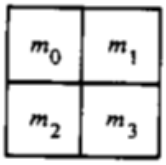
Class Notes on Karnaugh Map (KMap)

- The map method provides a simple procedure for minimizing Boolean functions.
- The map is made up of squares where each square represents a minterm.
- We represent each minterm of an equation by '1's
- We group the adjacent '1's in groups of 2, groups of 4 or groups of 8 and so on.
- In those groupings try to find the common literal.
- The map method is often known as Karnaugh Map or Veitech Diagram. In short we will call it K-map.

Two Variable Maps

There are 4 minterm of 2 variables; hence the map is four squares, one for each minterm. We name the boxes with each of the minterms.



(a)

Now we label the rows and columns.

- In the first row, the values of the two boxes are x'y' and x'y. Here, x' is common. So the 1st row is labeled x'.
- In the second row, the values of the two boxes are xy' and xy. Here, x is common. So the 2nd row is labeled x.
- In the first column, the values of the two boxes are x'y' and xy'. Here, y' is common. So the 1st column is labeled y'.
- In the second column, the values of the two boxes are x'y and xy. Here, y is common. So the 2nd column is labeled y.

	y'	y
x'	x'y'	x'y
x	xy'	xy

So our k-map for 2 variables looks like this:

	y'	y
x'	0	1
x	2	3

Three Variable Maps

There are 8 minterms of 3 variables; hence the map is eight squares, one for each minterm. We name the boxes with each of the minterms.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

Now we label the rows and columns.

- In the first row, x' is common. So the 1st row is labeled x'.
- In the second row, x is common. So the 2nd row is labeled x.
- In the first column, the values of the two boxes are x'y'z' and xy'z'. Here, y' and z' are common. So the 1st column is labeled y'z'.

- In the second column, the values of the two boxes are x'y'z and xy'z. Here, y' and z are common. So the 2nd column is labeled y'z.
- In the third column, the values of the two boxes are x'yz and xyz. Here, y and z are common. So the 3rd column is labeled yz.
- In the fourth column, the values of the two boxes are x'yz' and xyz'. Here, y and z' are common. So the 4th column is labeled yz'.

	y' z'	y' z	y z	yz'
x'	x'y'z'	x'y'z	x'yz	x'yz'
x	xy'z'	xy'z	xyz	xyz'

So our k-map for 3 variables looks like this:

	y' z'	y' z	y z	yz'
x'	0	1	3	2
x	4	5	7	6

Four Variable Maps

There are 16 minterms for 4 variables; hence the map is 16 squares, one for each minterm. We name the boxes with each of the minterms.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

Now we label the rows and columns.

- In the first row, w' and x' are common. So the 1st row is labeled w'x'.
- In the second row, w' and x are common. So the 2nd row is labeled w'x.
- In the third row, w and x are common. So the 3rd row is labeled wx.
- In the fourth row, w and x' are common. So the 4th row is labeled wx'.

- In the first column, y’ and z’ are common. So the 1st column is labeled y’z’.
- In the second column, y’ and z are common. So the 2nd column is labeled y’z.
- In the third column, y and z are common. So the 3rd column is labeled yz.
- In the fourth column, y and z’ are common. So the 4th column is labeled yz’.

	y'z'	y'z	yz	yz'
w'x'	w'x'y'z' 0	w'x'y'z 1	w'x'yz 3	w'x'yz' 2
w'x	w'xy'z' 4	w'xy'z 5	w'xyz 7	w'xyz' 6
wx	wxy'z' 12	wxy'z 13	wxyz 15	wxyz' 14
wx'	wx'y'z' 8	wx'y'z 9	wx'yz 11	wx'yz' 10

So our k-map for 4 variables looks like this:

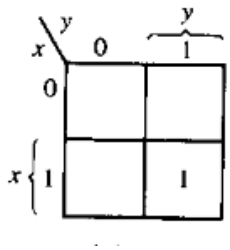
	y'z'	y'z	yz	yz'
w'x'	0	1	3	2
w'x	4	5	7	6
wx	12	13	15	14
wx'	8	9	11	10

Representing function in K-map

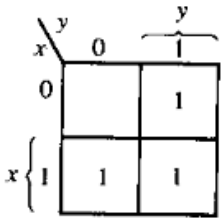
For a specific function, we put 1 in the minterm positions of the kmap. For finding minterms from an expression, we need its canonical form.

