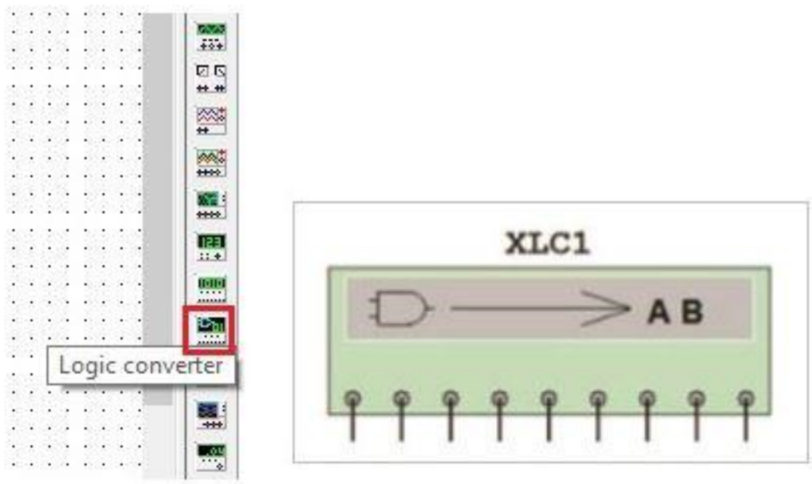
Lab Manual:
Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 3: Logic Gates Explored and Boolean Algebra

Lab 3: Logic Gates Explored and Boolean Algebra

In the previous lab, we were introduced to the two basic logic gates – AND and OR in detail. Building on these, we can create a few other types of logic gates. These are: **Inverters (NOT), NAND, NOR, XOR, and XNOR.** Let's take a look at each one in greater detail.

Learning Objectives

In this lab, students will:

1. Explore the function of various different logic gates
2. Create circuits with varying logic gates in theory and in practice.
3. Calculate and build combinational logic circuits from Sum-of-Products and Product-of-Sums derived from truth tables.

Required Tools and Technology

Platform: NI ELVIS III	✓ View User Manual: http://www.ni.com/en-us/support/model.ni-elvis-iii.html ✓ View Tutorials: https://www.youtube.com/playlist?list=PLvcPIuVaUMIWm8ziaSxv0gwtshBA2dh_M
Hardware: Digilent Digital Electronics Board for NI ELVIS III	✓ View NI Digital Electronics Board Manual: http://www.ni.com/pdf/manuals/376627b.pdf
Software: NI Multisim 14.0.1 Education Version or newer	✓ Install Multisim: http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US ✓ View Help: http://www.ni.com/multisim/technical-resources/
Software: NI LabVIEW FPGA Vivado 2014.4	✓ Install: http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/ **Note:** Digilent Driver (The installer above automatically downloads the installer below onto your computer) ✓ Navigate to: C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_drivers\nt64\digilent ✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- SOP and POS Boolean expressions
- Screenshot, picture, or sketch of circuits
- Truth Table
- Saved circuits
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

