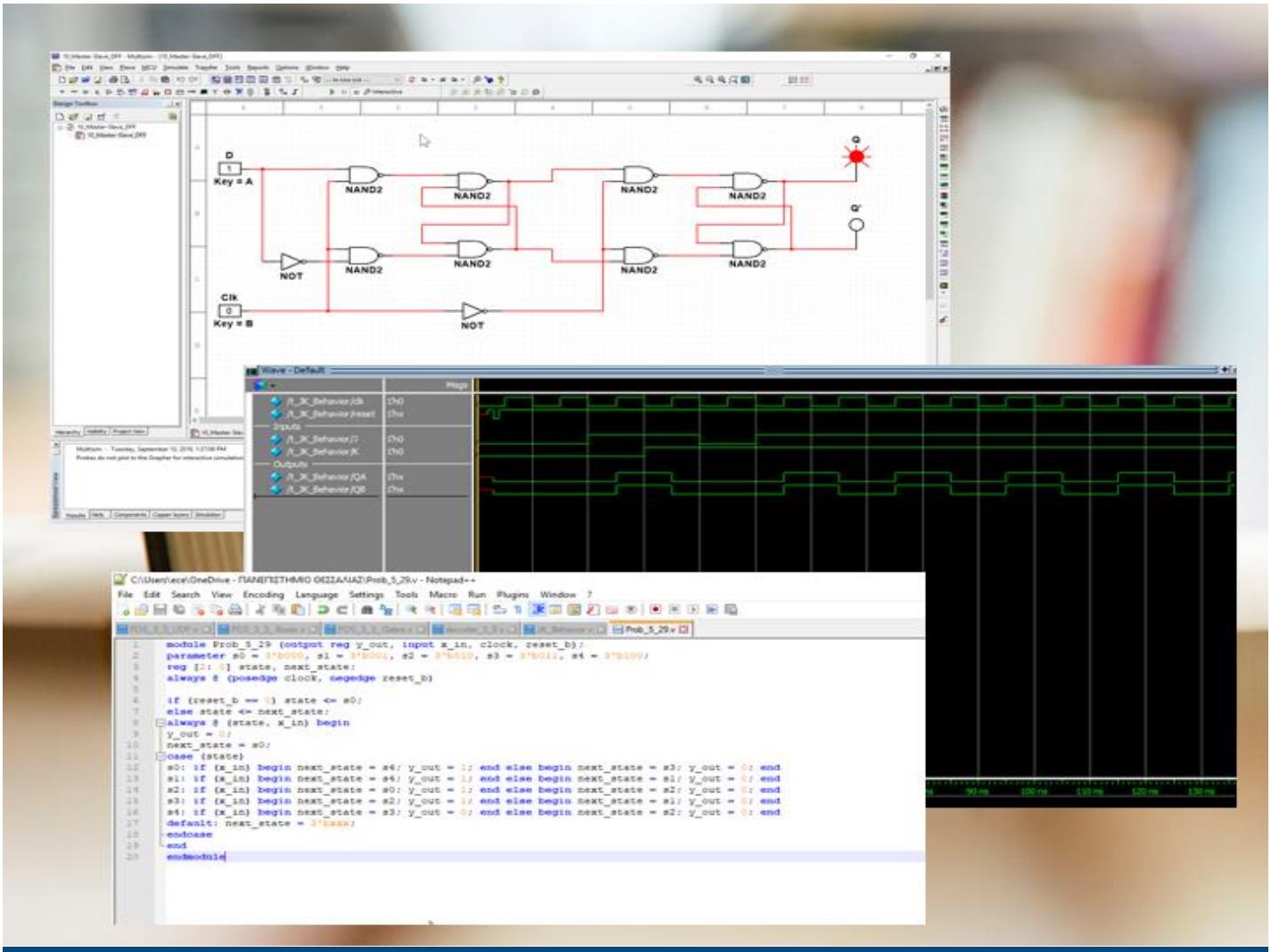
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
χρόνια δημιουργίας
1988 — 2018

