2.X OTHER LOGIC GATES

There are 2^{2^n} different combinations (Functions) for n binary variables.

For $n=2$;

x y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

1) Two functions that produce a constant 0 or 1

$F_0 = 0$ (Null) $F_{15} = 1$ (Identity)

2) Four functions with unary operations: Complement and Transfer.

$F_3 = x$ Transfer (x)

$F_5 = y$ Transfer (y)

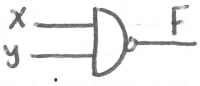
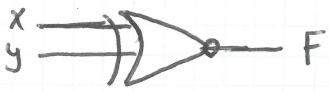
$F_{10} = y'$ Complement (Not y)

$F_{12} = x'$ Complement (Not x)

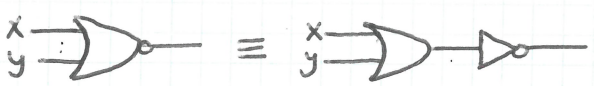
3) Ten functions with binary operators:

	Another notation	Name	comments
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x but not y
$F_4 = x'y$	y/x	Inhibition	y but not x
$F_6 = x'y + xy'$	$x \oplus y$	Exclusive-OR (XOR)	x or y but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x+y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$x \odot y$	Exclusive-NOR Equivalence (X-NOR)	x equals y
$F_{11} = x + y'$	$x \subset y$	Implication	If y then x
$F_{13} = x' + y$	$x \supset y$	Implication	If x then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND

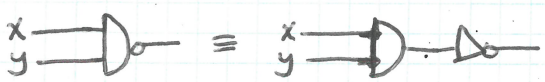
Out of the above 16 functions, only the complement, transfer AND, OR, NAND, NOR, & XOR, and Exclusive-NOR are used.

Name	Graphic Symbol	Function	Truth table
AND		$F = xy$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x+y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = X$	x | F 0 | 0 1 | 1
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x+y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR XOR		$F = x \oplus y$ $= xy' + x'y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR XNOR Equivalence		$F = x \odot y$ $= xy + x'y'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Note that;



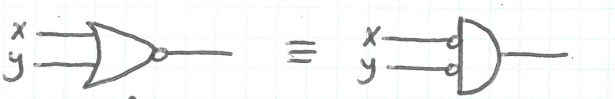
; similarly



$(xy)' = x'+y'$ as DeMorgan

$(x+y)' = x' \cdot y'$

NAND ≡ Negative-OR



NOR ≡ Negative-AND

2.4 Boolean Functions:

A Boolean function F, is an expression formed with binary variables, the binary operators OR and AND, unary operator NOT, parentheses, and an equal sign. For a given values of the variables, the function can be either 0 or 1.

Example:

$F_1 = xyz'$, $\Rightarrow$ $F_1 = 1$ if $x=1$, $y=1$, and $z'=1$ $(z=0)$; otherwise $F_1 = 0$

Consider the functions:

$F_2 = x + y'z$ $\qquad$ $F_3 = x'y'z + x'yz + xy'$ $\qquad$ $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

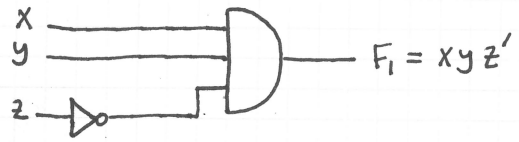
Is it possible to find two algebraic expressions to specify the same function? YES

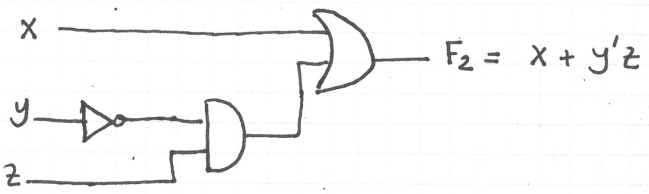
$F_3 \equiv F_4$

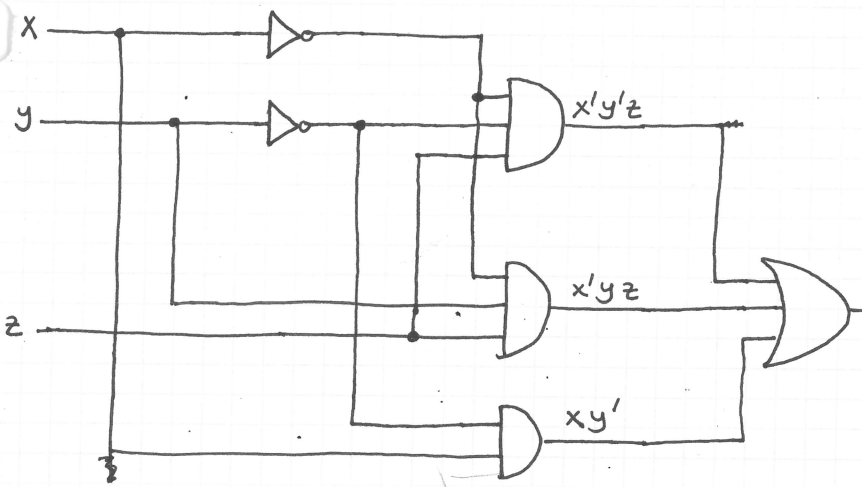
$F_3 = x'y'z + x'yz + xy'$

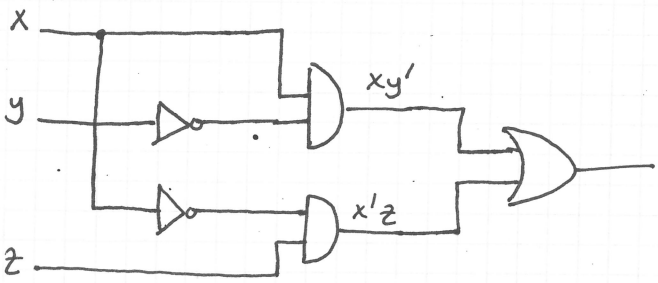
$= x'z(\underbrace{y'+y}_{1}) + xy' = x'z + xy'$

- A Boolean function may be transformed from an algebraic expression into a logic diagram.









Since $F_3 \equiv F_4$; it is more economical to implement F_4 because it has fewer gates $\Rightarrow$ we should simplify the function as much as possible.

Algebraic Manipulation:

Defn: A literal is a single variable within a term which may or may not be complemented. for example, x, y, x', y'

The number of literals in a Boolean function can be minimized by algebraic manipulations. ~~using some~~ Unfortunately, there are no specific rules to follow that will guarantee the final answer.

