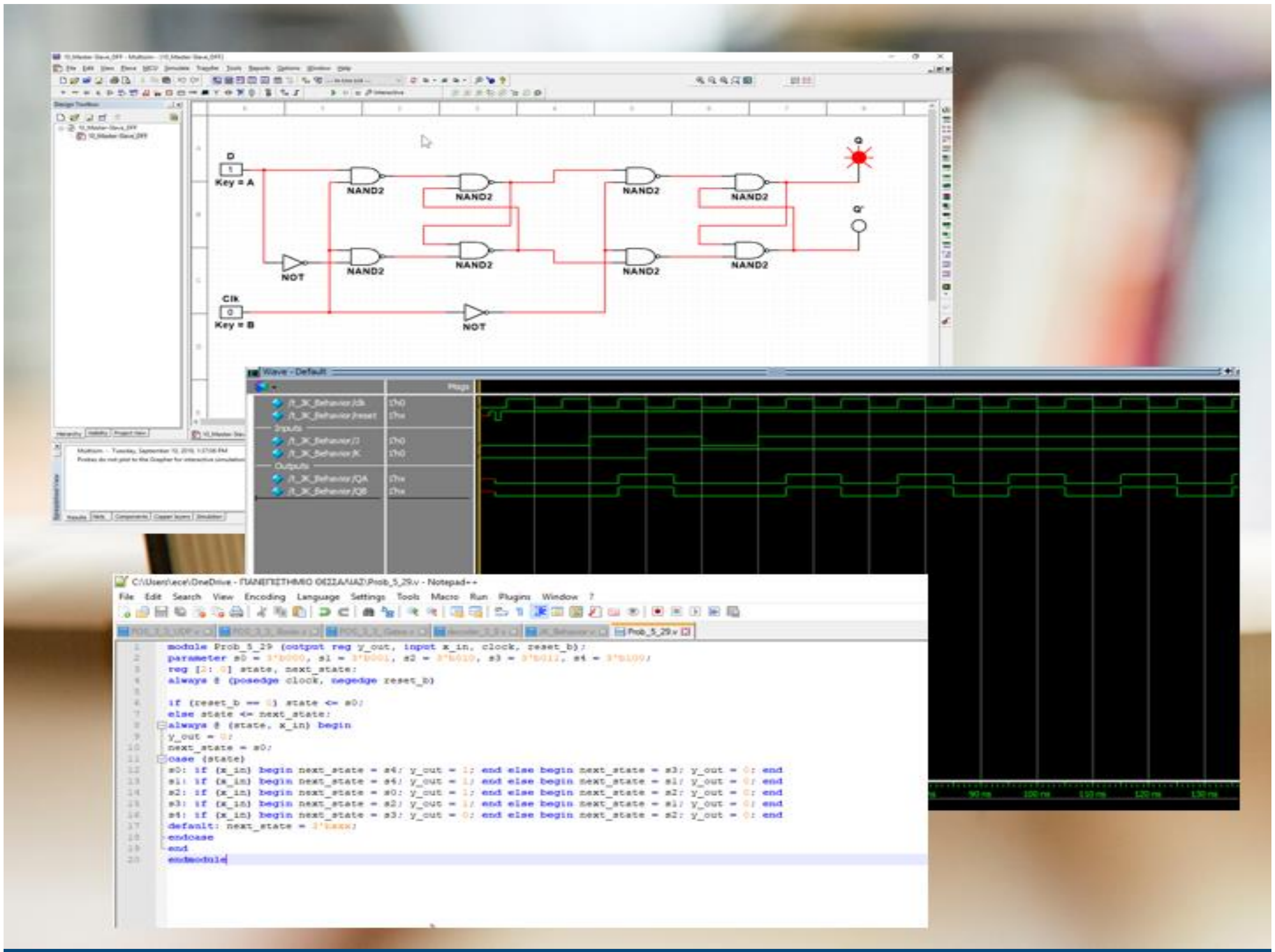
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
χρόνια δημιουργίας