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ



ECE119 Ψηφιακή Σχεδίαση

Εργαστηριακές ασκήσεις, Multisim - Verilog

Lab 4: Karnaugh Maps

Καραμπερόπουλος Δημήτρης

Σεπτέμβριος 2023

Lab 4

Required Tools and Technology 1
Lab 4: Karnaugh Maps 2
Learning Objectives 3
Expected Deliverables 3
4.1 Theory and Background 4
Seven Segment Display *5*
Truth Tables *6*
4.2 Exercise: Creating a Karnaugh Map from a Combinational Logic Circuit and simplify the circuit 8
Combinational Logic Circuit *8*
4.3 Implement: Using Karnaugh Maps in Seven Segment Displays 10
Using Karnaugh Maps in Seven Segment Displays and simplify circuits *10*
4.4 Conclusion 12
4.5 Exercise: Εύρεση πύλης & HDL - Verilog 13
4.6 Exercise: Συνδυαστικό κύκλωμα 14
4.7 Exercise: HDL - Verilog , Αλγεβρικές Εκφράσεις 16
4.8 Exercise: Οικουμενικότητα της πύλης NAND 17

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/

MultisimLive

- ✓ https://www.multisim.com/

Lab 4: Karnaugh Maps

The ***Karnaugh map (K-map)*** is a tool and procedure used for minimizing Boolean functions. It is a graphical method that can be used for the manual design of simple logic functions having a small number of variables. K-mapping usually requires fewer steps than algebraic simplification and it always produces a minimum expression.

The Karnaugh map of a function is actually its truth table written as a grid. The rows and the columns of the map correspond to the possible values of the inputs and each cell represents the outputs of the function for the correlated inputs.

The simplified expressions are always in one of the two standard forms:

- **Sum-of-Products**
- **Product-of-Sums**

The cells are formed in a square or rectangle fashion and arranged such that neighboring cells have a single variable difference, otherwise known as ***Gray code ordering***. For simplicity, the input values are placed as column and row labels. Each cell corresponds to a row in the truth table.

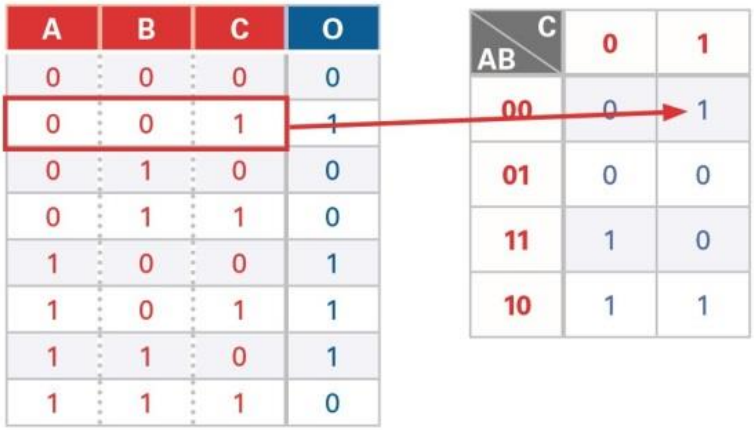


Figure 4-1 Truth Table

Learning Objectives

In this lab, students will:

1. Simplify a Boolean expression using Karnaugh maps
2. Use a circuit with inputs to derive:
    - The output experimentally and using Boolean algebra
    - The Karnaugh map
    - Simplified Combinational Logic Circuit

Expected Deliverables

In this lab you will collect the following deliverables:

- Boolean expressions
- Analysis of gates for Combinational Logic Circuits
- Truth tables
- Conclusion questions