- We represent each minterm of an equation by '1's
- We group the adjacent '1's in groups of 2, groups of 4 or groups of 8 and so on.
- In those groupings we try to find the common literal.

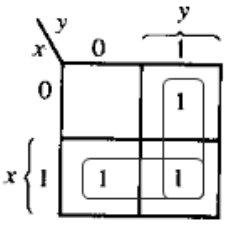
For example, if a function, F(x,y) = xy, then x=1 and y=1. So the minterm is 3.



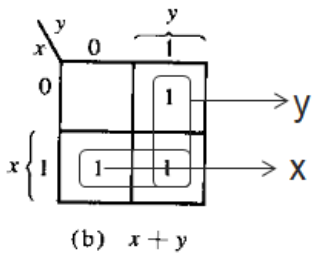
If a function F(x,y) = xy'+xy+x'y, then our minterm is 2, 3 and 1.



Now we make groups of the adjacent 1s. We can have 2^n numbers of 1s in a single group. For 2 variables, we can make a group of 2 and 4 ones. For 3 variables, we can make a group of 2,4 and 8 ones. For 4 variables, we can make a group of 2, 4, 8 and 16 ones.



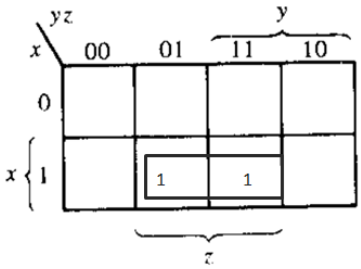
Next, we find the common literals in each group. It is allowed to use the same '1' more than once.



(b) $x + y$

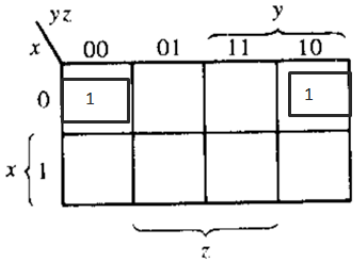
We sum up the common literals of each group and get our simplified answer.

Example 1: Simplify the function using Kmap: F1=Σ(5,7)



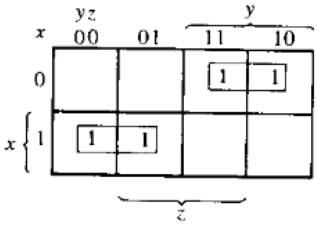
Answer: xz

Example 2: Simplify the function using Kmap: F1=Σ(0,2)



Answer: x'z'

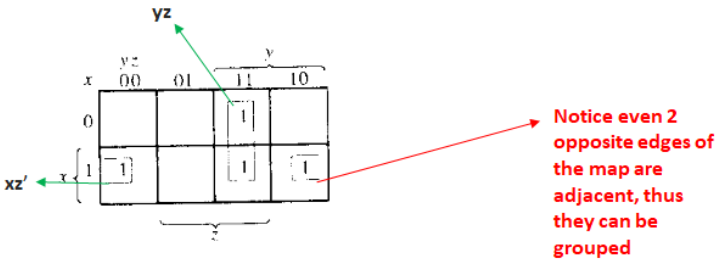
Example 3: Simplify the function using Kmap: F1=Σ(2,3,4,5)



Answer: x'y + xy'

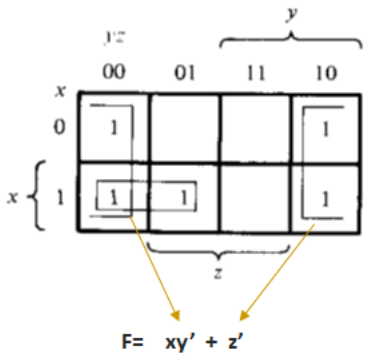
Example 4: Simplify the function using Kmap: F1=x'yz+xy'z'+xyz+xyz'