Examples:

1) $x + x'y = (x+x')\cdot(x+y) = 1\cdot(x+y) = x+y$ — Axioms: 4(b), 5(a), 2(b)

2) $x(x'+y) = xx' + xy = 0 + xy = xy$ — Axioms: 4(a), 5(b), 2(a)

3) $x'y'z + x'yz + xy' = x'z(y'+y) + xy'$
$= x'z\cdot 1 + xy' = x'z + xy'$ — Axioms: 4(a), 5(a), 2(b)

4) $xy + x'z + yz = xy + x'z + yz(x+x')$ — Axioms: 5(a), 2(b)
$= xy + x'z + yzx + yzx'$ — 4(a)
$= xy(1+z) + x'z(1+y)$ — 4(a)
$= xy + x'z$ — Theorem 2(a)

5) $(x+y)(x'+z)(y+z) = (x+y)(x'+z)(y+z+xx')$ — 5(b), 2(a)
$= (x+y)(x'+z)(x+y+z)(x'+y+z)$ — 4(b)
$= (x+y)(x+y+z)(x'+z)(x'+y+z)$ — 3(b)
$= (x+y)(x'+z)$ — Theorem 6(b)

6) $x'y'z' + x'y'z + xy'z' + xy'z + xyz' = x'y'(z'+z) + xy'(z'+z) + xyz'$
$= x'y' + xy' + xyz'$
$= (x'+x)y' + xyz'$
$= y' + xyz' = y' + y\cdot(xz')$
$= (y'+y)\cdot(y'+xz')$
$= y' + xz'$ //

Complement of a function:

The complement of a function F is F' and is obtained from an interchange of 0's and 1's in the value of F and by applying the generalized DeMorgan's theorem.

$(A+B+C+D+\cdots+G)' = A'\cdot B'\cdot C'\cdot D'\cdot\cdots\cdot G'$

$(ABC\cdots G)' = A'+B'+\cdots+G'$

Examples:

1) Find the complement of $F_1 = x'yz' + x'y'z$ by using DeMorgan's Theorem

$$F_1' = (x'yz' + x'y'z)' = (x'yz')' \cdot (x'y'z)'$$
$$= (x+y'+z) \cdot (x+y+z')$$

2) Find the complement of F_1 using the dual method {take the dual then, take complement of each part}

$F_1 = (x'yz') + (x'y'z)$

$\Rightarrow$ dual of $F_1 = (x'yz') \cdot (x'y'z)$

$\Rightarrow F_1' = (x'yz')' \cdot (x'y'z)' = (x+y'+z) \cdot (x+y+z')$

Find the complement of $F_2 = (A'+B) + CD$

$F_2' = ((A'+B) + CD)' = (A'+B)' \cdot (CD)' = (AB') \cdot (C'+D')$

4) Repeat 3 with dual method

$F_2 = (A'+B) + (CD)$

$\Rightarrow$ dual of $F_2 = (A'+B) \cdot (CD)$

$\Rightarrow F_2' = (A'+B)' \cdot (CD)' = (AB') \cdot (C'+D')$

5) Find the complement of $F_3 = X(y'z'+yz)$

$F_3' = [X(y'z'+yz)]' = X' + (y'z'+yz)'$

$= X' + [(y'z')' \cdot (yz)']$

$= X' + (y+z) \cdot (y'+z')$

6) Repeat 5 by the dual method.

$F_3 = X \cdot (y'z'+yz)$

$\Rightarrow$ dual of $F_3 = X + (y'z'+yz)$

$\Rightarrow F_3' = X' + (y'z'+yz)'$

$= X' + (y'z')' \cdot (yz)' = X' + (y+z) \cdot (y'+z')$

7) Find the complement of $F_4 = (A+B) \cdot C'D' + E + F'$

$F_4' = ((A+B) \cdot C' \cdot D' + E + F')' = ((A+B)C'D')' \cdot (E)' \cdot (F')'$

$= (A+B)' + C + D) \cdot E' \cdot F = (A'B' + C + D)E'F$

8) Repeat 7 using Dual method.

$F_4 = ((A+B)\bar{C}\bar{D}) + E + \bar{F} \Longrightarrow$ dual $= (A \cdot B + \bar{C} + \bar{D}) \cdot E \cdot F'$ (or) $F_4' = (A'B' + C + D) \cdot E' \cdot F$

$\Rightarrow$ dual of $F_4 = ((A+B)C'D') \cdot E \cdot F'$

$\Rightarrow F_4' = ((A+B)C'D')'' \cdot E' \cdot F$

$= ((A+B)' + C + D) E' F$

$= (A'B' + C + D) E' F$

Minterms and Maxterms:

- Any boolean expression may be expressed in terms of either minterms or maxterms.
- A minterm is the product of N distinct literals where each literal occurs exactly once. Each minterm is obtained from an AND term of the variables, with each variable being primed if the corresponding bit is a "0" and unprimed if "1".
- A maxterm is the sum of N distinct literals where each literal occurs exactly once. Each maxterm is obtained from an OR term of the variables, with each variable being unprimed if the corresponding bit is a "0" and primed if a "1".

⇒ a maxterm is the complement of its corresponding minterm, and vice versa.

Example: with 3-variables

	Minterms		Maxterms	
x y z	Term	Symbol	Term	Symbol
0 0 0	$x'y'z'$	m_0	$x+y+z$	M_0
0 0 1	$x'y'z$	m_1	$x+y+z'$	M_1
0 1 0	$x'yz'$	m_2	$x+y'+z$	M_2
0 1 1	$x'yz$	m_3	$x+y'+z'$	M_3
1 0 0	$xy'z'$	m_4	$x'+y+z$	M_4
1 0 1	$xy'z$	m_5	$x'+y+z'$	M_5
1 1 0	xyz'	m_6	$x'+y'+z$	M_6
1 1 1	xyz	m_7	$x'+y'+z'$	M_7

This allows us to represent expressions in either Sum of Products or Product of Sums forms; which are called the standard forms.

Sum of Products: (SOP)

- The sum of products (standard form) form represents an expression as a sum of minterms (canonical form).
- To derive the SOP form from a truth table, OR together all of the minterms which give a value of 1.

Example: Consider the truth table

x y z	F1	F2
0 0 0	0	0
0 0 1	1	0
0 1 0	0	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

$\Rightarrow F_1 = x'y'z + x'y z' + xyz$ ← SOP form (standard)

$= m_1 + m_4 + m_7$ ↔ canonical form

$= \sum(1,4,7)$

$F_2 = x'yz + xy'z + xyz' + xyz$

$= m_3 + m_5 + m_6 + m_7 = \sum(3,5,6,7)$

Products of Sums: (POS)

- The POS (standard form) form represents an expression as a Product of maxterms (canonical form).
- To derive the POS form from a truth table, AND together all of the maxterms which give a value of 0.

for the above example,

$F_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$ ← POS form (standard)

$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$ ↔ canonical form

$= \Pi(0,2,3,5,6)$

Important: In fact, we can obtain POS from SOP and vice versa.

