Digital Electronics and systems

19ECE204

Name : J. Rithika Sri

Roll No : CB.EN.U4CSE19025

Date : 08.09.2020

Home Assignment - 1

1) The lecture by Prof Anant Agarwal of MIT is about the impact of noise in digital signal.

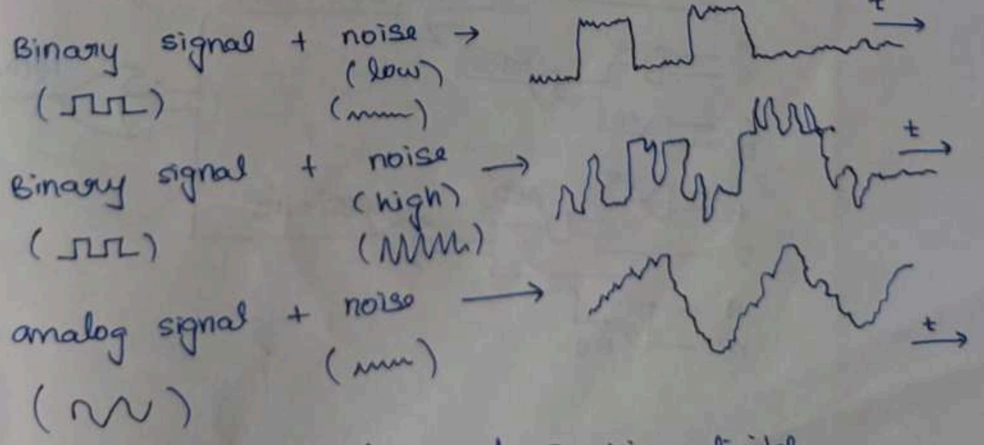
Digital signals are signals used to represent data as a sequence of discrete values at a given time.

↳ analog signals have continuous range of values

↳ binary signals have two possible values - '0' and '1'.

Impact of Noise :

Noise signals are unwanted interference That degrades a communication signal. It can interfere with both analog and binary. However, very large amount of noise is necessary to affect the binary signal. Since analog signal represent an Infinite range of values, The noise signal causes the fluctuation in the message communicated.



Refered to → wikipedia and Precision digital

2) ~~F(A,B,C,D)~~

(a)

$$F(A,B,C,D,E) = A'B'C'D' + A'BC'D' + A'B'CD' + A'B'D'E + AB'CD + ABD'E' + ABD'E + AB'CDE$$

$$= A'C'D'(B+B') + AB'CD(1+E) + ABD'(E+E') + A'B'CD' + A'B'D'E$$

$$= A'C'D' + AB'CD + ABD' + A'B'CD' + A'B'D'E$$

$$= A'D'(C' + \cancel{A}B') + AB'CD + ABD' + A'B'D'E$$

$$= A'D'C + A'D'B' + AB'CD + ABD' + A'B'D'E$$

$$= A'D'C' + A'D'B' + AB'CD + ABD'$$

$$= A'D'C' + D'(AB + A'B') + AB'CD$$

$$= A'D'C' + ABD' + A'B'D' + AB'CD$$

(b)

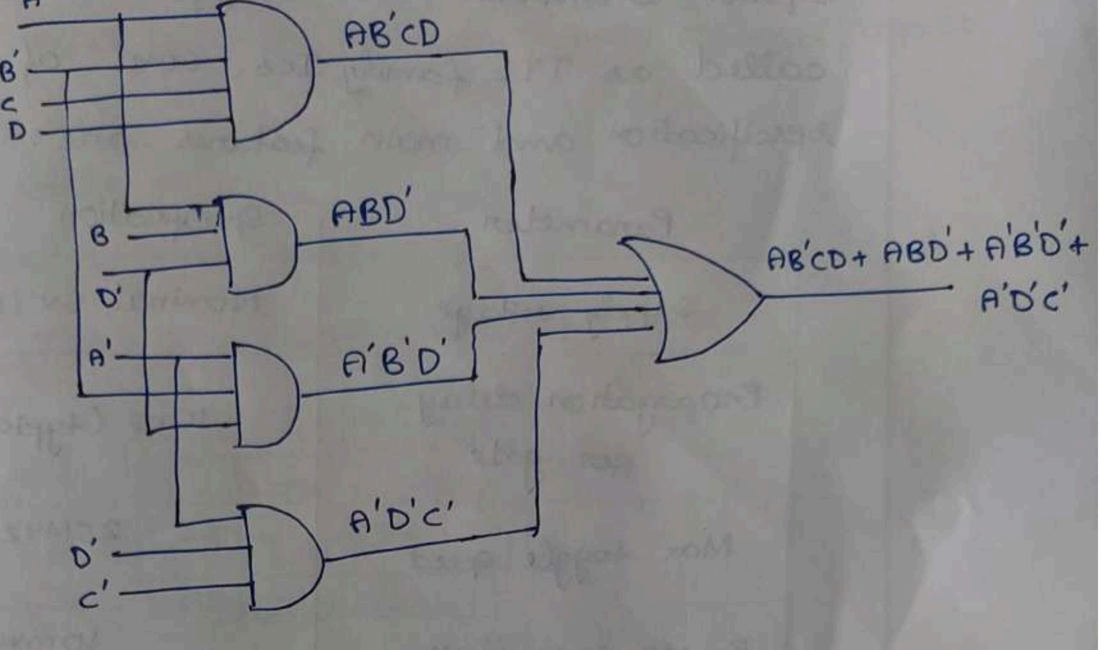
$$F(A,B,C,D,E) = A'B'C'D'E + A'B'C'D'E' + A'BC'D'E + A'BC'D'E' + A'B'CD'E + A'B'CD'E' + AB'CDE + AB'CDE' + ABCD'E' + ABC'D'E' + \cancel{AB'CDE}\ ABCD'E + ABC'D'E$$

$$= 00001 + 00000 + \overset{01001}{\cancel{001010}} + 01000 + 00101 + 00100 + 10111 + 10110 + 11100 + 11000 + 11101 + 11001$$

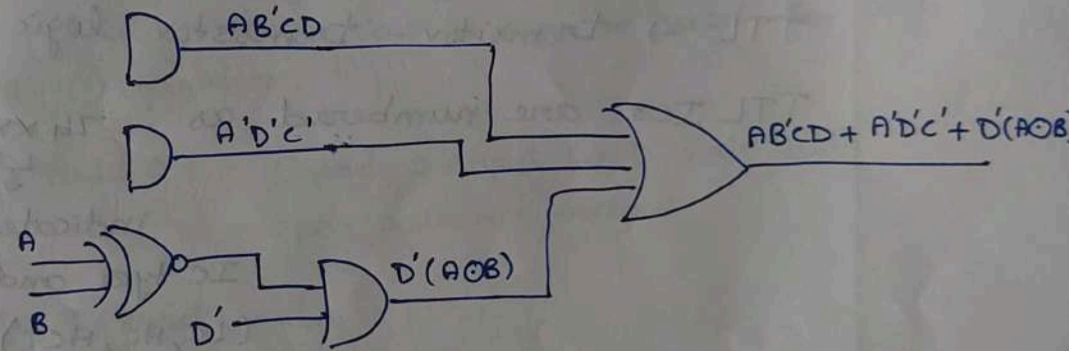
$$= 1 + 0 + 9 + 8 + 5 + 4 + 23 + 22 + 28 + 24 + 29 + 25$$

Minterms : $m1, m0, m9, m8, m5, m4, m23, m22, m28, m24, m29, m25$

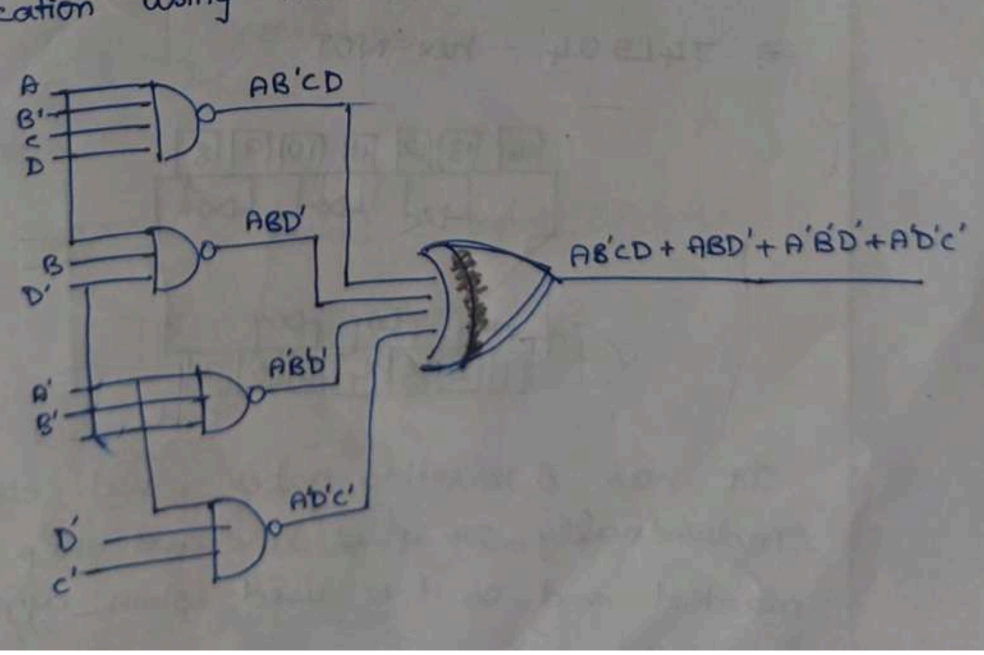
(c) 2 Level realization of F(A,B,C,D,E):



(d) Simplification using XOR / XNOR :



(e) simplification using NAND :



3) The 74 series Ics are fabricated by using bipolar transistor technology, so they are called as TTL family Ics. some of its specification and main features are:

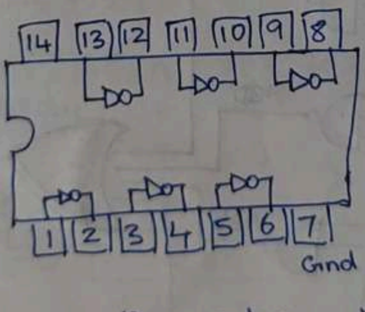
Parameter	Specification
supply voltage	Nominal 5v (4.25-5.25)
Propagation delay per gate	10ns (typically)
Max toggle speed	25MHZ
Power consumption per gate	10mw

TTL → transitor - transistor logic

TTL ICs are numbered as : 74xx00
↳ indicates The IC type and specification (LC, HC, HCT)

(a) NOT Gate :

⇒ 74LS04 - hex NOT



It has 6 inverting gates which can be used individually. It is a low cost chip available in market and so it is used when application cost is low.

Applications : Servers, memory units, networking, PCs and notebooks.

other TTL ICs for NOT :

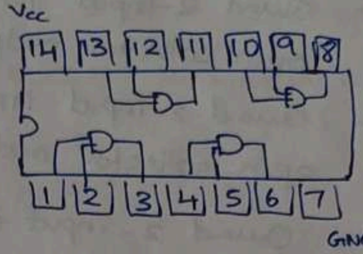
⇒ 74 LS05 - hex NOT with open collector outputs

⇒ 74LS14 - hex NOT with schmitt Trigger inputs

(b) AND Gate :

Quad 2-input gates :

⇒ 74LS08 - Quad 2-input AND gate



It has four AND gates in the chip. The gates in the chip are designed by schottky transistors to make switching delay less. so this can be used for high speed AND operation.

⇒ 74LS09 - Quad 2-input AND Gate with open collector output

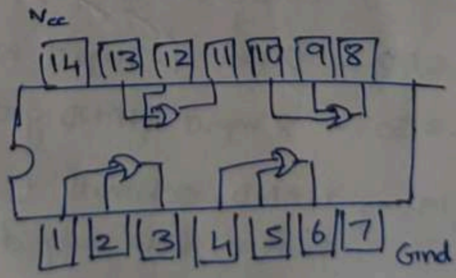
Triple 3-input gates:

⇒ 74LS11 - Triple 3 input AND

⇒ 74 LS 15 - Triple-3 input AND gate, open collector outputs

(c) OR Gate :

⇒ 74LS32 - Quad 2-input OR gate

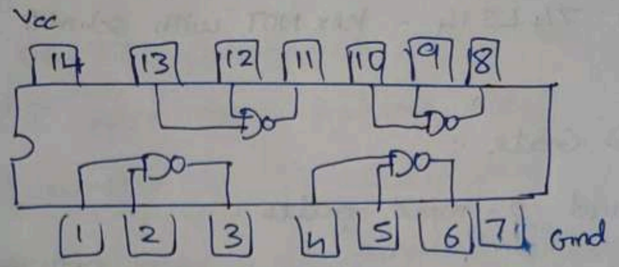


It has four OR gates.

Application : Oscillator circuits, encoder and decoder, multiplexer and de- multiplexer.

(d) NAND Gate :

=> 74LS00 - Quad 2-input NAND gate



=> 74LS01 - Quad 2-input NAND gate, open collector output

=> 74LS03 - Quad 2-input NAND gate with open collector output

=> 74LS132 - Quad 2-input NAND gate with schmitt trigger inputs

=> 74LS26 - Quad 2-input NAND gate, OC (15V)

=> 74LS28 - Quad 2-input NAND gate with OC (15V)

=> 74LS38 - Quad 2-input NAND gate, open collector outputs

=> 74LS10 - triple 3-input NAND

=> 74LS12 - triple 3-input NAND with open collector outputs

=> 74LS13 - Dual 4-input NAND schmitt triggers

=> 74LS20 - dual 4-input NAND

=> 74LS22 - Dual 4-input NAND gate, open collector outputs

=> 74LS40 - Dual 4-input NAND gate

=> 74LS30 - 8 input NAND gate

=> 74LS19 - NAND schmitt trigger, totem Pole output

=> 74LS39 - 4x two input NAND, open collector

PTO →

Refered to : electronichub.org , electronics - tutorials.ws

(e) NOR Gate:

⇒ 74LS02 - Quad 2-input NOR

⇒ 74LS37, 74LS32, 74LS28 - Quad 2-input NOR

⇒ 74LS33 - Quad 2-input NOR, open collector output

⇒ 74LS38 - Quad - 2 input NOR, open collector outputs

⇒ 74LS27 - triple 3-input NOR

⇒ 74LS23 - 2 × four input NOR with strobe

⇒ 74LS25 - 2 × four input NOR with strobe

(f) XOR Gate:

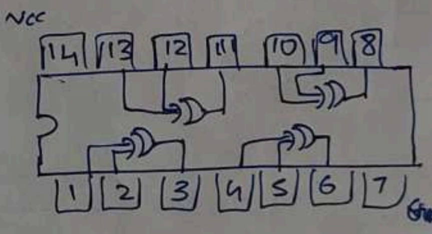
⇒ 7486 - quad 2-input XOR gate

⇒ 741G86 - single 2 input XOR gate

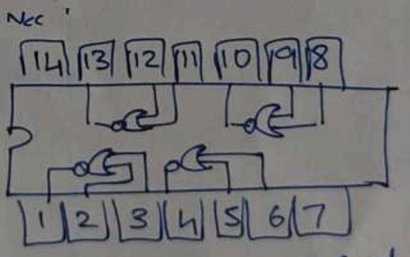
⇒ 74135 - quad exclusive -or

⇒ 74136 - quad 2-input XOR with open collector gate

⇒ 74386 - quad 2-input XOR



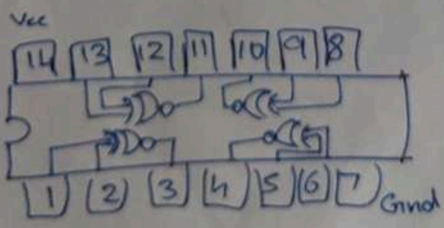
→ 7486



→ 74LS02

(e) XNOR Gate:

⇒ ~~7486~~ 74266 - quad 2-input XNOR gate with open collector output



⇒ 747266 - quad 2-input XNOR gate

4)

(i) $F(a,b,c,d,\cancel{e}) = \Pi M(0,2,4,5,10,11,13,15)$

ab \ cd	00	01	11	10
00	M0 0	M1	M3	M2 0
01	M4 0	M5 0	M7	M6
11	M12	M13 0	M15 0	M14
10	M8	M9	M11 0	M10 0

PI - (M0,M2), (M2,M10), (M0,M4), (M4,M5), (M5,M13), (M13,M15), (M15,M11), (M11, M10)

EPI - ~~(M0,M2), (M4,M5), (M13,M15), (M11,M10)~~ None

y=

(ii) $G(x1, x2, x3, x4) = \Sigma m(0,1,2,3,4,10,12,14)$

x1 x2 \ x3 x4	00	01	11	10
00	m0 1	m1 1	m3 1	m2 1
01	m4 1	m5	m7	m6
11	m12 1	m13	m15	m14 1
10	m8	m9	m11	m10 1

PI - (m0,m1,m3,m2), (m0,m4), (m4,m12), (m14,m10), (m14,m12), (m~~0~~10, m2)

EPI - (m0, m1, m3, m2), ~~(m4, m12)~~, ~~(m14, m10)~~

(iii) $H(a,b,c,d) = \Pi M(0,1,3,5,7,8,10)$

ab \ cd	00	01	11	10
00	M0 0	M1 0	M3 0	M2
01	M4	M5 0	M7 0	M6
11	M12	M13	M15	M14
10	M8 0	M9	M11	M10 0

PI - (M1, M3, M5, M7), (M0,M1), (M0,M8), (M8,M10)

EPI - (M1, M3, M5, M7), ~~(M0, M1)~~, (M8,M10)

(iv) $K(x_1, x_2, x_3, x_4) = \Sigma m(0, 1, 2, 3, 5, 8, 9, 10, 11, 12)$

x_1x_2 \ x_3x_4	00	01	11	10
00	m0 1	m1 1	m3 1	m2 1
01	m4	m5 1	m7	m6
11	m12 1	m13	m15	m14
10	m8 1	m9 1	m11 1	m10 1

P.I - (m0, m1, m3, m2, m8, m9, m11, m10), (m12, m8), (m1, m5),

EPI - (m0, m1, m3, m2, m8, m9, m11, m10), (m12, m8), (m1, m5).

5)

(a) 4 bit input (b1, b2, b3, b4) is sent to output (y1, y2, y3, y4) only if the entered password is same as P1, P2, P3, P4, P5.

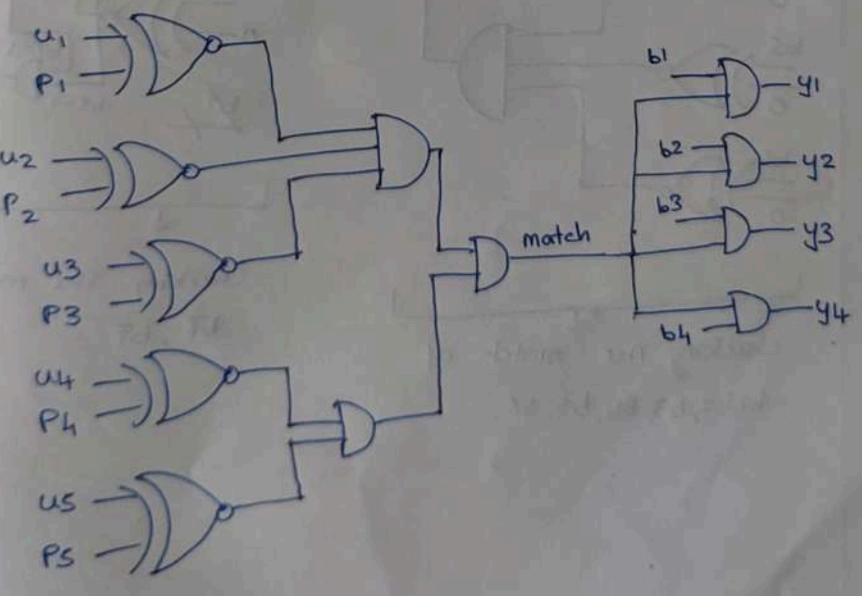
User entered password - u1, u2, u3, u4, u5

~~Hint~~:

→ passwords can be matched using XNOR gates

→ only if passwords match, b1b2b3b4 is sent to y1y2y3y4.

Logic circuit:



(b)

→ To check if The 8 bit ASCII input corresponds to The letters - 'a', 'b' or 'c'.