Minterms: x'yz = 011_2 = 3, xy'z' = 100_2 = 4, xyz = 111_2 = 1, xyz' = 110_2 = 6

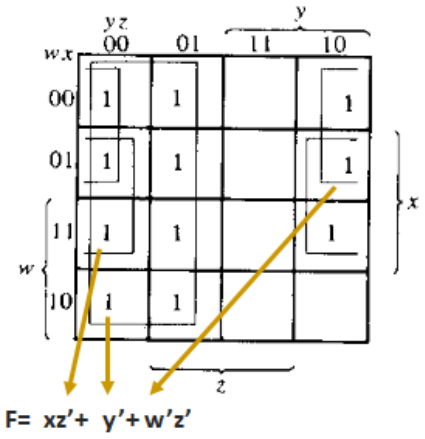


Answer = xz' + yz

Example 5: Simplify the function using Kmap: F=Σ(0,2,4,5,6)



Example 6: Simplify the function using Kmap: F=Σ(0,1,2,4,5,6,8,9,12,13,14)



Example 7: Simplify the function using Kmap: F=A'B'C'+B'CD'+A'BCD'+AB'C'

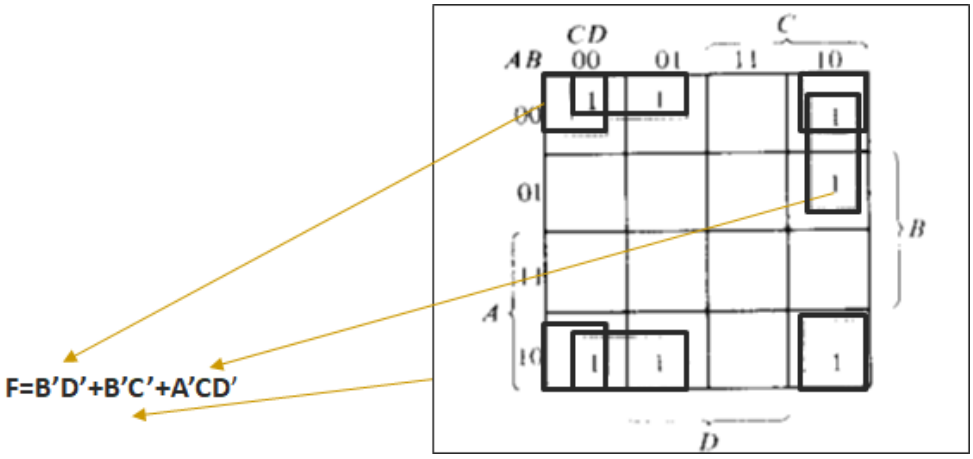
Since F is not in its Canonical form, we cannot find the minterm. So first we convert F to its Canonical form.

F = A'B'C' + B'CD' + A'BCD' + AB'C'

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

= A'B'C'D + A'B'C'D' + AB'CD' + A'B'CD' + A'BCD' + AB'C'D + AB'C'D'

= Σ(0,1,2,6,8,9,10)



Product of Sum (POS) simplification

- F is represented with the '1's
- F' is represented with the '0's
- (F')' =F so find F' and then find its complement using DeMorgan's law.
- In other words, '1's represent the minterms whereas '0's represent the maxterms.
- If Equation is provided in SOP format, mark the '1's and then mark the remaining with '0's. On the other hand, if Equation is provided in POS format, mark the '0's and then mark the remaining with '1's. Once marking is done, function can be simplified in SOP or POS format.

Example 1:

SOP->

F=Σ(0,1,2,5,8,9,10)

Combining '1's we get

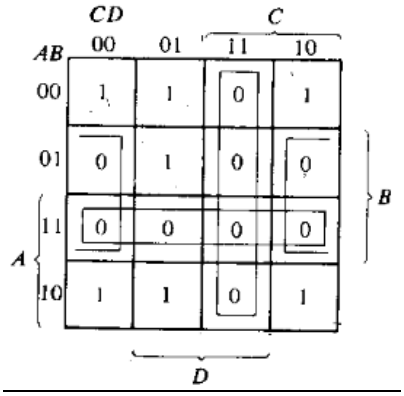
F=B'D'+B'C'+A'C'D

Combining '0's we get

F'=AB+CD+BD'