For the above example:

$F_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$

$F_1 = (F_1')' = (x'y'z' + x'yz' + x'yz + xy'z + xyz')'$

$= (x'y'z')' \cdot (x'yz')' \cdot (x'yz)' \cdot (xy'z)' \cdot (xyz')'$

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

$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$

Example: Consider the truth table;

x y z	F	
0 0 0	0	M_0
0 0 1	1	m_1
0 1 0	0	M_2
0 1 1	0	M_3
1 0 0	1	m_4
1 0 1	1	m_5
1 1 0	1	m_6
1 1 1	1	m_7

$\Rightarrow F = x'y'z + xy'z' + xy'z + xyz' + xyz = \sum(1,4,5,6,7)$

or $F = (x+y+z) \cdot (x+y'+z) \cdot (x+y'+z') = \Pi(0,2,3)$

or $F' = x'y'z' + x'yz' + x'yz$

$F = (F')' = (x'y'z')' \cdot (x'yz')' \cdot (x'yz)$

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

similarly to obtain SoP from POS;

$F' = (x+y+z')(x'+y+z)(x'+y+z')(x'+y'+z)(x'+y'+z')$

$\Rightarrow F = (F')' = (x+y+z')' + (x'+y+z)' + (x'+y+z')' + (x'+y'+z)' + (x'+y'+z')'$

$= x'y'z + xy'z' + xy'z + xyz' + xyz$

$= m_1 + m_4 + m_5 + m_6 + m_7 = \sum(1,4,5,6,7)$

Example: Express the function $F = xy + x'z$ in POS form.

$F = xy + x'z$

$= xy(z+z') + x'z(y+y')$

$= \underbrace{xyz}_{m_7} + \underbrace{xyz'}_{m_6} + \underbrace{x'yz}_{m_3} + \underbrace{x'y'z}_{m_1}$

$F' = m_0 + m_2 + m_4 + m_5$

$= x'y'z' + x'yz' + xy'z' + xy'z$

$F = (F')' = (x'y'z')' \cdot (x'yz')' \cdot (xy'z')' \cdot (xy'z)'$

$= \underbrace{(x+y+z)}_{M_0} \cdot \underbrace{(x+y'+z)}_{M_2} \cdot \underbrace{(x'+y+z)}_{M_4} \cdot \underbrace{(x'+y+z')}_{M_5} = M_0 \cdot M_2 \cdot M_4 \cdot M_5$

Another solution: form the truth table and proceed!

Exercise!
Example: Express the function: $(AB+CD)(A'B'+C'D')$ in SOP

$(AB+CD)(A'B'+C'D') = \underbrace{AB\,A'B'}_{0} + ABC'D' + A'B'CD + \underbrace{CD\,C'D'}_{0}$

$= ABC'D' + A'B'CD$

Extension to Multiple Inputs:

- AND and OR operations are both commutative and Associative:

i.e $x+y = y+x$; $x \cdot y = y \cdot x$ {commutative}

$(x+y)+z = x+(y+z) = x+y+z$; $(x \cdot y) \cdot z = x \cdot (y \cdot z) = x \cdot y \cdot z$ {associative}

- NAND and NOR functions are commutative, but not associative

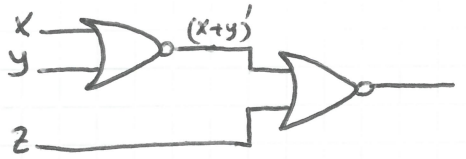
Ex! $[(x+y)'+z]' = (x+y) \cdot z' = xz' + yz'$

$[x+(y+z)']' = x' \cdot (y+z) = x'y + x'z$

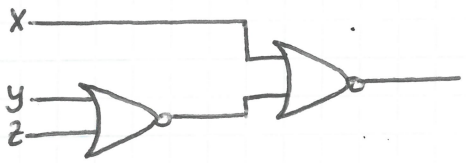
$\Rightarrow ((x+y)'+z)' \neq (x+(y+z)')'$

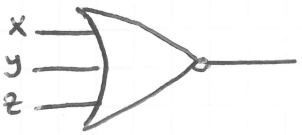
$\Rightarrow$ To overcome the difficulty, define multiple ~~NOR and NAND~~ inputs gates

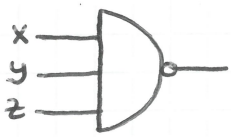
$((X+y)'+z)' \equiv$

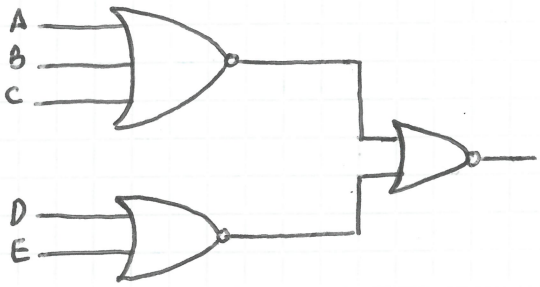


$(x+(y+z)')'$



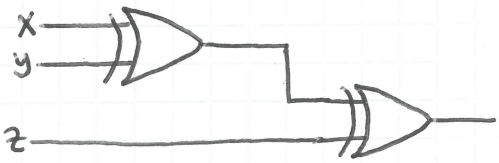


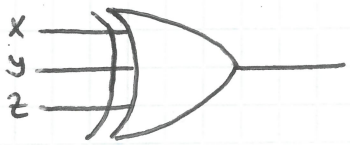




$F = \left((A+B+C)' + (D+E)'\right)' = (A+B+C)\cdot(D+E)$

- The XOR and XNOR functions are both commutative and associative and they can be extended to more than two inputs.





x y z	F=x⊕y⊕z	
0 0 0	0	0
0 0 1	1	1
0 1 0	1	1
0 1 1	0	0
1 0 0	1	1
1 0 1	0	0
1 1 0	0	0
1 1 1	1	1

Integrated Circuits: (IC) Read 2.8

⊙	
0 0	1
0 1	0
1 0	0
1 1	1

⊕	
0 0	0
0 1	1
1 0	1
1 1	0

Chapter - 3
Simplification of Boolean Functions

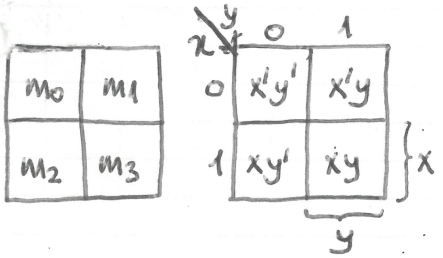
The Map Method:

Boolean functions may be simplified by means of the map method which is a simple straightforward procedure. The map method is also known as the "Karnaugh map". It relies on two-dimensional tables where

* Each square represents one minterm.