'a' - 01100001 (97)

'b' - 01100010 (98)

'c' - 01100011 (99)

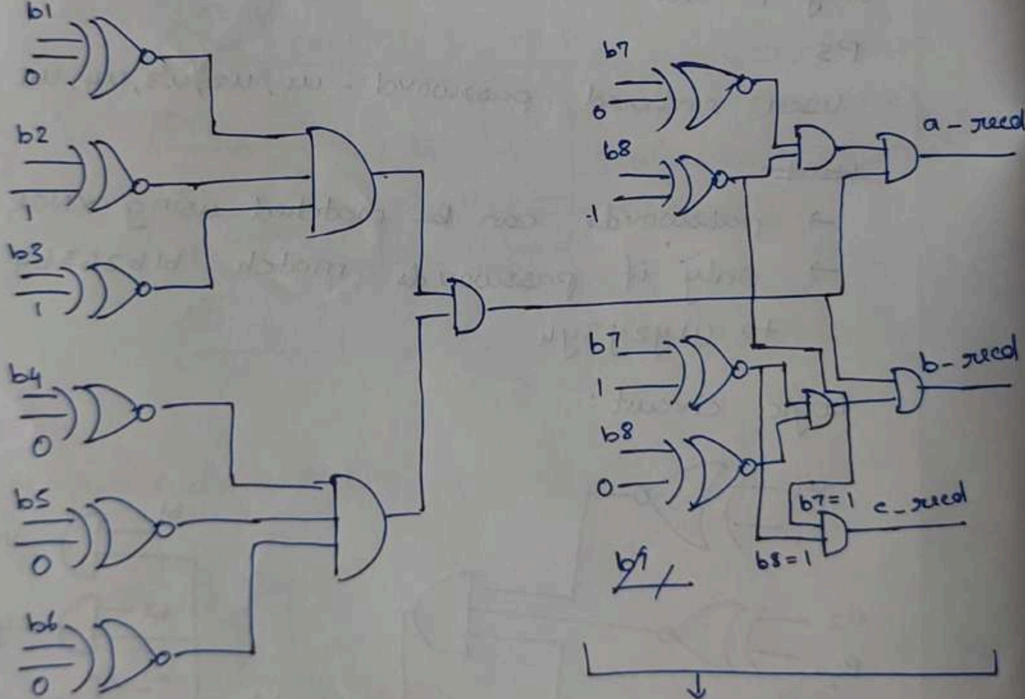
↳ The last two bits are changing - b7, b8

↳ The first six bits are same - b1, b2, b3, b4, b5, b6

→ so only after The first 6 bits are matched we proceed to check b7, b8 for 'a', 'b', 'c'

b7 b8	letter
01	a
10	b
11	c

b1	b2	b3	b4	b5	b6
0	1	1	0	0	0

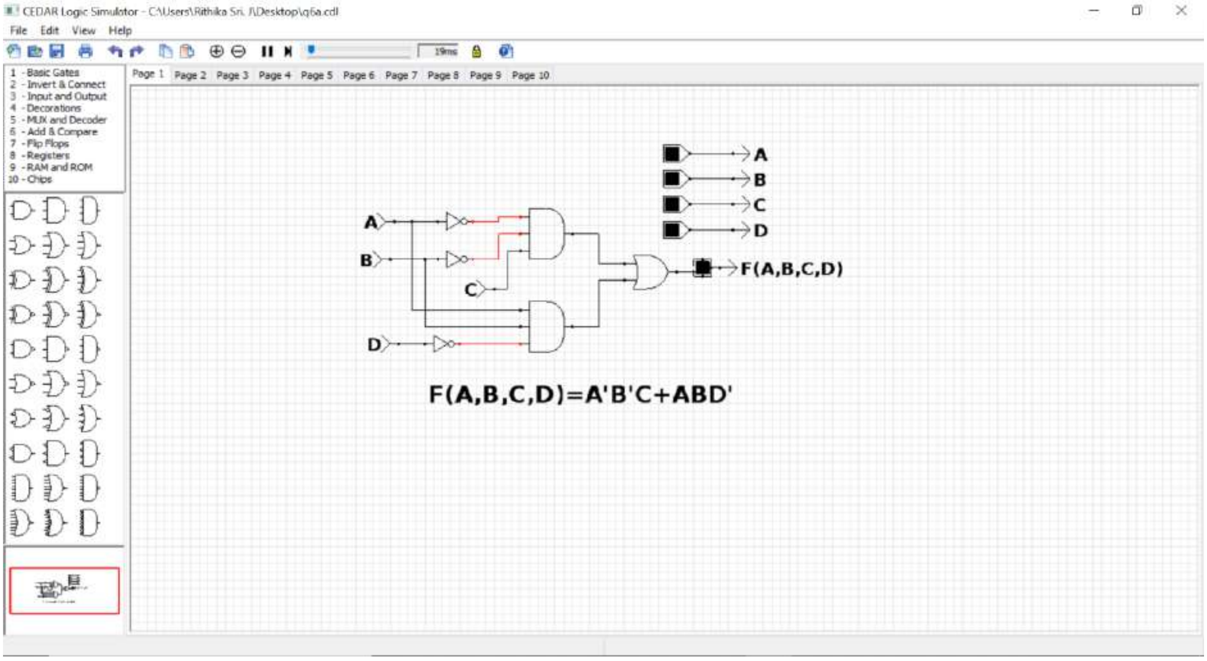


Checking The match of b8, b8

checking The match of b1, b2, b3, b4, b5, b6.

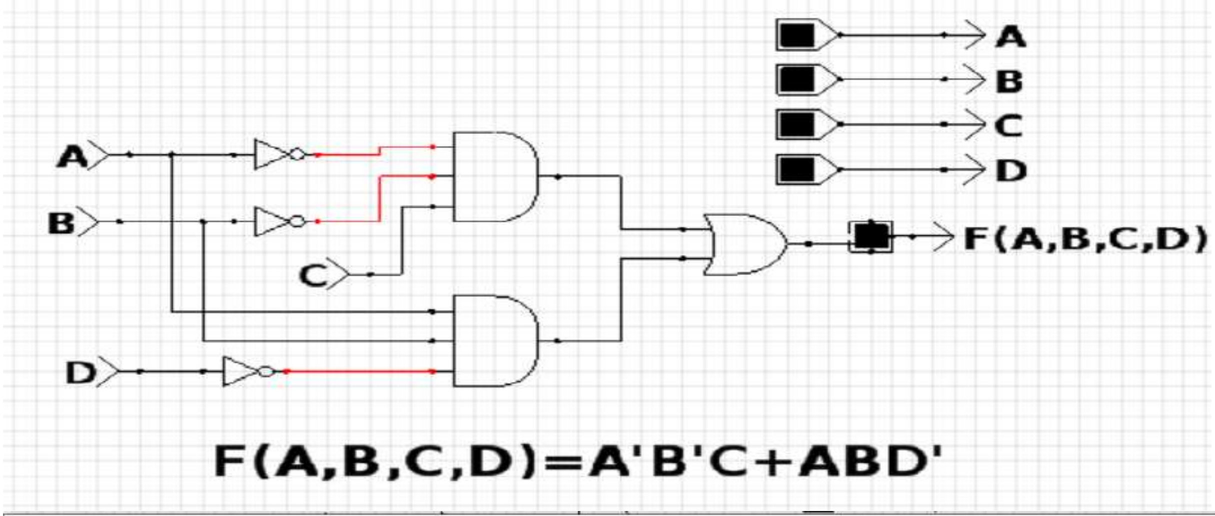
6)

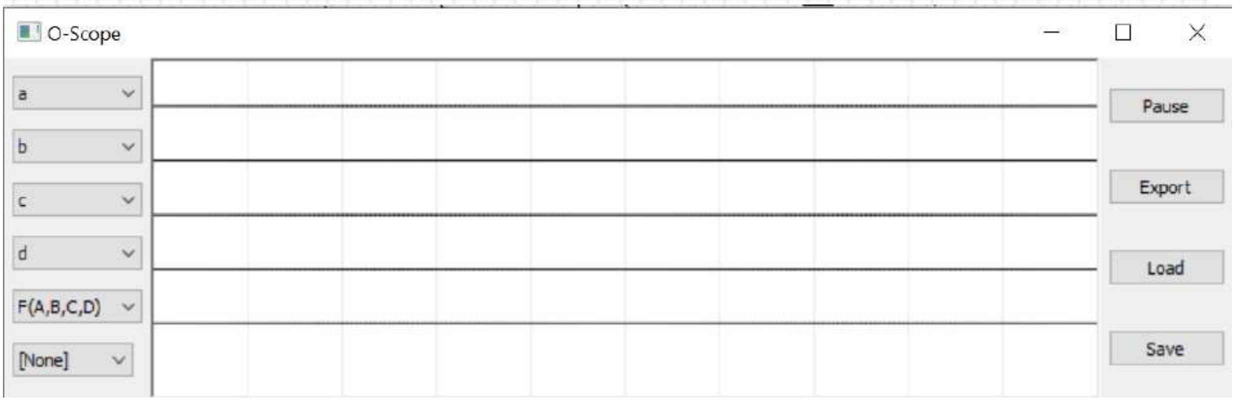
(a)F(A,B,C,D) = A’B’C + ABD’



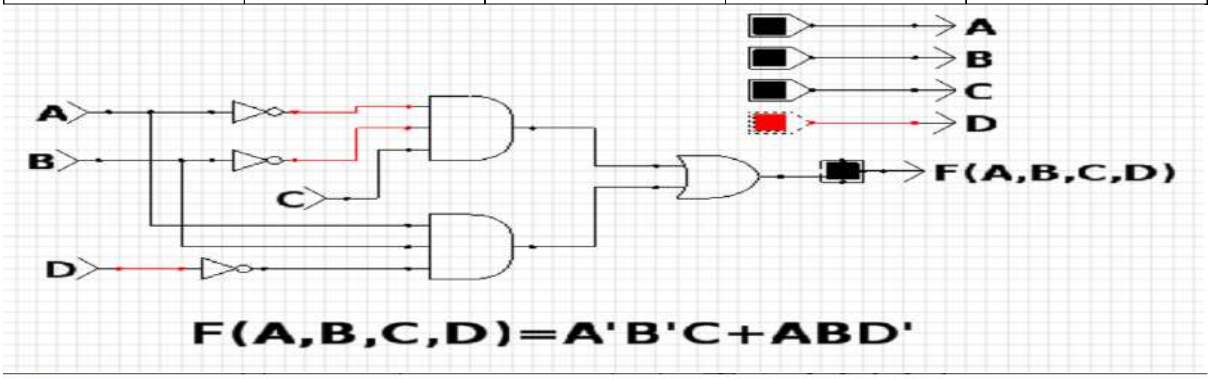
TRUTH TABLE:

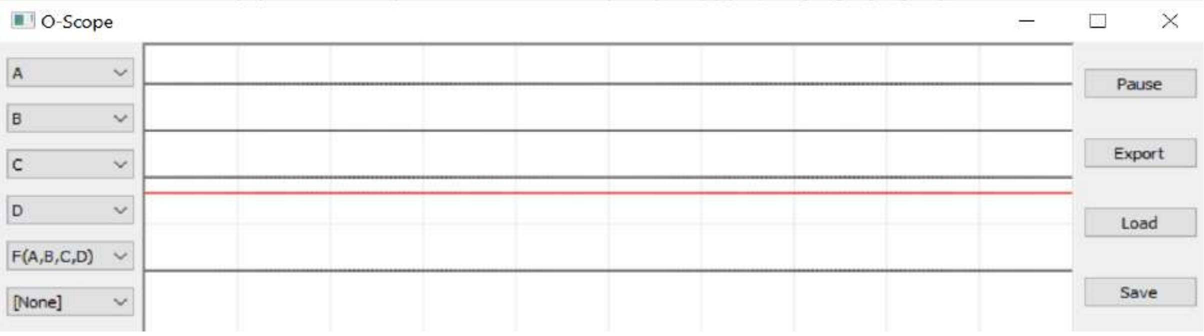
A	B	C	D	F
0	0	0	0	0



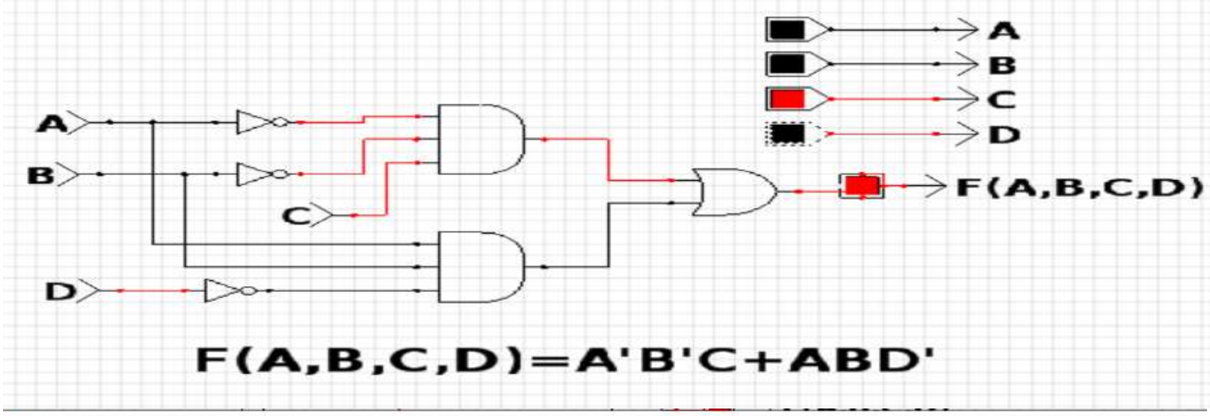


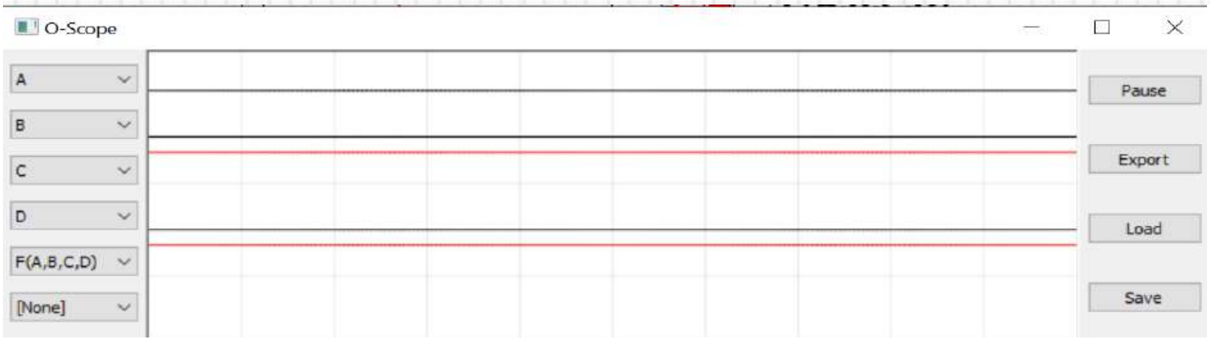
A	B	C	D	F
0	0	0	1	0



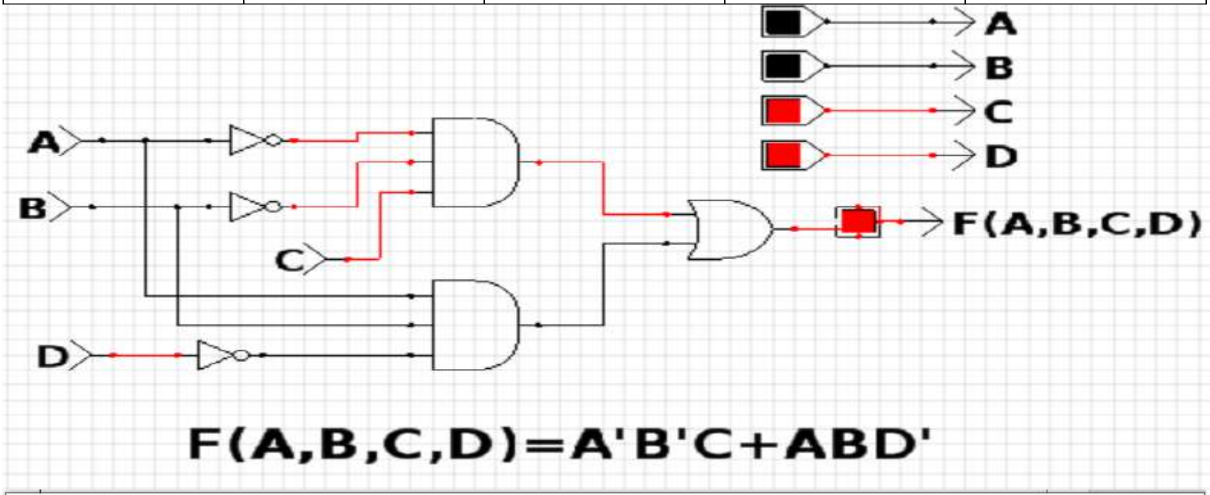


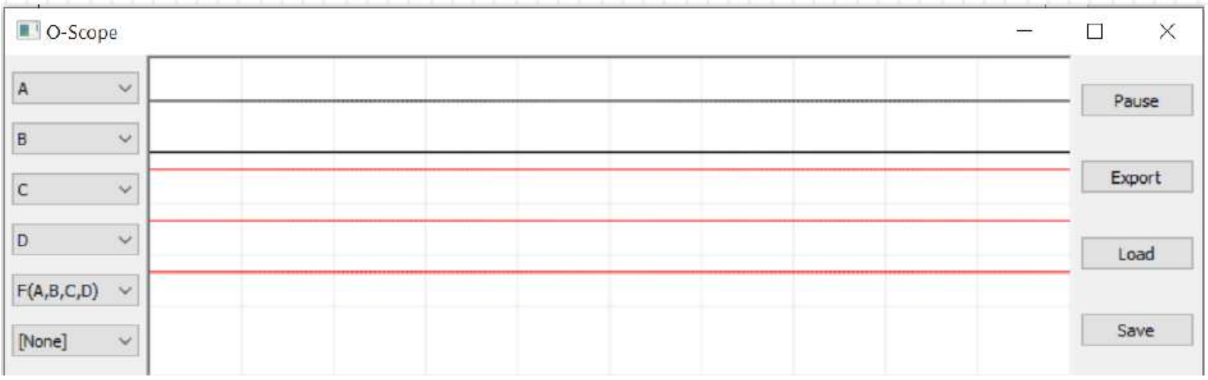
A	B	C	D	F
0	0	1	0	1



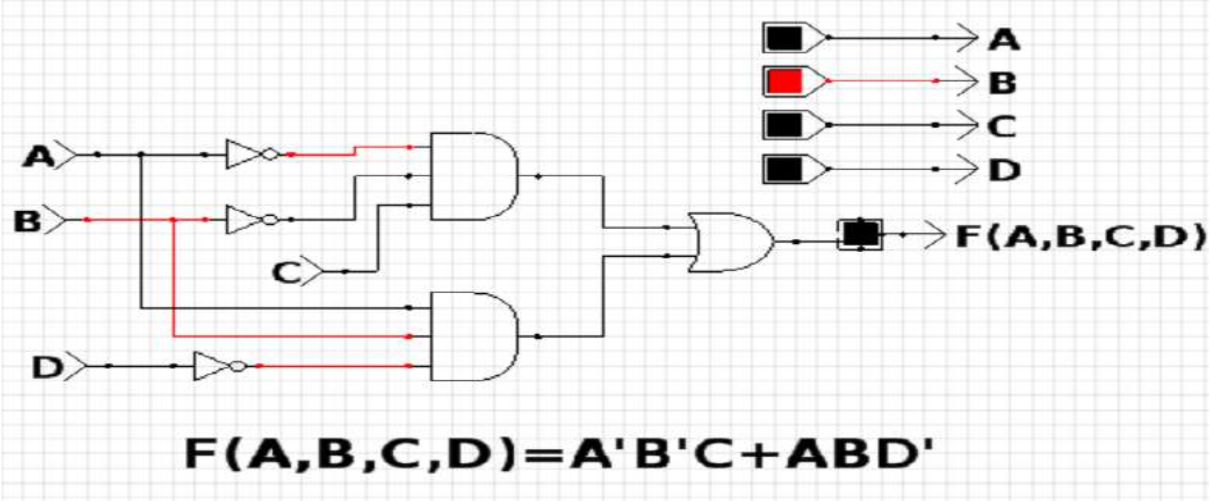


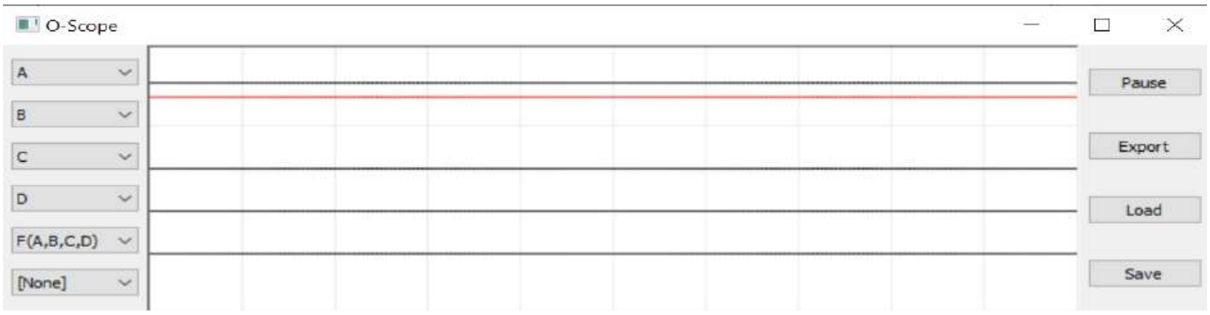
A	B	C	D	F
0	0	1	1	1



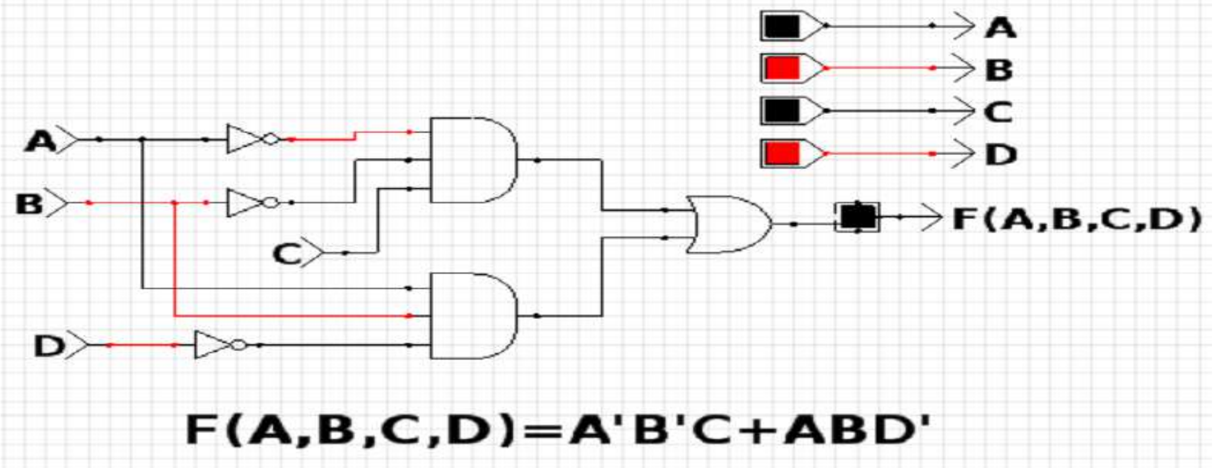


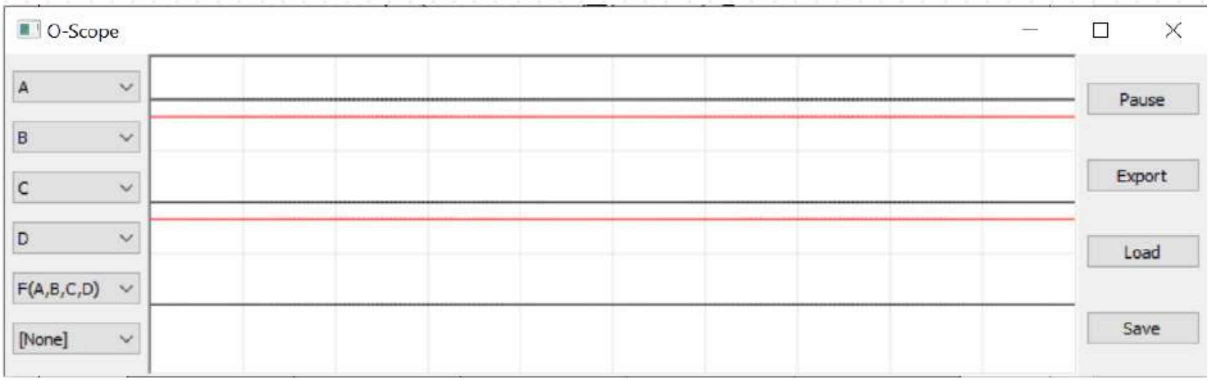
A	B	C	D	F
0	1	0	0	0



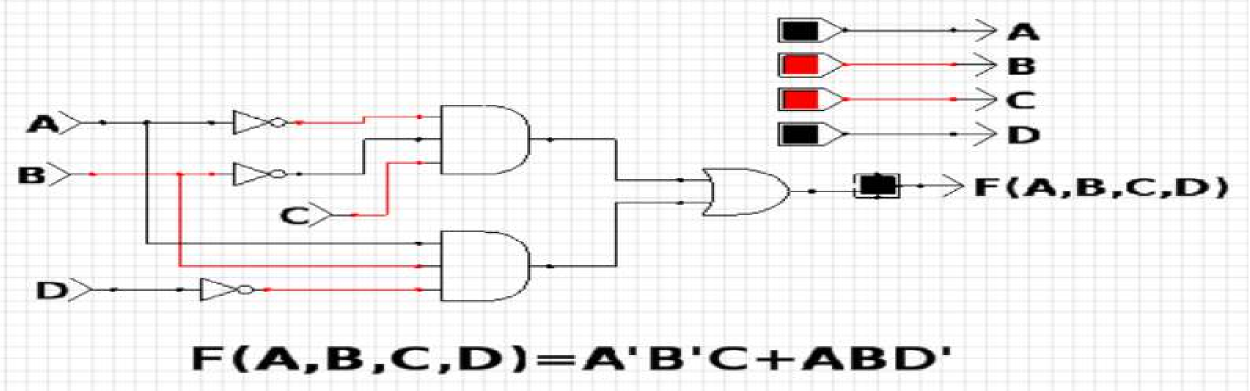


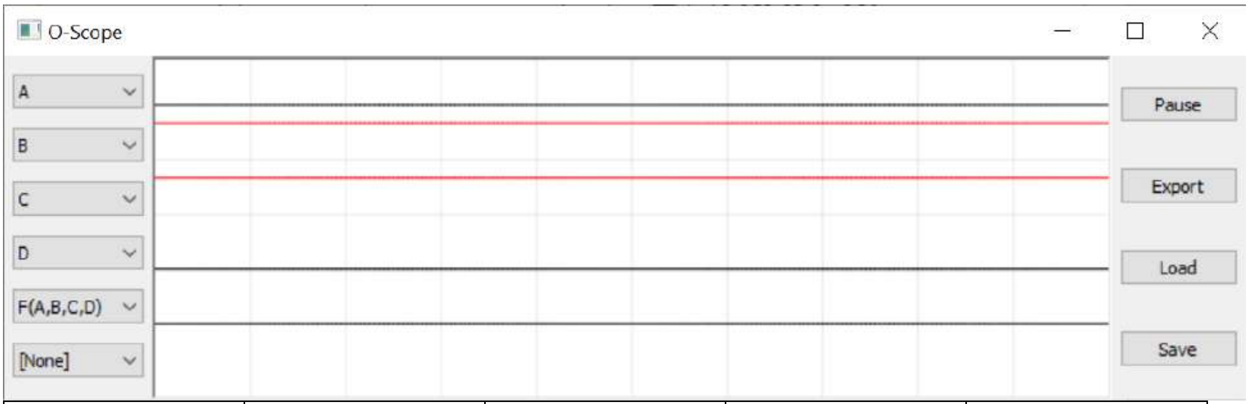
A	B	C	D	F
0	1	0	1	0



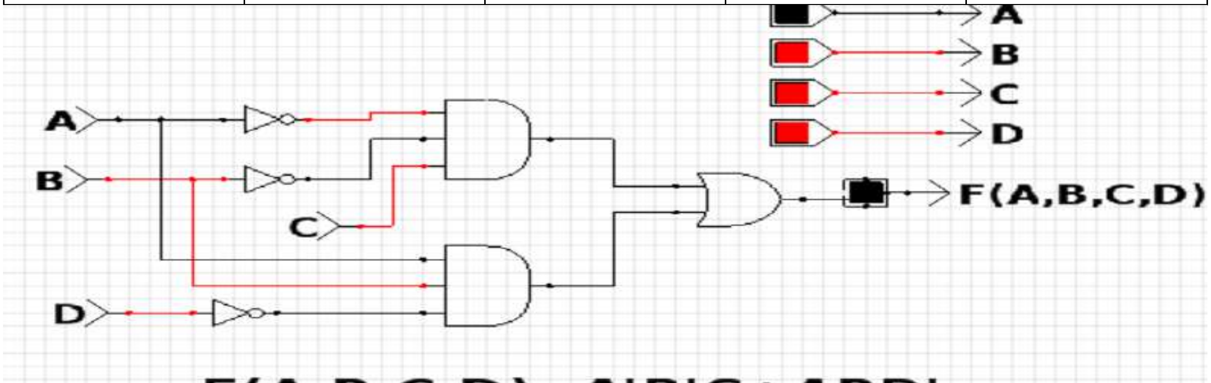


A	B	C	D	F
0	1	1	0	0

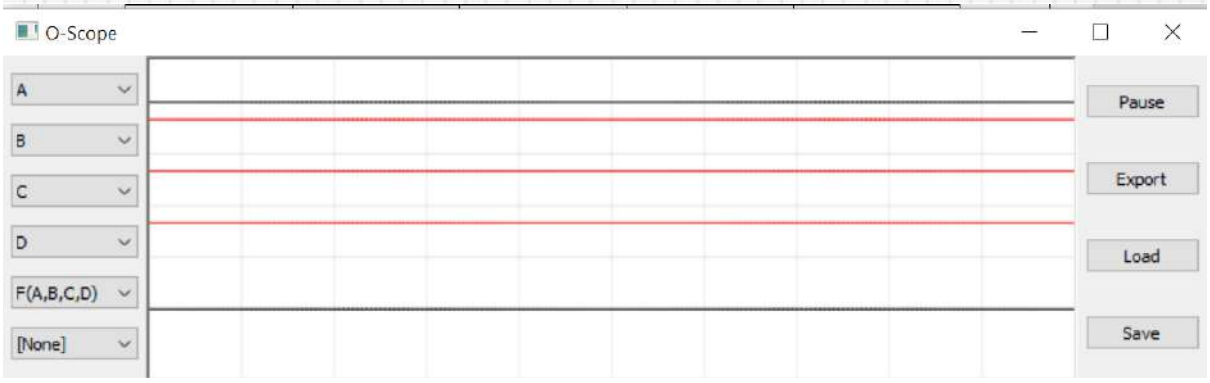




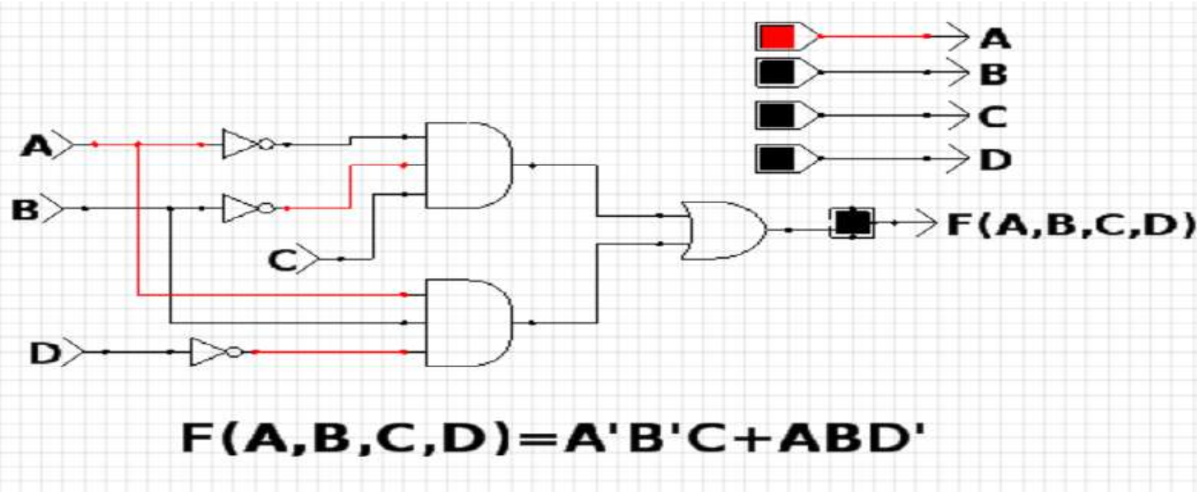
A	B	C	D	F
0	1	1	1	0



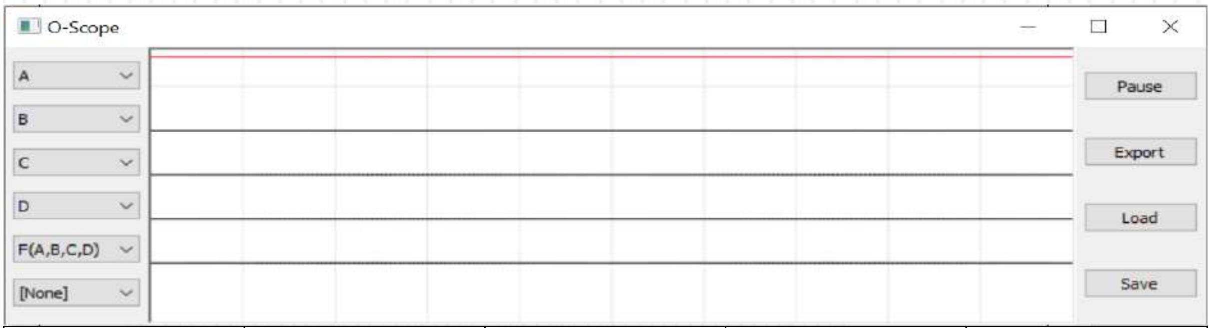
F(A,B,C,D)=A'B'C+ABD'



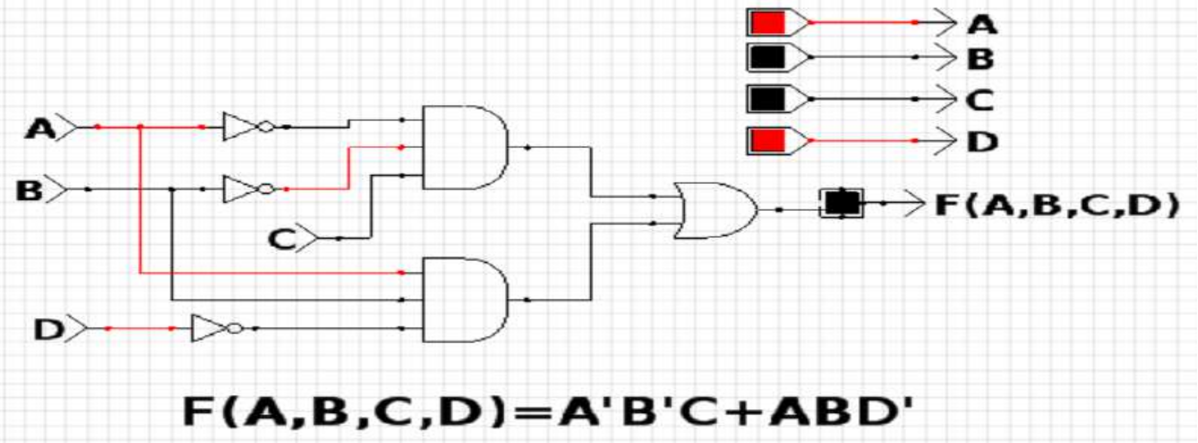
A	B	C	D	F
1	0	0	0	0



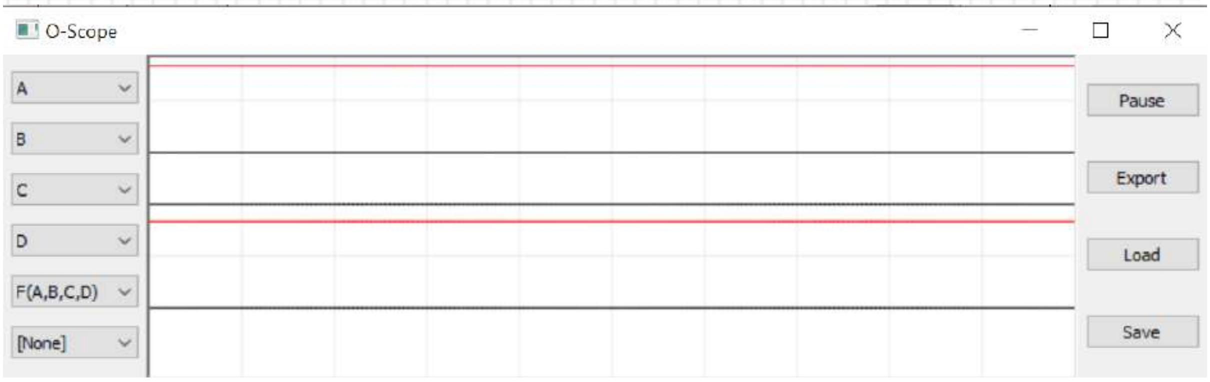
F(A,B,C,D)=A'B'C+ABD'



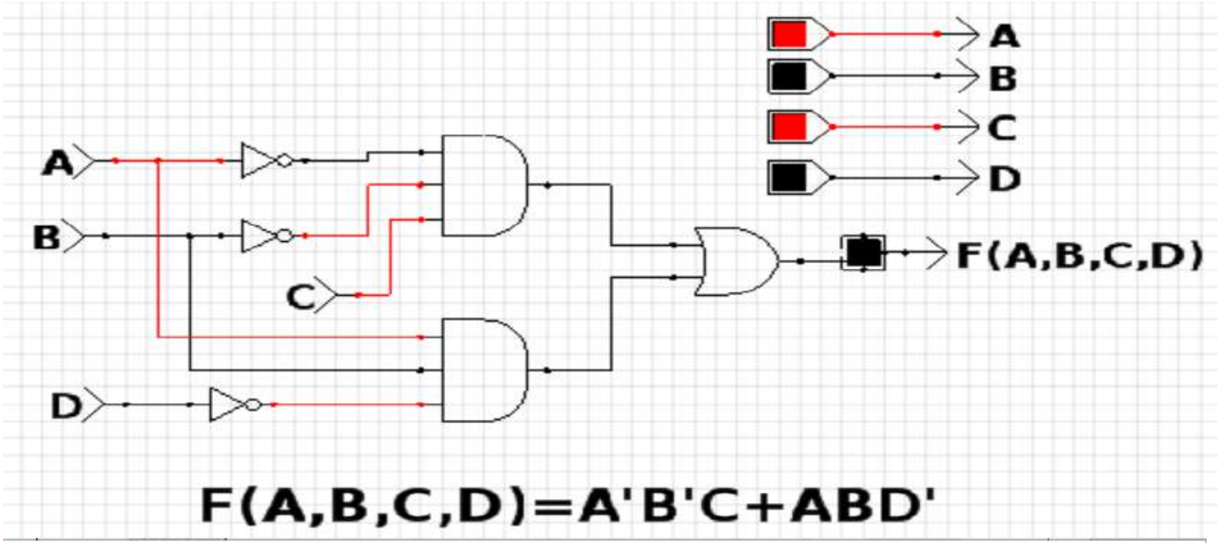
A	B	C	D	F
1	0	0	1	0



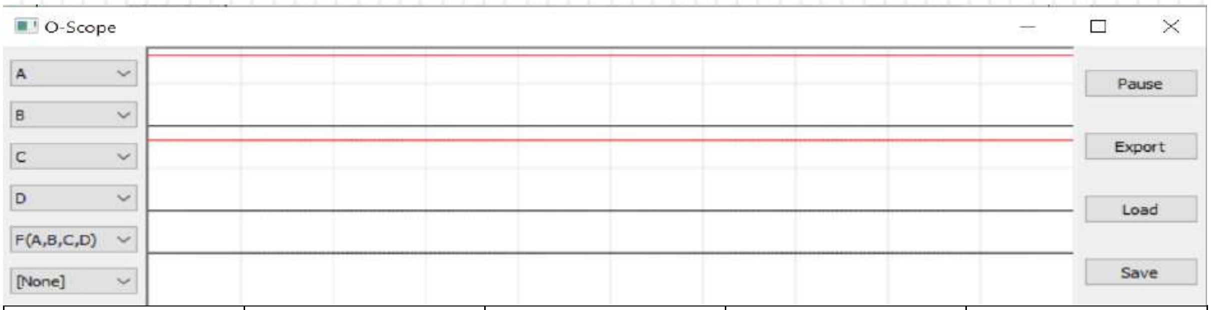
F(A,B,C,D)=A'B'C+ABD'



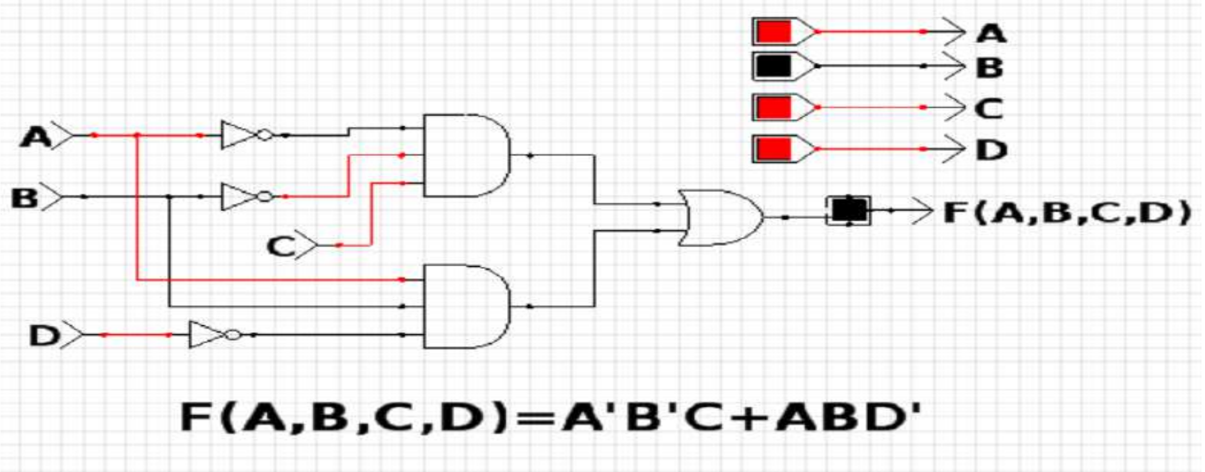
A	B	C	D	F
1	0	1	0	0



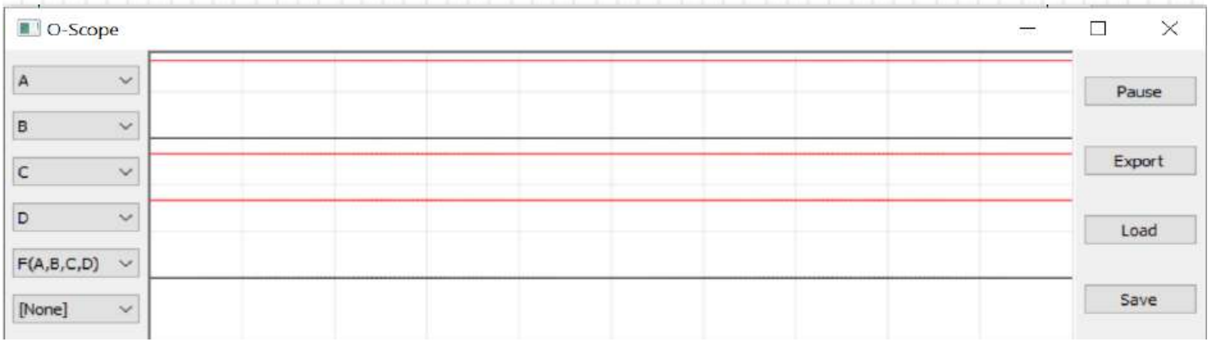
F(A,B,C,D)=A'B'C+ABD'



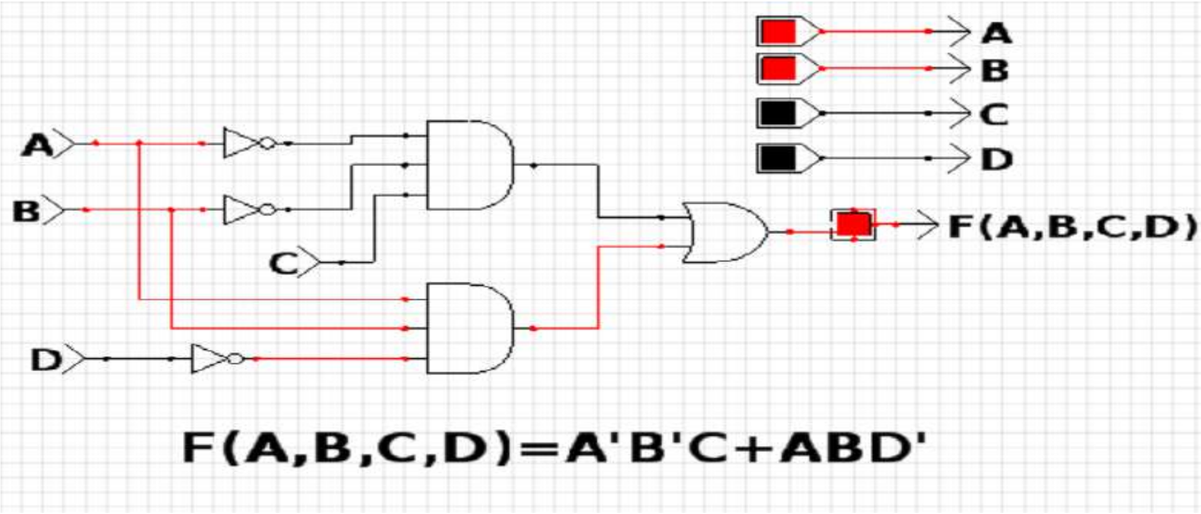
A	B	C	D	F
1	0	1	1	0

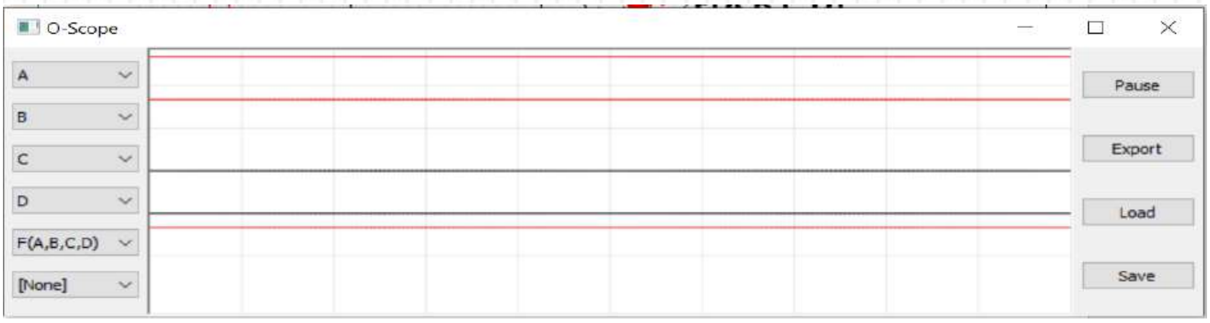


F(A,B,C,D)=A'B'C+ABD'