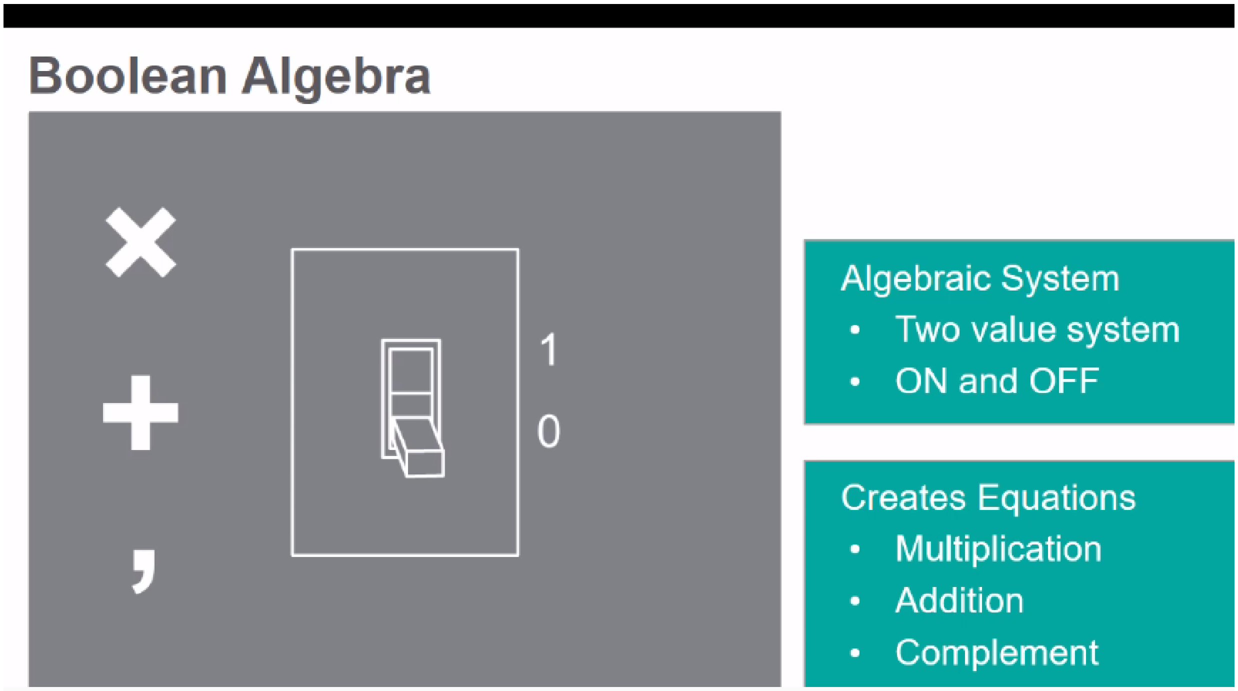


Figure 1-1 Video. View the video here: https://youtu.be/1wnztS6et0w

Video Summary

- NAND, NOR, XOR, and XNOR gates have at least two inputs, one output, and a unique truth table
- NOT gate output is the inverse of the input
- Boolean algebra is a system where two values are used to represent the properties of bi-stable electrical switching circuits, namely on and off

Inverters

- Inverters are also known as *NOT* gates.
- They have only one input and one output.
- The truth table for an inverter is simple. The output is always the *opposite* of the input.
- For example, if the input is 1, the output will be 0 and vice versa. Visually this is depicted by a circle at the input and/or output ends of the logic gates.
- In this situation, the circle is at the output, which means that the output is inverted. If it was at the input, then it is the input that would be inverted.
- Circuits with more than one input can use NAND or NOR logic gates which we will explore next.

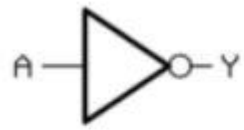


Figure 1-2 Inverter

NAND Logic Gates

- *NAND* gates invert the output of the AND gate.
- The inputs do not change from those of the AND truth table, but the output is the opposite.
- As a rule, if any of the inputs are 0, the output will always be 1.
- See below for the truth table and the symbol.

A	B	O
0	0	1
0	1	1
1	0	1
1	1	0

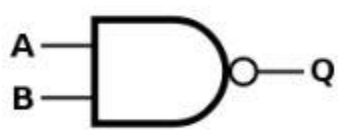


Figure 1-3 NAND gate truth table and symbol

NOR Logic Gates

- The *NOR* logic gate inverts the output of the OR gate.
- The inputs of the truth table for the OR gate do not change, but the output is the opposite.
- As a rule, if any of the inputs are 1, the output will always be 0.
- See below for the truth table and symbol.

A	B	O
0	0	1
0	1	0
1	0	0
1	1	0

Figure 1-4 NOR gate truth table and symbol

XOR Logic Gates

- An *XOR* gate is also known as an exclusive OR gate.
- The output will be 1 if only one of the inputs is 1. The output will be 0 if both inputs are 0 or both are 1.
- See below for the truth table and symbol.

A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

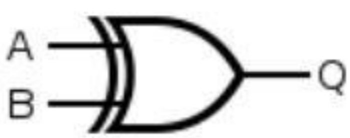


Figure 1-5 XOR gate truth table and symbol

XNOR Logic Gates

- The *XNOR* gate does the opposite of the XOR gate.
- The output will be 1 if the inputs are the same and the output will be 0 if the inputs are not the same.
- See below for the truth table.

A	B	O
0	0	1
0	1	0
1	0	0
1	1	1

Figure 1-6 XNOR gate truth table

Combinational Logic Circuits (CLCs)

CLCs are a classification of circuits whose output is only dependent on the current inputs and are implemented by Boolean circuits. Using combinations of logic gates, different results can be achieved. A truth table is often used to define the behavior of a CLC, but sometimes we start with a truth table and need to design a CLC.

Boolean Algebra

Boolean algebra is an algebraic system where two values are used to represent the properties of bistable electrical switching circuits, namely on and off, or simply 1 and 0. The rules for the two binary operators (addition and multiplication) and complement (’) for a two-valued Boolean algebraic expression are presented in the tables below.

