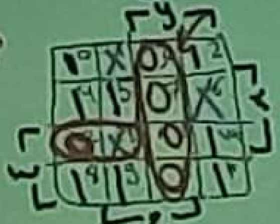
i) Simplify the following Function and then implement it using 2-Level NOR Gates only?

a. $F(W,X,Y,Z) = (0, 2, 4, 5, 8, 9, 10, 14)$, $d(W,X,Y,Z) = (1, 6, 13)$

→ F = product of sums

OR

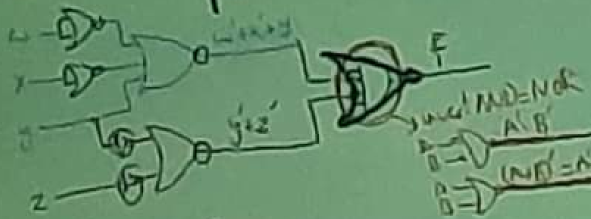


$F' = yz + wxy' \rightarrow$ SOP

$F = (yz + wxy')'$

$F = (y'+z') \cdot (w'+x'+y)$

POS



$A'B'$

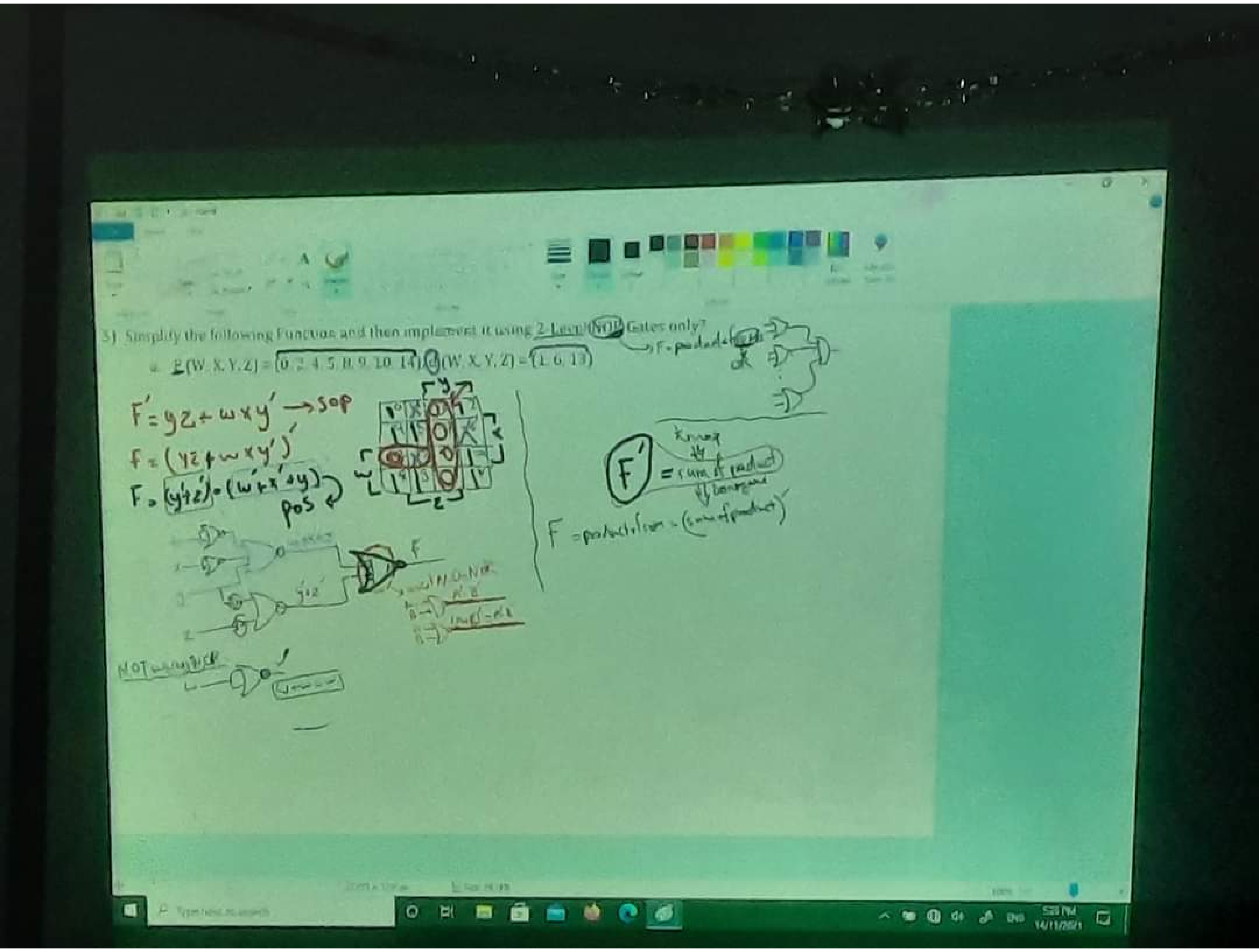
$(A+B)' = A'B'$

NOT using NOR

F' = sum of product (Kmap)