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ


D
Key = A
NAND2
NOT
Clk
Key = B
Q
Q'
Wave - Default
Msgs
Inputs
Outputs
C:\Users\ece\OneDrive - ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ\Prob_5_29.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Prob_5_29.v

```
module Prob_5_29 (output reg y_out, input x_in, clock, reset_b);
parameter s0 = 3'b000, s1 = 3'b001, s2 = 3'b010, s3 = 3'b011, s4 = 3'b100;
reg [2: 0] state, next_state;
always @ (posedge clock, negedge reset_b)

if (reset_b == 0) state <= s0;
else state <= next_state;
always @ (state, x_in) begin
y_out = 0;
next_state = s0;
case (state)
s0: if (x_in) begin next_state = s4; y_out = 1; end else begin next_state = s3; y_out = 0; end
s1: if (x_in) begin next_state = s4; y_out = 1; end else begin next_state = s1; y_out = 0; end
s2: if (x_in) begin next_state = s0; y_out = 1; end else begin next_state = s2; y_out = 0; end
s3: if (x_in) begin next_state = s2; y_out = 1; end else begin next_state = s1; y_out = 0; end
s4: if (x_in) begin next_state = s3; y_out = 0; end else begin next_state = s2; y_out = 0; end
default: next_state = 3'bxxx;
endcase
end
endmodule
```

ECE119 Ψηφιακή Σχεδίαση

Εργαστηριακές ασκήσεις, Multisim - Verilog

Lab 3: Logic Gates Explored and Boolean Algebra

Καραμπερόπουλος Δημήτρης Σεπτέμβριος 2023 **Lab 3**

Required Tools and Technology

Software: NI Multisim 14.0 or newer

- ✓ **Install Multisim:** http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ **View Help:** http://www.ni.com/multisim/technical-resources/

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.
4. Learn how to write a Combinational Logic Circuit (CLC) in Verilog.
5. Learn how to test a module and take True Table in Verilog.

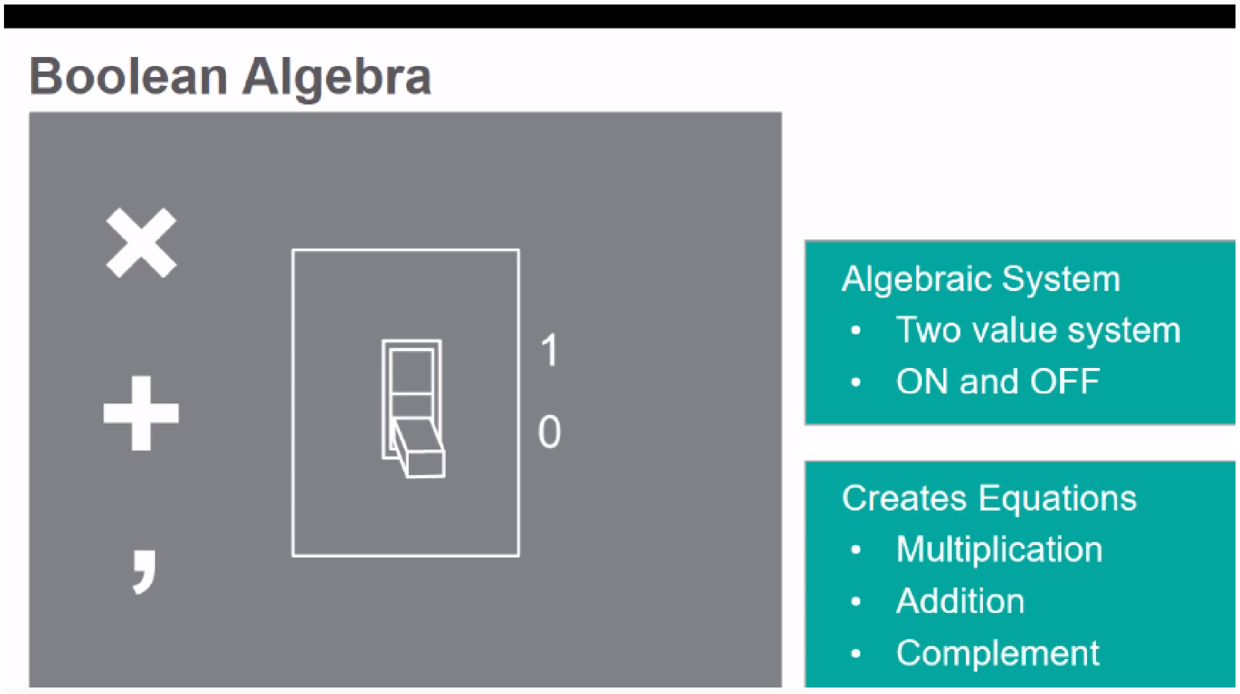
Expected Deliverables

In this lab, you will collect the following deliverables:

- SOP and POS Boolean expressions
- Screenshot, picture, or sketch of circuits
- Truth Tables
- Multisim Files
- Conclusion questions
- Verilog File

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

3.1 Theory and Background


Boolean Algebra

Algebraic System
- Two value system
- ON and OFF

Creates Equations
- Multiplication
- Addition
- Complement

Figure 3-1 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.

