	FALSE	0	0	FALSE
0	1	TRUE	0	1	TRUE
1	0	FALSE	1	0	FALSE
1	1	TRUE	1	1	TRUE

Part 1.4.1: Quartus II Simple Circuit Simulation

Simulation Output



Circuit



Part 1.4.2: Building Circuits, Expressions, and Truth Tables

1.

a)

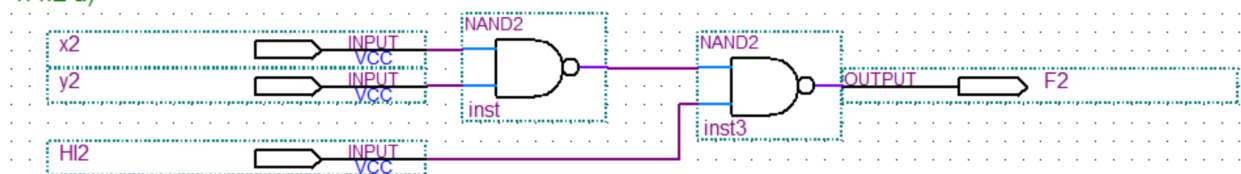
i) Truth Table:

Table 2: Logic Circuit 2 Truth Table

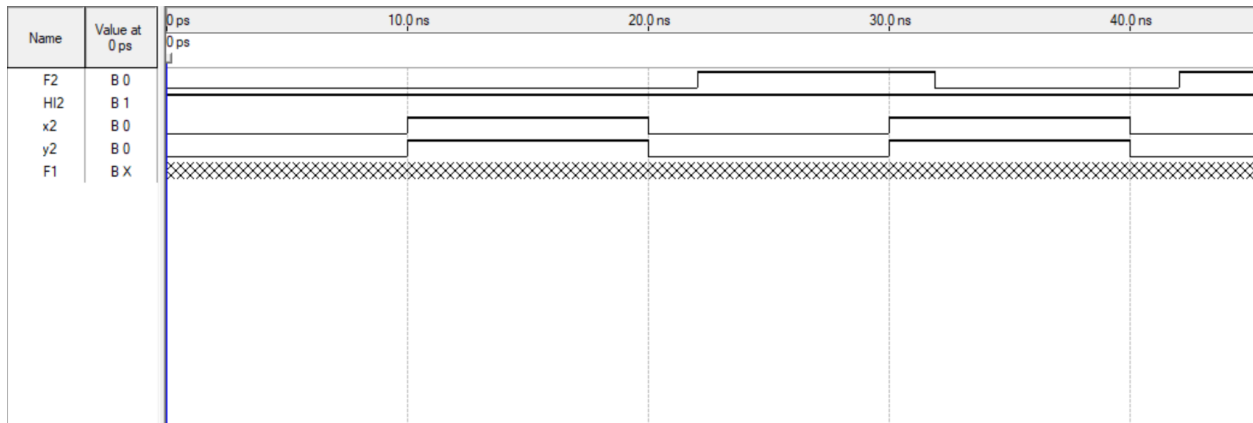
X	Y	HI	F2
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

ii) Circuit:

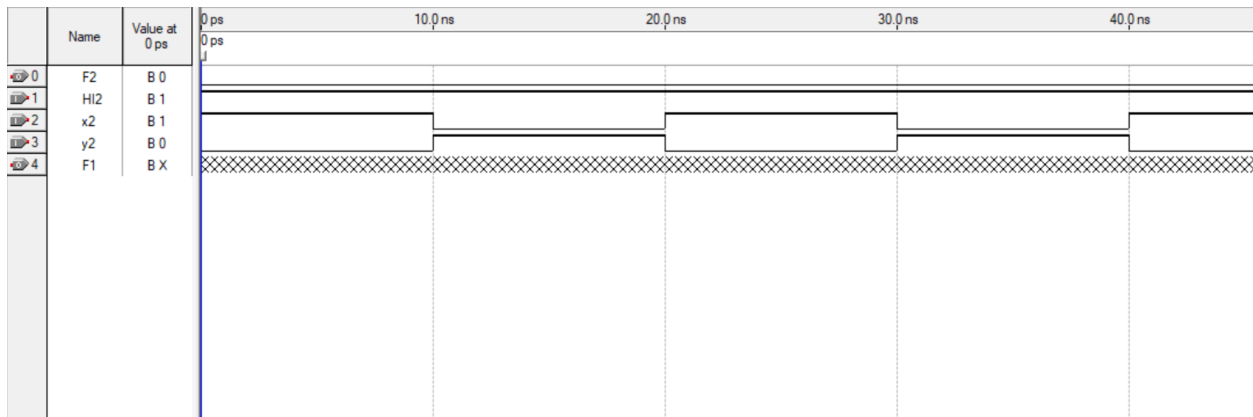
1.4.2 a)