Two-Variable Map:

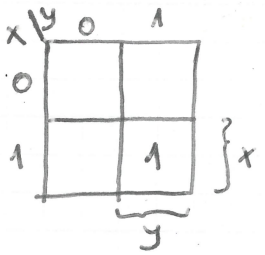
There are four minterms for two variables, hence the map consists of four squares (2x2 table):



x y	Minterms
0 0	$m_0 = x'y'$
0 1	$m_1 = x'y$
1 0	$m_2 = xy'$
1 1	$m_3 = xy$

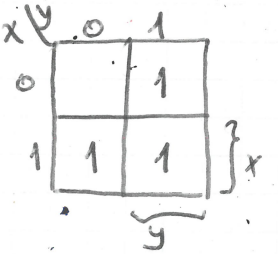
Example: Consider the function $F_1 = xy$

x y	$F_1 = xy$	
0 0	0	
0 1	0	
1 0	0	
1 1	1	m_3



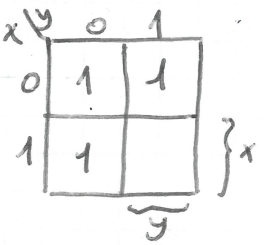
Example: $F_2 = x + y$ $\Rightarrow$

x y	F_2
0 0	0
0 1	1
1 0	1
1 1	1



Example: $F_3 = (xy)'$ $\Rightarrow$

x y	
0 0	1
0 1	1
1 0	1
1 1	0



Three-Variable Map

There are eight minterms ~~(squares)~~ for 3 variables, hence the map consists of 8 squares (2x4).

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

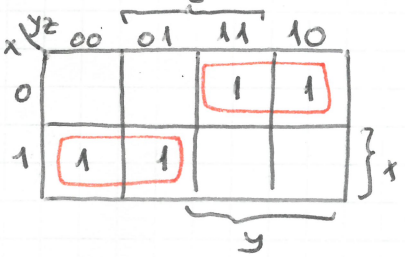
x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

(z over 01 and 11; x beside rows; y under columns)

x y z	Minterms	
0 0 0	$x'y'z'$	m_0
0 0 1	$x'y'z$	m_1
0 1 0	$x'yz'$	m_2
0 1 1	$x'yz$	m_3
1 0 0	$xy'z'$	m_4
1 0 1	$xy'z$	m_5
1 1 0	xyz'	m_6
1 1 1	xyz	m_7

- Note that the combination of yz follows the sequence 00, 01, 11, 10 not the numerical sequence 00, 01, 10, 11. This ensures that adjacent entries in the Karnaugh Map differs only by one variable.
- The map should be considered as being drawn on a surface where edges touch each other. (for example in case of 3 variables map m_0 is adjacent to m_2 ; m_4 is adjacent to m_6).

Example: Simplify the Boolean function $F(x,y,z) = \sum(2,3,4,5)$



- Main idea: Combine the adjacent 1's in the largest possible group of power of 2. (overlap may occurs).

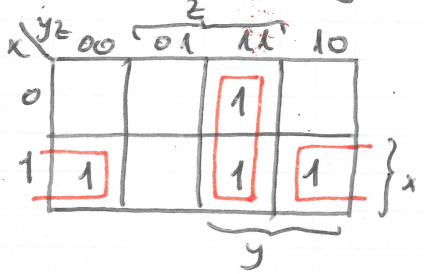
$\Rightarrow F = x'y + xy'$

Remember (previous chapter):

$F = x'yz' + x'yz + xy'z' + xy'z$

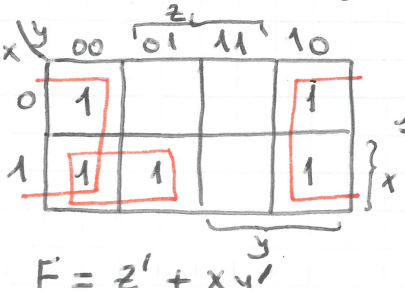
$= x'y(z'+z) + xy'(z'+z) = x'y + xy'$

Example: Simplify the Boolean function $F(x,y,z) = \sum(3,4,6,7)$



$\Rightarrow F = xz' + yz$

Example: Simplify the Boolean function $F(x,y,z) = \sum(0,2,4,5,6)$



Notes:
- 1 square represents a product term of 3 literals
- 2 adjacent squares " " " " 2 literals
- 4 " " " " " one. "

$F = z' + xy'$

Example: Given the Boolean function $F = A'C + A'B + AB'C + BC$

(a) Express F in sum of minterms

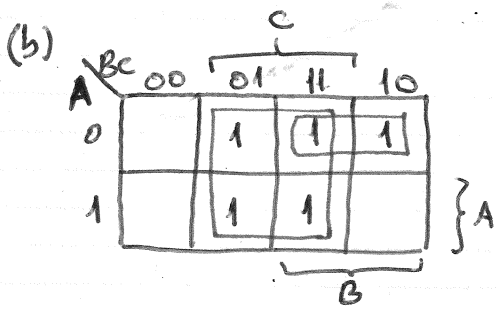
(b) Find the minimal SOP expression

(a)

A	B	C	F	
0	0	0	0	
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	0	
1	0	1	1	m_5
1	1	0	0	
1	1	1	1	m_7

$\Rightarrow F = \sum(1,2,3,5,7)$

(b)

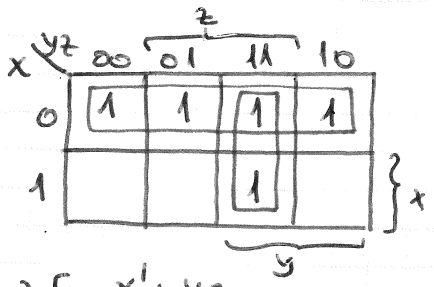


$\Rightarrow F = C + A'B$

Example: Given the Boolean function $F = x'y' + yz + x'yz'$

x	y	z	F	
0	0	0	1	m_0
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	m_7

$\Rightarrow F = \sum(0,1,2,3,7)$

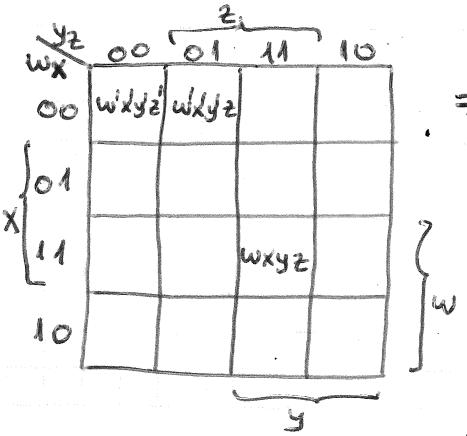


$\Rightarrow F = x' + yz$

Four-Variable Map

There are 16 minterms for 4 variables, the map has 16 squares (4x4)

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}



$\Rightarrow$ • one square represents a term of 4 lit.