- It can be seen that the binary addition, multiplication and complement are the same as the OR, AND and NOT logic operations.
- For the complement, several notations are used: apostrophe after the variable, exclamation mark, tilde or the word NOT before the variable or an overbar on top of it.
- Because it works with digital systems with only the values 0 and 1, the algebra used is simply called “binary logic”.
- Any logic function, no matter how complex it is, can be implemented using only the three basic logic operations.
- A function represented by a truth table can be expressed using different methods.

- Knowing the logic expression and the function, the circuit can be then realized.

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1-7 Binary Multiplication (AND logic operation)

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

Figure 1-8 Binary Addition (OR logic operation)

x	x′
0	1
1	0

Figure 1-9 Compliment (NOT logic operation)

Sum-of-Products

A simple method for converting a truth table into a CLC is found in a standard form of Boolean expression called the *Sum-of-Products (SOP).*

- An SOP expression is literally a sum of Boolean terms called *minterms*.
- A minterm is a multiplicative combination of Boolean variables whose output equals 1.
- An example of an SOP expression is ABC + BC + DE, where ABC, BC, and DE are minterms.
- SOP expressions may be generated from truth tables using the following steps:
    1. Determine which rows of the table have an output of 1.
    2. Derive each row's minterm, such that the output is 1 given that row's input state.
    3. Sum the minterms.

Below is an example of a truth table conversion to an SOP expression.

A	B	C	O	
0	0	0	0	
0	0	1	1	$A'B'C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$AB'C'$
1	0	1	1	$AB'C$
1	1	0	1	ABC'
1	1	1	0	

$$O = A'B'C + AB'C' + AB'C + ABC'$$

Figure 1-10 SOP truth table

Product-of-Sums

Product-of-Sums (POS) expressions are another way of representing truth tables.

- A POS expression is a product of Boolean terms called *maxterms*.
- A maxterm is a summation of Boolean variables whose output equals 0.
- To generate a POS expression from a truth table, perform the following steps:
    1. Determine which rows of the table have an output of 0.
    2. Derive each row's maxterm, such that the output is 0 given that row's input state.
    3. Multiply the maxterms.

A	B	C	O	
0	0	0	0	$A + B + C$
0	0	1	1	
0	1	0	0	$A + B' + C$
0	1	1	0	$A + B' + C'$
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	0	$A' + B' + C'$

$$O = (A + B + C)(A + B' + C)(A + B' + C')(A' + B' + C')$$

Figure 1-11 POS truth table

The SOP and POS standard Boolean forms are powerful tools when applied to truth tables. They can be used to derive a Boolean expression—and ultimately, an actual logic circuit. When creating a circuit from SOPs, it would be constructed of AND gates feeding into an OR gate. When creating a circuit from POSs, it would be constructed of OR gates feeding into an AND gate.

Check Your Understanding

Note: The following questions are meant to help you self-assess your understanding so far. You can view the answer key for all "Check your Understanding" questions at the end of the lab.

1-1 Write the SOP and POS Boolean expression derived from the truth tables of the two circuits (O_1 and O_2).

A	B	C	O_1	O_2
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1

Figure 1-12 Truth table for two circuits

1.2 Simulate: Building a CLC Circuit - Example 1

Circuit 1

Using the SOP for the first circuit (O_1), build a circuit which will implement the truth table defined.

- Place as many **OR gates** and **AND gates** as needed from the **Misc Digital** group.
- Place three **INTERACTIVE_DIGITAL_CONSTANTs** from the **Sources** group.
- Place one **PROBE_DIG_RED** from the **Indicators** group.
- Wire them together, as necessary.

Note: Take a picture or draw a sketch of your circuit and attach it to your completed lab.

- Click the **Run** button to begin simulating the circuit.

Figure 1-13 Run button

- Using the **A**, **B**, and **C** keys, vary the inputs into the circuit.

1-2 Record the results, as indicated by the probe, in the following truth table.

A	B	C	O
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- When you're done, stop the simulation by clicking the **Stop** button.

Figure 1-14 Stop button

Note: Save your circuit as **SOP-1.ms14**.

1-3 Does the behavior of your circuit match the truth table from question 1-1? If not, what might be wrong?

1.3 Simulate: Building a CLC Circuit - Example 2

Circuit 2

Create a new circuit design.

- Using the POS for the second circuit of question 1-1 (O_2), build a circuit which will implement the truth table defined.
- Place as many **OR gates** and **AND gates** as needed from the **Misc Digital** group.
- Place three **INTERACTIVE_DIGITAL_CONSTANTs** from the **Sources** group.
- Place one **PROBE_DIG_RED** from the **Indicators** group.
- Wire them together, as necessary.