iii) Simulation Result:



The simulation above was testing the output F2 when the inputs X and Y are low/high at the same time.



The simulation above was testing the output F2 when the inputs X and Y are low/high at different times.

iv) Brief Description:

We used the simulation tool and tested for different combinations of the inputs. We set the fixed input value and then checked if the output was equivalent with the logical output that was determined in the truth table. As shown in the example above, we can notice that the output is 0 when both x and y inputs are low, and when the inputs x is 1 and y is 0, we receive an output of 0.

v) Logical Operation:

$$F2 = ((X * Y)' * HI)'$$

F2 is equivalent to an AND gate

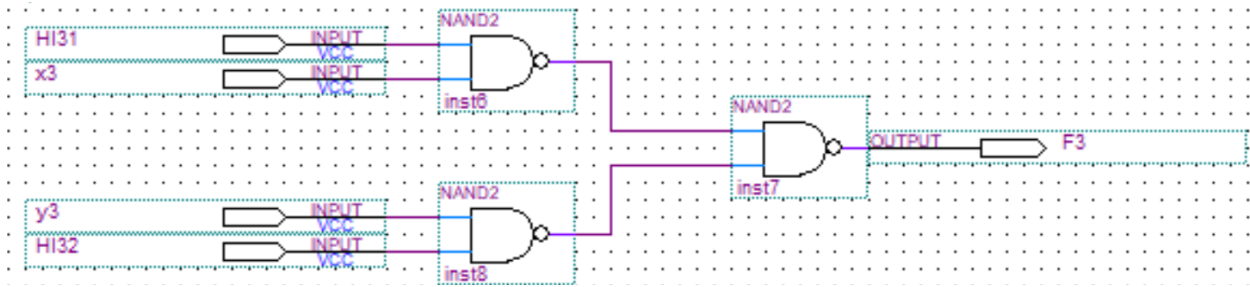
b)

i) Truth Table:

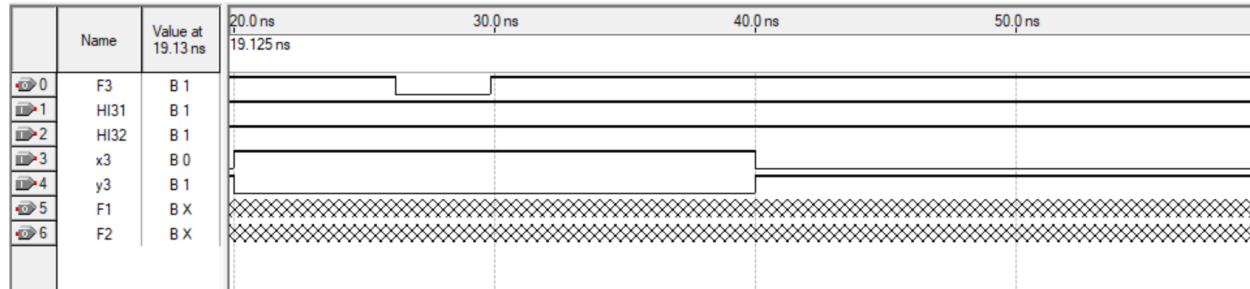
Table 3: Logic Circuit 3 Truth Table

X	Y	HI	F3
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

ii) Circuit:



iii) Simulation Result:



iv) Brief Description:

We tested some unique combinations of the inputs from the truth table and checked if the simulation results matched with the logical output. The result we received for F3 was 0 when we tested x is equal to 0 and y is equal to 1.