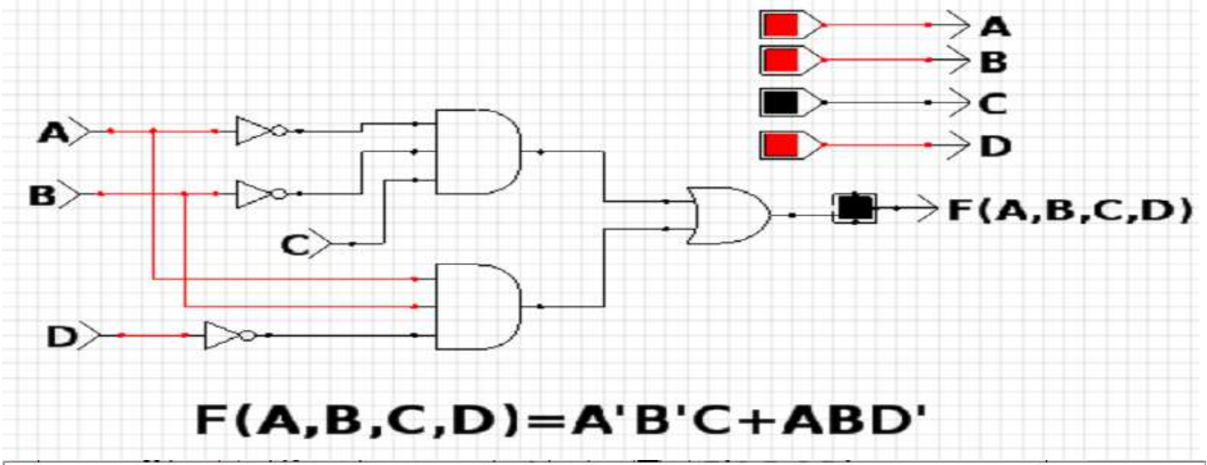


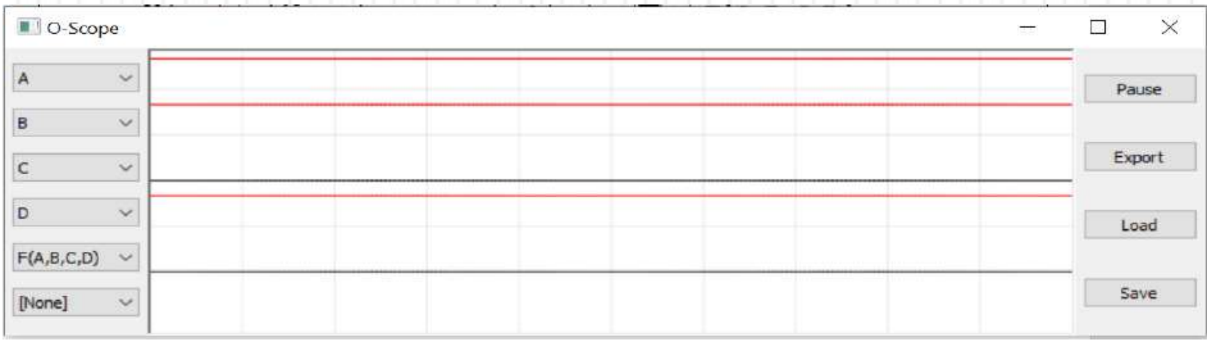
A	B	C	D	F
1	1	0	0	1





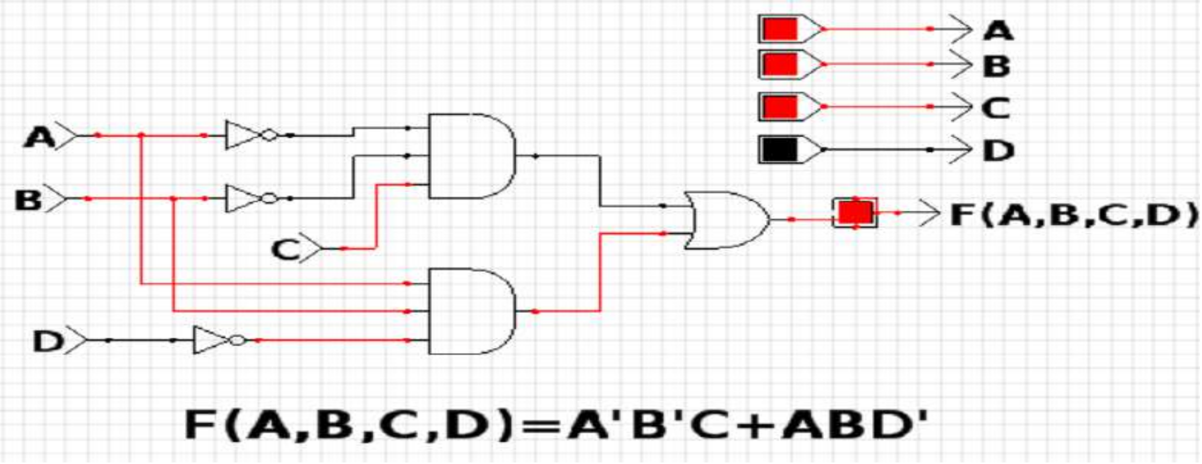
A	B	C	D	F
1	1	0	1	0

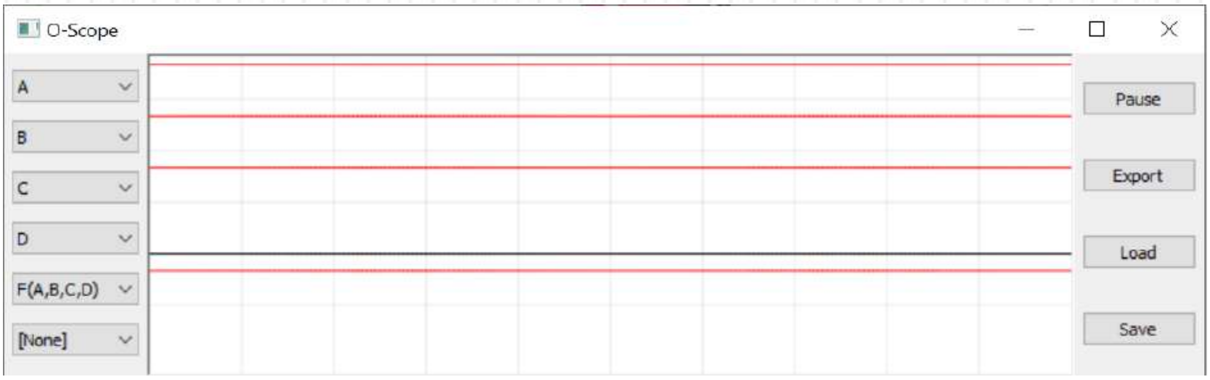




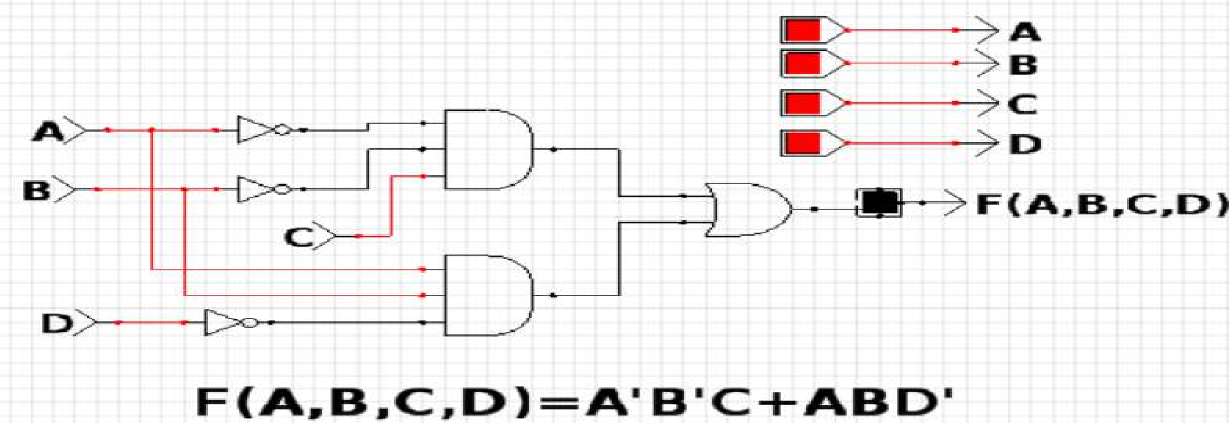
A	B	C	D	F

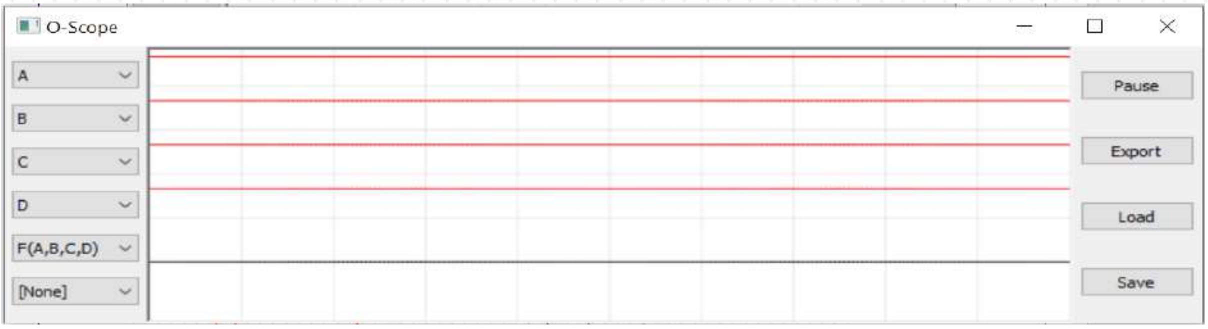
1	1	1	0	1



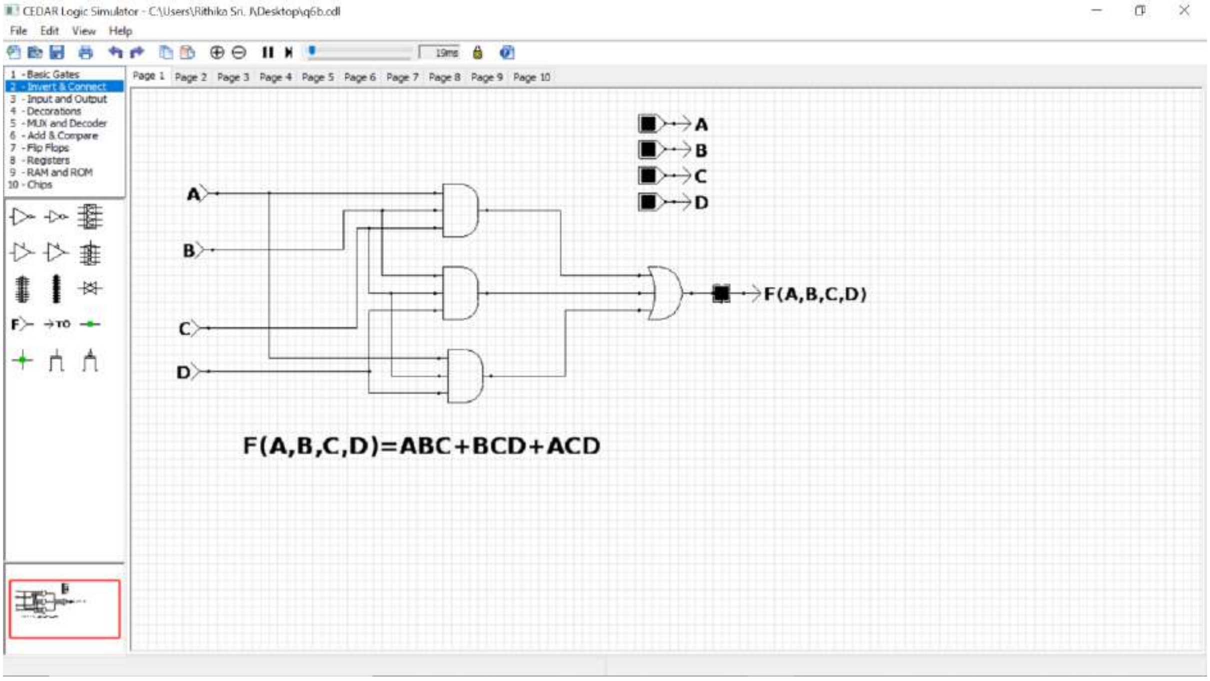


A	B	C	D	F
1	1	1	1	0





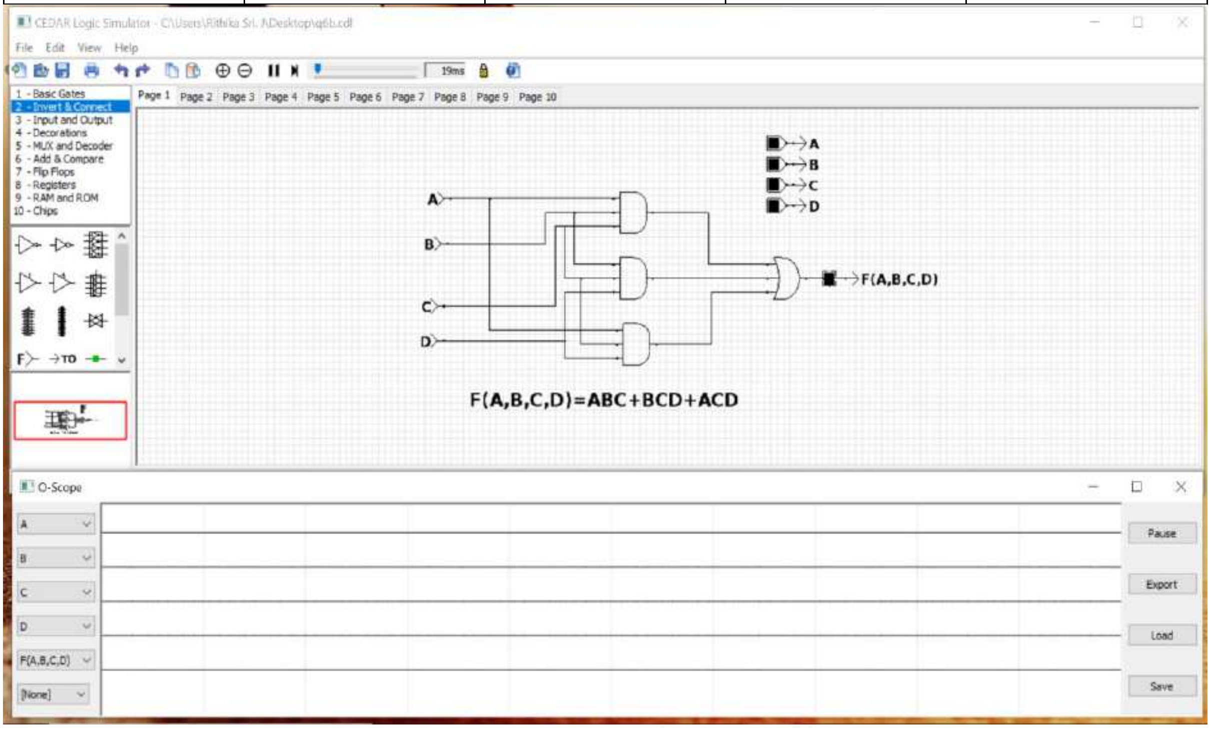
(b) F(A,B,C,D) = ABC + BCD + ACD


CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

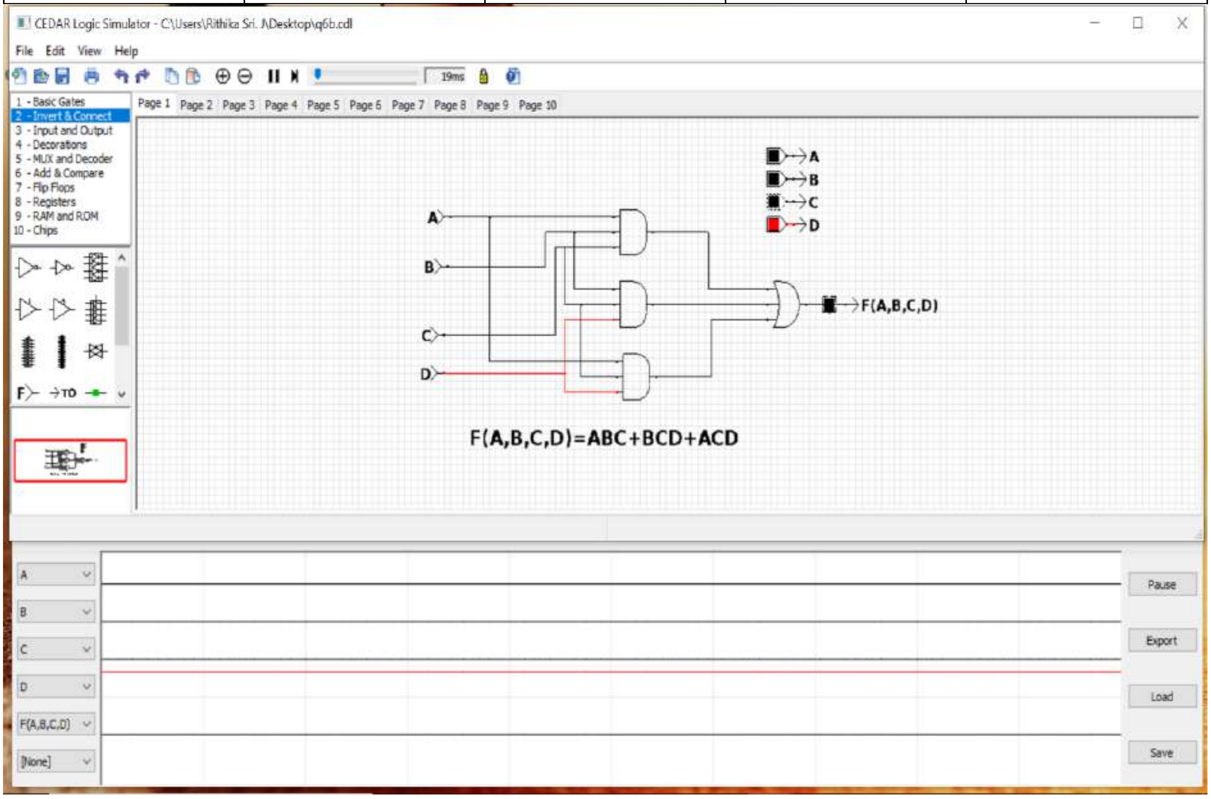
TRUTH TABLE:

A	B	C	D	F
0	0	0	0	0

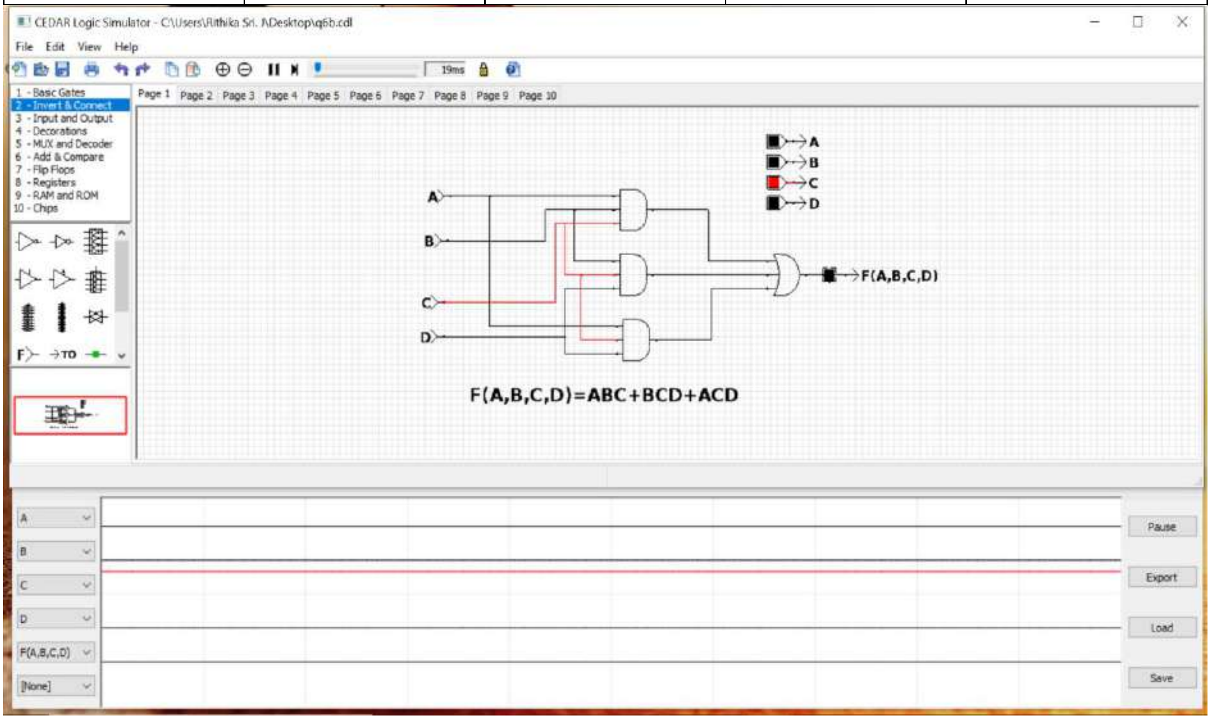

CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

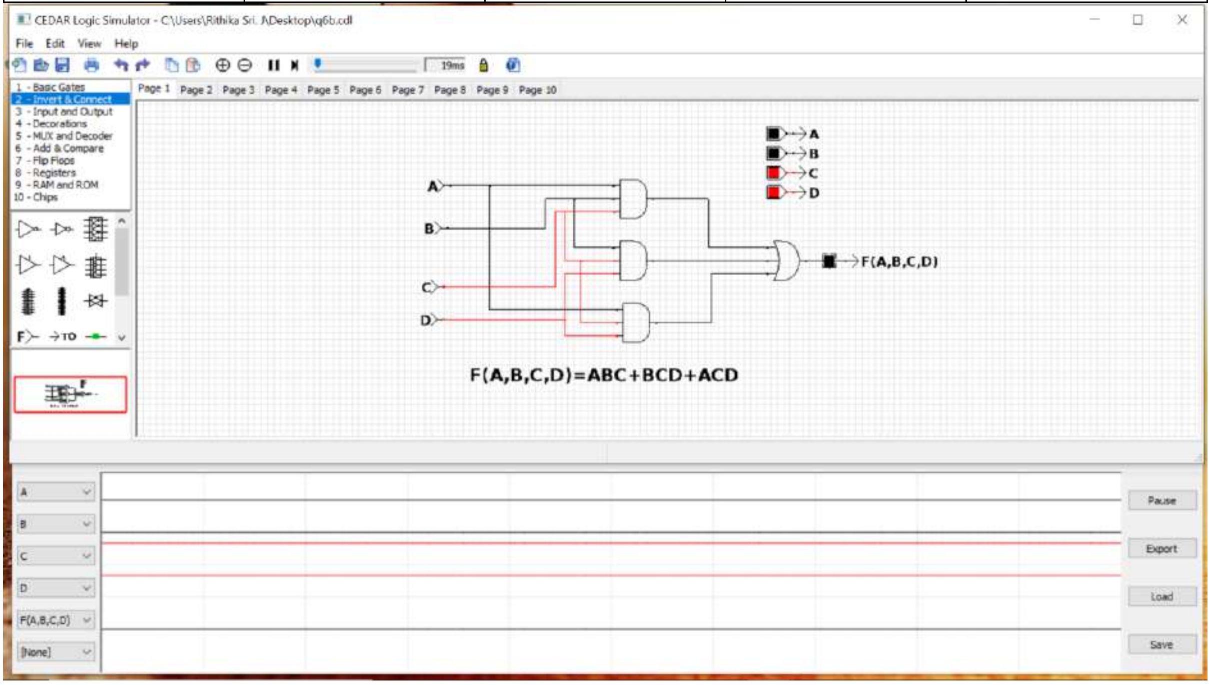
A	B	C	D	F
0	0	0	1	0



A	B	C	D	F
0	0	1	0	0

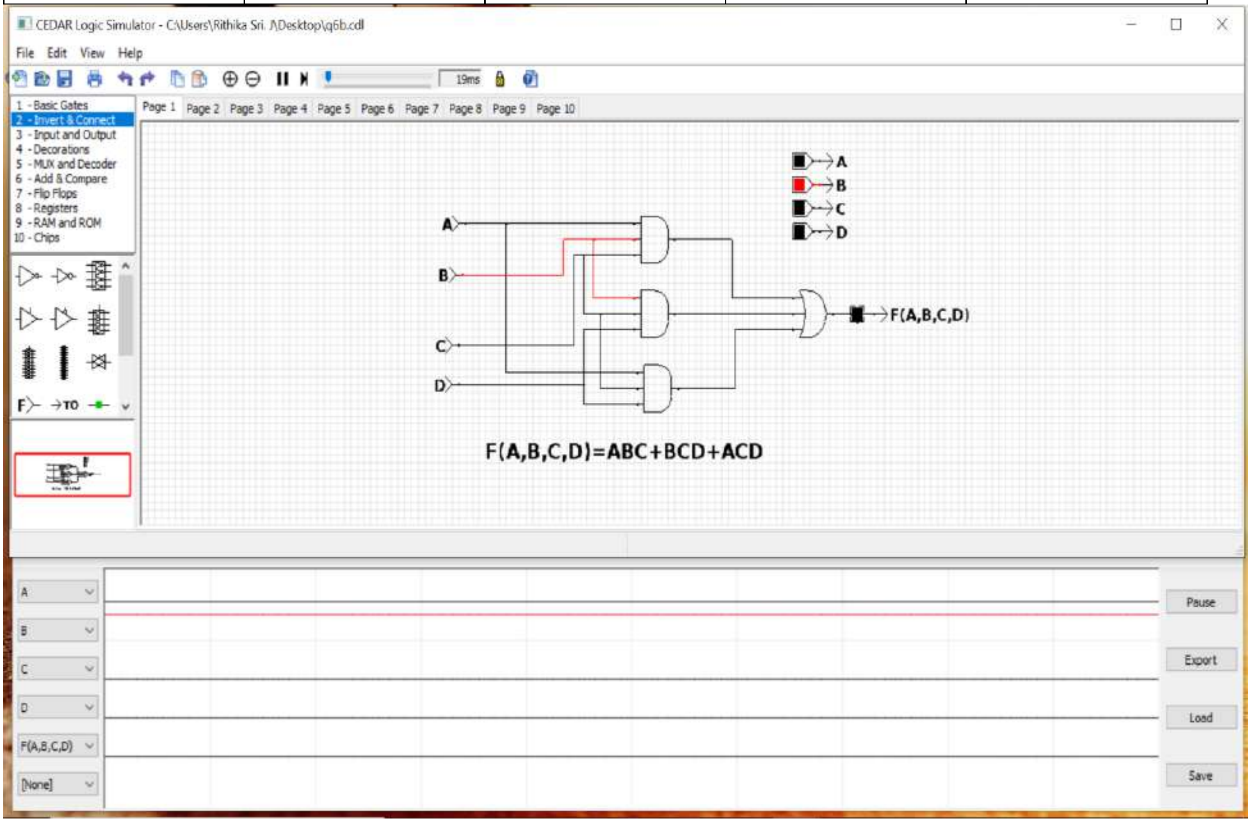


A	B	C	D	F
0	0	1	1	0



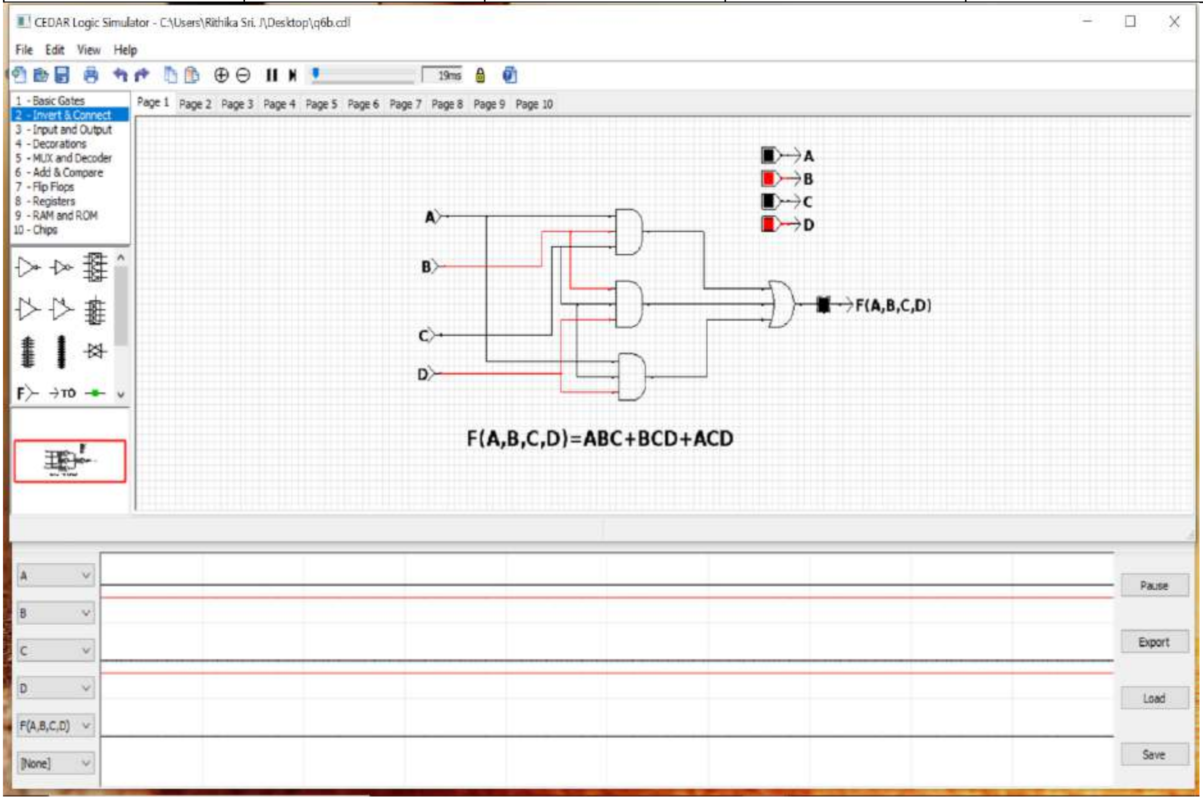
F(A,B,C,D)=ABC+BCD+ACD

A	B	C	D	F
0	1	0	0	0

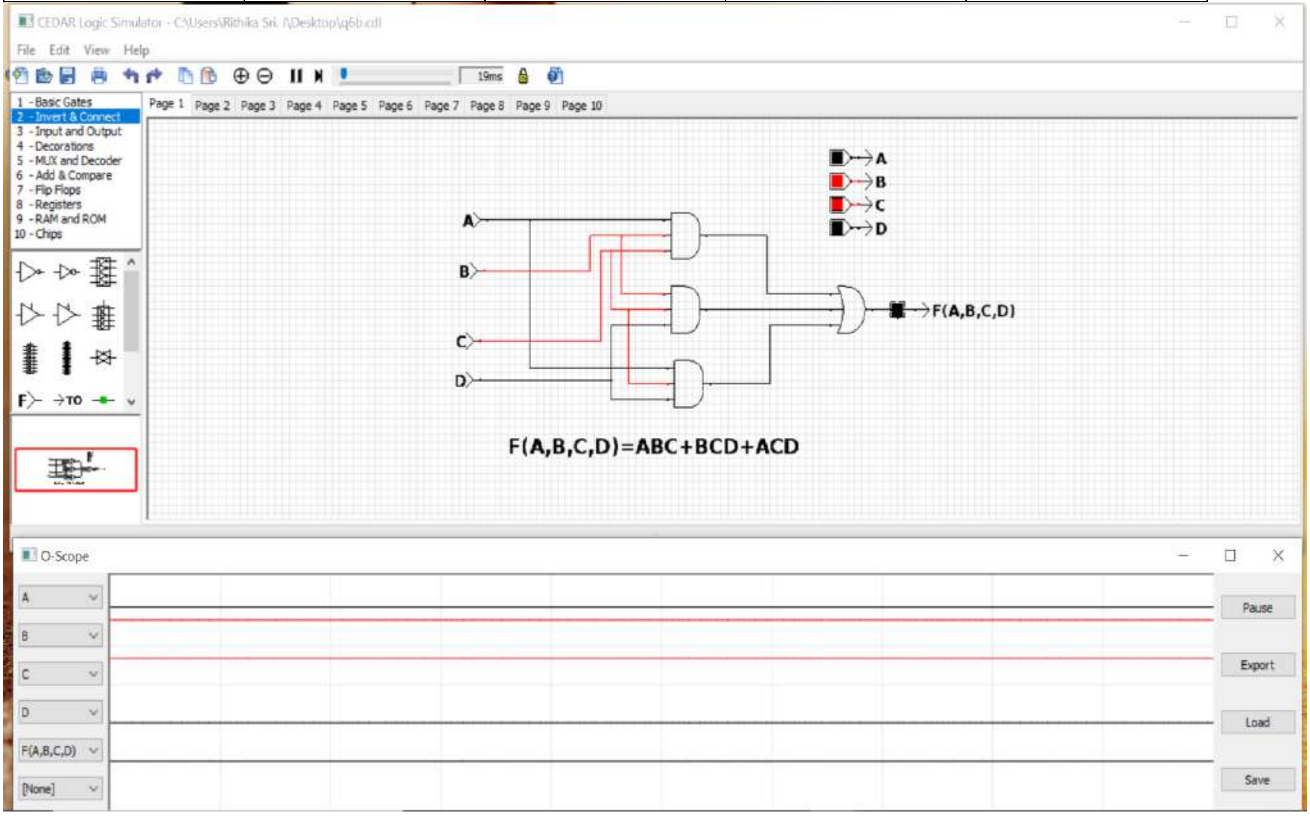


F(A,B,C,D)=ABC+BCD+ACD

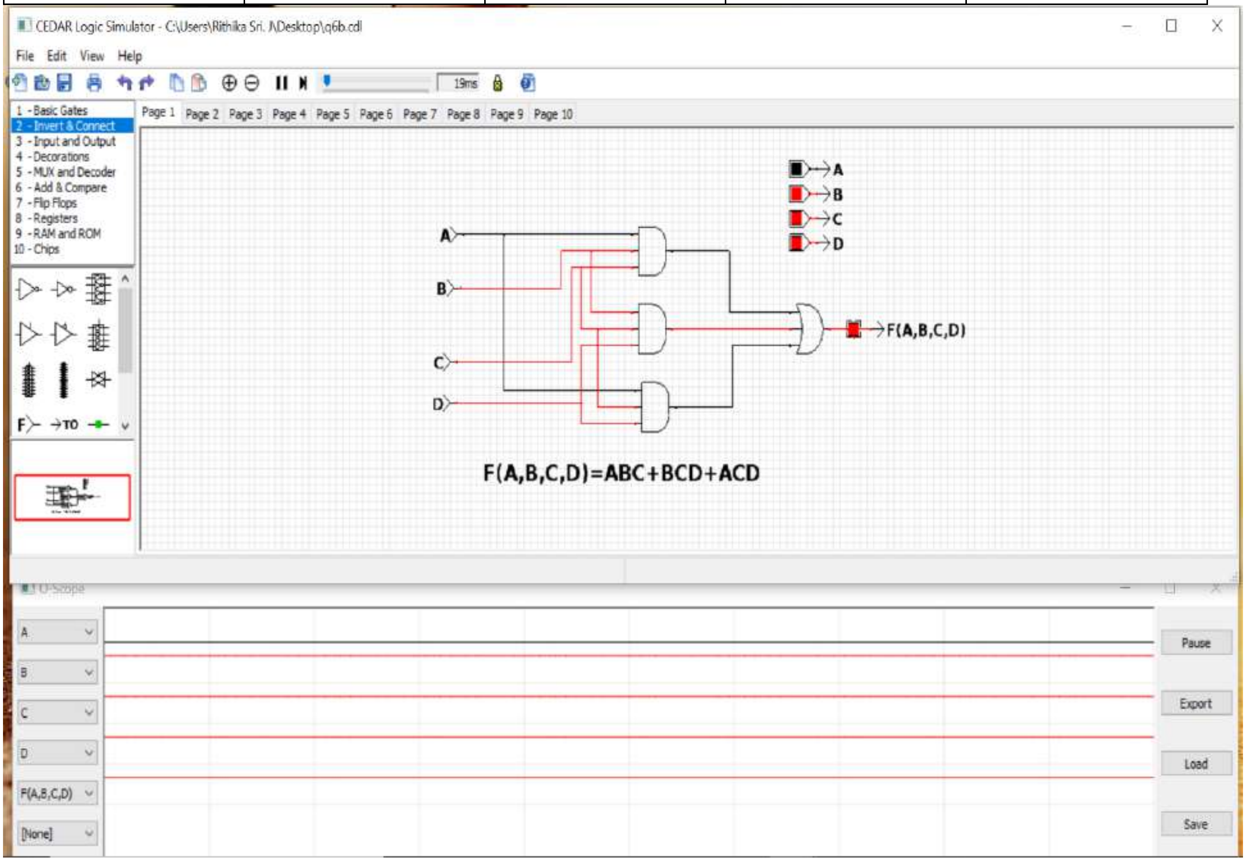
A	B	C	D	F
0	1	0	1	0



A	B	C	D	F
0	1	1	0	0



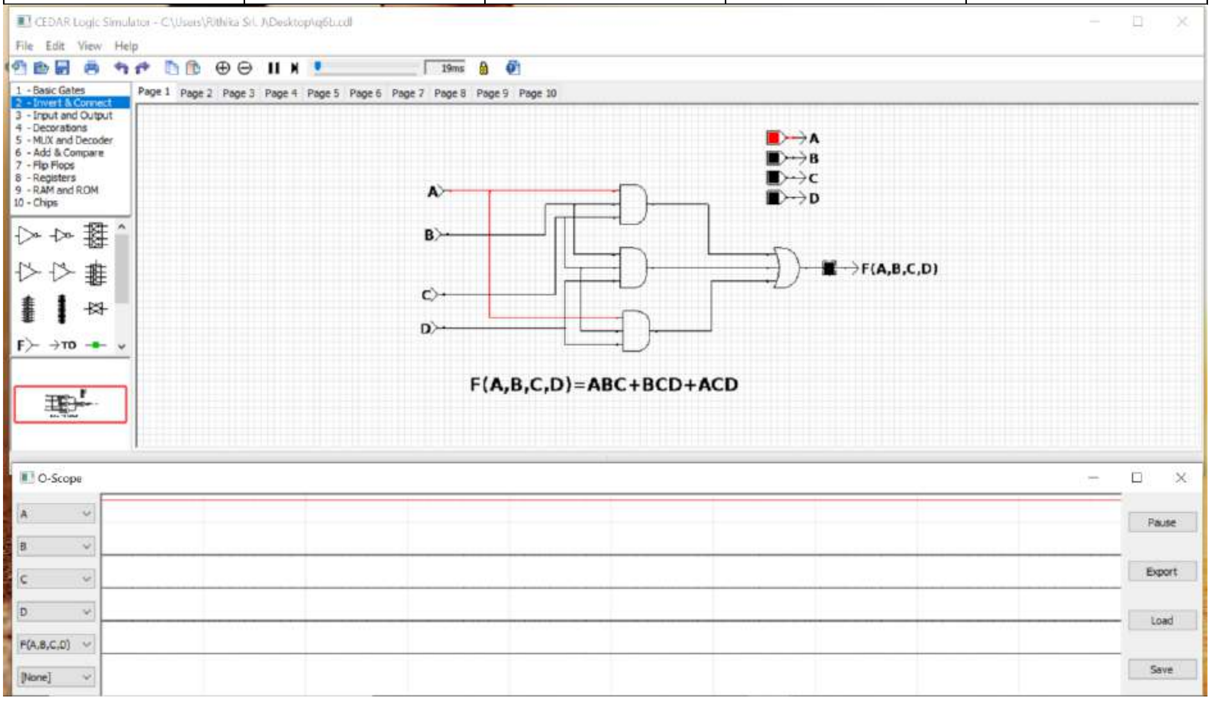
A	B	C	D	F
0	1	1	1	1



CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

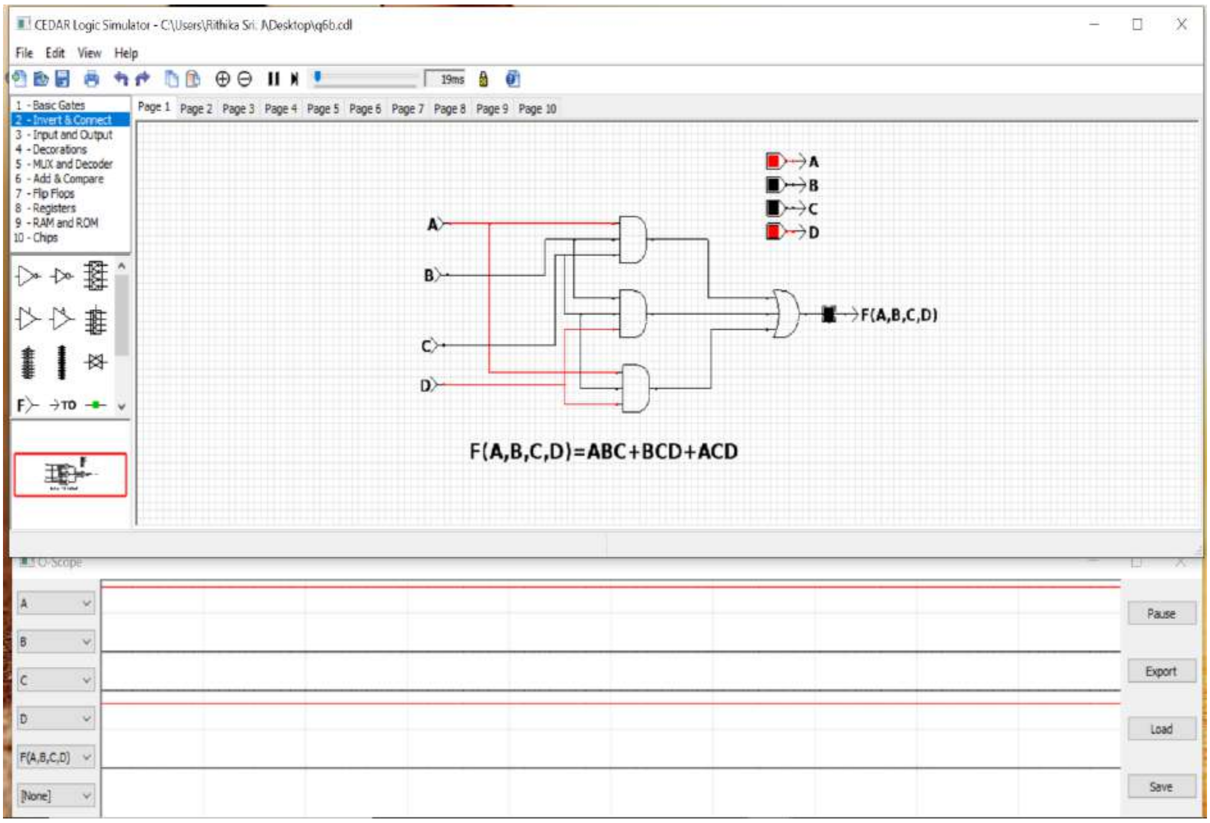
A	B	C	D	F
1	0	0	0	0



CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

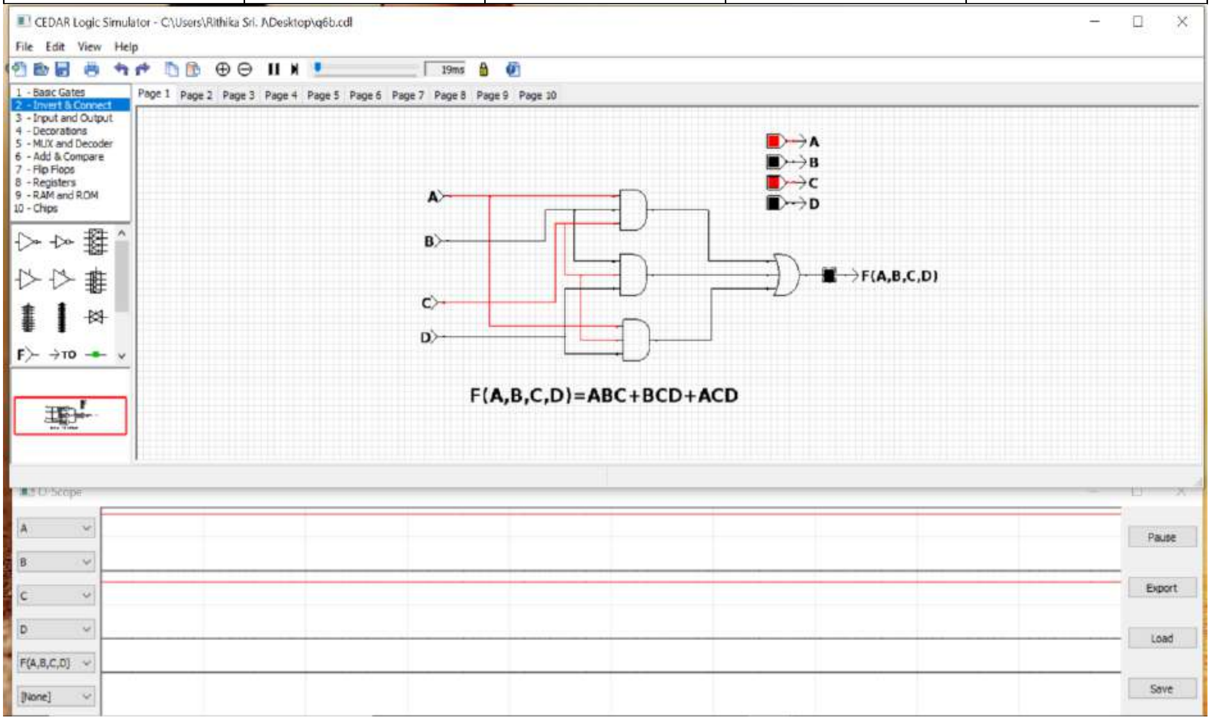
A	B	C	D	F
1	0	0	1	0



CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

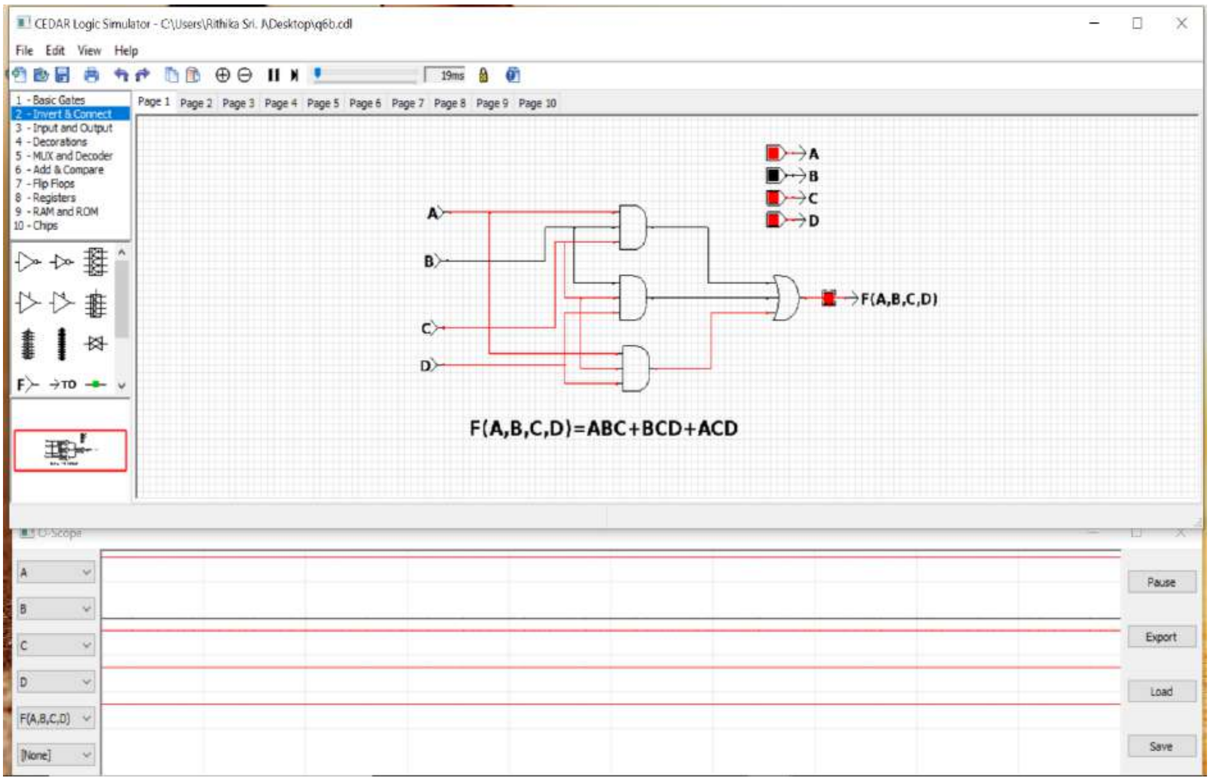
A	B	C	D	F
1	0	1	0	0



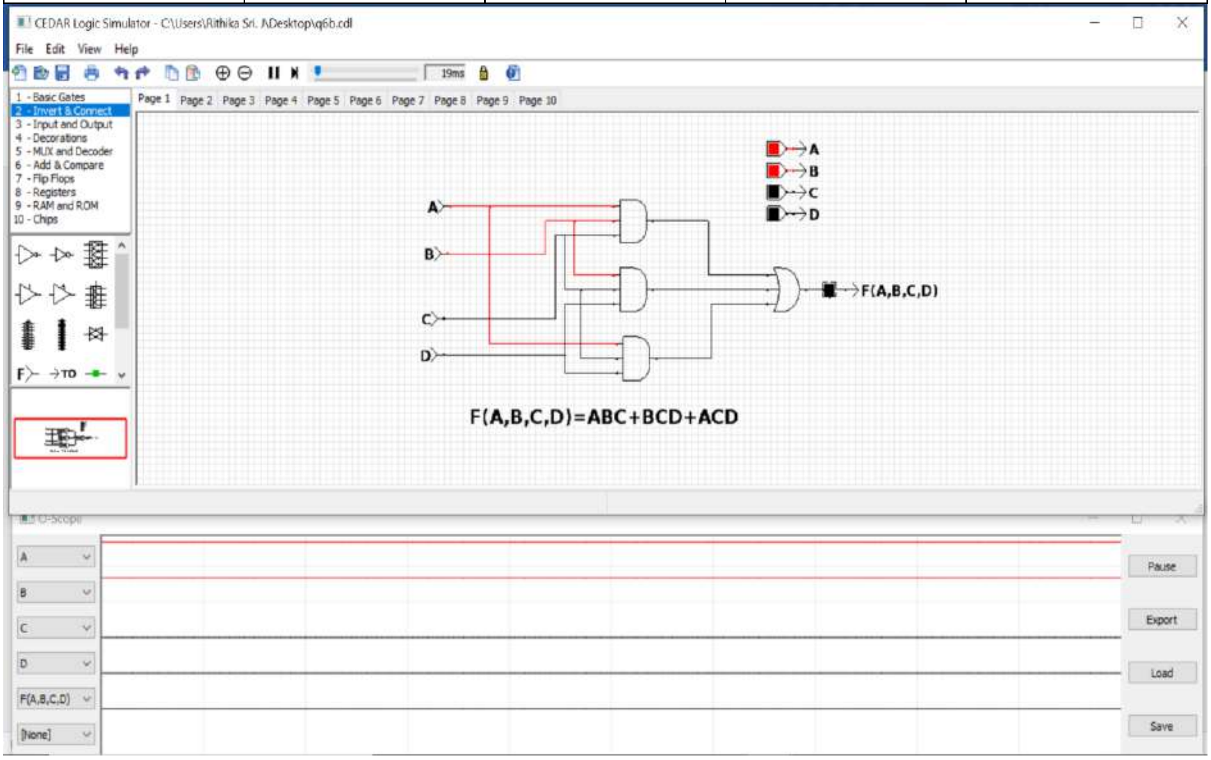
CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

A	B	C	D	F
1	0	1	1	1

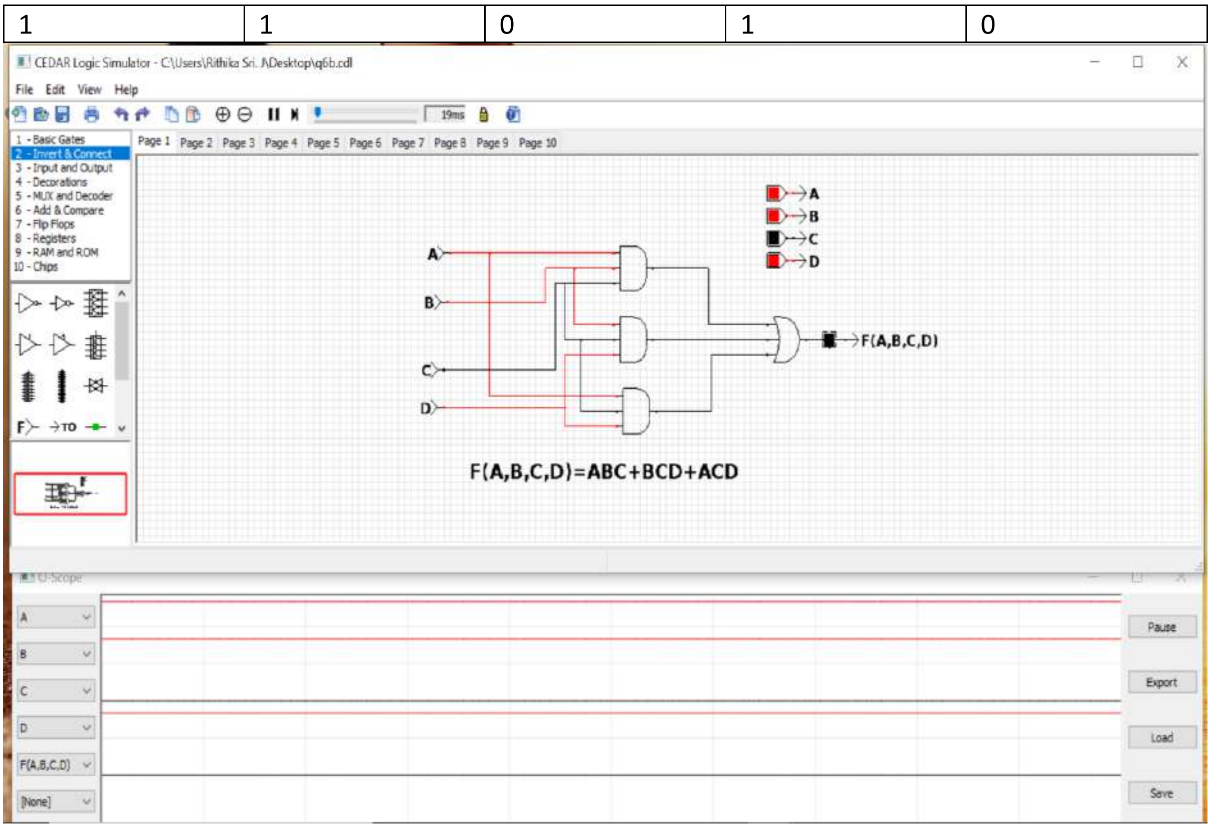


A	B	C	D	F
1	1	0	0	0



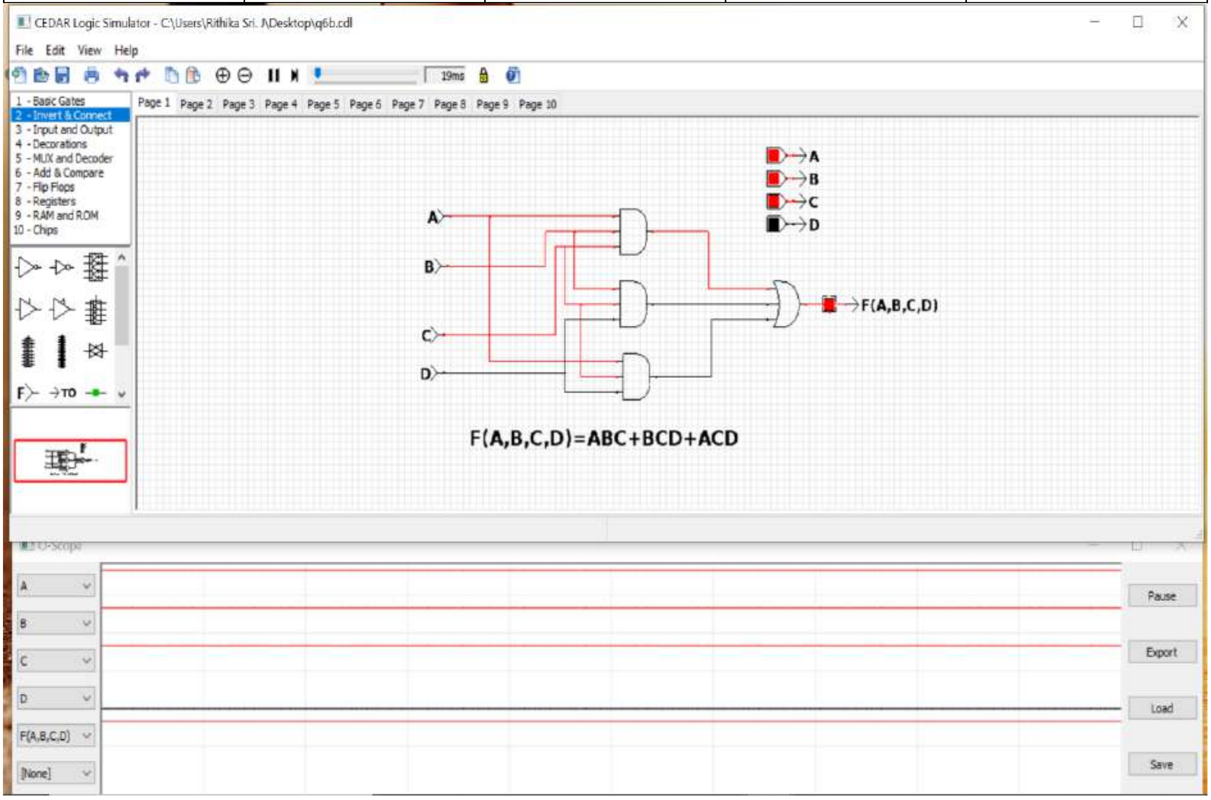
A	B	C	D	F

1	1	0	1	0



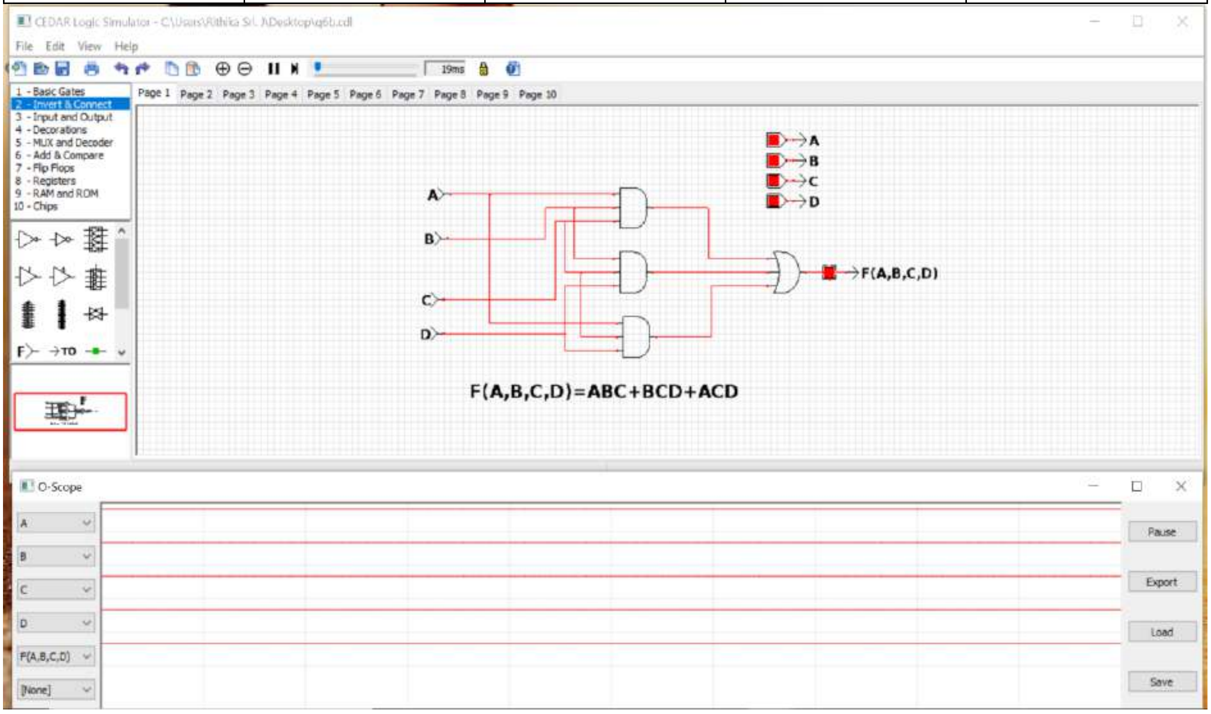
F(A,B,C,D)=ABC+BCD+ACD

A	B	C	D	F
1	1	1	0	1



F(A,B,C,D)=ABC+BCD+ACD

A	B	C	D	F
1	1	1	1	1



CEDAR Logic Simulator - C:\Users\Rithika Sri. J\Desktop\q6b.cdl

F(A,B,C,D)=ABC+BCD+ACD

7) $F = \Sigma m(0,1,5,8,9)$