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

4.1 Theory and Background

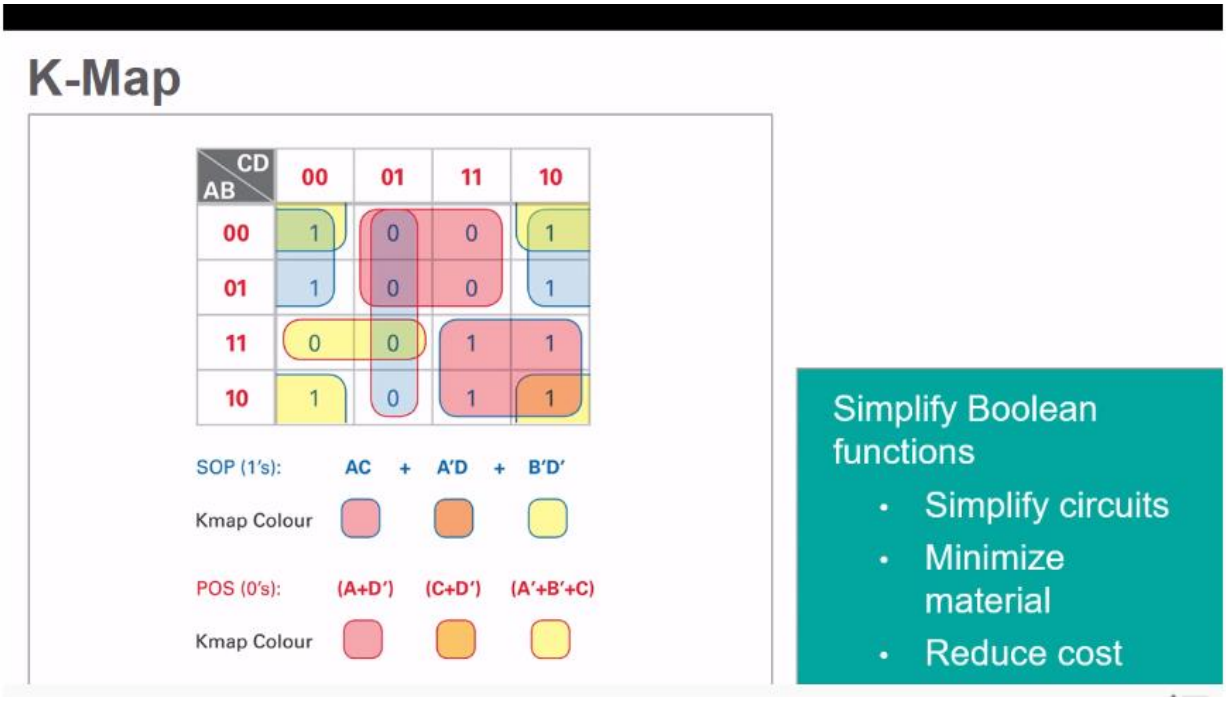


Figure 4-2 Video. View the video here: https://youtu.be/hVifB2hvJic

Video Summary

- Karnaugh maps (K-maps) are used to simplify Boolean functions
- Simplified Boolean functions mean the number of logic gates needed is minimized
- K-maps are built by taking a truth table and converting it into a grid
- K-maps are useful if the logic functions have a small number of variables

After transferring the truth table to a Karnaugh map, cells with common output values, either all 0s or all 1s, are grouped into the largest possible rectangles. Cells are grouped in functions of 2^n either horizontally or vertically. Cells can be used more than once only if this generates the ***least*** number of groups. Also, all cells sharing the chosen common output must be contained within a grouping.

The groups generated can be converted to a Boolean expression. Each group represents a minimal minterm (1s) or maxterm (0s). The term is minimized by eliminating variables whose non-inverted and inverted forms appear within the same cell group. The terms left are then converted to a Boolean expression and used to derive an SOP or POS expression, which may then be converted to a combinational logic circuit.

When choosing the POS form, the variable combination is supposed to equal 0, as is the case when deriving a POS from a truth table. Therefore, the variable's inverse (1) is used when determining a maxterm.