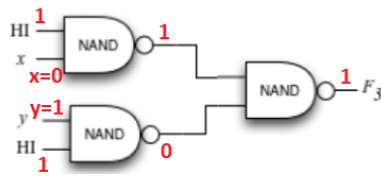


Figure 1.3: Logic Circuit 3

v) Logical Operation:

$$F3 = ((X * HI)' * (Y * HI))'$$

F3 is logically equivalent to an OR gate.

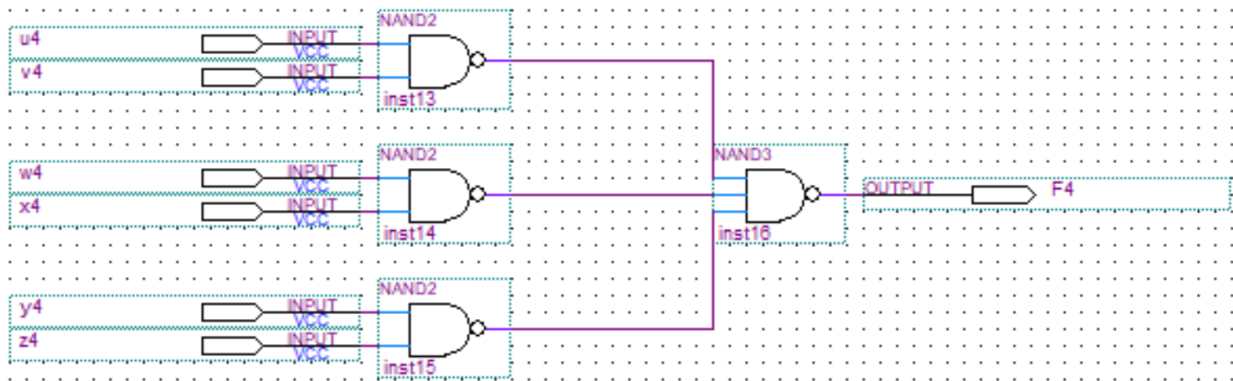
c)

i) Truth Table:

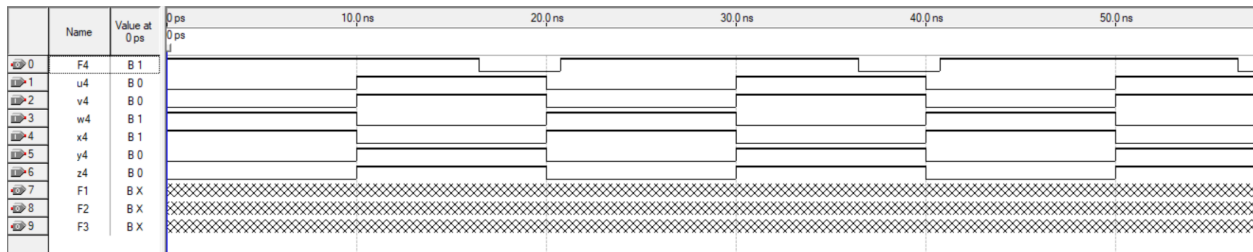
Table 4: Logic Circuit 4 Truth Table

u	v	w	x	y	z	F4
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	1
0	0	0	1	0	0	0
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	0	1	0
0	0	1	0	1	0	0
0	0	1	0	1	1	1
0	0	1	1	0	0	1
0	0	1	1	0	1	1
0	0	1	1	1	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	1	0	0
0	1	0	0	1	1	1
0	1	0	1	0	0	1
0	1	0	1	0	1	1
0	1	0	1	1	0	0
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	0
0	1	1	0	1	0	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
0	1	1	1	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	0	0	1	0
1	0	0	0	1	0	0
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	0	1	0	1	0
1	0	0	1	1	0	0
1	0	0	1	1	1	1
1	0	1	0	0	0	0
1	0	1	0	0	1	0
1	0	1	0	1	0	0
1	0	1	0	1	1	1
1	0	1	1	0	0	0
1	0	1	1	0	1	0
1	0	1	1	1	0	0
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	0	0	1	0
1	1	0	0	1	0	0
1	1	0	0	1	1	1
1	1	0	1	0	0	0
1	1	0	1	0	1	0
1	1	0	1	1	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	0	0	1	0
1	1	1	0	1	0	0
1	1	1	0	1	1	1
1	1	1	1	0	0	0
1	1	1	1	0	1	0
1	1	1	1	1	0	0
1	1	1	1	1	1	1

ii) Circuit:

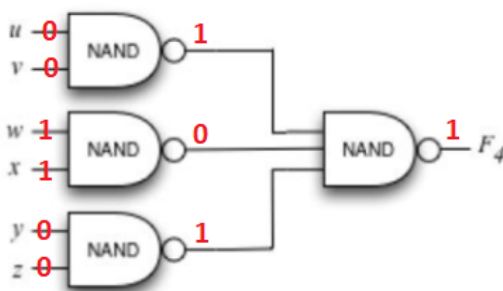


iii) Simulation Result:



iv) Brief Description:

We tested some unique combinations of the inputs from the truth table and checked if the simulation results matched with the logical output. The result we received for F4 was 1 when we tested w and x is 1 and the rest as 0.



v) Logical Operation:

$$F4 = ((u * v)' * (w * x)' * (y * z)')'$$

$$F4 \text{ (simplified)} = (a * b * c)' = a' + b' + c' = (u * v) + (w * x) + (y * z)$$

The logical operation is a combination of AND and OR.

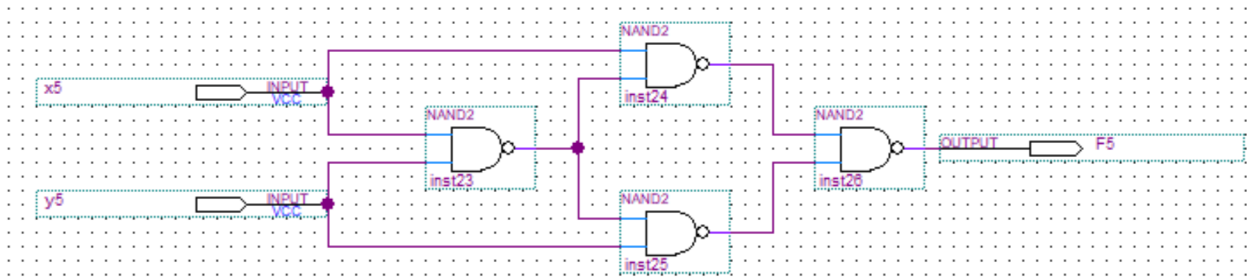
d)

i) Truth Table:

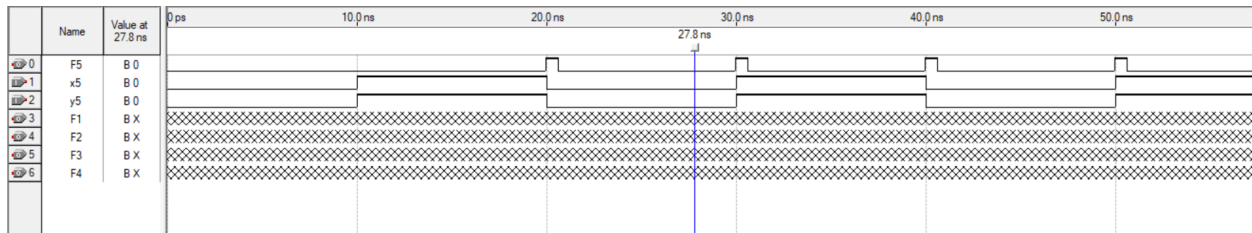
Table 5: Logic Circuit 5 Truth Table

X	Y	F5
0	0	0
0	1	1
1	0	1
1	1	0

ii) Circuit:



iii) Simulation Result:



iv) Brief Description:

We tested some unique combinations of the inputs from the truth table and checked if the simulation results matched with the logical output. The result we received for F5 was 0 when we tested x and y is 0.