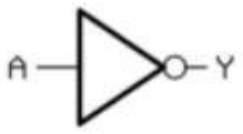
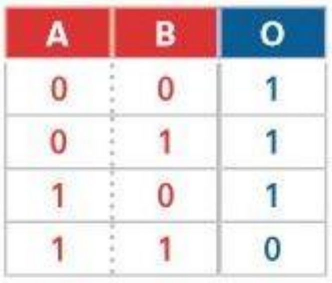


Figure 3-2 Inverter

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

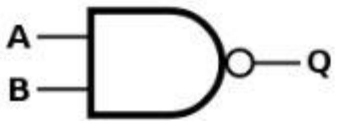
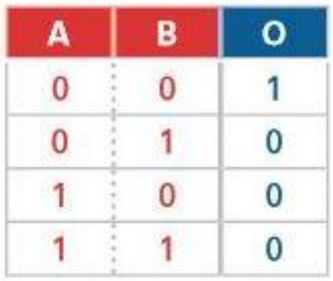


Figure 3-3 NAND gate truth table and symbol

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

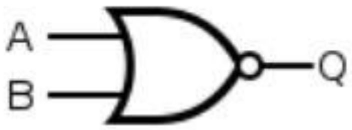
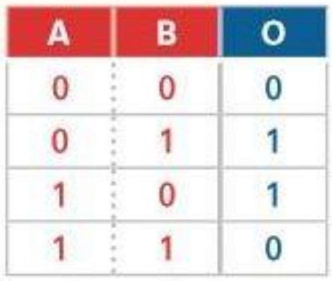


Figure 3-4 NOR gate truth table and symbol

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

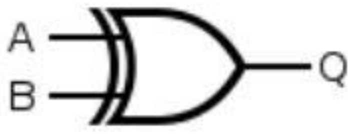
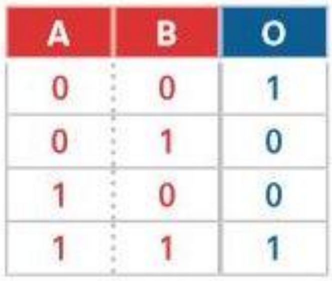


Figure 3-5 XOR gate truth table and symbol

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 3-6 XNOR gate truth table

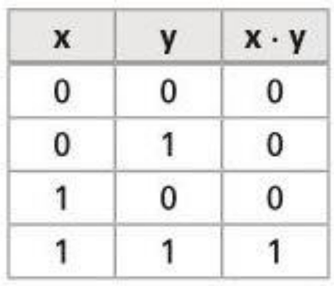
Combinational Logic Circuits (CLCs)

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

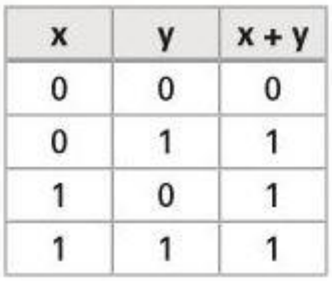
Boolean algebra is an algebraic system where two values are used to represent the properties of bi-stable electrical switching circuits, namely on and off, or simply 1 and 0. The rules for the two binary operators (addition and multiplication) and complement (’) for a two-valued Boolean algebraic expression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



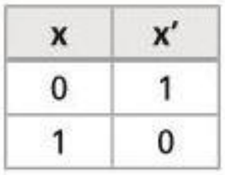
x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3-7 Binary Multiplication (AND logic operation)