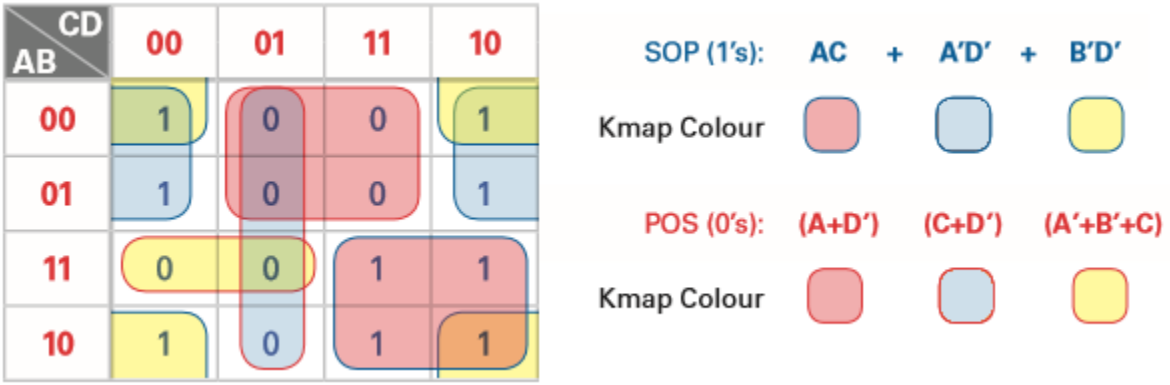


Figure 4-3 Karnaugh Map

Seven Segment Display

Karnaugh maps are useful for minimizing the number of logic gates needed in a circuit. In a practical sense, this reduction also results in a decrease in cost for a manufacturer since fewer components are needed to create an equivalent circuit.

A common way we work with Karnaugh maps is the ***Seven Segment Display (SSD)***. An SSD is an electronic device used for displaying numerical values. The device typically consists of seven segments arranged in a figure 8. Any digit, as well as some alphabet letters, can be displayed when the correct segments are activated. An example of an SSD as well as possible outputs can be seen in the image shown.

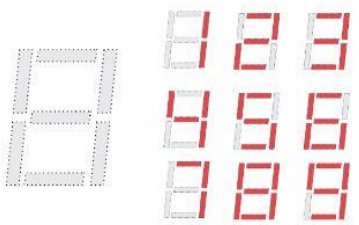


Figure 4-4 Seven segment display

Truth Tables

The truth table for an SSD consists of four inputs and seven outputs:

A	B	C	D	a	b	c	d	e	f	g	Numeric Output
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	X	X	X	X	X	X	X	10
1	0	1	1	X	X	X	X	X	X	X	11
1	1	0	0	X	X	X	X	X	X	X	12
1	1	0	1	X	X	X	X	X	X	X	13
1	1	1	0	X	X	X	X	X	X	X	14
1	1	1	1	X	X	X	X	X	X	X	15

Figure 4-5 SSD Truth Table

Looking at this table, we can make several observations:

- Inputs are noted by capital letters (A-D).
- Outputs are denoted by lower case letters (a-g).
- The Numeric Outputs correspond to the numbers visible on the display.
- For numbers 10-15, “X” values are visible in the table. These are known as ***don’t care conditions***.

The numerical outputs of 0-9 are necessary in an SSD, but outputs 10-15 are said to be illegal. In a Karnaugh map, don’t care conditions can be treated either as 0 or a 1, depending on which one produces a larger block. See below for an example.

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
10	x	x	x	x
11	1	1	x	x

Figure 4-6 Karnaugh Map for output "a"

Note: Similar Karnaugh maps can be created for all other segments.

4-1 Create Boolean expressions (SOP and POS) from the Karnaugh map.