⇓ Demorgan

F = product of sum = (sum of product)'

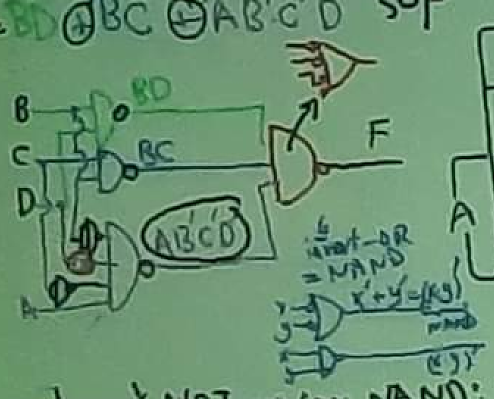


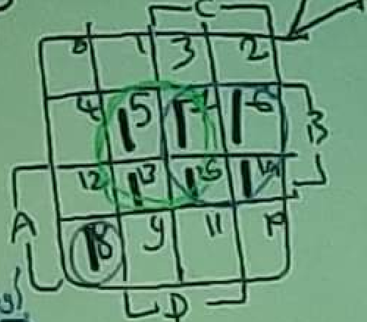
3) Simplify the following Function and then implement it using 2-Level NOR Gates only?
a. Σ(W, X, Y, Z) = (0, 2, 4, 5, 8, 9, 10, 14) d(W, X, Y, Z) = (1, 6, 13)
F' = yz + wxy' → sop
F = (yz + wxy')'
F = (y'+z')·(w'+x'+y)
pos
F

2) Simplify the following Function and then implement it using 2-Level N

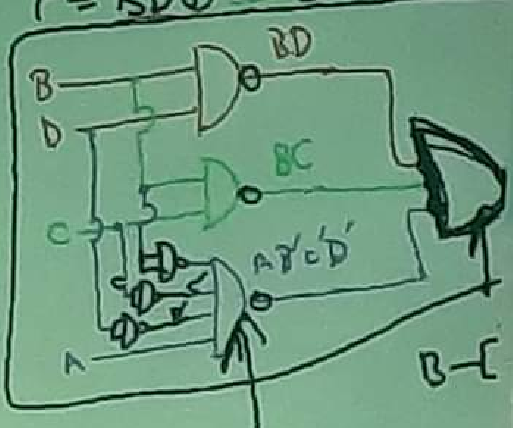
a. $F(A, B, C, D) = (5, 6, 7, 8, 13, 14, 15)$