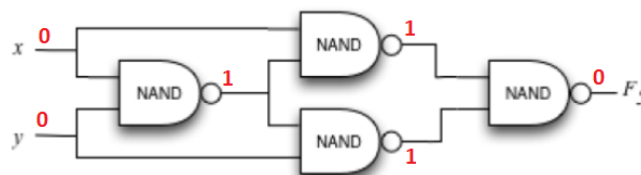


Figure 1.5: Logic Circuit 5

v) Logical Operation:

$$F4 = ((x * (x * y))' * (y * (x * y))')'$$

F4 is equivalent to an XOR gate.

Part 1.4.3 Logical Equivalence

i) Brief Description:

Table 6: Logic Circuit 6 Truth Table

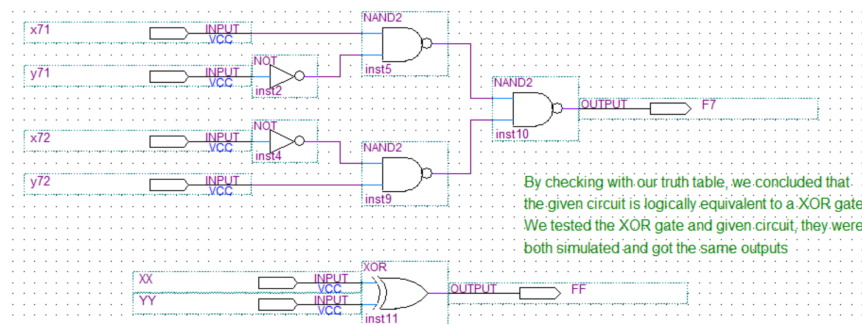
X	Y	F6
0	0	0
0	1	1
1	0	1
1	1	0

After we created a truth table for the given circuit, we concluded that the circuit is logically equivalent to an XOR gate. By definition, Logical equivalence is when two statements have identical truth tables. As seen below, both truth tables are identical, which proves that they are logically equivalent.

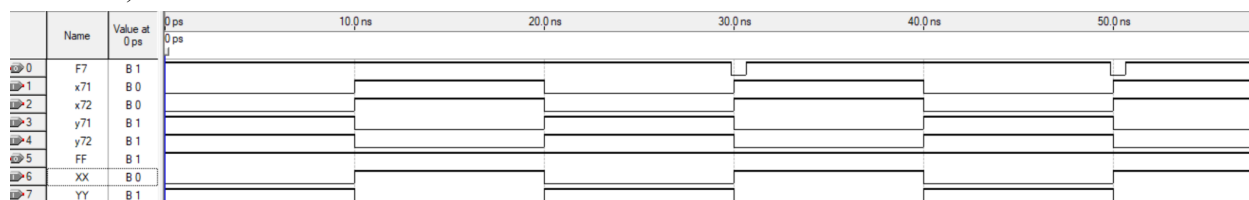
Table 7: XOR Gate Truth Table

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

ii) Circuit



iii) Simulation Result:



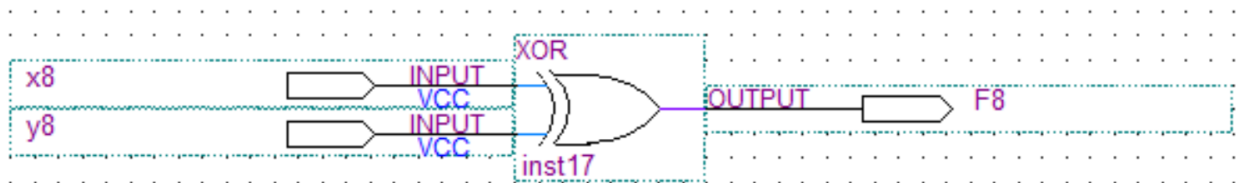
Part 1.4.4: Enable/Disable Inverter

In the context of digital logic, an inverter is a logical negation (NOT logic gate) as a result it inverts the input. An XOR gate can be simplified as $F = X'Y + XY' = X \oplus Y$, if both inputs are 0 or 1, then the output