AB \ C	0	1
00	0	1
01	0	0
11	1	0
10	1	1

Figure 4-7 Karnaugh Map

SOP: _____

POS**:** _____

4-2 Identify how many and which gates are needed to create the simplified Combinational Logic Circuit for the SOP expression.

_____ AND Gates
_____ OR Gates
_____ NOT Gates

4-3 Identify how many and which gates are needed to create the simplified Combinational Logic Circuit for the POS expression.

_____ AND Gates
_____ OR Gates
_____ NOT Gates

4.2 Exercise: Creating a Karnaugh Map from a Combinational Logic Circuit and simplify the circuit

Combinational Logic Circuit

Create the following circuit in Multisim:

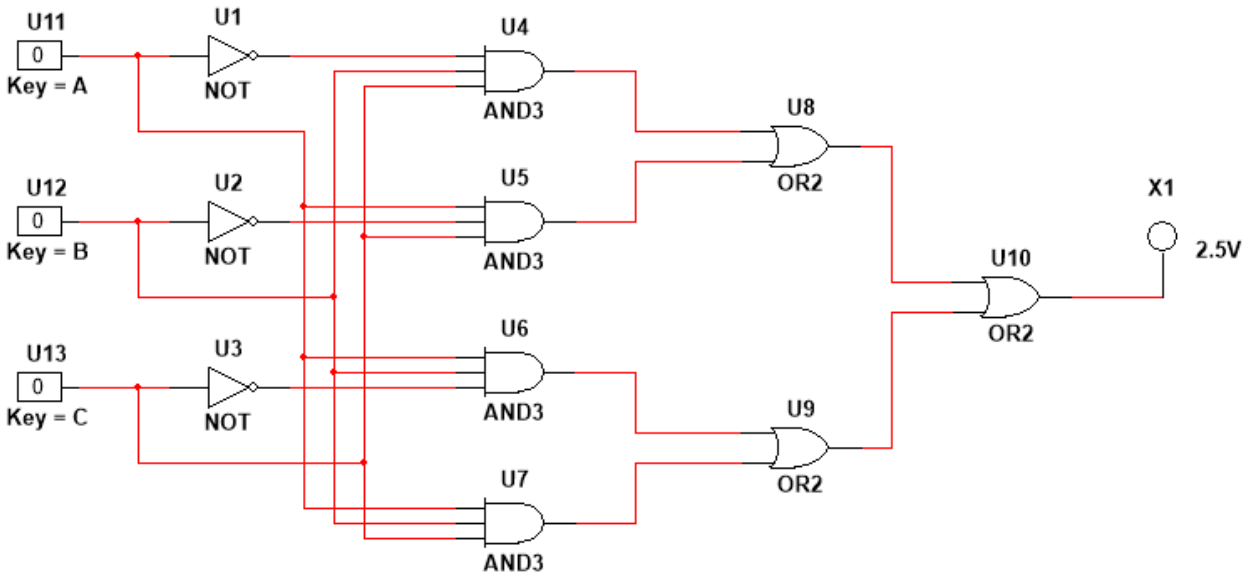


Figure 4-8 Circuit

- Multisim:	Όνομα αρχείου “**4_2_exercise.ms14**”. Προσθήκη στο zip file με όνομα ”Lab04_ονοματεπώνυμο_ΑΜ.zip”

4-4 Vary the inputs as per the truth tables and fill in the output.

A	B	C	Output
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

- **Simplify the Output** using Karnaugh maps.

4-5 What is the resulting Output_simplified expression using Boolean algebra (SOP)?

Output_simplified = _____

4-6 Create the simplified circuit from the simplified expression:

- Multisim: Όνομα αρχείου "**4_2_exercise_simplified.ms14**".
Προσθήκη στο zip file με όνομα "Lab04_ονοματεπώνυμο_AM.zip"

4-7 Vary the inputs as per the truth tables and fill in the output of the simplified circuit.