$F = BD \oplus BC \oplus AB'C'D'$ SOP

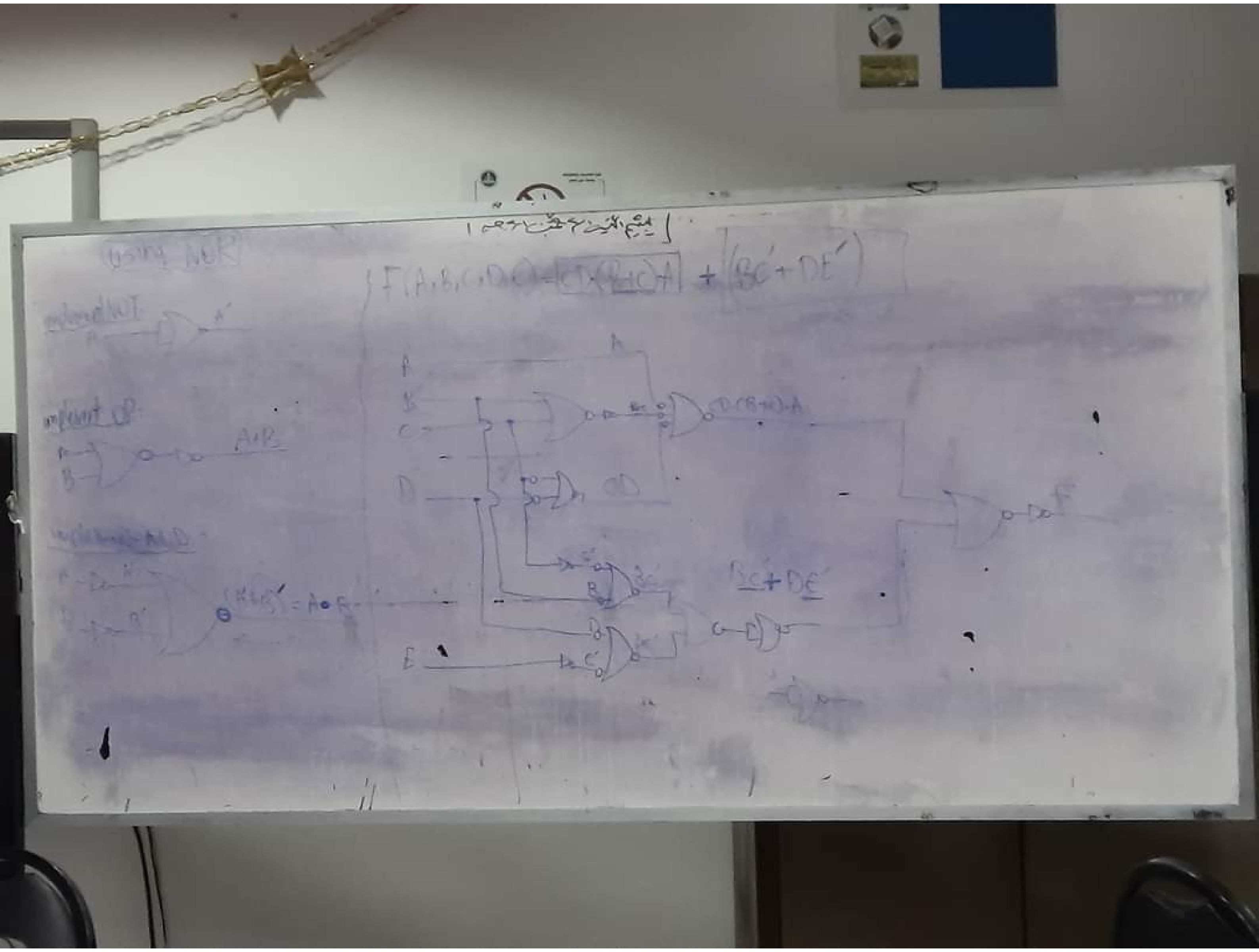


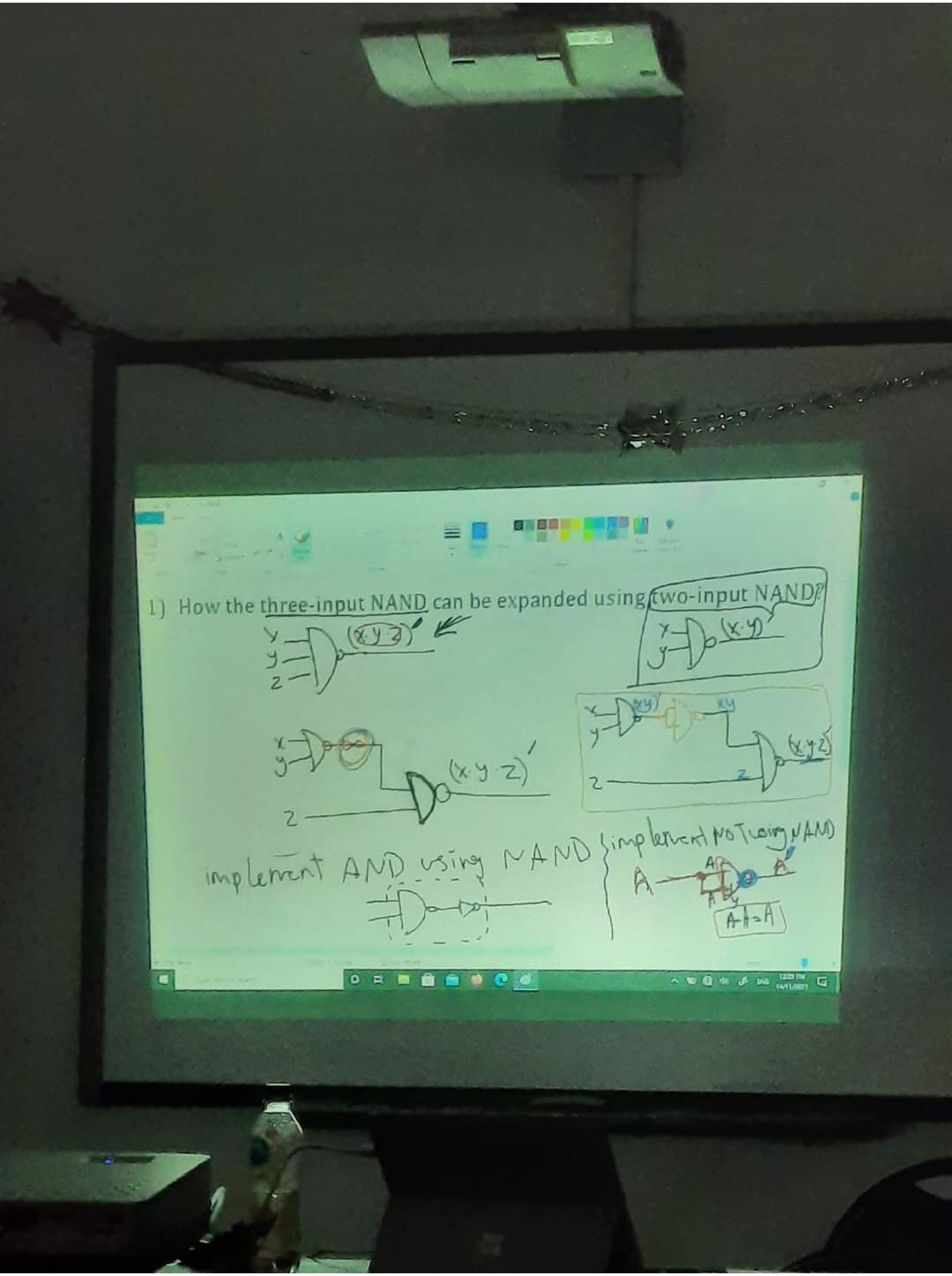


$F = BD \oplus BC \oplus AB'C'D'$



implement NOT using NAND:

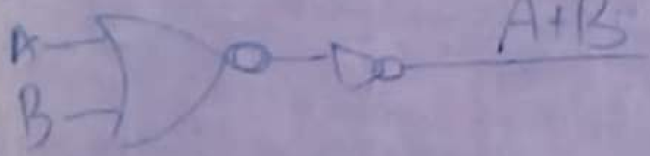

Using NOR
BC'+DE'


1) How the three-input NAND can be expanded using two-input NAND?
(x·y·z)'
(x·y)'
(x·y·z)'
implement AND using NAND
implement NOT using NAND
A·A = A

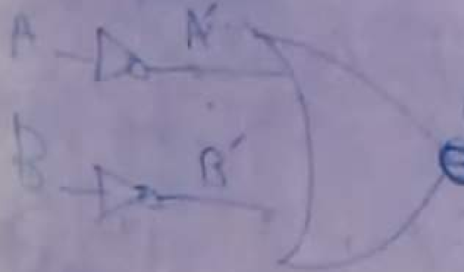
Using NOR

implement NOT.