Applying De-morgan

F=(A'+B')(C'+D')(B'+D) <-POS



Example 2:

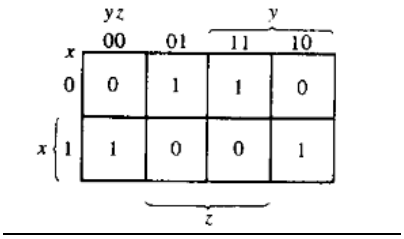
If SOP: Σ(1,3,4,6)

then POS: Π(0,2,5,7)

So grouping '1's, F=x'z+xz'

Grouping '0's, F'=xz+x'z'

therefore, F= (x'+z')(x+z)

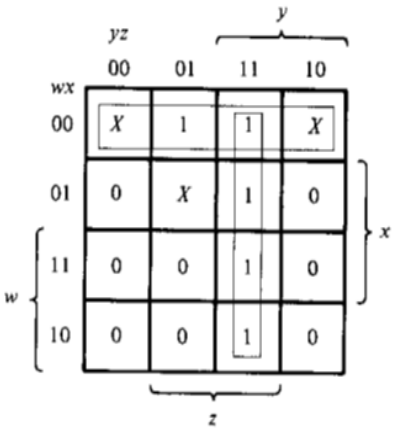


Don't Care condition

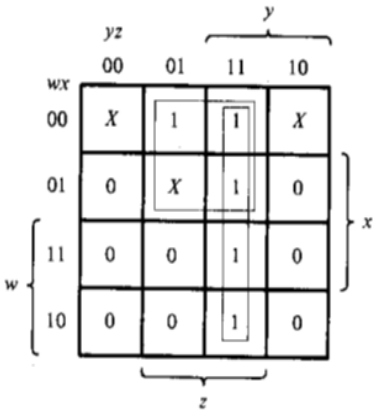
- Don't care in a Karnaugh map, or truth table, may be either 1s or 0s, as long as we don't care what the output is for an input condition we never expect to see.
- We plot these cells with a special sign, 'X', among the normal 1s and 0s.
- When forming groups of cells, treat the don't care cell as either a 1 or a 0, or ignore the don't cares. This is helpful if it allows us to form a larger group than would otherwise be possible without the don't care.
- There is no requirement to group all or any of the don't cares. (i.e. do not make any group only consisting of the don't cares). Only use them in a group if it simplifies the logic.

Example 1: Simplify F(w,x,y,z)= Σ(1,3,7,11,15) and don't care d(w,x,y,z)= Σ(0,2,5)

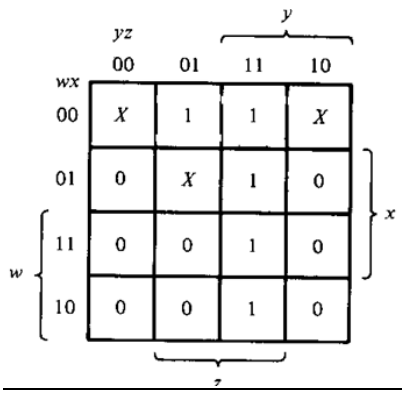
Solution: Even though 2 solutions are not the same, either 1 is acceptable. Such scenarios are only applicable for Don't care scenarios!