A	B	C	Output
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

4-8 Does the behavior of the simplified circuit match the expected result?
Do the original and the simplified circuit have the same truth table? (Yes / No)

4.3 Implement: Using Karnaugh Maps in Seven Segment Displays

Using Karnaugh Maps in Seven Segment Displays and simplify circuits

- Using the truth table for a Seven Segment Display in the introduction, create Karnaugh maps for segments **a, b,** and **c.**

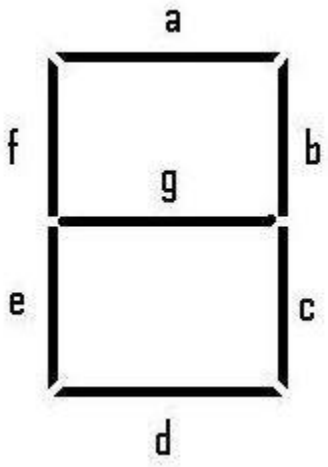


Figure 4-9 Image for Karnaugh maps

4-9 Create simplified Boolean expressions from the Karnaugh map (SOP).

a = _____

b = _____

c = _____

(Η απλοποίηση να γίνει με χάρτες Karnaugh, τους οποίους **θα πρέπει να επιδείξετε** κατά την διάρκεια του εργαστηρίου)

- Create a circuit (one sheet) which will output 3 values to control **a, b** and **c** segments. Then, if it is possible, **minimized them.**
  Use only gates with **1 or 2 inputs**.

- Multisim:	Όνομα αρχείου “**4_SSD_abc.ms14**”. Προσθήκη στο zip file με όνομα ”Lab04_ονοματεπώνυμο_ΑΜ.zip”

- Run the simulation in Multisim.

4-10 Does the behavior of the simplified circuit match the expected result? (Yes / No)

4-11 Would it be possible to simplify this circuit? If so, how?

Use only gates with 1 or 2 inputs.

_____

_____

_____

4-12 How many gates do you need in order to implement the simplified circuit?

Use only gates with 1 or 2 inputs.

_____

4.4 Conclusion

4-13 What are Karnaugh maps (K-maps) used for?

A. Graphically minimizing Boolean functions
B. Taking simple expressions of Boolean functions and expanding them
C. Grouping opposite output values together (0's and 1's)
D. Determining product-of-sums only

4-14 How many cells can be grouped together in the simplification of Karnaugh maps?

A. No more than 4
B. 2
C. 2^n
D. As many as possible

4-15 Creating Boolean expressions from Karnaugh maps typically leads to _____ in the number of logic gates needed in a circuit.

A. An increase
B. A decrease
C. No change
D. None of the above

4-16 How are 'don't' care conditions' treated in Karnaugh maps?

A. They are ignored
B. They are combined with the value on their right
C. They can be treated as either a 0 or 1 depending on the situation
D. They are always given a value of 0

4-17 A Seven Segment Display (SSD):

A. Consists of seven segments arranged in a figure 8
B. Can display any number between 0 and 9 digitally
C. Can be simulated with logic gates
D. All of the above

4-18 Δώστε έναν ορισμό της έννοιας του συνδυαστικού κυκλώματος.

A. CLCs are a classification of circuits whose output is dependent on the current inputs and/or the current outputs and are implemented by Boolean circuits.
B. CLCs are a classification of circuits whose output is only dependent on the current inputs and are implemented by Boolean circuits.

4.5 Exercise: Εύρεση πύλης & HDL - Verilog

Βρείτε την συνάρτηση που υλοποιεί το παρακάτω κύκλωμα (Figure 4-10) και απλοποιείστε την ώστε να δείτε με ποια γνωστή λογική πύλη ισοδυναμεί.