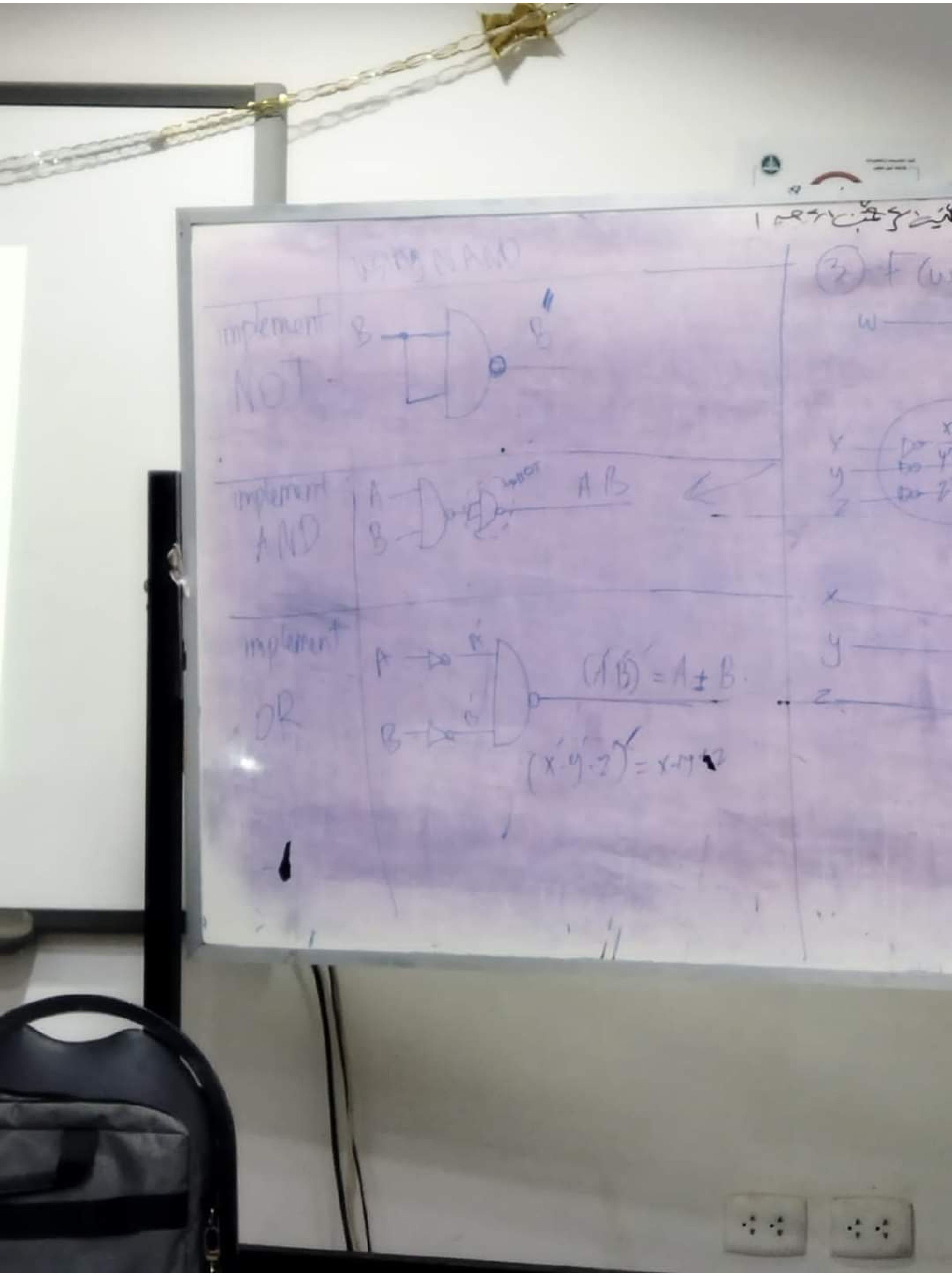
implement OR.



implement AND.



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


using NAND
implement NOT
implement AND
implement OR
A B

Parity checker

step1 no. of inputs : x y z P

output : ~~[illegible]~~ F

step2 :

x	y	z	P	F?	false		A $x \oplus y$	B $z \oplus P$	$A \odot B$
0	0	0	0	1	0	m0	0	0	1
0	0	0	1	0	1	m1	0	1	0
0	0	1	0	0	0	2	0	1	0
0	0	1	1	1	1	3	0	0	1
0	1	0	0	0	0	4	1	0	0
0	1	0	1	1	1	5	1	1	1
0	1	1	0	1	0	6	1	1	1
0	1	1	1	0	1	7	1	0	0
1	0	0	0	0	0	8	1	0	0
1	0	0	1	1	1	9	1	1	1
1	0	1	0	1	0	10	1	1	1
1	0	1	1	0	1	11	1	0	0
1	1	0	0	1	0	12	0	0	1
1	1	0	1	0	1	13	0	1	0
1	1	1	0	0	1	14	0	1	0
1	1	1	1	1	0	15	0	0	1