4 - bit input variables → a, b, c, d

don't cares → d(10, 11, 12, 13, 14)

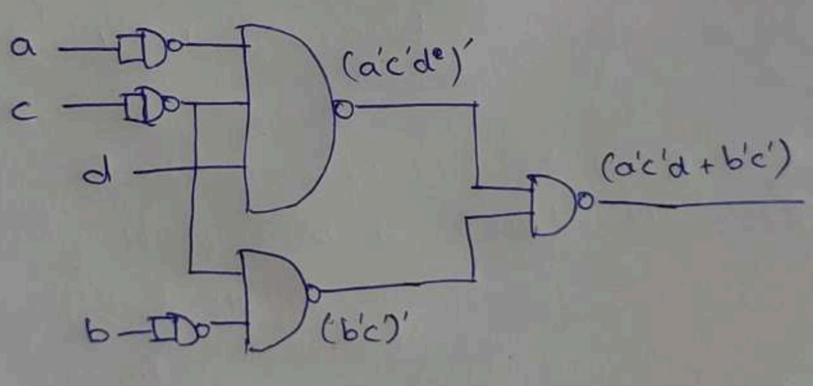
But using don't cares changes the output of the function.

ab \ cd	00	01	11	10
00	1	1		
01		1		
11				
10	1	1		

∴ SOP : $\bar{a}\bar{c}d + \bar{b}\bar{c}$

POS : $(a + c + \bar{d}) . (b + c)$

NAND Realization :



8) 4 bit input variables → a, b, c, d

From The question,

LED ON → m_1, m_3, m_5, m_7, m_9 (odd no.s)

→ m_{10}, m_{11}, m_{12}, m_{13}, m_{14}, m_{15} (no.s > 9)

LED OFF → m_0, m_2, m_4, m_6, m_8

$\therefore F(a,b,c,d) = \sum m(1,3,5,7,9,10,11,12,13,14,15)$

Step 1:

m1	0001
m3	0011
m5	0101
m9	1001
m10	1010
m12	1100
m_7	0111
m_{11}	1011
m13	1101
m14	1110
m15	1111

Step 2:

1,3	00-1	✓
1,5	0-01	✓
1,9	-001	✓
3,7	0-11	✓
3,11	-011	✓
5,7	01-1	✓
5,13	-101	✓
9,11	10-1	✓
9,13	1-01	✓
10,11	101-	✓
10,14	1-10	✓
12,13	110-	✓
12,14	11-0	✓
7,15	-111	✓
11,15	1-11	✓
13,15	11-1	✓
14,15	111-	✓

Step 3:

1,3,5,7	0--1	✓
1,3,9,11	~~0~~-0-1	✓
1,5,9,13	--01	✓
3,7,11,15	--11	✓
5,7,13,15	-1-1	✓
9,11,13,15	1--1	✓
10,11,14,15	1-1-	–
12,13,14,15	11--	✓

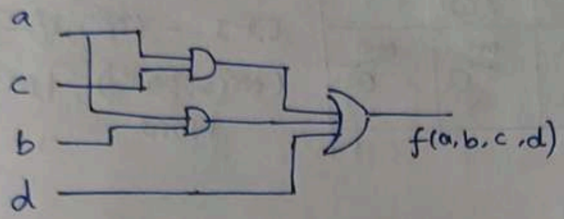
Step 4:

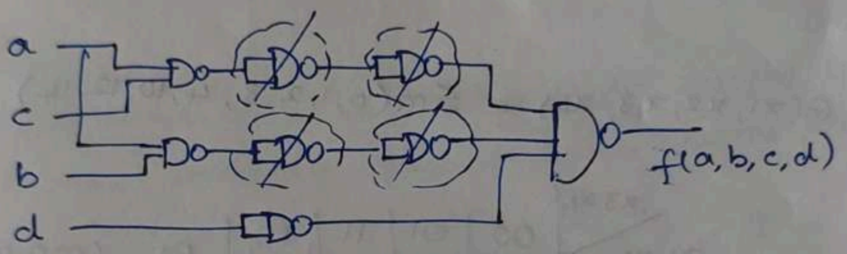
P.I

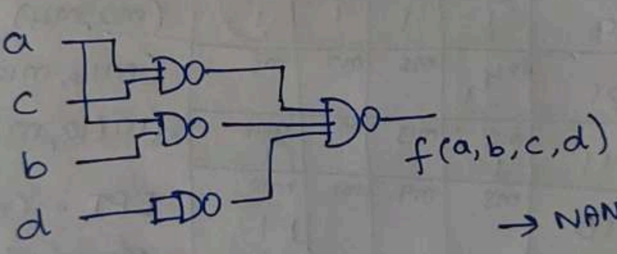
1,3,5,7,9,11,13,15	---1	d
~~1,5,9,13,3,7,11,15~~		
10,11,14,15	1-1-	ac
12,13,14,15	11--	ab

PI	m1	m3	m5	m7	m9	m10	m11	m12	m13	m14	m15
1, 3, 5, 7, 9, 11, 13, 15	⊛	⊛	⊛	⊛	⊛		*		*		*
10, 11, 14, 15						⊛	*			*	*
12, 13, 14, 15								⊛	*	*	*