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

Figure 3-8 Binary Addition (OR logic operation)



x	x′
0	1
1	0

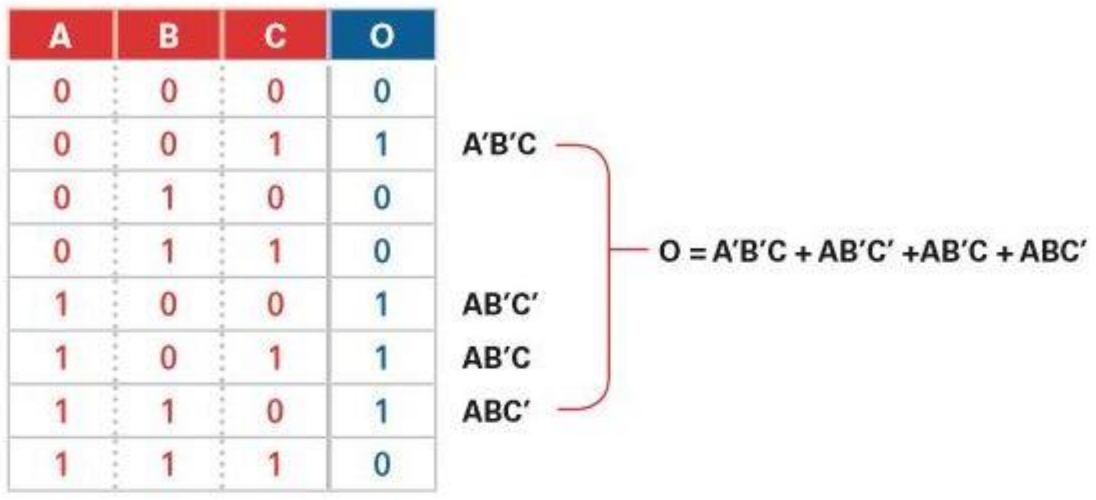
Figure 3-9 Compliment (NOT logic operation)

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 + AB’C’, where ABC, AB’C’ 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
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

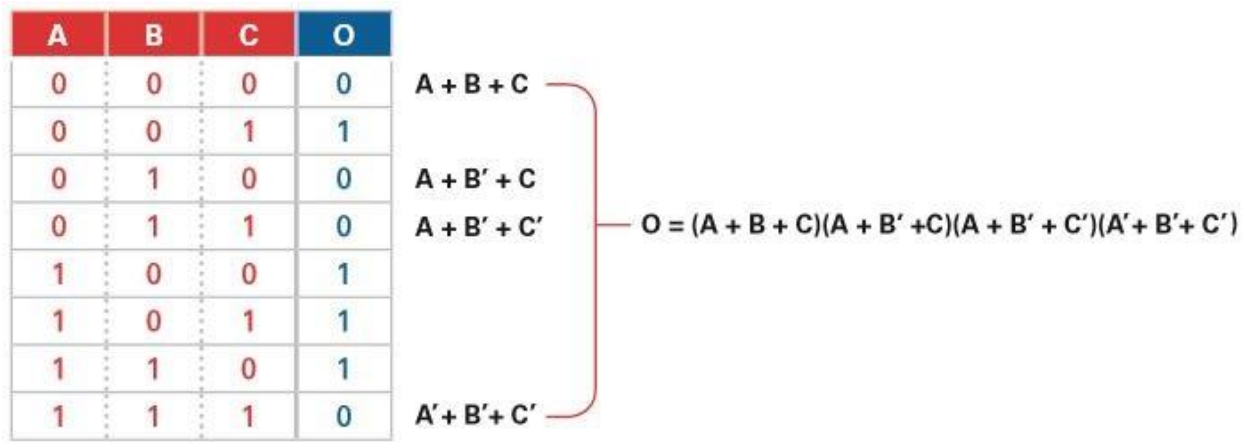
A’B’C
AB’C’
AB’C
ABC’
O = A’B’C + AB’C’ +AB’C + ABC’

Figure 3-10 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
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

A + B + C
A + B′ + C
A + B′ + C′
A′ + B′ + C′
O = (A + B + C)(A + B′ + C)(A + B′ + C′)(A′ + B′ + C′)