Step3 $f = \Sigma(0, 3, 5, 6, 9, 10, 12, 15)$

z

1^{0}	1	1^{3}	2	
4	1^{5}	7	1^{6}	y
1^{12}	13	1^{15}	14	y
8	1^{9}	11	1^{10}	

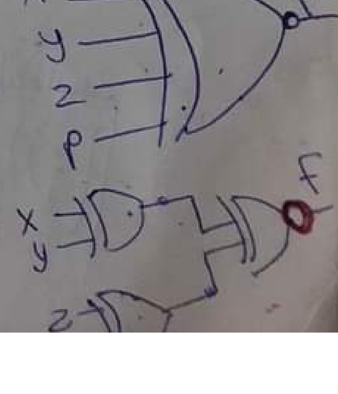
x

P

$f = x \odot y \odot z \odot P$

$= ((x \oplus y) \oplus (z \oplus P))'$

step 4



4) Design an odd Parity generator / Parity checker circuits

step 1 {Parity generator}

3 inputs x, y, z
one output P

step 2
truth table

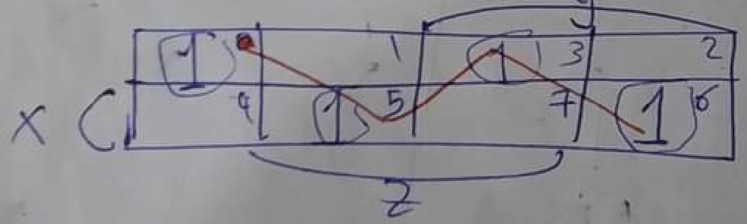
	x	y	z	P		$x \oplus y$
even	0	0	0	1	m_0	0
	0	0	1	0	m_1	0
	0	1	0	0	2	1
even	0	1	1	1	3	1
	1	0	0	0	4	1
even	1	0	1	1	5	1
	1	1	0	1	6	0
	1	1	1	0	7	0

Note: P72

$F6 = xy' + x'y$
$F6 = x \oplus y$
Exclusive-OR
(x or y, but not both)
(odd function)

$F9 = xy + x'y'$
$= (x \oplus y)'$
$= x \odot y$
Equivalence
Exclusive-NOR
(x equals y).
(even function)

step 3 $P = \Sigma(0, 5, 6, 3)$

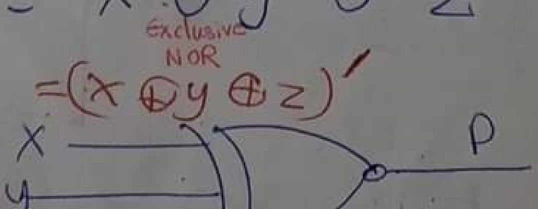


$P = x \odot y \odot z$

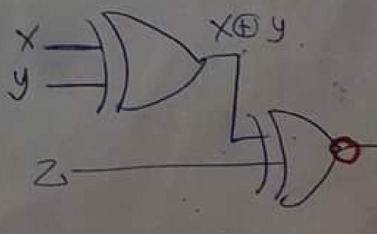
Exclusive NOR

$= (x \oplus y \oplus z)'$

step 4



Even function

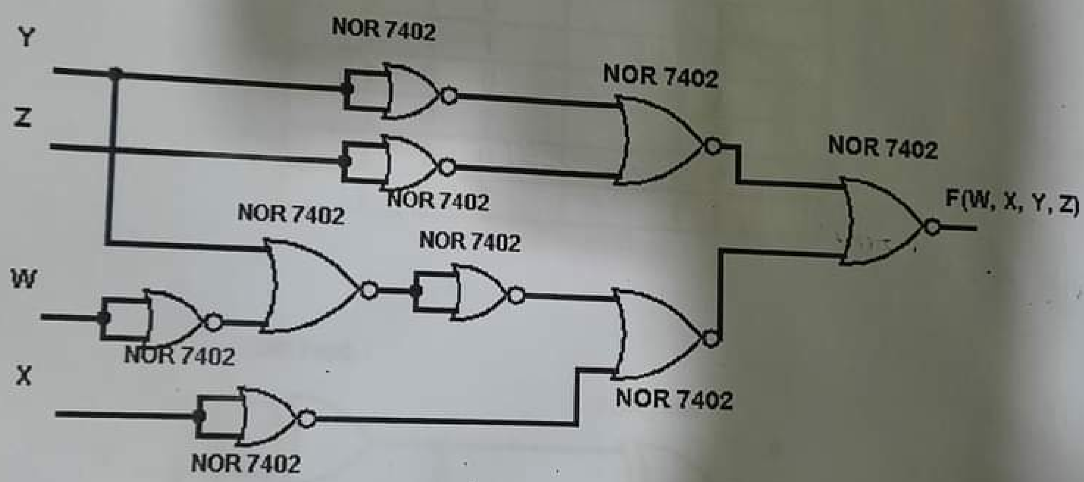


Lab 5 (NOR only)

Objective: implementing functions using NOR only.

Task to do in lab: implementing practically the following function using NOR only.

$$F\ (W, X, Y, Z) = (Y'+Z')(W'+X'+Y)$$



$$F\ (W, X, Y, Z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 10, 14)$$