$F(a,b,c,d) = d + ac + ab$





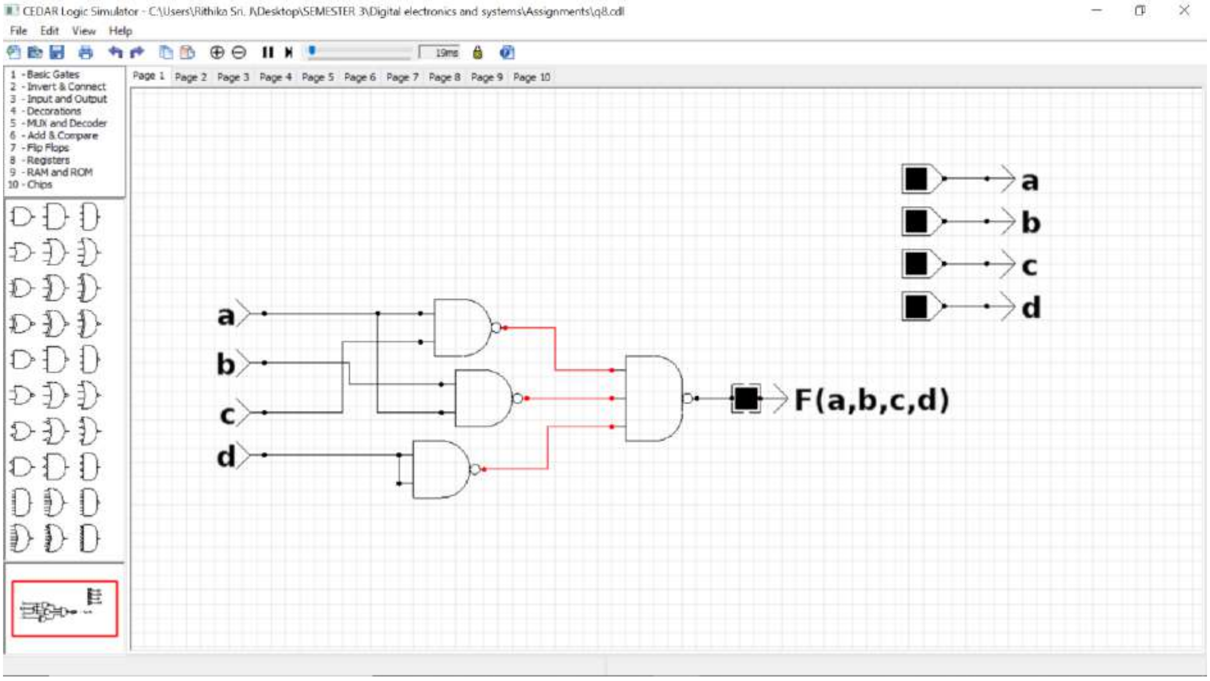


→ NAND- NAND network

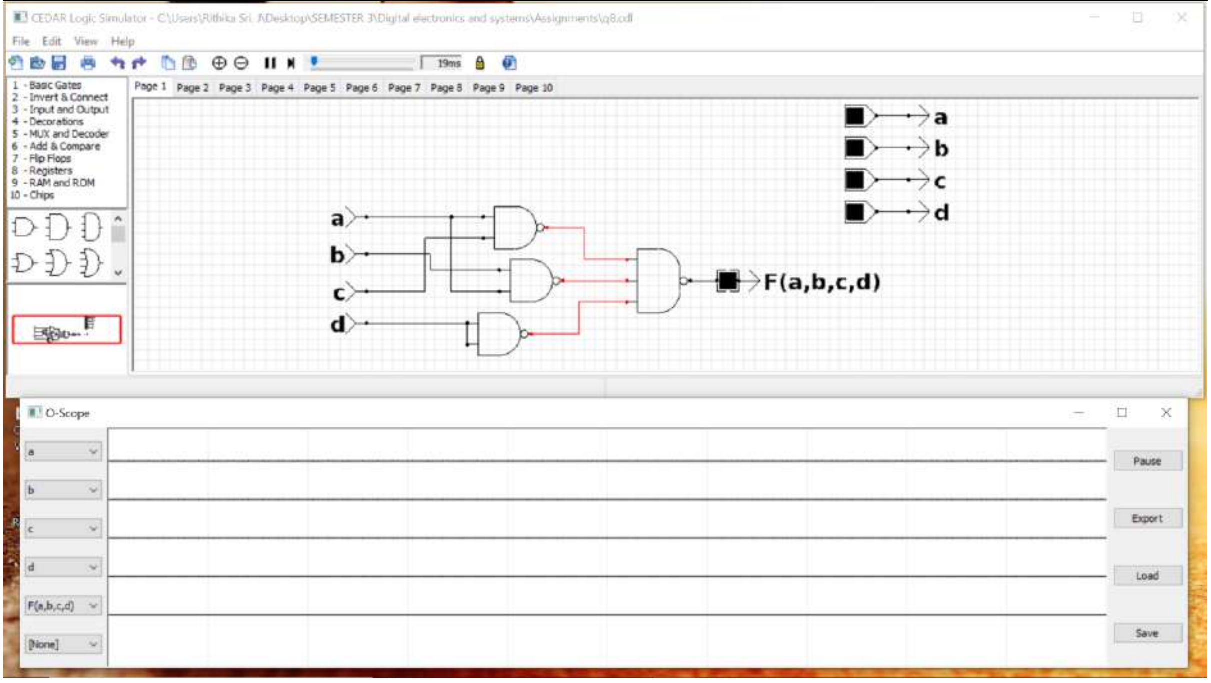
8)

F(a,b,c,d) = d+ac+ab

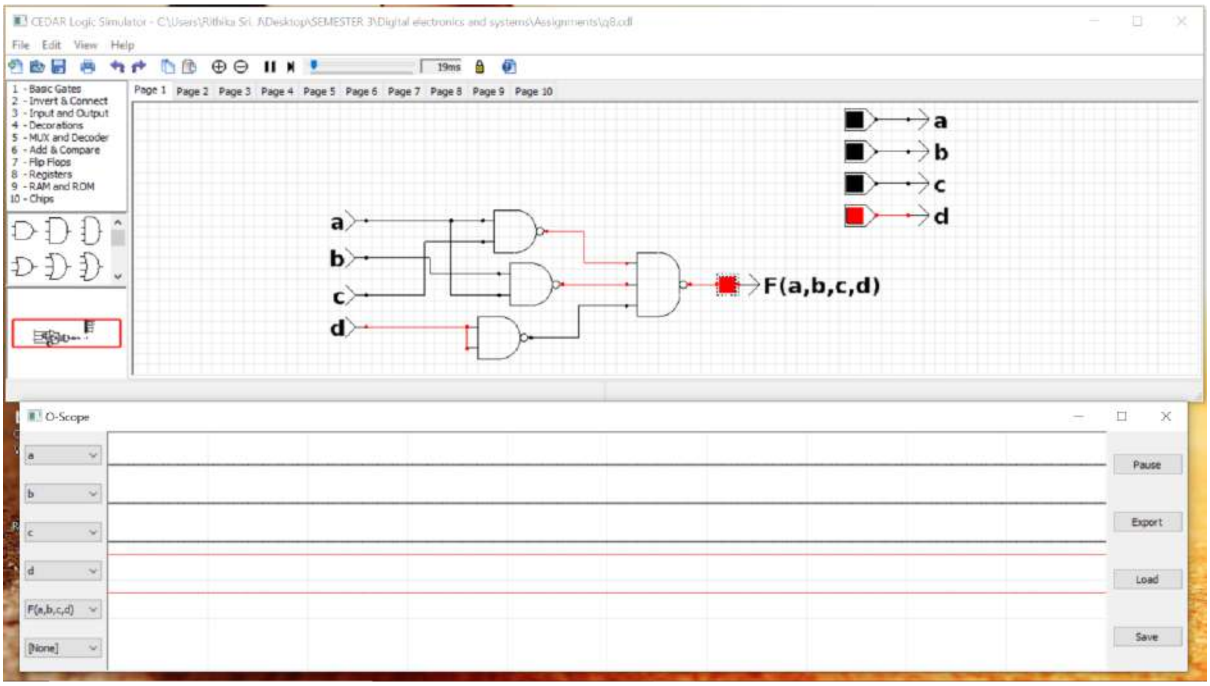
NAND-NAND network:



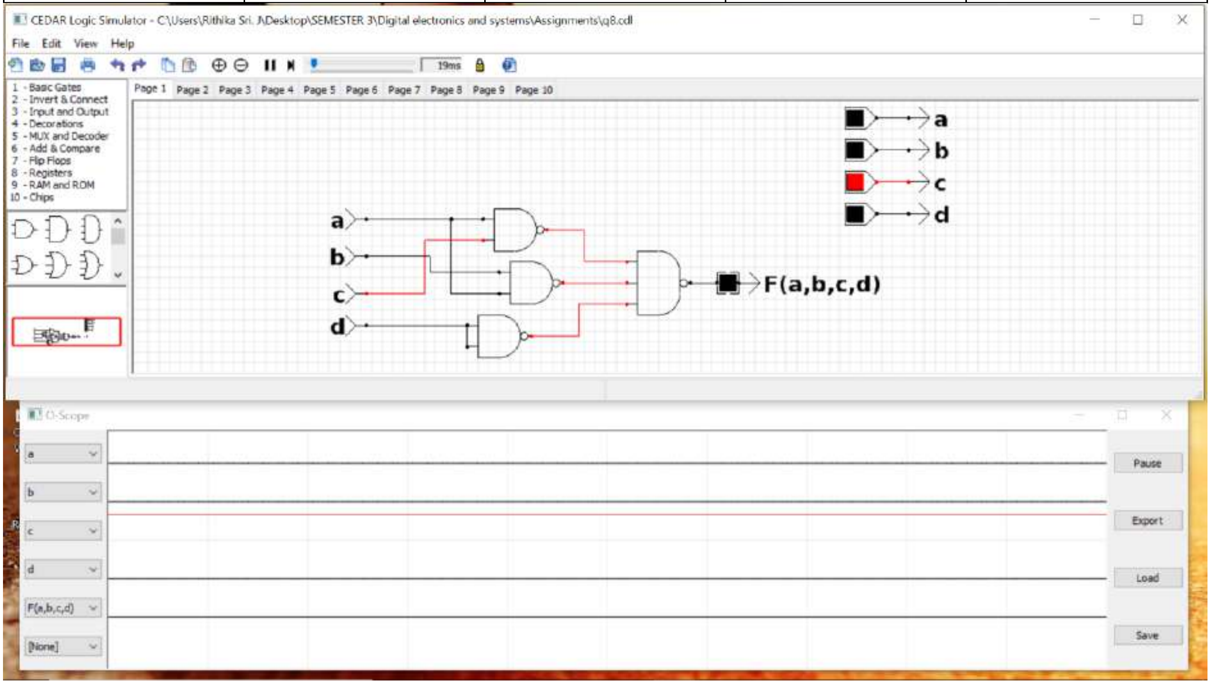
A	B	C	D	F
0	0	0	0	0



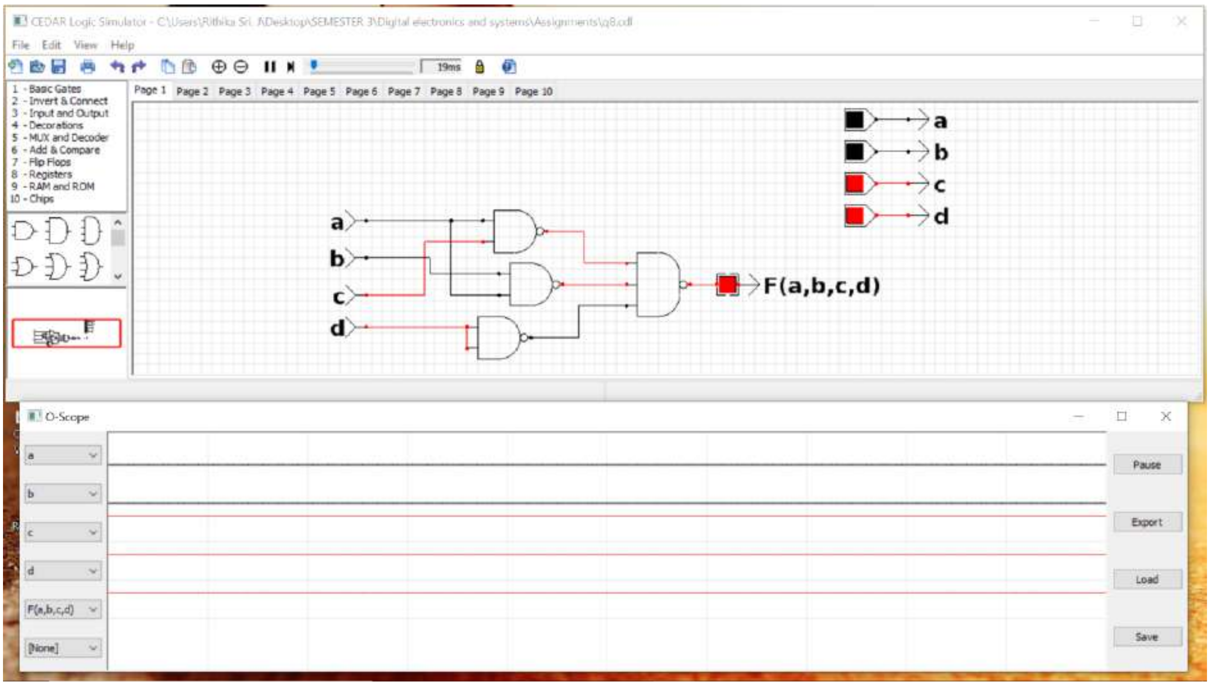
A	B	C	D	F
0	0	0	1	1



A	B	C	D	F
0	0	1	0	0

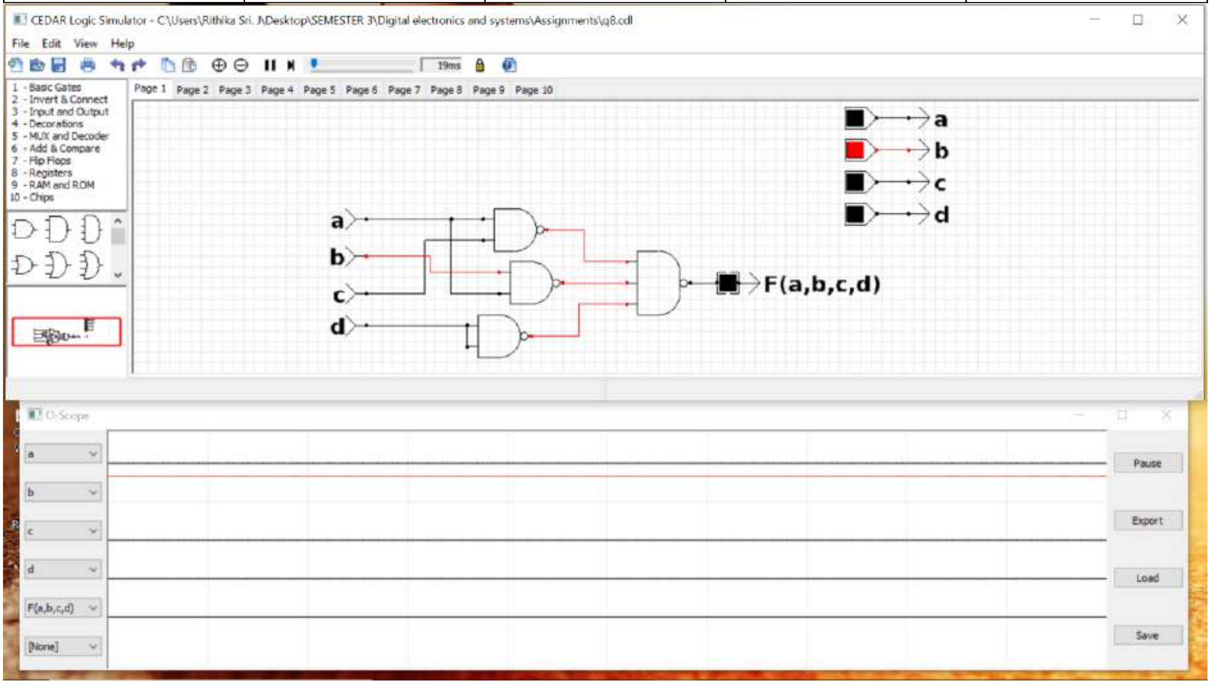


A	B	C	D	F
0	0	1	1	1



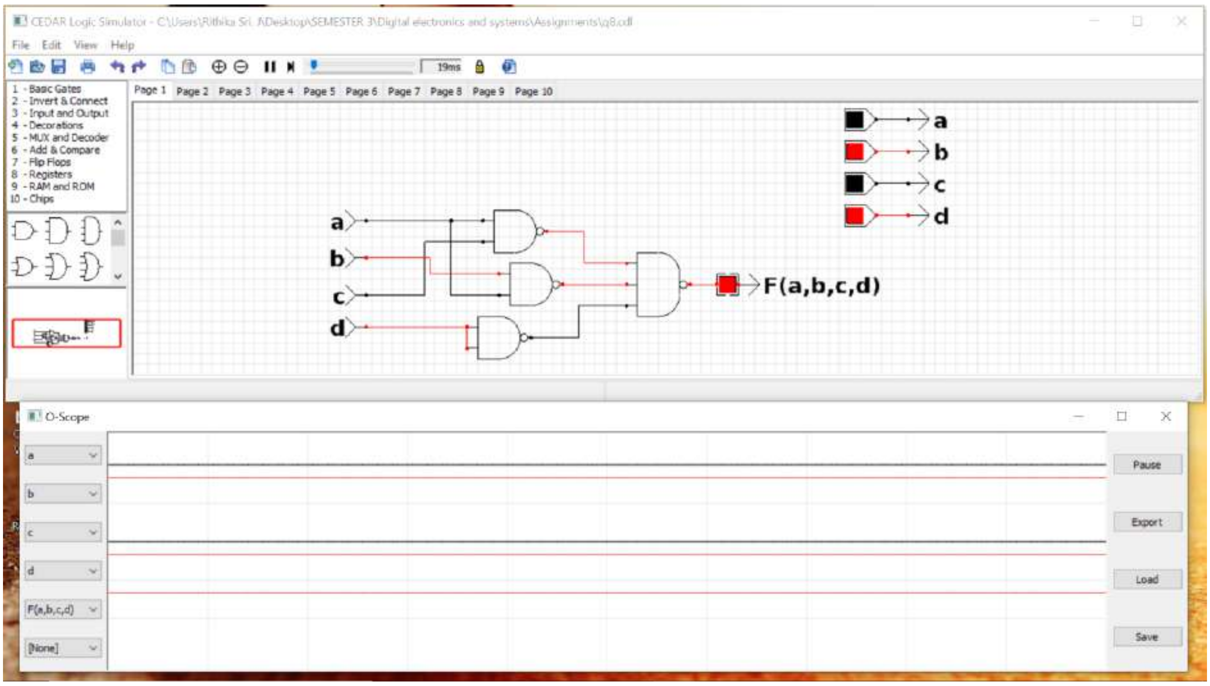
CEDAR Logic Simulator - C:\Users\Rithika Sri. J\Desktop\SEMESTER 3\Digital electronics and systems\Assignments\q8.cdl

A	B	C	D	F
0	1	0	0	0

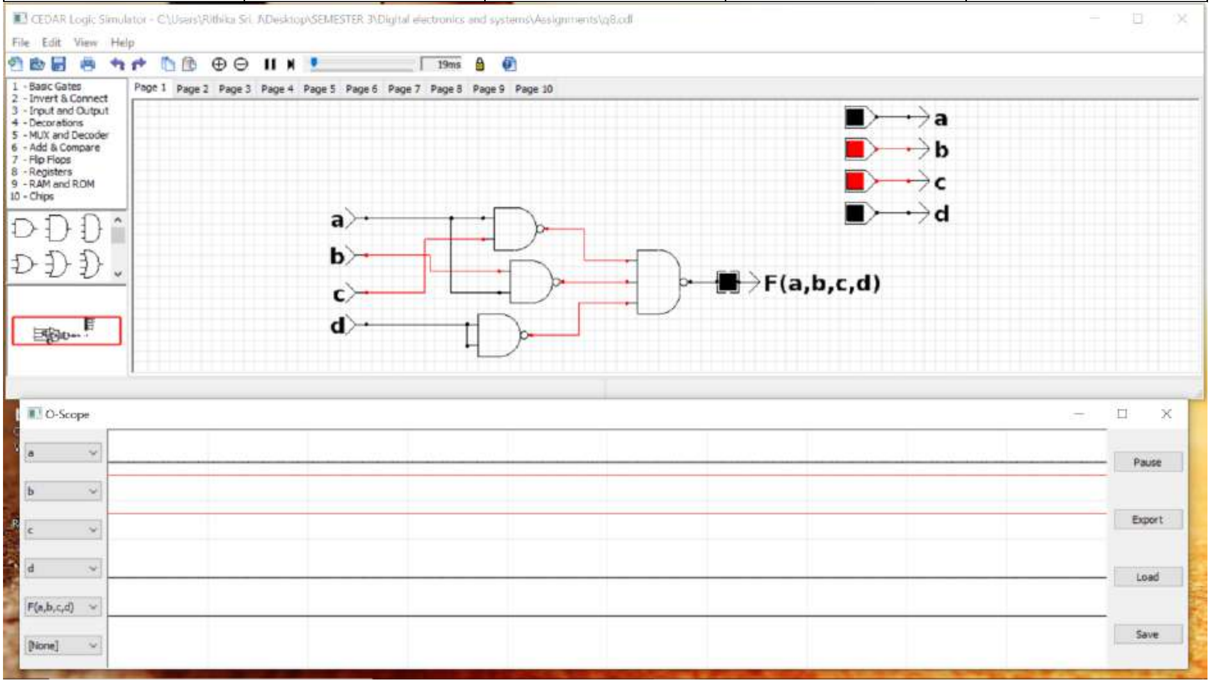


CEDAR Logic Simulator - C:\Users\Rithika Sri. J\Desktop\SEMESTER 3\Digital electronics and systems\Assignments\q8.cdl