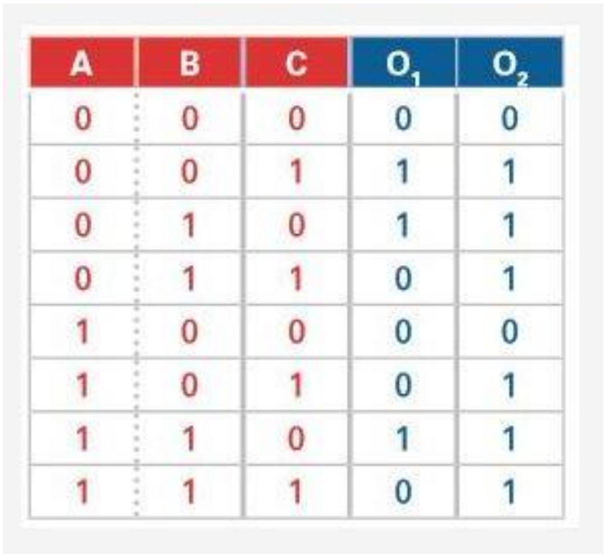
Figure 3-11 POS truth table

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.

3-1 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 3-12 Truth table for two circuits

O1 (SOP) = ____

O2 (POS) = ____

3.2 Simulate: Building a CLC Circuit

Create a new circuit design.

- Using the POS for the second circuit of question 3-1 (O_2), build a circuit which will implement the truth table defined.
- Place as many **NOT gates, 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.

Multisim .ms14

- Multisim: Όνομα αρχείου "**3_Figure_3_12_POS.ms14**".
Προσθήκη στο zip file με όνομα "Lab03_ονοματεπώνυμο_AM.zip"

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

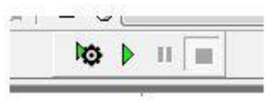


Figure 3-15 Run button

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

3-2 Record the results, as indicated by the probe, in the following truth table.

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

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

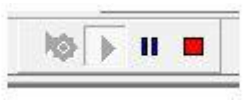


Figure 3-14 Stop button

3-3 Does the behavior of your circuit match the truth table from question 3-1? (Yes/No)

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

Logic Converter

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**.

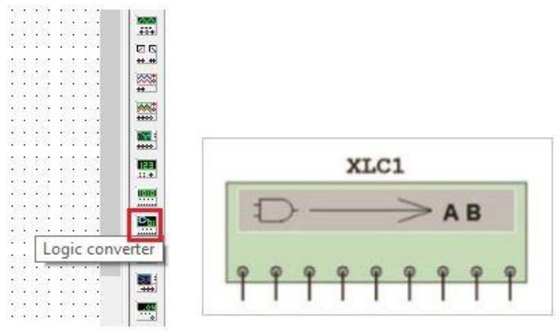


Figure 3-17 Logic converter

3-4 Does the truth table generated match the truth table in question 3-1? If so, is this confirmation that you SOP 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.

- 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**.
- 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.

- Multisim: Όνομα αρχείου “**3_Figure_3_12_SOP_Simpl.ms14**”.
Προσθήκη στο zip file με όνομα ”Lab03_ονοματεπώνυμο_AM.zip”

3-5 Does the circuit generated match the circuit that you could built at the beginning of this lab, if you had built a circuit which implement the truth table using the SOP for the first circuit (O_1);

A. Yes
B. No

3.4 Exercise: Building an XOR Logic Gate in Multisim

XOR Gate Circuit

Build the following circuit using an XOR gate:

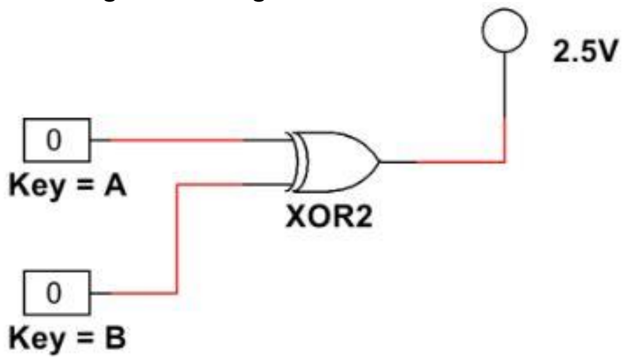


Figure 3-18 XOR gate circuit

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.

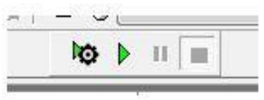


Figure 3-19 Run button

3-6 Vary the inputs of the gate to complete the truth table below.

