Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Digital Lab (ENCS2110)

Experiment No.1 Post-Lab

Title: Combinational Logic Circuits

Prepared by:

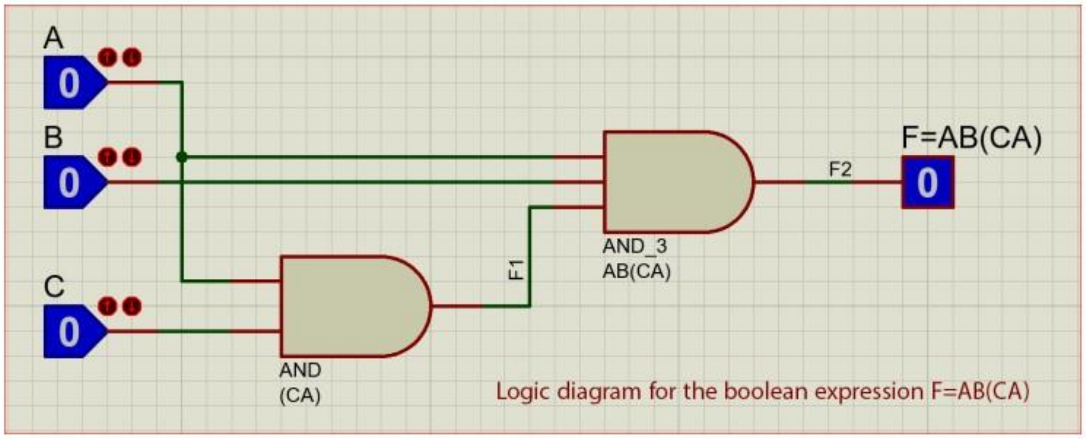
Name: Aya Dahbour **Number:** 1201738

Instructor: Dr. Bilal Karaki **TA:** Eng. Ali Hamoudeh

Post-lab questions' solution:

*All circuits were connected using Proteus.

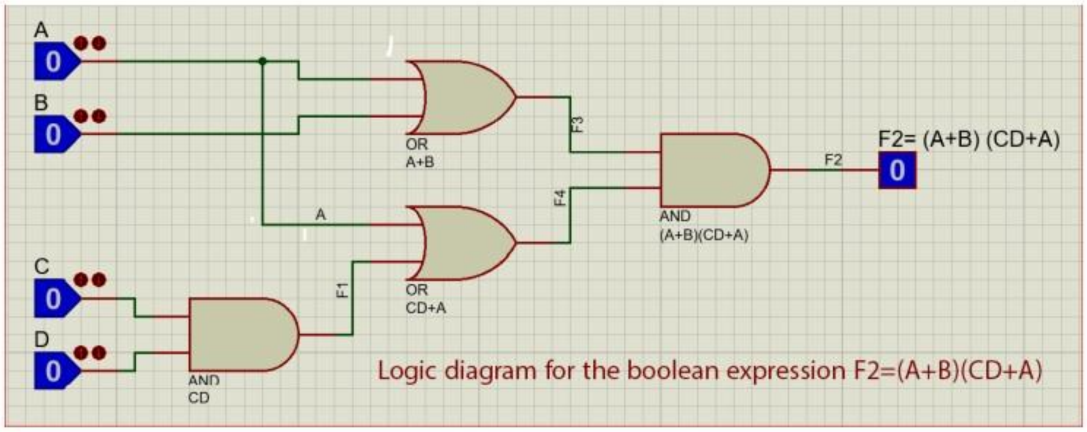
1. F=AB(CA) using AND gates



Figure(1): F logic diagram

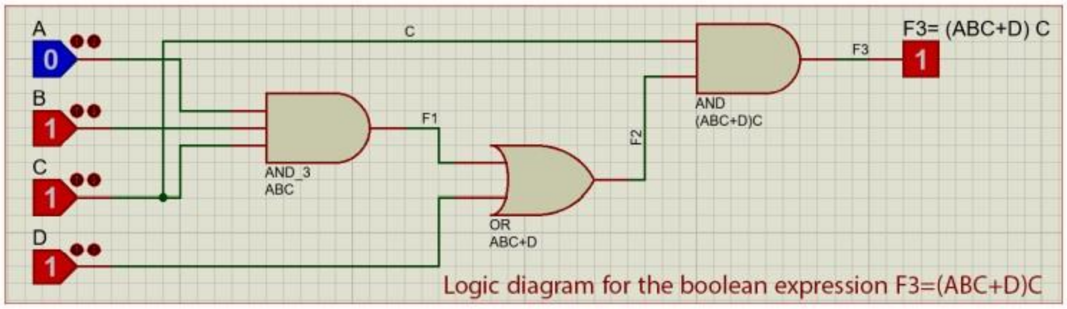
2.