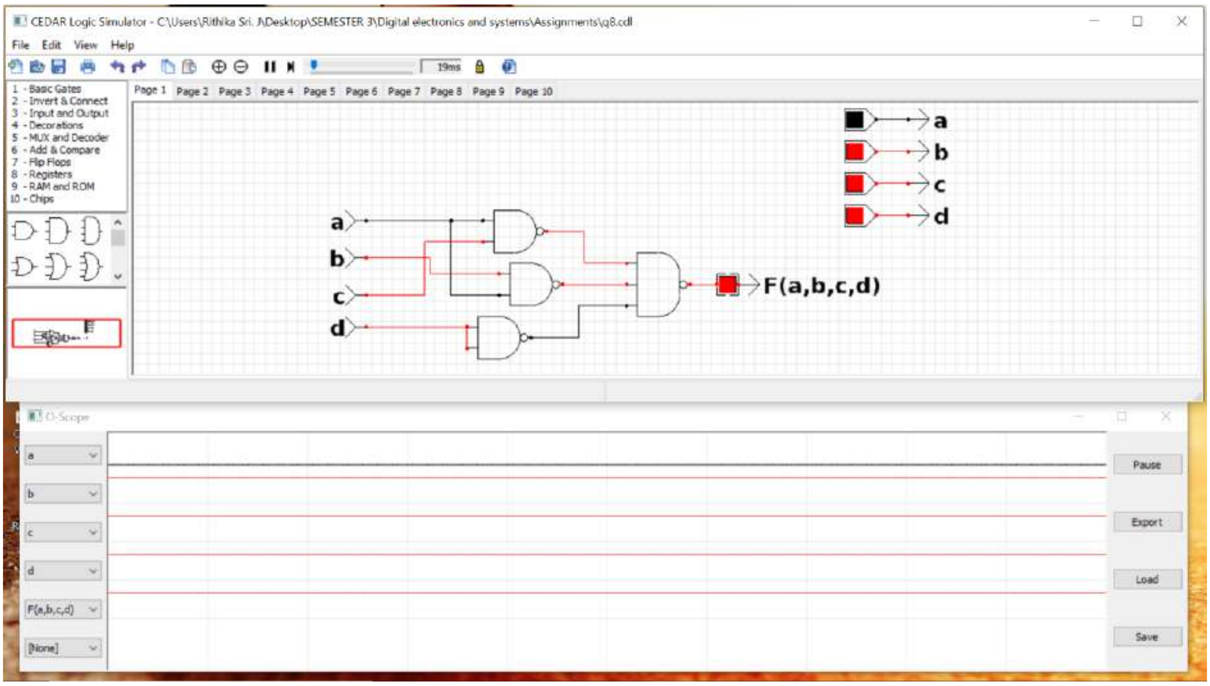
A	B	C	D	F
0	1	0	1	1



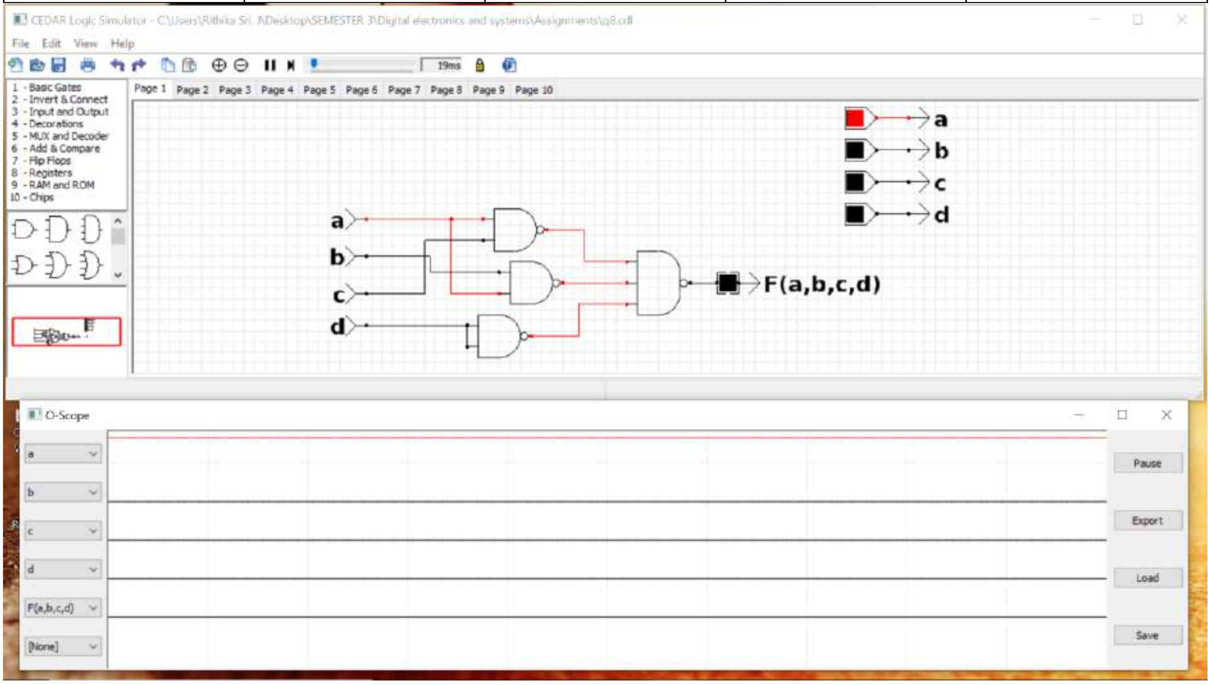
A	B	C	D	F
0	1	1	0	0



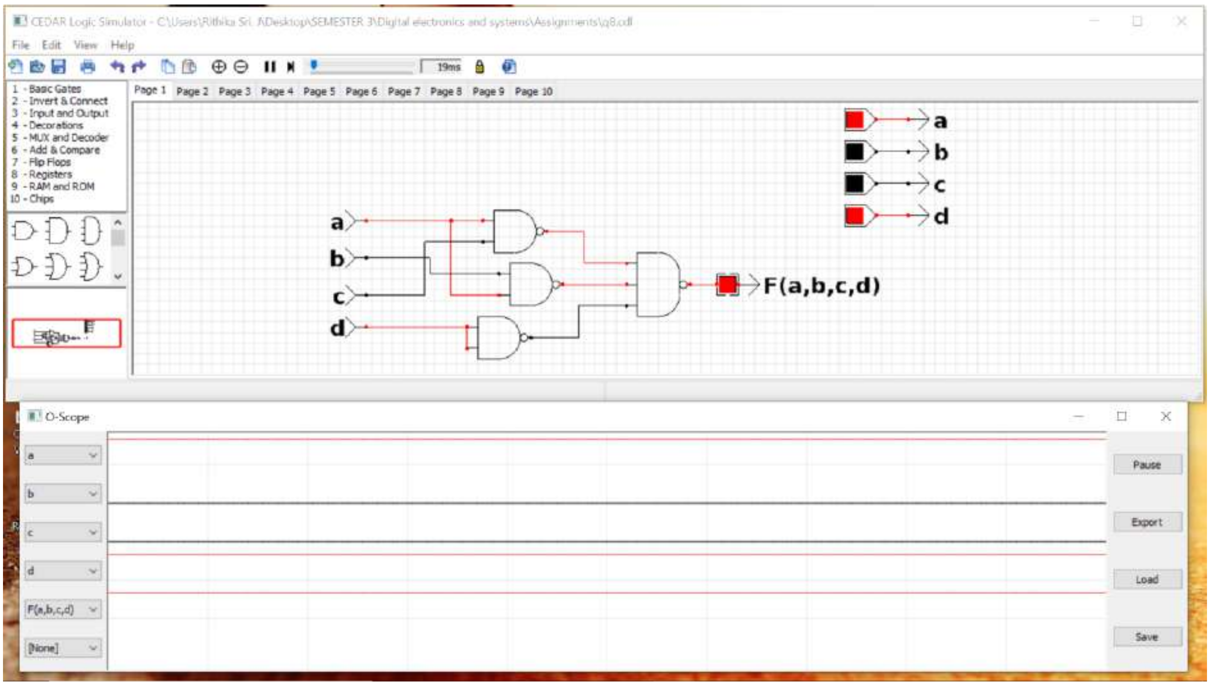
A	B	C	D	F
0	1	1	1	1



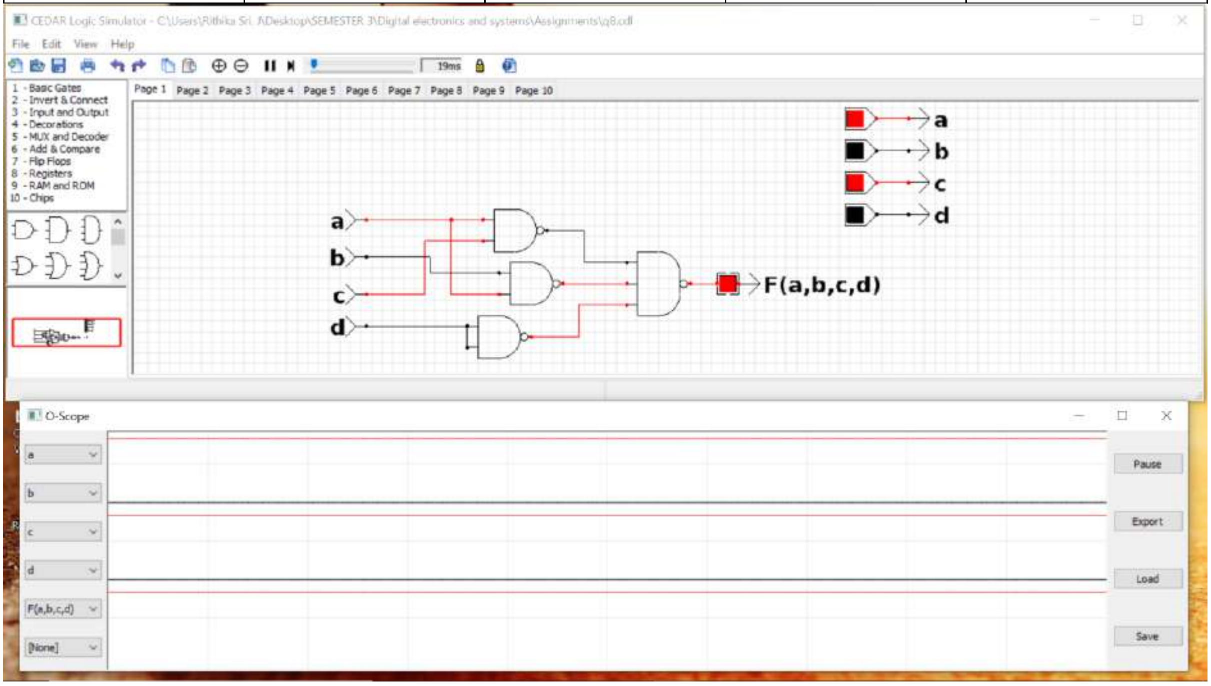
A	B	C	D	F
1	0	0	0	0



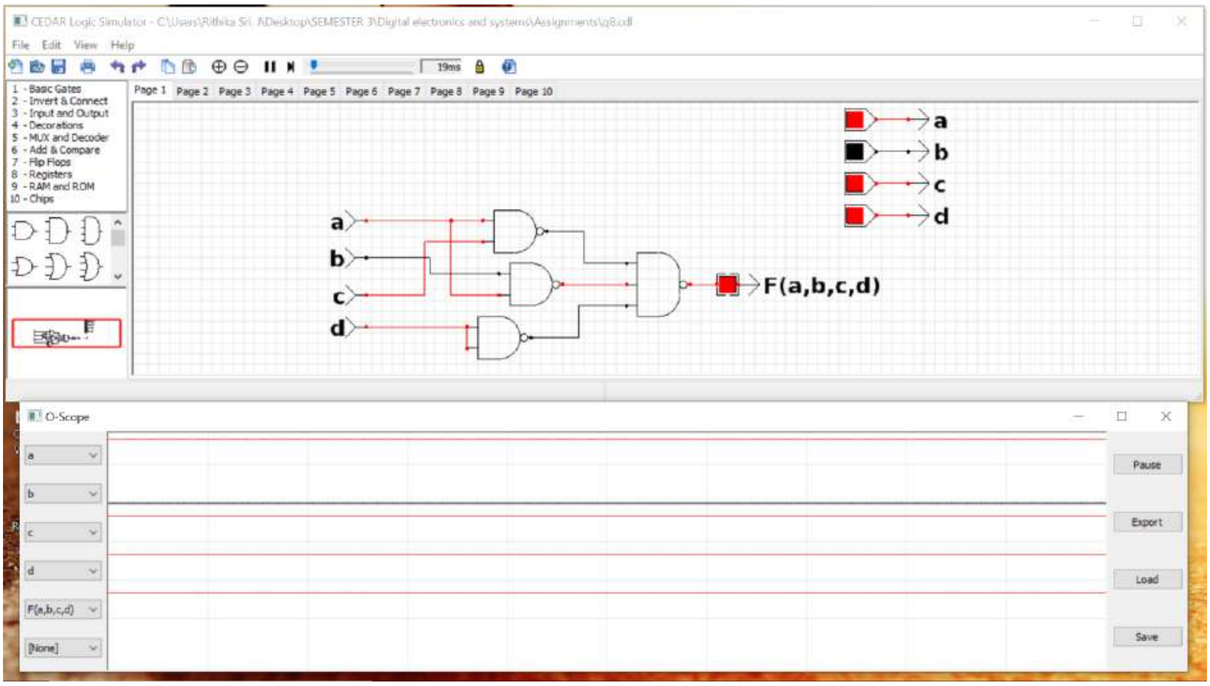
A	B	C	D	F
1	0	0	1	1



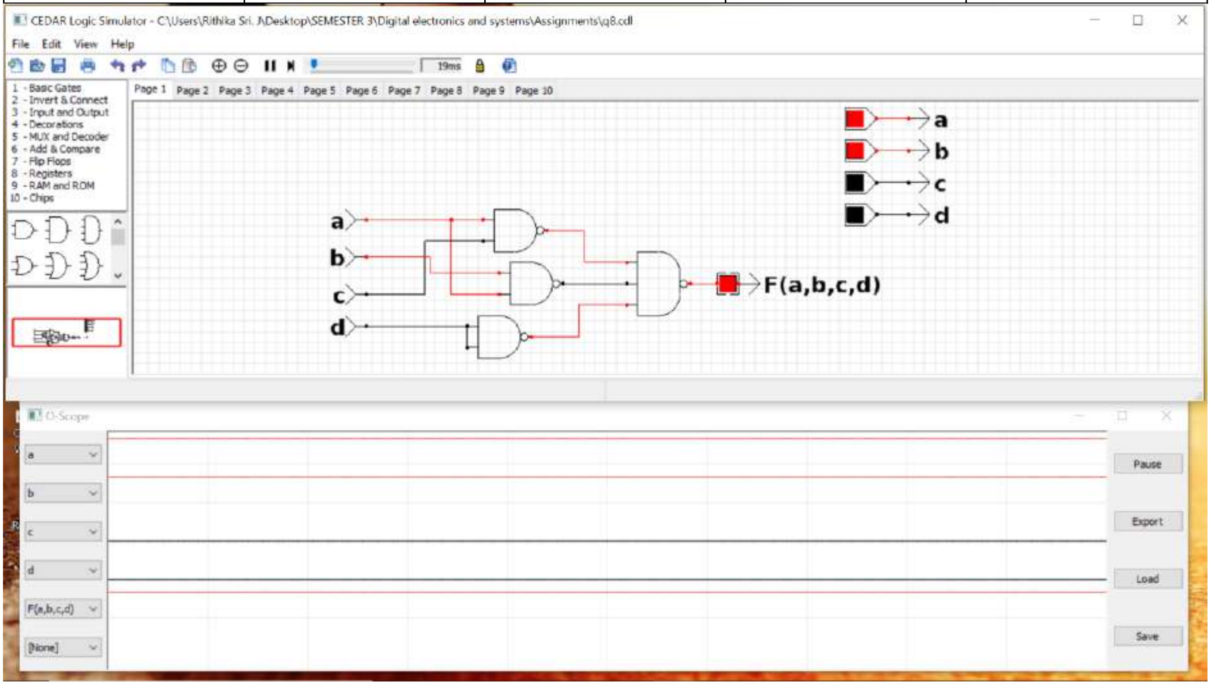
A	B	C	D	F
1	0	1	0	1



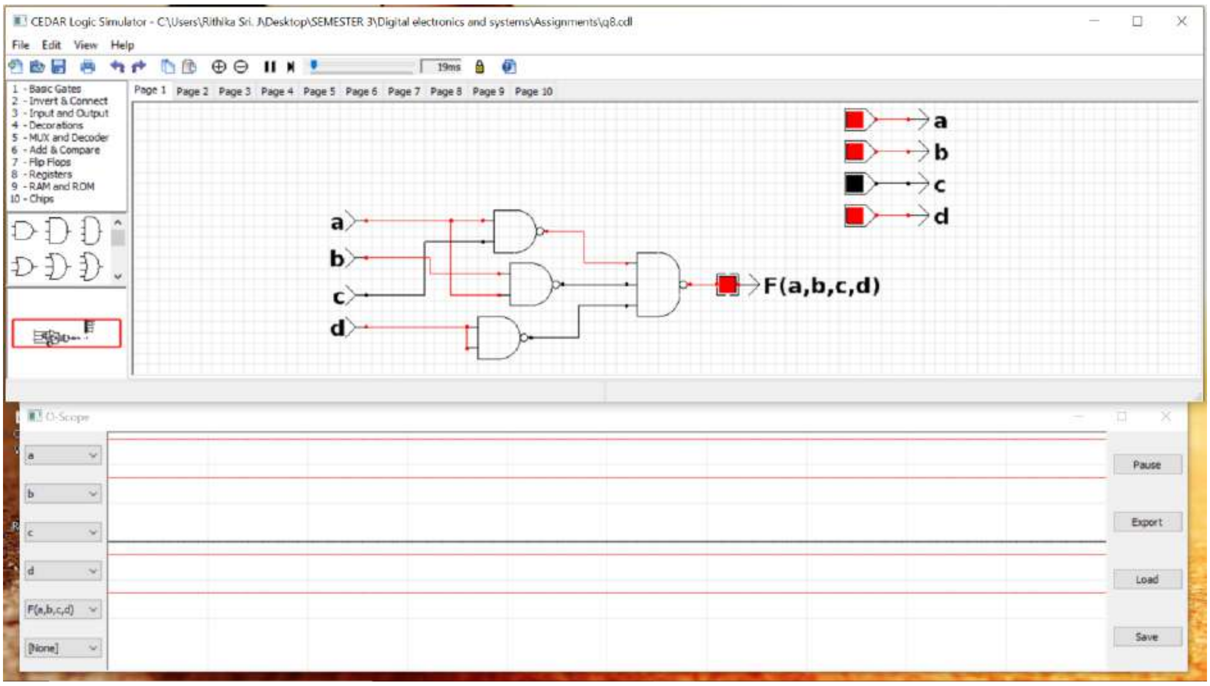
A	B	C	D	F
1	0	1	1	1



A	B	C	D	F
1	1	0	0	1

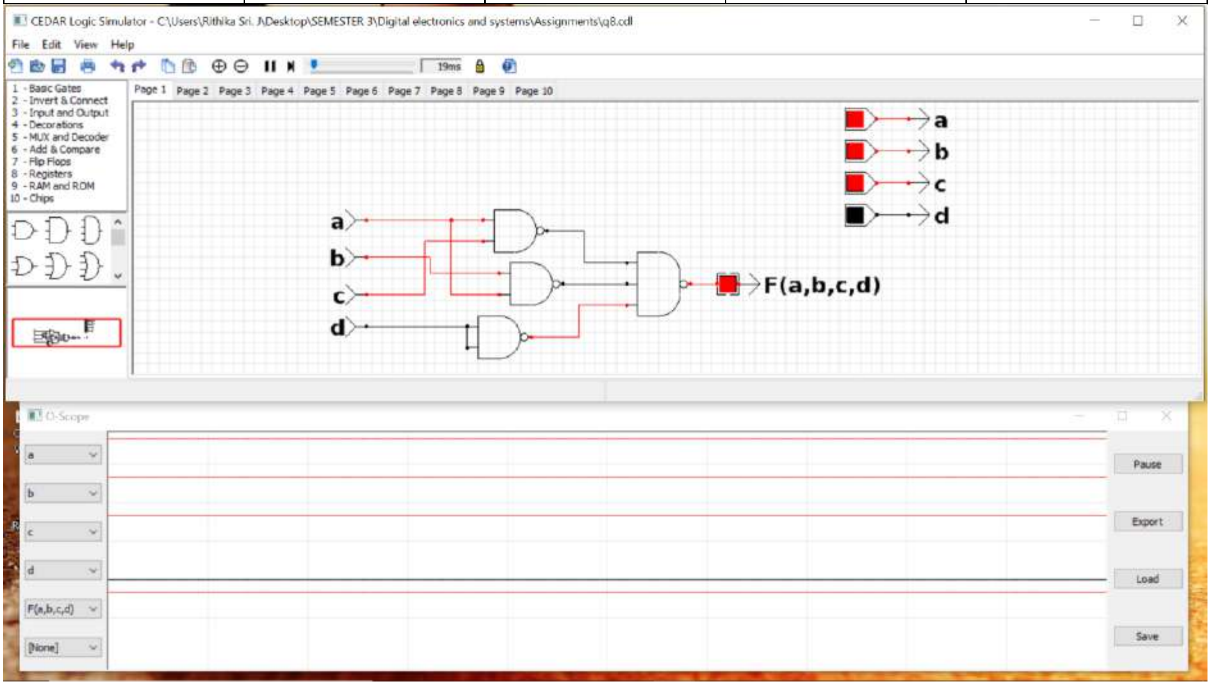


A	B	C	D	F
1	1	0	1	1



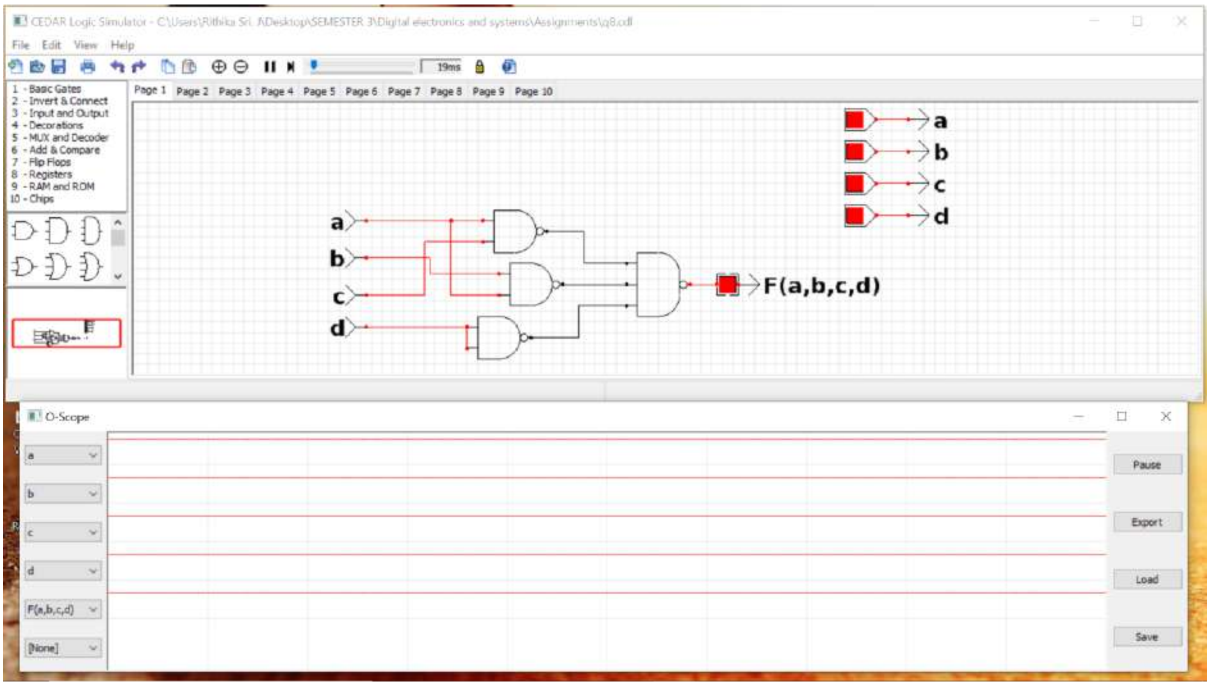
CEDAR Logic Simulator - C:\Users\Rithika Sri. J\Desktop\SEMESTER 3\Digital electronics and systems\Assignments\q8.cdl

A	B	C	D	F
1	1	1	0	1



CEDAR Logic Simulator - C:\Users\Rithika Sri. J\Desktop\SEMESTER 3\Digital electronics and systems\Assignments\q8.cdl

A	B	C	D	F
1	1	1	1	1

CEDAR Logic Simulator - C:\Users\Rithika Sri. A\Desktop\SEMESTER 3\Digital electronics and systems\Assignments\q8.cdl
File
Edit
View
Help
19ms
1 - Basic Gates
2 - Invert & Connect
3 - Input and Output
4 - Decorations
5 - MUX and Decoder
6 - Add & Compare
7 - Flip Flops
8 - Registers
9 - RAM and ROM
10 - Chips
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7
Page 8
Page 9
Page 10
a
b
c
d
a
b
c
d
F(a,b,c,d)
O-Scope
a
b
c
d
F(a,b,c,d)
[None]
Pause
Export
Load
Save