Note: Take a screenshot, take a picture, or draw a sketch of your circuit and include it with your completed lab.

- Click the **Run** button to begin simulating the circuit.

Figure 1-15 Run button

- Using the **A**, **B**, and **C** keys, vary the inputs into the circuit.

1-4 Record the results, as indicated by the probe, in the following truth table given to define the behavior of this circuit.

A	B	C	O
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- When you're done, stop the simulation by clicking the **Stop** button.

Figure 1-16 Stop button

Note: Save your circuit as **POS-2.ms14**.

1-5 Does the behavior of your circuit match the truth table from question 1-1? If not, what might be wrong?

1.4 Exercise: Verify SOP and POS Expressions Using a Logic Converter

Logic Converter Circuit 1

The *Logic Converter* is a great tool for checking truth tables and logic expressions. To build a Logic Converter circuit:

- Place the **Logic Converter** from the instruments toolbar on the right screen onto the circuit.
- Double click the **Logic Converter** to open its user interface.
- For the first circuit (O_1), enter the SOP expression that you calculated in the text field at the bottom of the window.
- Click the fourth button, **Expression to Truth Table**.

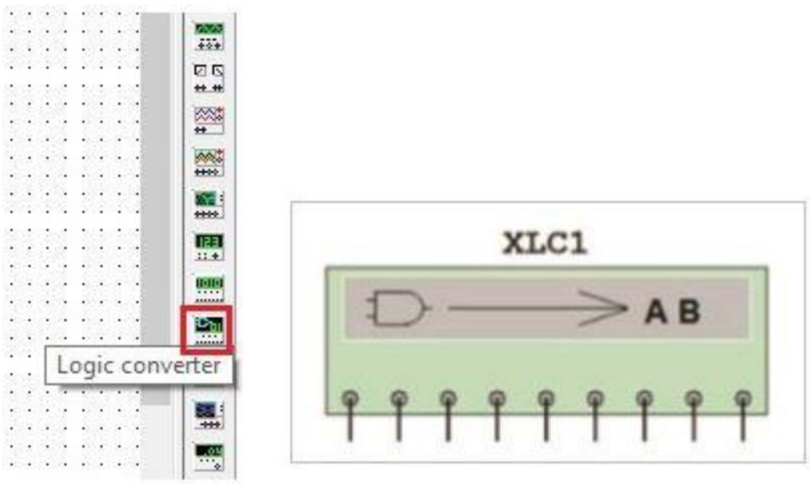


Figure 1-17 Logic converter

1-6 Does the truth table generated match the truth table in question 1-1? If so, is this confirmation that you POS expression is correct?

A. Yes
B. No

Logic Converter Circuit 2

For the second circuit (O_2):

- Enter the POS expression that you calculated in the text field at the bottom of the window.
- Click the fourth button, **Expression to Truth Table**.

1-7 Does the truth table generated match the truth table in question 1-1? If so, is this confirmation that you POS expression is correct?

A. Yes
B. No

Logic Converter Functions

The Logic Converter can also generate circuits from POS and SOP expressions. This can save some time from doing the work manually.

- Click the third button, **Truth Table to Simplified Expression**. This will simplify the expression if it can be simplified.
- Next, click the fifth button, **Expression to Circuit**.
  - Place the circuit that it generates.
  - Take a screenshot, take a picture, or draw a sketch of the circuit and include it with your completed lab.

1-8 Does the circuit generated match the circuit that you built at the beginning of this lab? If not, why might it be different?

1.5 Exercise: Building an XOR Logic Gate in Multisim

XOR Gate Circuit

Build the following circuit using an XOR gate:

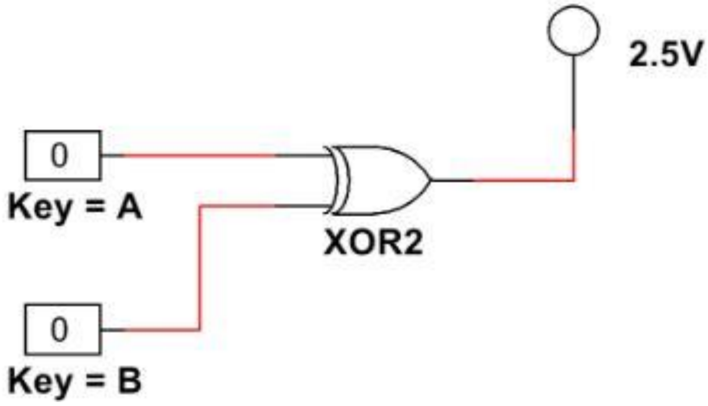


Figure 1-18 XOR gate circuit

Configure the Digital Constants:

- Double-click the top **Digital Constant**.
- In the window that appears, select '**A**' from the Key for toggle dropdown.
- Change the second constant to toggle with the '**B**' key.
- Click the **Run** to begin simulating the circuit.