A. F2 = (A+B)(CD+A)



Figure(2): F2 logic diagram

B. F3 = (ABC+D)C

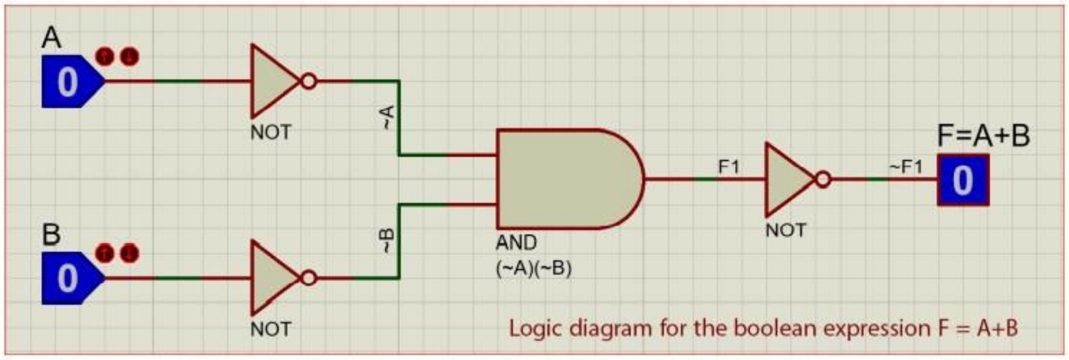


Figure(3): F3 logic diagram

3. OR operation implemented using AND, NOT gates.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Table(1): OR truth table