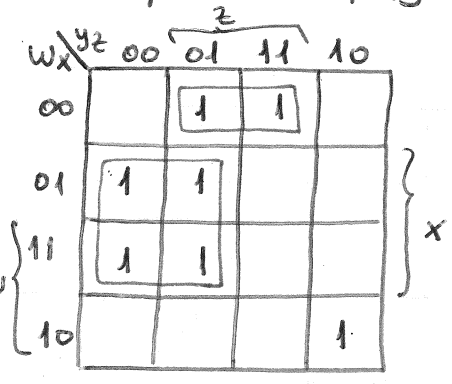
• Two squares " a term of 3 literals

• Four " " " " 2 "

• Eight " " " " 1 "

(w, x, y, z)

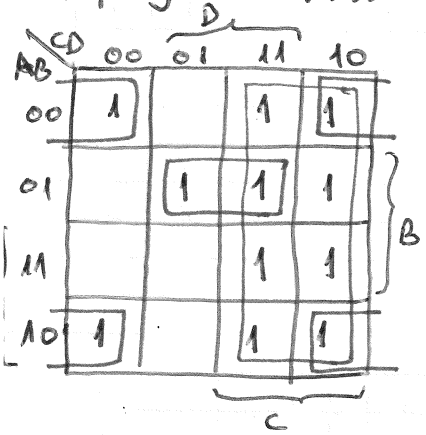
Example: Simplify the function $F = \sum(1,3,4,5,10,12,13)$



$\Rightarrow F = xy' + w'x'z + wx'yz'$

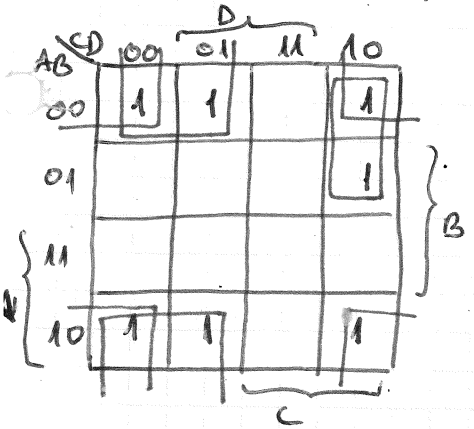
Example:

Simplify $F(A,B,C,D) = \sum(0,2,3,5,6,7,8,10,11,14,15)$



$\Rightarrow F = C + B'D' + A'BD$

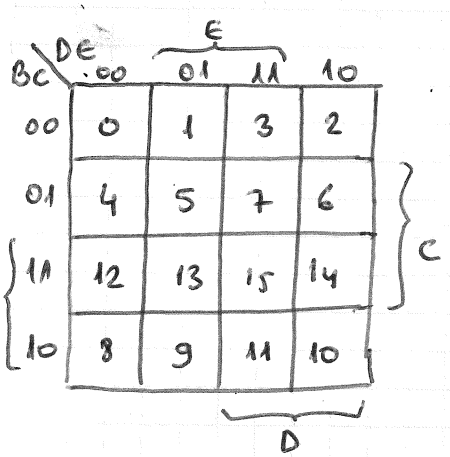
Example: Simplify $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

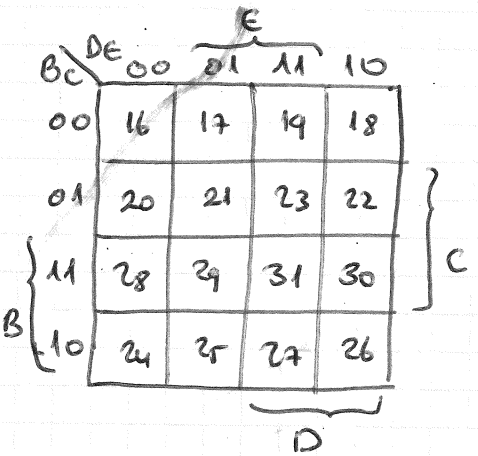


$\Rightarrow F = B'C' + B'D' + A'CD'$

Five-Variable Map

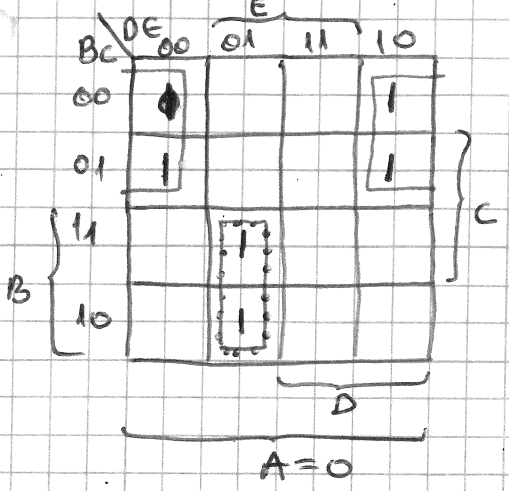
There are 32 minterms for 5 variables organized as two separate 16 squares. (e.g $F(A,B,C,D,E) \Rightarrow A$ distinguishes between the two four-variable maps).

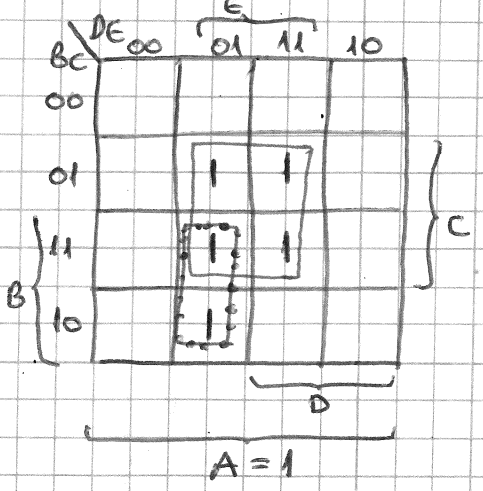




- 1 square represents a product term of 5 literals
- 2 " (s) " " " " " 4 "
- 4 " " " " " " 3 "
- 8 " " " " " " 2 "
- 16 " " " " " " 1 "

Example: Simplify the function $F(A,B,C,D,E) = \sum(0,2,4,6,9,13,21,23,25,29,31)$

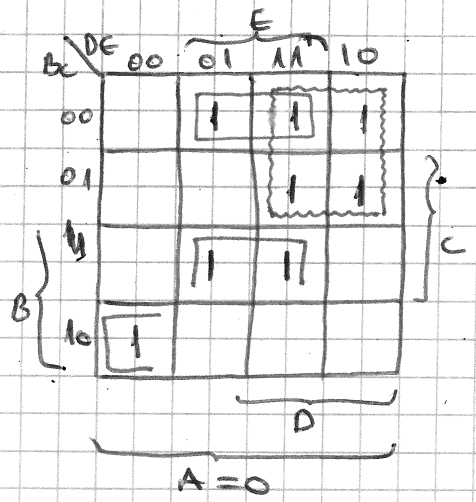


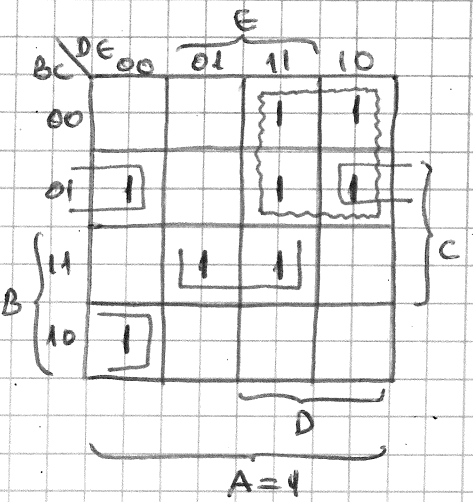


$\Rightarrow F = A'B'E' + ACE + BD'E$

- $m_0 + m_2 + m_4 + m_6 = A'B'E'$
- $m_{21} + m_{23} + m_{29} + m_{31} = ACE$
- $m_9 + m_{13} + m_{25} + m_{29} = BD'E$