A	B	O
0	0	
0	1	
1	0	
1	1	

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

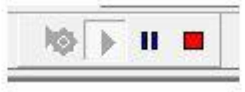


Figure 3-20 Stop button

3.5 Exercise: Building a NOR Logic Gate in Multisim

NOR Gate Circuit

Build the following circuit using an NOR gate:

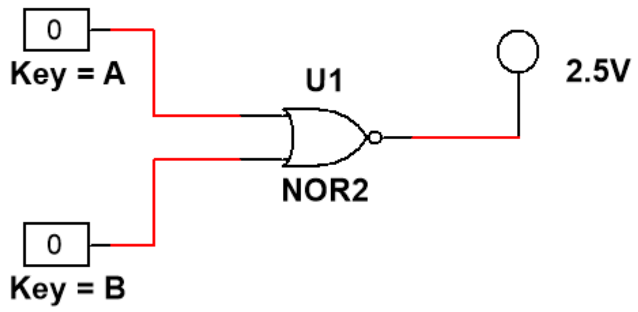


Figure 3-21 NOR gate circuit

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.

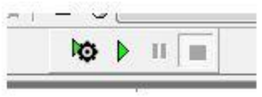


Figure 3-22 Run button

3-7 Vary the inputs of the gate to complete the truth table below.

A	B	O
0	0	
0	1	
1	0	
1	1	

3.6 Conclusion

3-8 What is the difference between SOP and POS? Consider truth tables and logic gates.

A. POS: Άθροισμα γινομένων (Ελαχιστόρων). Οι ελαχιστόροι που παρουσιάζονται στο άθροισμα αυτό είναι εκείνοι που αντιστοιχούν στα ”1” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες AND που τροφοδοτούν μία πύλη OR.

   SOP: Γινόμενο Αθροισμάτων (Μεγιστόρων). Οι μεγιστόροι που παρουσιάζονται στο γινόμενο αυτό είναι εκείνοι που αντιστοιχούν στα ”0” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες OR που τροφοδοτούν μία πύλη AND.

B. SOP: Άθροισμα γινομένων (Ελαχιστόρων). Οι ελαχιστόροι που παρουσιάζονται στο άθροισμα αυτό είναι εκείνοι που αντιστοιχούν στα ”1” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες AND που τροφοδοτούν μία πύλη OR.

   POS: Γινόμενο Αθροισμάτων (Μεγιστόρων). Οι μεγιστόροι που παρουσιάζονται στο γινόμενο αυτό είναι εκείνοι που αντιστοιχούν στα ”0” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες OR που τροφοδοτούν μία πύλη AND.

C. SOP: Άθροισμα γινομένων (Ελαχιστόρων). Οι ελαχιστόροι που παρουσιάζονται στο άθροισμα αυτό είναι εκείνοι που αντιστοιχούν στα ”0” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες AND που τροφοδοτούν μία πύλη OR.

   POS: Γινόμενο Αθροισμάτων (Μεγιστόρων). Οι μεγιστόροι που παρουσιάζονται στο γινόμενο αυτό είναι εκείνοι που αντιστοιχούν στα ”1” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες OR που τροφοδοτούν μία πύλη AND.

D. SOP: Άθροισμα γινομένων (Ελαχιστόρων). Οι ελαχιστόροι που παρουσιάζονται στο άθροισμα αυτό είναι εκείνοι που αντιστοιχούν στα ”1” του πίνακα αληθείας της συνάρτησης.
   Κατασκευάζονται από πύλες AND που τροφοδοτούν μία πύλη AND.

   POS: Γινόμενο Αθροισμάτων (Μεγιστόρων). Οι μεγιστόροι που παρουσιάζονται στο γινόμενο αυτό είναι εκείνοι που αντιστοιχούν στα ”0” του πίνακα αληθείας της

συνάρτησης.
Κατασκευάζονται από πύλες OR που τροφοδοτούν μία πύλη OR.

3-9 Can more than one circuit design produce the same truth table behavior? (Yes/No)

3-10 Why can some SOP and POS expressions be reduced to simpler terms? (Πολλαπλές Απαντήσεις)

A. Κάποιες είσοδοι μπορεί να μην επηρεάζουν την έξοδο

B. Κάποιες έξοδοι μπορεί να μην επηρεάζουν την είσοδο

C. Δεν μπορεί να συμβεί αυτό

D. Κάθε συνάρτηση μπορούμε να την γράψουμε και με διαφορετικό τρόπο

3-11 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

3-12 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

3-13 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

3.7 Exercise: NOT - Gate

Θέλουμε να μετρήσουμε και να συγκρίνουμε την **καθυστέρηση διάδοσης** του σήματος από την είσοδο μέχρι την έξοδο δύο αντιστροφέων διαφορετικής τεχνολογίας.