Figure 1-19 Run button

- Press the '**A**' key on the keyboard to change the value of that input to **1**.

1-9 Does the probe turn on?

A. Yes
B. No

- Press the ‘**A**’ key again to change the top input back to **0**.
- Press the ‘**B**’ key to change the second input to **1**.

1-10 Does the probe turn on?

A. Yes
B. No

- Press the ‘**A**’ key, so that both inputs are equal to **1**.

1-11 Does the probe turn on?

A. Yes
B. No

1-12 How would you describe the behavior of this gate?

____

____

____

____

- When you’re done, stop the simulation by clicking the **Stop** button.

Figure 1-20 Stop button

1.6 Exercise: Building a NOR Logic Gate on the Digital Electronics Board

PLD Design

Create a new PLD design.

- Select **File>>New**.
- In the window that appears, select the **PLD design** and click **Create**.
- Use the standard configuration for the NI Digital Electronics Board.
- In the second step, name the PLD Design ‘**NOR Gate**’, and click **Next**.
- In the third step, leave the default settings and click **Finish**.

Using the switches, LEDs, and logic gates, create the following circuit:

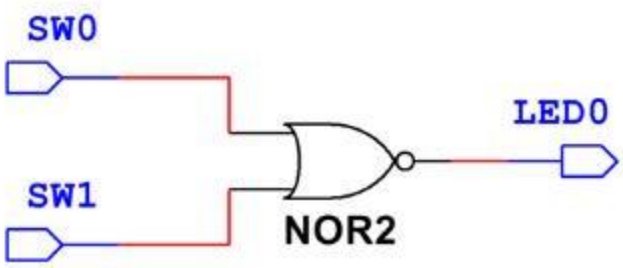


Figure 1-21 PLD design

Deploy the Board

To run this circuit, we need to deploy it to the board.

- Select **Transfer>>Export to PLD**.
- In the window that appears, leave the default settings, as seen, and click **Next**.
- Configure the second step:
  - Choose the **Xilinx Vivado Design Suite** tool.
  - Select a location for the file to export.
  - Select the **XC7Z020**.
  - Click the **Refresh** button to see if your device is detected.
  - Click the **Finish** button to deploy to the Digital Electronics Board.

Note: The Digital Electronics Board is now programmed and the switches **SW0** and **SW1** can be toggled. Depending on their values, **LED0** turns on or off.

1-13 Vary the outputs into the gate to complete the truth table below.

SW1	SW0	O
0	0	
0	1	
1	0	
1	1	

1-14 Does the truth table match the NOR truth table from the *Theory and Background* section?

____

____

1.7 Conclusion

1-15 What is the difference between SOP and POS? Consider truth tables and logic gates.

1-16 Can more than one circuit design produce the same truth table behavior? If so, give an example.

1-17 Why can some SOP and POS expressions be reduced to simpler terms?

1-18 What is a combinational logic circuit (CLC)?

A. A circuit that consists of two or more logic gates
B. A circuit that uses Boolean algebra and depends only on the current input
C. A circuit that can only be run in Multisim
D. A circuit that has two or more inputs

1-19 Minterms are used to determine the following:

A. Truth tables
B. Sum of Products
C. Product of Sums
D. The number of NAND gates in a circuit

1-20 When creating a circuit from the Product of Sums, it would be constructed of:

A. NOR gates feeding into a NAND gate
B. NAND gates feeding into a NOR gate
C. OR gates feeding into an AND gate
D. AND gates feeding into an OR gate

1-21 In Multisim, logic gate components are found in the _________ family within the Misc. Digital group

A. TIL
B. Microcontrollers
C. Memory
D. Line driver

1-22 What are interactive digital constants replaced with when a circuit is created in PLD?

A. LEDs
B. FPGAs
C. Logic gates
D. Switches

Answer Key – Check Your Understanding Questions Only

Check Your Understanding

1-1

<u>O1 – SOP:</u> A'B'C + A'BC' + ABC'

<u>O1 – POS:</u> (A + B + C)(A + B' + C')(A' + B + C)(A' + B + C')(A' + B' + C')

<u>O2 – SOP:</u> A'B'C + A'BC' + A'BC + AB'C + ABC' + ABC

<u>O2 – POS:</u> (A + B + C)(A' + B + C)