Example: Simplify $F(A,B,C,D,E) = \sum(1,2,3,6,7,8,13,15,18,19,20,22,23,24,29,31)$





- $m_2 + m_3 + m_6 + m_7 + m_{18} + m_{19} + m_{22} + m_{23} = B'D$
- $m_{13} + m_{15} + m_{29} + m_{31} = BCE$
- $m_8 + m_{24} = BC'D'E'$
- $m_1 + m_3 = A'B'C'E$
- $m_{20} + m_{22} = AB'CE'$.

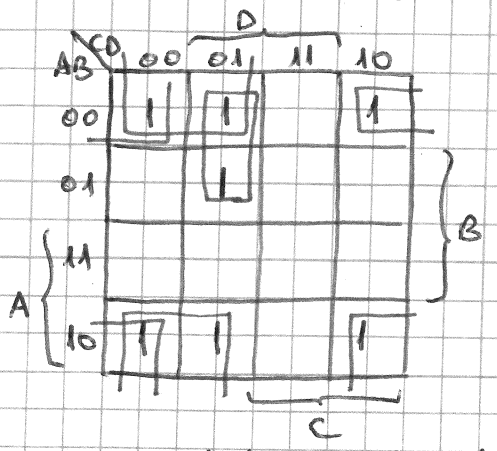
$\Rightarrow F = B'D + BCE + BC'D'E' + A'B'C'E + AB'CE'$

Product of sums Simplification:

- Up to now, we combined 1's in the Karnaugh map and we obtained F in SOP form.
- $\Rightarrow$ If we combine the 0's in the Karnaugh map, we will obtain $\underline{F'}$ in SOP form.
- $\therefore$ The complement of F' (i.e $(F')'$) gives us F in POS form.

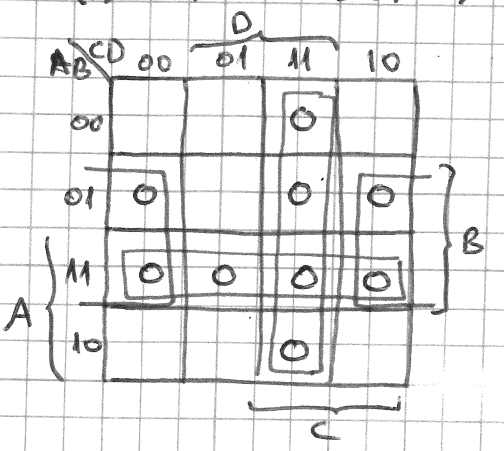
Example: Simplify the Boolean function $F(A,B,C,D) = \sum(0,1,2,5,8,9,10)$ in (a) Sum of Products (SOP) (b) Product of sums (POS)

(a) F in SOP ⇒ combine 1's



$\Rightarrow F = B'C' + B'D' + A'C'D$

(b) F' in SOP ⇒ combine 0's



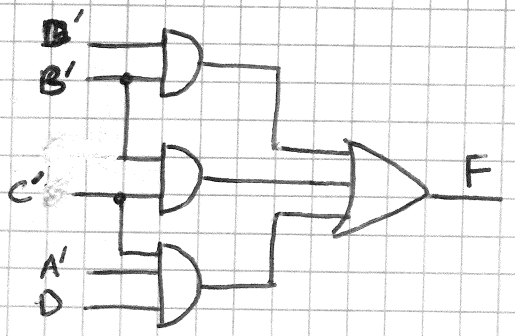
$F' = AB + CD + BD'$

$\Rightarrow F = (F')' = (AB + CD + BD')'$

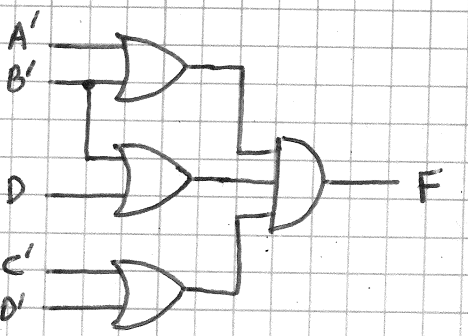
$= (A'+B')(C'+D')(B'+D)$

Assume that the complements of the input variables are available ⇒

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



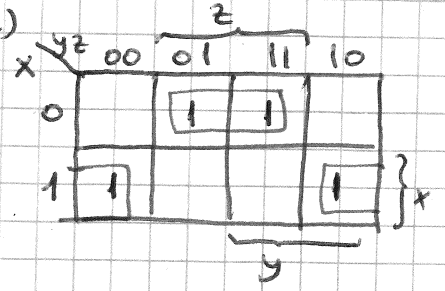
$F = (A'+B')(B'+D)(C'+D')$



Important Note: The procedure for obtaining the POS simplification is also valid when the function is originally expressed in the product of maxterms.

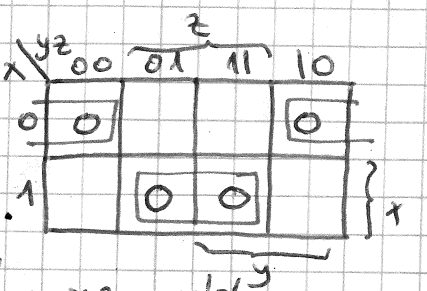
Example: Simplify the function $F(x,y,z) = \Pi(0,2,5,7)$ in (a) SOP and (b) POS

(a)



$\Rightarrow F = x'z + xz'$

(b)



$F' = xz + x'z'$

$\Rightarrow F = (F')' = (x'+z')(x+z)$

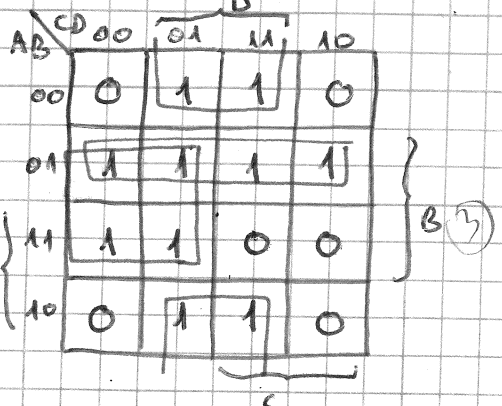
Note:

In order to enter a function expressed in product of sums in the map, the complement of the function is taken and from F', the squares to be marked with 0's are found.

