Q1) Using K-maps find the simplest form in sum of products of the function H = *f* · *g*, where *f* and *g* are, respectively:

$$f(X,Y,Z,W) = XY + YZ' + Y'W' + X'Z'$$

AND $$g(X,Y,Z,W) = (X + Y + W)(Y + Z)(X' + Y' + Z')$$

f

Z

1	1	0	1
1	1	0	0
1	1	1	1
1	0	0	1

Y, X, W

g

Z

0	0	1	0
1	1	1	1
1	1	0	0
0	0	1	1

Y, X, W

f · g is the superset of all zero!

Y

0	0	0	0
1	1	0	0
1	1	0	0
0	0	0	1

X, W, Z

f · g = YZ' + XY'ZW'
= (Y+Z)(X+Y)(Z'+W')(Y'+Z')

Q2) The Boolean function F(A,B,C,D) = ∑ m(0,2,3,4,8,10,14) has the following don't care conditions d(A,B,C,D) = ∑m(9,11,12,15)

Implement F using minimum number of gates as:

F

C

1	0	1	1
1	0	0	0
X	0	X	1
1	X	X	1

B, A, D

F in (SOP) = C'D' + B'C + AC

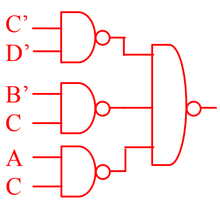
F in (POS) = (C + D')(A + B' + C')

F' in (SOP) = C'D + A'BC

F' in (POS) = (C + D) (B + C') (A' + C')

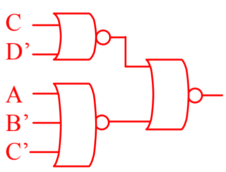
a) NAND-NAND

We use the SOP expression of F:



b) NOR-NOR

We use the SOP expression of F:

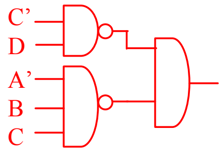


c) NAND-AND

Start with POS expression of F: F = (C + D’)(A + B’ + C’)

Now double invert each SUM terms (won’t change the function) F = (C + D’)’’ (A + B’ + C’)’’

Expand the 1st inversion → F = (C’D)’ (A’BC)’ (NAND-AND)

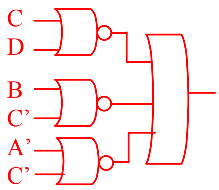


d) NOR-OR

Start with SOP expression of F: F = C’D’ + B’C + AC

Now double invert each AND terms (won’t change the function) F = (C’D’)’’ + (B’C)’’ + (AC)’’

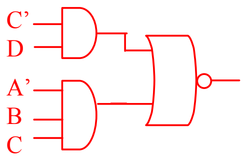
Expand the 1st inversion → F = (C+D)’ + (B+C’)’ + (A’+C’)’



e) AND-NOR

AND-OR-Invert → We use F’ SOP form for the AND-OR, then the Invert will get us F!

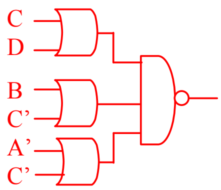
F’ in (SOP) = C’D + A’BC → F = (C’D + A’BC)’



f) OR-NAND

OR-AND-Invert → We use F’ POS form for the OR-AND, then the Invert will get us F!

F’ in (POS) = (C + D) (B + C’) (A’ + C’) → F = [(C + D) (B + C’) (A’ + C’)]’



g) Which of the above implementations are equivalent (i.e. logically the same)

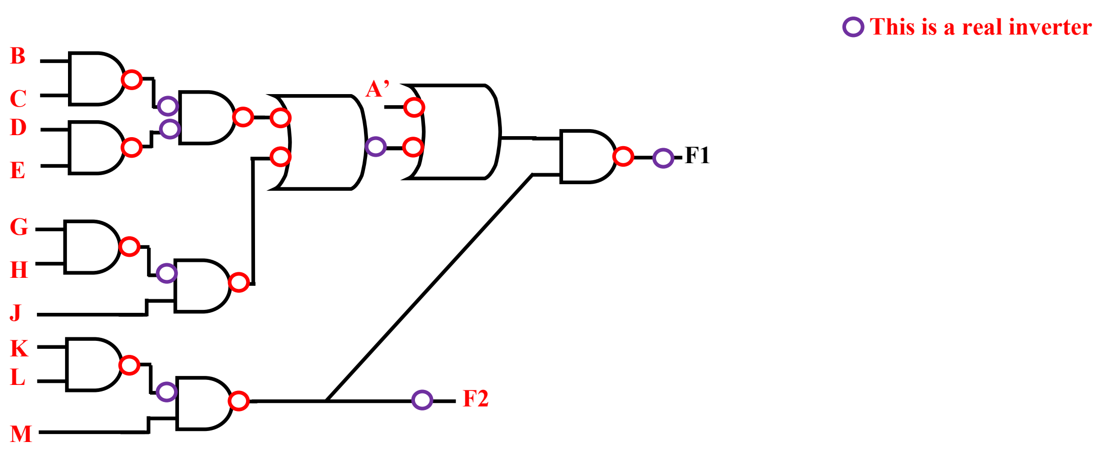
Since we have not used any Don’t Care condition as 0 and 1 to obtain all the above expressions, then all expressions are equivalent (logically identical)

If someone uses the same X as a 0 and then as a 1 to obtain two expressions, these expressions won’t be equivalent.

Q.3. Implement the logic diagram below:

a) Using 2-input NAND gates only

We need first to convert all gates (ANDs and ORs) to 2-input gates, then we do the logic transformations: AND-Invert is a NAND, and Invert-OR is also a NAND .. All bubbles are inserted in pairs not to alter the logic!



b) Using 2-input NOR gates only

We need first to convert all gates (ANDs and ORs) to 2-input gates, then we do the logic transformations: OR-Invert is a NOR, and Invert-AND is also a NOR .. All bubbles are inserted in pairs not to alter the logic!