will be zero, but for example, if X can be controlled high and low, the output of the circuit can also be controlled. When X is set to low, the output is Y (as seen below). When X is set to high, the output is inverted to Y' . Therefore an XOR gate is considered to be a “controllable inverter”.

$$\begin{aligned}
 &\text{If } X = 0 \text{ then,} \\
 F &= (0)'Y + (0)Y' \\
 &= (1)Y + (0)Y' \\
 &= Y + 0 = Y
 \end{aligned}$$

$$\begin{aligned}
 &\text{If } X = 1 \text{ then,} \\
 F &= (1)'Y + (1)Y' \\
 &= (0)Y + (1)Y' \\
 &= 0 + Y' = Y'
 \end{aligned}$$

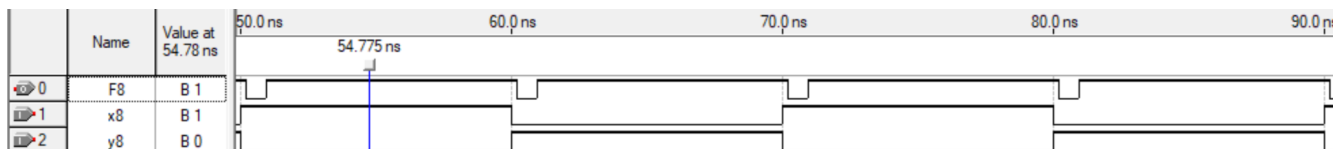


Truth Table:

Table 8: Logic Circuit 7

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Simulation:



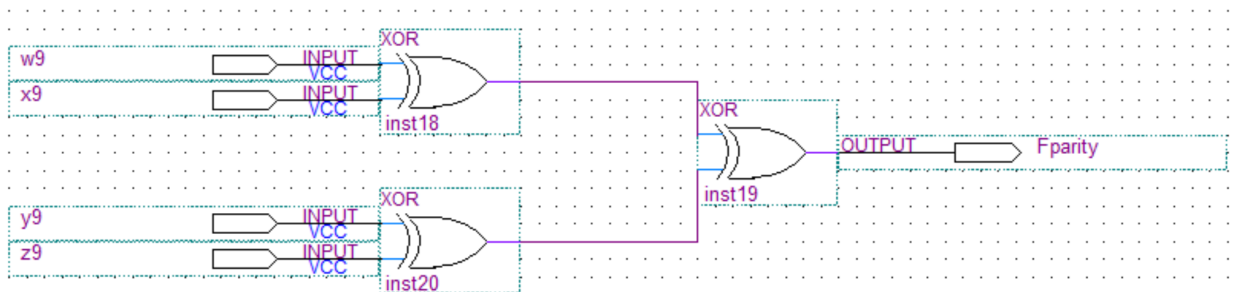
Part 1.4.5: Parity Generator

Truth Table:

Table 9: Logic Circuit 8

W	X	Y	Z	Fparity
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Circuit:



Simulation Results:

	Name	Value at 70.0 ns	0 ps	10.0 ns	20.0 ns	30.0 ns	40.0 ns
0	Fparity	B 1					
1	w9	B 1					
2	x9	B 0					
3	y9	B 1					
4	z9	B 1					

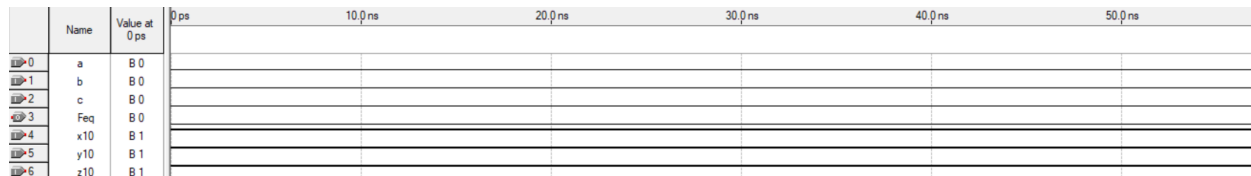
Part 1.4.6: Equality Detection

Truth Table:

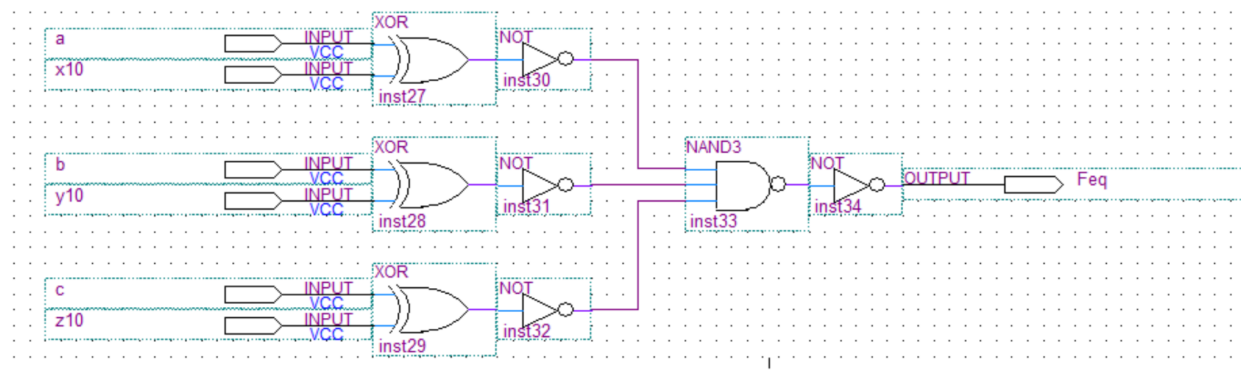
Table 10: Logic Circuit 9 Truth Table

A	B	C	X	Y	Z	Equality	A	B	C	X	Y	Z	Equality
0	0	0	0	0	0	1	0	1	1	1	1	1	0
0	0	0	0	0	1	0	0	1	1	1	1	1	0
0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	1	0	0	0	1	1	0
0	0	0	1	1	0	0	1	0	0	1	0	0	1
0	0	0	1	1	1	0	1	0	0	1	1	0	0
0	0	1	0	0	0	0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	1	0	1	0	0
0	0	1	1	0	0	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	1	0	1	1	1	0	0
0	0	1	1	1	0	0	1	0	1	1	1	0	0
0	0	1	1	1	1	0	1	0	1	1	1	0	0
0	1	0	0	0	0	0	1	1	0	0	0	0	0
0	1	0	0	0	1	0	1	1	0	0	1	0	0
0	1	0	0	1	0	1	1	1	0	0	1	0	0
0	1	0	0	1	1	0	1	1	0	0	1	0	0
0	1	0	1	0	0	0	1	1	0	1	0	0	0
0	1	0	1	0	1	0	1	1	0	1	1	0	0
0	1	0	1	1	0	0	1	1	0	1	1	0	0
0	1	0	1	1	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	0	1	1	1	0	0	0	0
0	1	1	0	0	1	0	1	1	1	0	1	0	0
0	1	1	0	1	0	0	1	1	1	0	1	0	0
0	1	1	0	1	1	0	1	1	1	0	1	0	0
0	1	1	1	0	0	0	1	1	1	1	0	0	0
0	1	1	1	0	1	1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	1	1	1	1	1	0	0
0	1	1	1	1	1	0	1	1	1	1	1	0	0
1	0	0	0	0	0	0	1	1	1	1	1	1	1

Simulation:



Circuit:



Description:

The simulation results validate the logical operation for this circuit because of how the output is equivalent to the output that is determined using the truth table and step-by-step visual representation below when we forced specific inputs into the simulation.