Example: Simplify the function $F = (A'+B'+C')(B+D)$ in the two standard forms (i.e SOP, POS)

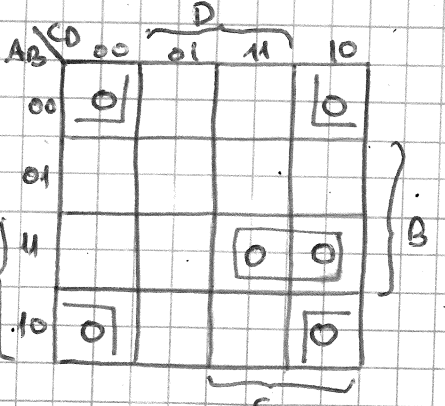
Take the complement $F' = ABC + B'D'$ {in order to see where are 0's}

(a)



$\Rightarrow F = A'B + BC' + B'D$ (4)

(b)



(1) $F' = ABC + B'D'$

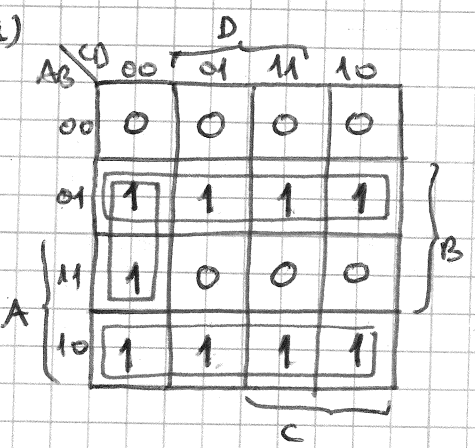
$\therefore F = (F')' = (A'+B'+C')(B+D)$

Example: Simplify the funchon $F = (A+B)(A'+B'+C')(A+B+C'+D)(A'+B'+C+D)$

in (a) Sum of Products (b) Product of sums.

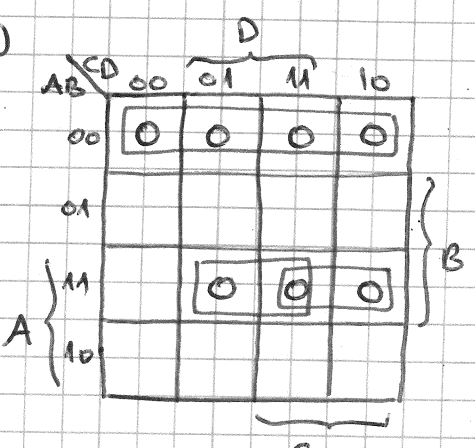
$F' = A'B' + ABC + A'B'CD' + ABC'D$

(a)



$\Rightarrow F = AB' + A'B + BC'D'$

(b)



$F' = A'B' + ABC + ABD$

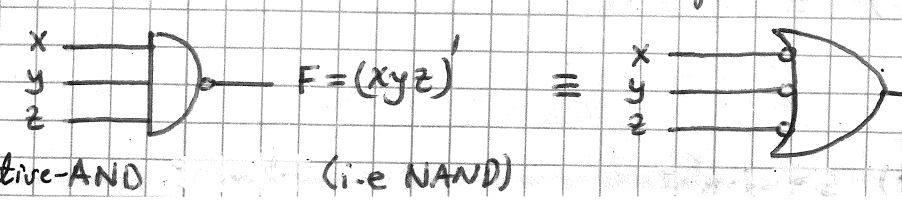
$\Rightarrow F = (A+B)(A'+B'+C')(A'+B'+D')$

Go to don't care

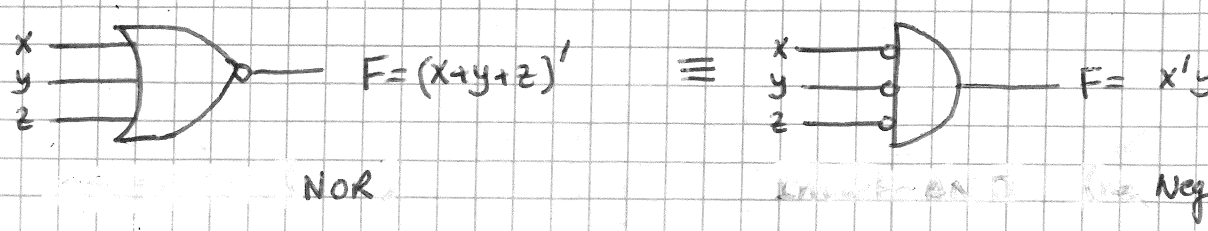
NAND and NOR IMPLEMENTATION:

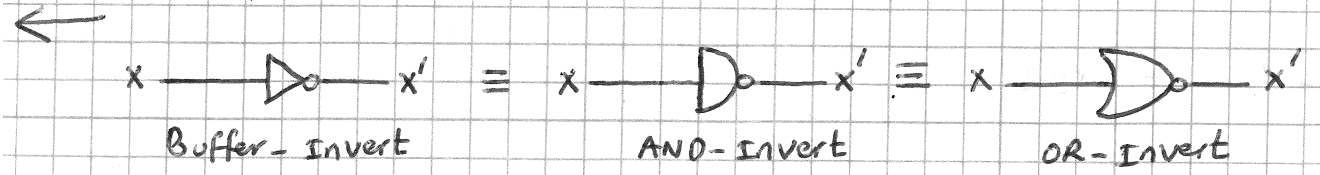
- Digital circuits are ~~most~~ more frequently implemented with NAND or NOR gates than AND and OR gates since NAND and NOR are easier to fabricate. Therefore we have to convert other gates to these gates.

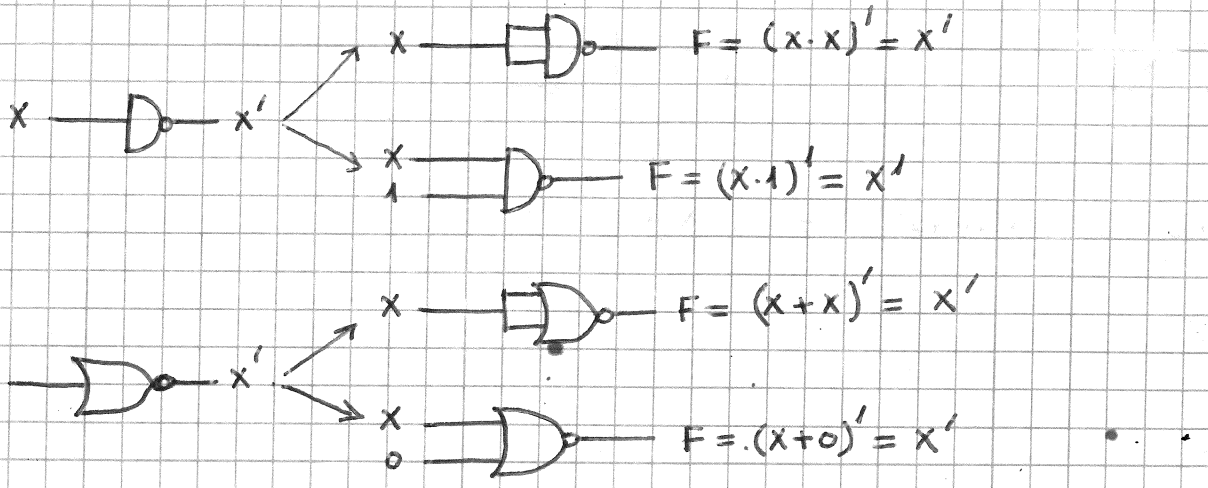
Remember:



Negative-AND (i.e NAND) Negative-OR



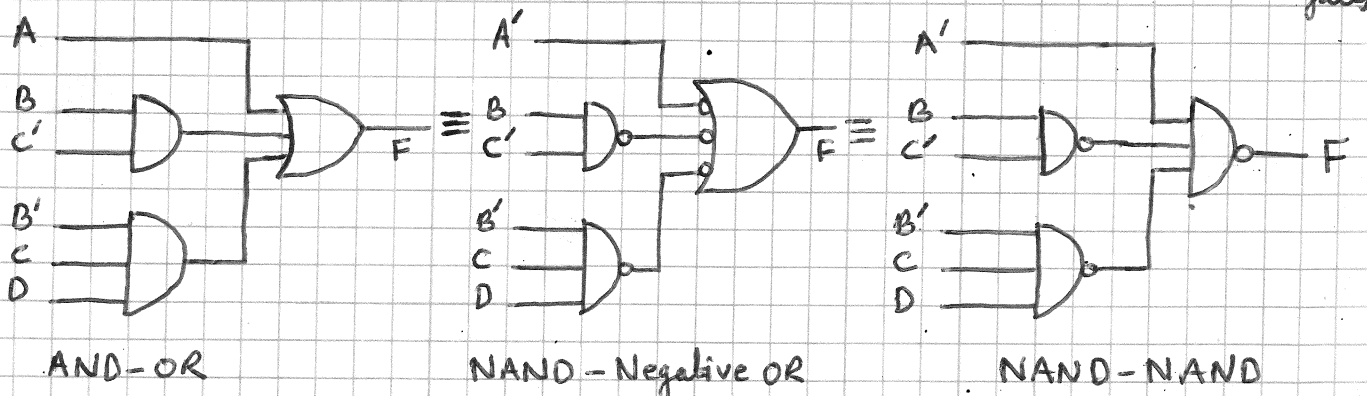




NAND Implementation:

The implementation of a function with NAND gates requires that the function must be simplified in the sum of products form.

Example: Implement the function $F = A + BC' + B'CD$ with NAND gates.

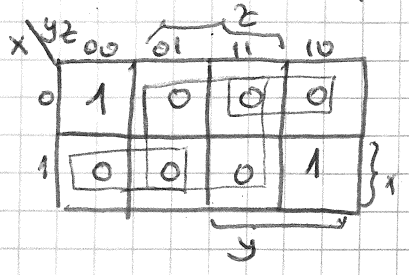


Procedure for designing a minimum two-level NAND-NAND logic diagram:

1) Simplify the function in sum of products form.
2) Draw a NAND gate for each product term that has at least two literals. The literals are the inputs.
3) Draw a single NAND gate in the second level, with inputs coming from the outputs of the first-level gates.
4) A term with a single literal will be complemented and applied as an

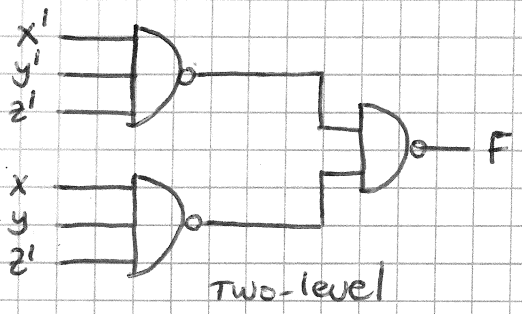
- There is a second method to implement a Boolean function with NAND gates: - Express F' in sop by combining 0's and implement it with two levels NAND gates
  - Insert an Inverter at the output to obtain F.

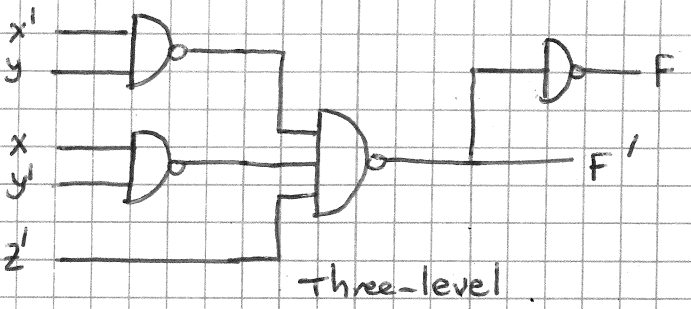
Example: Implement $F(x,y,z) = \sum(0,6)$ with NAND gates



$F = x'y'z' + xyz'$

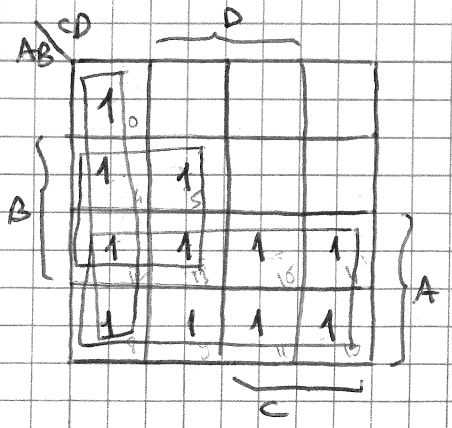
$F' = x'y + xy' + z$



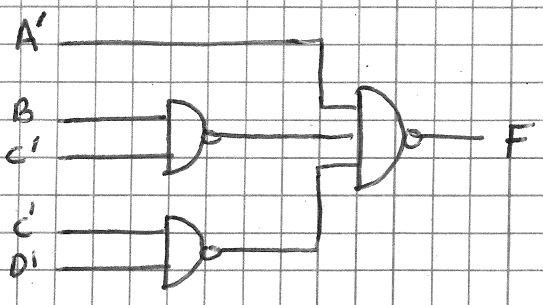


Example:

Implement the function $F(A,B,C,D) = AB' + ABD + ABD' + A'C'D' + A'BC'$ with two-level NAND gate circuit.



$F = A + BC' + C'D'$



NOR Implementation:

1- Simplify the function in product of sums form.

2- Draw a NOR gate for each sum that has at least two literals.

3- Draw a single NOR gate in the second level, with inputs coming from outputs of first-level NOR gates.

4- A term with single literal requires an inverter or may be complemented