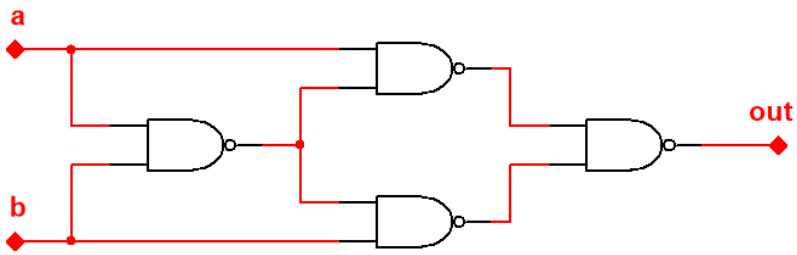


Figure 4-10

4-19 Αρχική συνάρτηση: **E =** _____

4-20 Απλοποιημένη συνάρτηση: **E =** _____

4-21 Ποια πύλη είναι; _____

1. Θέλουμε να επαληθεύσουμε την απλοποιημένη συνάρτηση που βρήκαμε από το Figure 4-10, καθώς και την πύλη που επιλέξαμε.

Για να το πετύχουμε αυτό θα πρέπει να ελέγξουμε την λειτουργικότητα του κυκλώματος που απεικονίζει το Figure 4-10.

Γράψτε λοιπόν σε γλώσσα Verilog την περιγραφή του προηγούμενου κυκλώματος (Figure 4-10). *(Η περιγραφή θα πρέπει να γίνει με τις 4 πύλες NAND που απεικονίζονται στο σχήμα και όχι με την απλοποιημένη συνάρτηση που υπολογίσαμε)*

Ονομάστε το module: "Which_gate_is".

2. Έπειτα κατασκευάστε μία μονάδα δοκιμής (test bench) όπου θα δοκιμάζετε το προηγούμενο κύκλωμά (module) που περιγράψατε.

Ονομάστε το module: "t_Which_gate_is".

Να χρησιμοποιήσετε την εντολή **$monitor** για να ελέγξετε τις εισόδους – εξόδους.

Να χρησιμοποιήσετε τις εντολές **$dumpfile** και **$dumpvars** ώστε να παραχθεί το αρχείο "vcd" με τις κυματομορφές των σημάτων.
Έπειτα να ανοίξετε αυτό το αρχείο με το **gtkwave** για να δείτε τις κυματομορφές και να ελέγξετε οπτικά τον τρόπο που μεταβάλλεται η έξοδος σε όλους του πιθανούς συνδυασμούς τις εισόδου.

- Verilog: Όνομα αρχείου “**Which_gate_is.v**”.
Προσθήκη στο zip file με όνομα ”Lab04_ονοματεπώνυμο_ΑΜ.zip”

4-22 Συμπληρώστε τον πίνακα αληθείας με βάση τις κυματομορφές:

A	B	Output
0	0	
1	0	
0	1	
1	1	

4-23 Είναι ο πίνακας αληθείας της πύλης που υποθέσατε στην προηγούμενη άσκηση;
(Ναι/Όχι) : _____

4.6 Exercise: Συνδυαστικό κύκλωμα

Σχεδιάστε ένα συνδυαστικό κύκλωμα το οποίο αποφασίζει εάν ένας αριθμός στην δυαδική του αναπαράσταση με χρήση τριών bits, είναι μικρότερος του 3. (εάν είναι να βγάζει έξοδο “1”)

4-24 Συμπληρώστε τον πίνακα αληθείας

x	y	z	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

4-25 Γράψτε την συνάρτηση της εξόδου σε μορφή SOP

F = _____

4-26 Γράψτε την απλοποιημένη μορφή της συνάρτησης εξόδου σε μορφή SOP

F = _____

4-27 Θέλουμε να επαληθεύσουμε την απλοποιημένη συνάρτηση που βρήκαμε.

Για να το πετύχουμε αυτό θα πρέπει να ελέγξουμε την λειτουργικότητα του κυκλώματος υλοποιεί την απλοποιημένη συνάρτηση της 4-26.