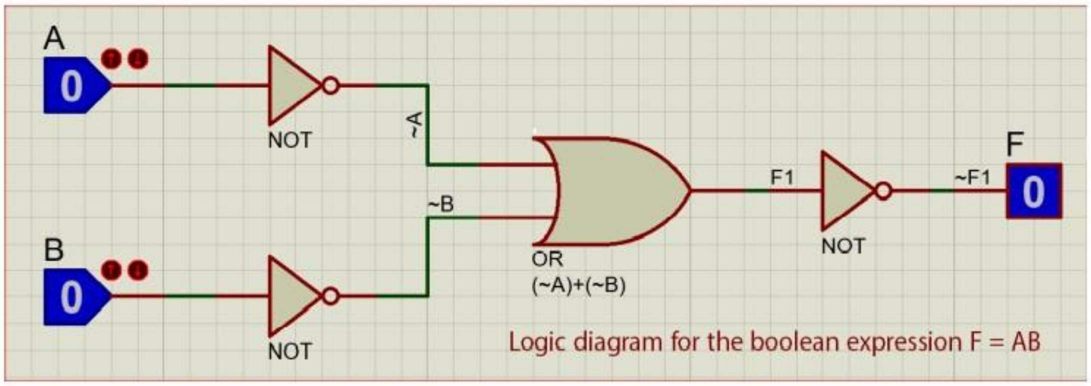


Figure(4): OR implementation using AND, NOT gates

4. AND gate implemented using OR, NOT gates.

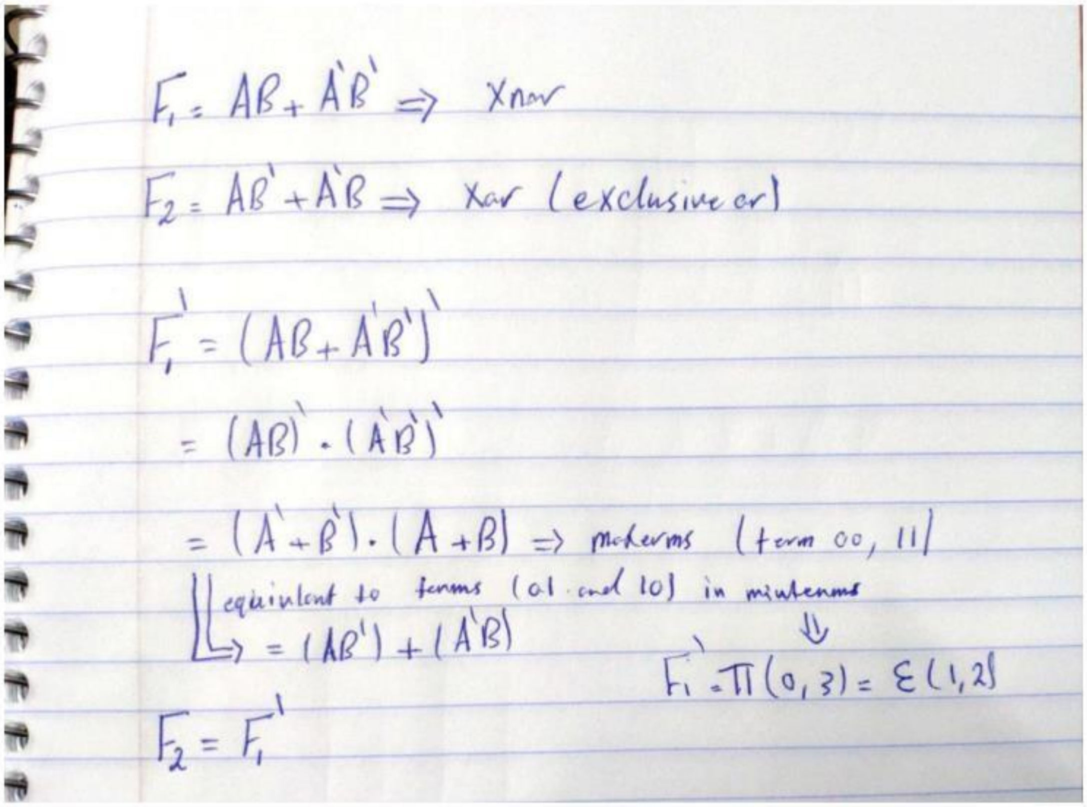
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Table(2): AND truth table



Figure(5): AND implementation using AND, NOT gates

5. Proving that XNOR is the inverse of XOR



$F_1 = AB + A'B' \Rightarrow$ Xnor

$F_2 = AB' + A'B \Rightarrow$ Xor (exclusive or)

$F_1' = (AB + A'B')'$

$= (AB)' \cdot (A'B')'$

$= (A' + B') \cdot (A + B) \Rightarrow$ maxterms (term 00, 11)

equivalent to terms (01 and 10) in minterms $\Downarrow$

$\Rightarrow = (AB') + (A'B)$

$F_1' = \Pi(0,3) = \Sigma(1,2)$

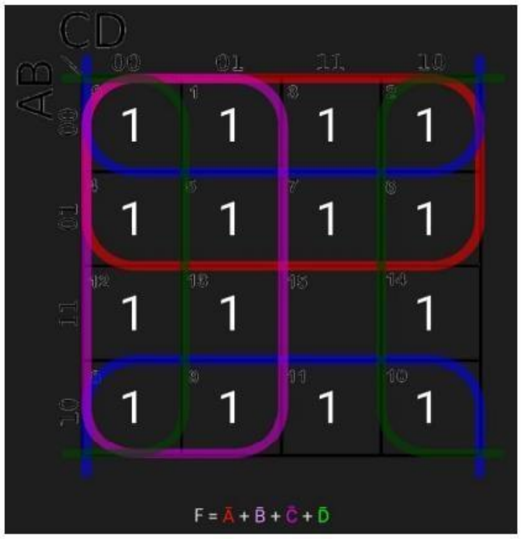
$F_2 = F_1'$

Figure(6): Proof of XNOR is inverse of XOR

6. NAND gate with four inputs implementation using two input NAND gates

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table(3): 4-input NAND truth table



Figure(7): K-map for F

Using boolean algebra rules, I found equivalent boolean expression for the F which can be used to implement 4-input NAND using 2-input NANDs