(a) $F = yz + w'x'$



(b) $F = yz + w'z$



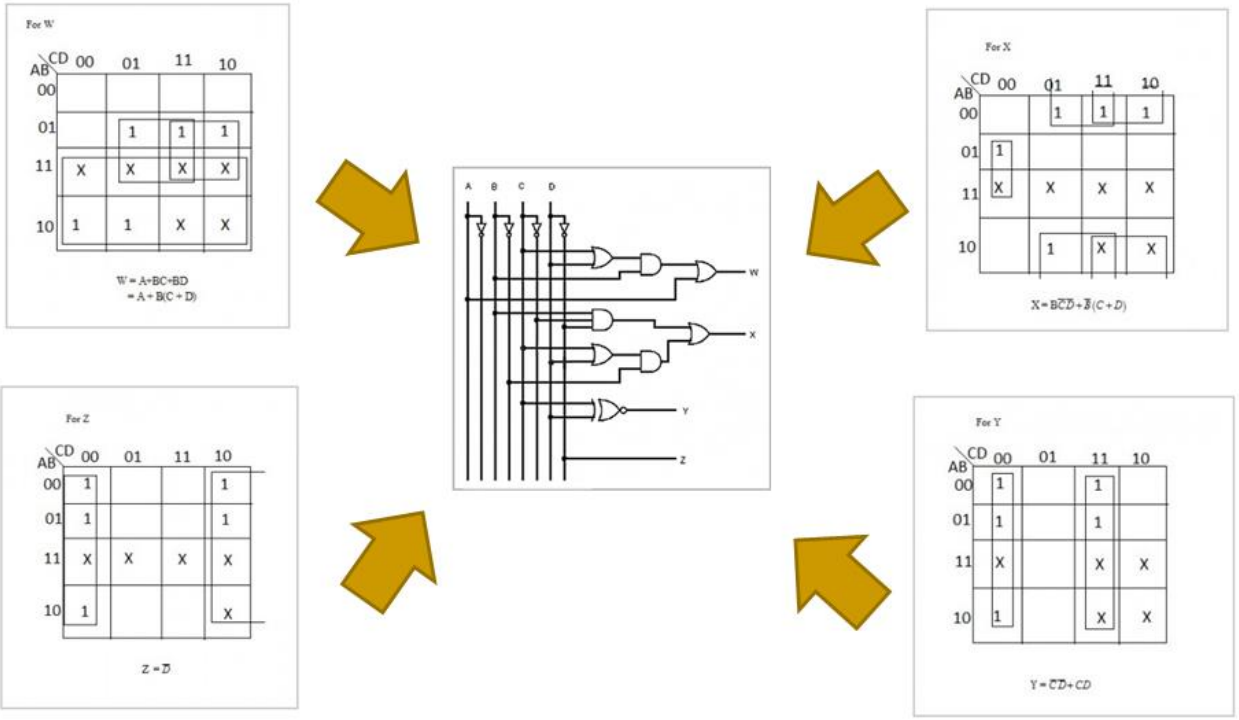
F'=z'+wy'
F=(z)(w'+y)

Applications of Kmap

Example 1: Design and draw the circuit diagram of BCD to Excess-3 code converter

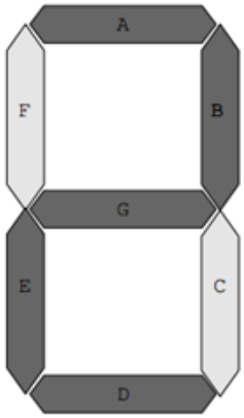
BCD Input				Excess-3 Output			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