Γράψτε λοιπόν σε γλώσσα Verilog την περιγραφή του προηγούμενου κυκλώματος (συνάρτηση 4-26).

Ονομάστε το module: "clc".

4-28 Έπειτα κατασκευάστε μία μονάδα δοκιμής (test bench) όπου θα δοκιμάζετε το προηγούμενο κύκλωμά (module) που περιγράψατε.

Ονομάστε το module: "t_clc".

Να χρησιμοποιήσετε την εντολή **$monitor** για να ελέγξετε τις εισόδους - εξόδους.

Να χρησιμοποιήσετε τις εντολές **$dumpfile** και **$dumpvars** ώστε να παραχθεί το αρχείο "vcd" με τις κυματομορφές των σημάτων.
Έπειτα να ανοίξετε αυτό το αρχείο με το **gtkwave** για να δείτε τις κυματομορφές και να ελέγξετε οπτικά τον τρόπο που μεταβάλλεται η έξοδος σε όλους του πιθανούς συνδυασμούς τις εισόδου.

- Verilog:	Όνομα αρχείου "**clc.v**". Προσθήκη στο zip file με όνομα "Lab04_ονοματεπώνυμο_ΑΜ.zip"

4-29 Συμπληρώστε τον πίνακα αληθείας με βάση τις κυματομορφές:

x	y	z	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

4-30 Είναι ο πίνακας αληθείας όμοιος με αυτόν που συμπληρώσατε στο ερώτημα 4-24;

(Ναι/Όχι) : _____

4.7 Exercise: HDL - Verilog , Αλγεβρικές Εκφράσεις

Υλοποιήστε σε γλώσσα Verilog τις παρακάτω αλγεβρικές εκφράσεις:

1. $f = a \cdot b + c'$
2. $f = d(e + a)' + b \cdot c'$

Στο φυσικό κύκλωμα που θα περιγράψετε, θεωρείστε ότι υπάρχουν οι εξής καθυστερήσεις διάδοσης των σημάτων στις πύλες:

nand, nor:	40 nsec
and:	30 nsec
or:	20 nsec
not:	10 nsec

- Verilog: Όνομα αρχείου “**Expressions_4.v**”.
Προσθήκη στο zip file με όνομα ”Lab04_ονοματεπώνυμο_ΑΜ.zip”

Θα περιλαμβάνει τις εξής 2 μονάδες (modules):

- **“Expression_4_1”**
- **“Expression_4_2”**

4.8 Exercise: Οικουμενικότητα της πύλης NAND

Όπως έχουμε δει και στη θεωρία του μαθήματος, με χρήση της πύλης NAND μπορούμε να υλοποιήσουμε κάθε λογική συνάρτηση που μας δίνεται. Χρησιμοποιώντας, λοιπόν, πύλες NAND υλοποιείστε:

- έναν αντιστροφέα
- μια πύλη AND δύο εισόδων
- μια πύλη OR δύο εισόδων
- μια πύλη NOR δύο εισόδων
- μια πύλη XOR δύο εισόδων

Για να επαληθεύσετε την ορθότητα κάθε σχεδίασής σας συμπληρώστε τον πίνακα αληθείας με τις τιμές που προκύπτουν από το κύκλωμα σας εάν αυτό προσομοιωθεί με τη χρήση κατάλληλων διακοπτών και λαμπτήρων στο Multisim.

- Multisim: Όνομα αρχείου “**4_All_Nand.ms14**”.
Προσθήκη στο zip file με όνομα ”Lab04_ονοματεπώνυμο_AM.zip”

NOT	
A	**Output**
0	
1	

AND		
A	**B**	**Output**
0	0	
1	0	
0	1	
1	1	

OR		
A	**B**	**Output**
0	0	
1	0	
0	1	
1	1	

NOR		
A	**B**	**Output**
0	0	
1	0	
0	1	
1	1	

XOR		
A	**B**	**Output**
0	0	
1	0	
0	1	
1	1	