Οι αντιστροφείς που θα συγκριθούν είναι οι εξής:

- CMOS / CMOS_5V / **4009BD_5V**
- TTL / 74STD / **7404N**

Συνδέστε στην είσοδο και των δύο πυλών ένα ρολόι παραγωγής τετραγωνικών παλμών με συχνότητα 1 MHz.

Χρησιμοποιώντας έναν **παλμογράφο** παρατηρείστε την είσοδο του παλμού στο κύκλωμα και την έξοδο της κάθε πύλης χρησιμοποιώντας τρία κανάλια της συσκευής.

Υπολογίστε τον χρόνο καθυστέρησης στη διάδοση του σήματος κάθε αντιστροφέα, για την μετάβασή του **από 0→1** καθώς και για μετάβαση **από 1→0**.

- Καθυστέρηση διάδοσης από 0→1 του 4009BD _______ **nsec**
- Καθυστέρηση διάδοσης από 0→1 του 7404N _______ **nsec**
- Καθυστέρηση διάδοσης από 1→0 του 4009BD _______ **nsec**
- Καθυστέρηση διάδοσης από 1→0 του 7404N _______ **nsec**

3-14 Ποιος αντιστροφέας είναι πιο γρήγορος;

A. 4009BD
B. 7404N

Η προσομοίωση να γίνει με το Multisim.

Multisim .ms14

- Multisim: Όνομα αρχείου “**3_Not.ms14**”.
Προσθήκη στο zip file με όνομα ”Lab03_ονοματεπώνυμο_AM.zip”

3.8 Exercise: HDL - Verilog, Product-of Sums (POS).

Γράψτε την περιγραφή HDL του κυκλώματος της παραγράφου 3.2 με τρεις τρόπους:
(Δηλαδή από το Figure 3-12, Συνάρτηση O_2)

Δημιουργήστε 3 αρχεία Verilog με τα αντίστοιχα modules:

1) Με τη δημιουργία στοιχειώδους κυκλώματος (user-defined primitive, UDP), κάνοντας χρήση του πίνακα αληθείας του κυκλώματος.

Ονομάστε το module: "POS_3_3_UDP".
(βλέπε Παράδειγμα HDL 3.5 – Morris Mano)

- Verilog: Όνομα αρχείου "**POS_3_3_UDP.v**".
Προσθήκη στο zip file με όνομα "Lab03_ονοματεπώνυμο_AM.zip"

2) Με εκφράσεις Boole.

Ονομάστε το module: "POS_3_3_Boole ".
(βλέπε Παράδειγμα HDL 3.4 – Morris Mano)

- Verilog: Όνομα αρχείου "**POS_3_3_Boole.v**".
Προσθήκη στο zip file με όνομα "Lab03_ονοματεπώνυμο_AM.zip"

3) Περιγραφή σε επίπεδο πυλών.

Ονομάστε το module: "POS_3_3_Gates".
(βλέπε Παράδειγμα HDL 4.10 – Morris Mano)

- Verilog: Όνομα αρχείου "**POS_3_3_Gates.v**".
Προσθήκη στο zip file με όνομα "Lab03_ονοματεπώνυμο_AM.zip"

3.9 Exercise: HDL - Verilog - Test Circuit

Δημιουργήστε μια υπομονάδα διέγερσης για την ανάλυση του κυκλώματος που κατασκευάσατε στην Exercise 3.8 (Module POS_3_3_Gates).

Ονομάστε το module: “t_POS_3_3_Gates”.
(βλέπε Παράδειγμα HDL 4.10 – Morris Mano)

- Verilog: Δημιουργήστε την στο αρχείο που κατασκευάσατε: “ **POS_3_3_Gates.v** ”

3-15 Συμπληρώστε τον Πίνακα Αληθείας

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

3-16 Does the truth table generated match the truth table in Figure 3-12?

A. Yes

B. No