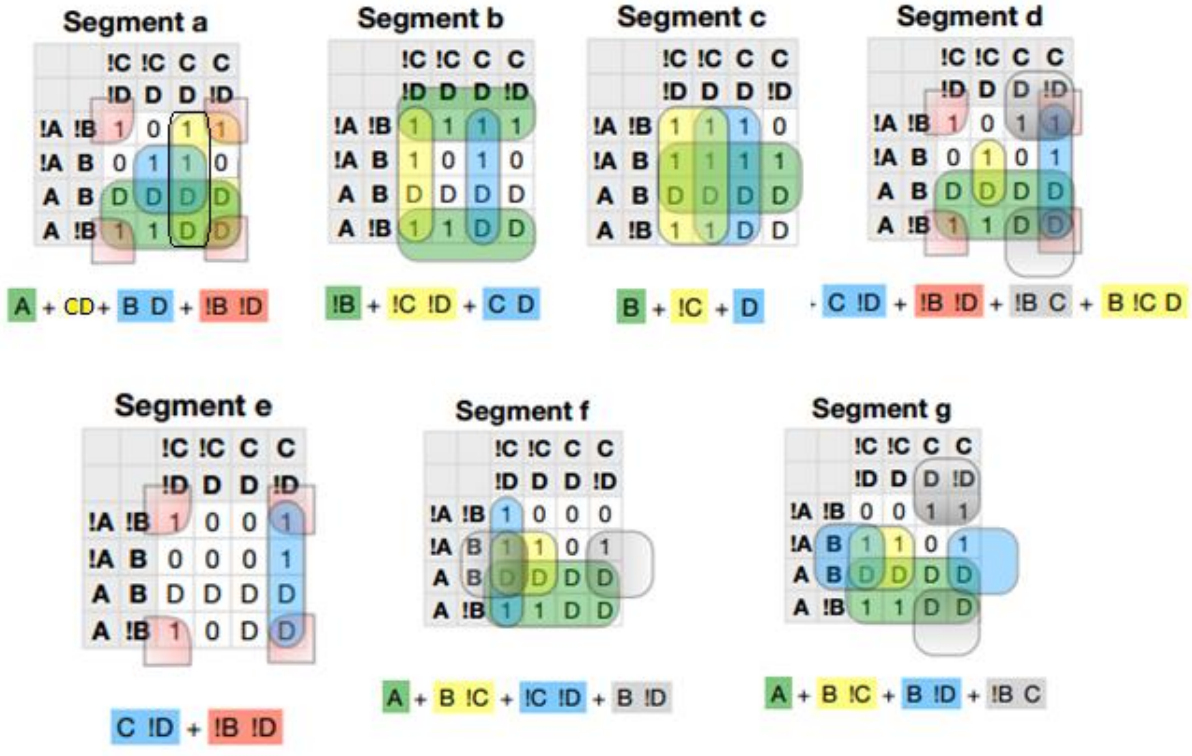
Any value higher than 9 will result into don't care output



Example 2: Write the boolean expressions of a 7 segment display board

Display digit	b8	b4	b2	b1	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
off	1	0	1	0	0	0	0	0	0	0	0
off	1	0	1	1	0	0	0	0	0	0	0
off	1	1	0	0	0	0	0	0	0	0	0
off	1	1	0	1	0	0	0	0	0	0	0
off	1	1	1	0	0	0	0	0	0	0	0
off	1	1	1	1	0	0	0	0	0	0	0





Example 3: A car garage has a front door and one window, each of which has a sensor to detect whether it is open. A third sensor detects whether it is dark outside. A security

system for the garage follows this rule: the alarm rings if the alarm switch is turned on and either the front door is not closed or it is dark and the side window is not closed.

Solution:

Assuming that,

- AlarmRing = 1 if Alarm rings, 0 otherwise.
- Switch = 1 if alarm switch is on, 0 otherwise.
- Door = 1 if door is not closed, 0 otherwise.
- Window = 1 if window is not closed.
- Dark = 1 if it is dark outside, 0 otherwise.

Condition: AlarmRing = 1 if [(Switch = 1) & {(Door = 0) + (Dark = 1)} & (Window =1)]

Switch	Door	Window	Dark	AlarmRing
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	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

	W'Da'	W'Da	WDa	WDa'
S'Do'	0	0	0	0
S'Do	0	0	0	0
SDo	0	0	1	1
SDo'	0	0	1	0

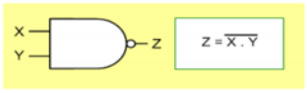
Answer: SWDa + SDoW

Universal Logic Gate

NAND and NOR gates are universal logic gates. Any gate can be constructed using only NAND or only NOR gates.