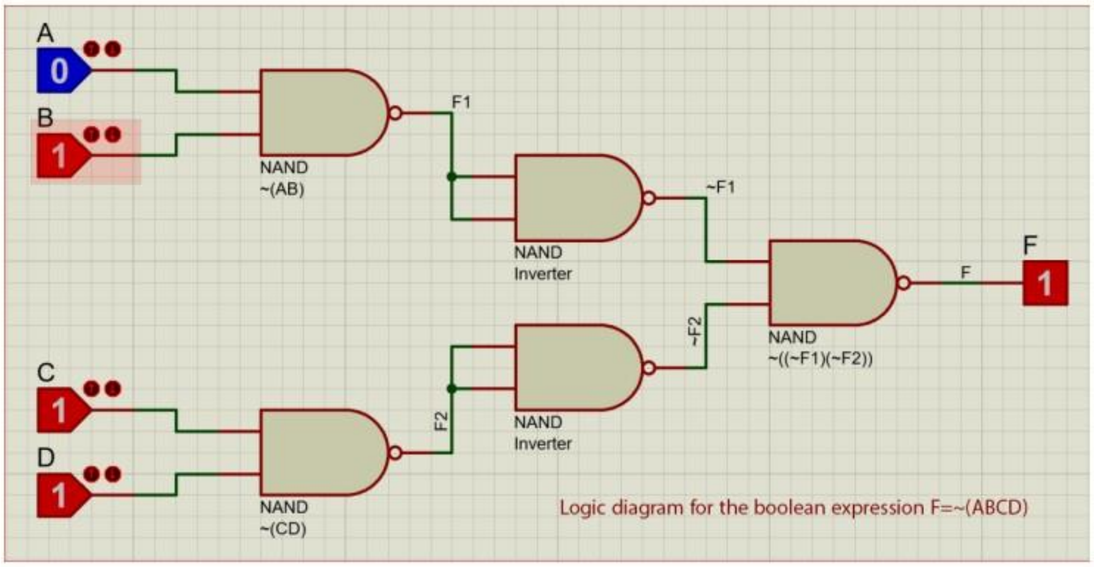
F = A′ + B′ + C′ + D′

= (A′+B′)+(C′+D′)

$= (AB)'+(CD)'$

$= \big(((AB)')'.((CD)')'\big)'$

This is the function which is going to be using in implementing 4-input NAND gate using 2-input NAND gates only.



Figure(8): 4-input NAND implementation using 2-input NAND

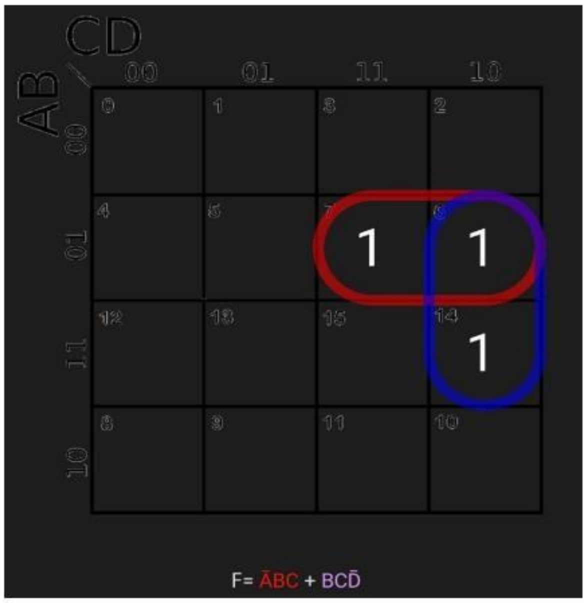
7.

A. F1 = A'BCD + ABCD' + A'BCD' + ABCD'

A	B	C	D	F1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

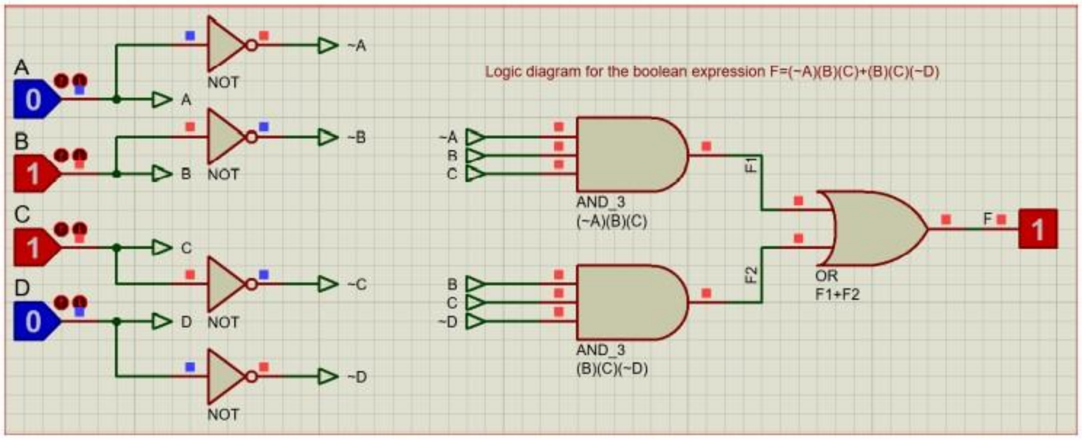
Table(4): F1 truth table

Using K-map:



Figure(9): K-map for F1

F1 = A'BC+BCD'



Logic diagram for the boolean expression F=(~A)(B)(C)+(B)(C)(~D)
A
0
B
1
C
1
D
0
NOT
~A
~B
~C
~D
AND_3
(~A)(B)(C)
AND_3
(B)(C)(~D)
F1
F2
OR
F1+F2
F
1

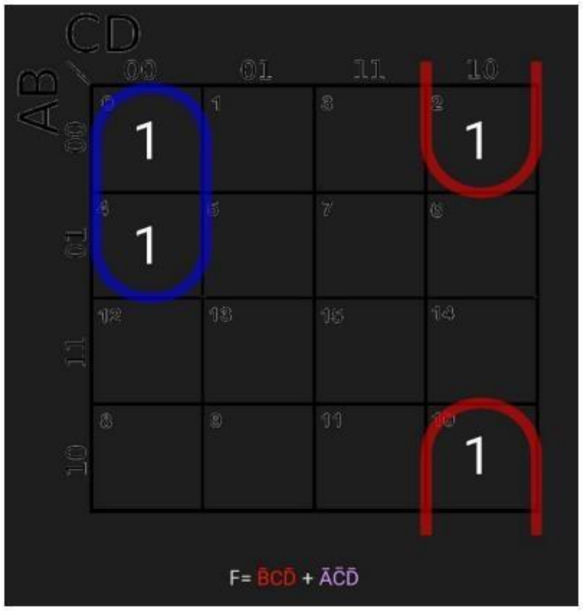
Figure(10): F1 logic diagram

B. F2 = A'B'C'D' + AB'CD' + A'B'CD' + A'BC'D'

A	B	C	D	F2
0	0	0	0	1
0	0	0	1	0
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	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

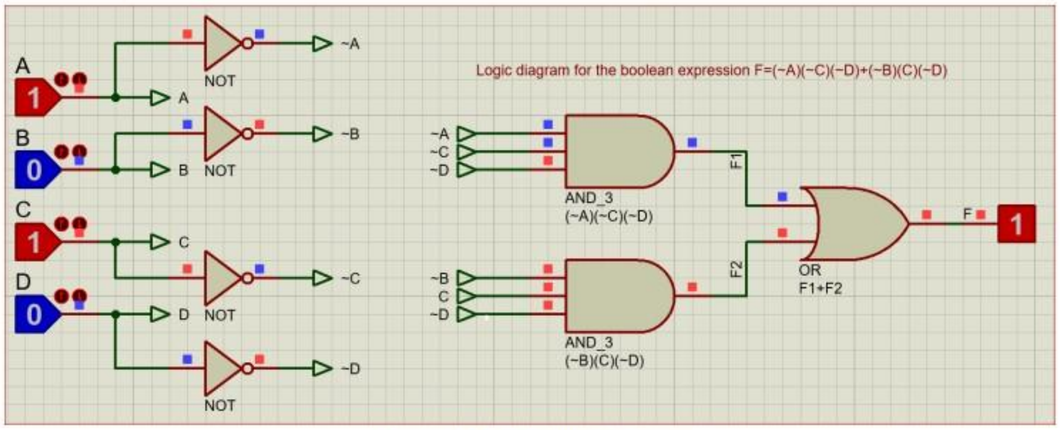
Table(5): F2 truth table

Using K-map



Figure(11): K-map for F2

F2 = A’C’D’+B’CD’


Logic diagram for the boolean expression F=(~A)(~C)(~D)+(~B)(C)(~D)
A
B
C
D
NOT
~A
~B
~C
~D
AND_3
(~A)(~C)(~D)
(~B)(C)(~D)
F1
F2
OR
F1+F2
F

Figure(12): F2 logic diagram