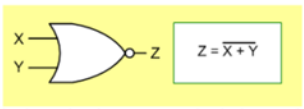
NAND

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

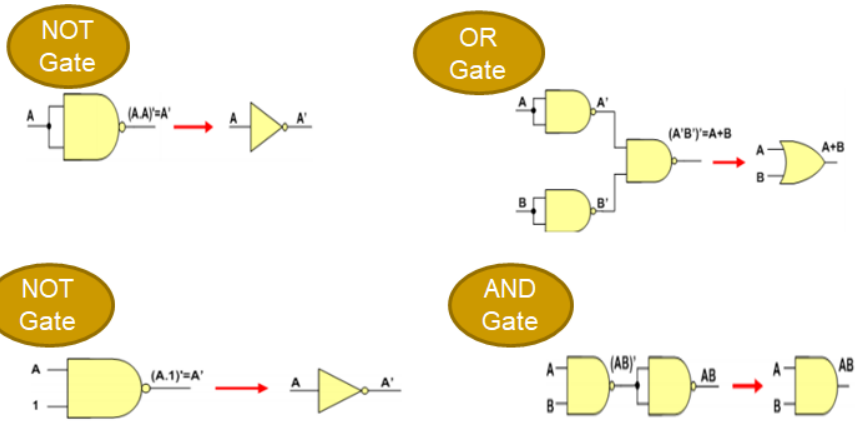


NOR

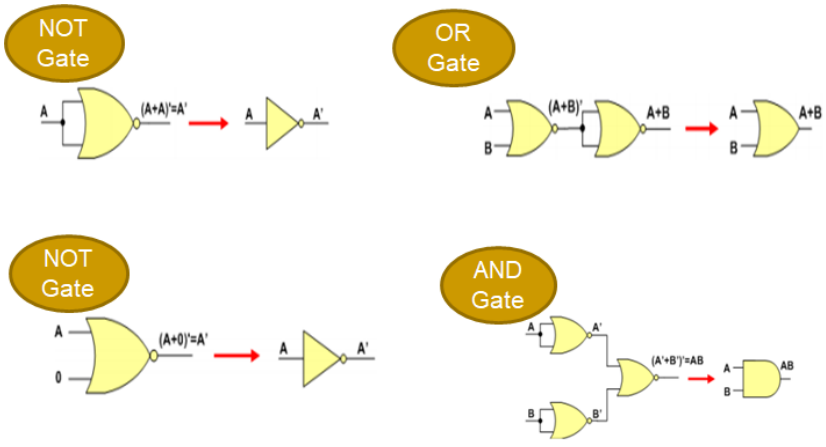
X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



Constructing other gates using only NAND gates:



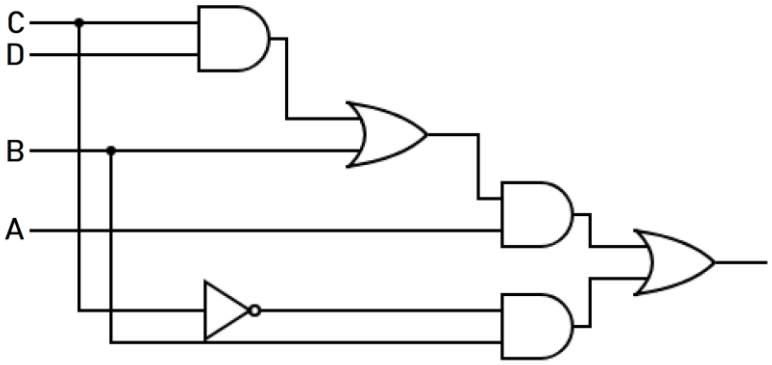
Constructing other gates using only NOR gates:



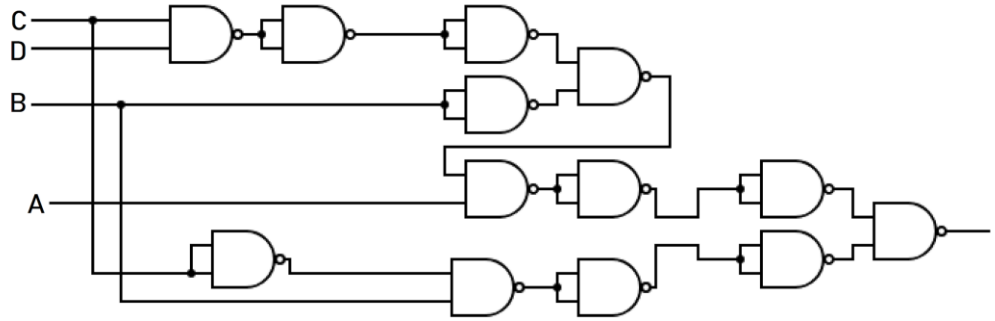
Example: Construct the following circuit using i)only NAND gates ii)only NOR gates

F=A(B+CD)+BC'

Normal Circuit:



Using only NAND gate:



Using only NOR gate: