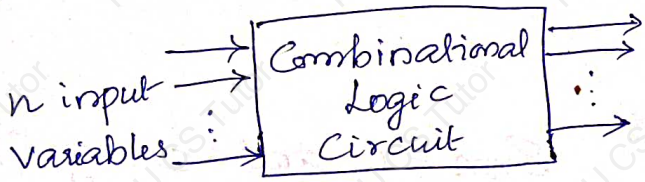
Combinational Logic Circuits

- Consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs.

Sequential circuits employ memory elements (binary cells), in addition to logic gates. Their outputs ~~employ m~~ are a function of the inputs and the state of the memory elements.

Combinational circuit consists of input variables, logic gates and output variables. The logic gates accept signals from the inputs and generate signals ~~to~~ the outputs.



- 2^n possible binary input combinations.
- for each i/p combination, only one output combination.

Design Procedure

1. The problem is stated
2. The number of available input variables and required output variables is determined.
3. Input and output variables are assigned letter symbols
4. Truth table that defined the required relationships between inputs and outputs is derived.
5. The simplified boolean function for each output is obtained.
6. The logic diagram is drawn.

Constrains to be considered while logic circuits design are :

① Minimum number of gates.
② Minimum number of inputs to a gate
③ Minimum propagation time of the signal through the circuit.
④ Minimum number of inter connections.
⑤ limitations of the driving capabilities of each gate.

Binary Adders

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10. Sum = 0 and carry = 1.

A combinational circuit that performs the addition of two bits is called a half adder and of three bits (two significant bits and a previous carry) is a full adder.

Half Adder

(Augend and audend)
- two binary inputs and two binary outputs. (Sum and Carry)
- two inputs x, y and two outputs S (for sum) and C (for carry)

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The simplified sum of products expressions for the outputs are:

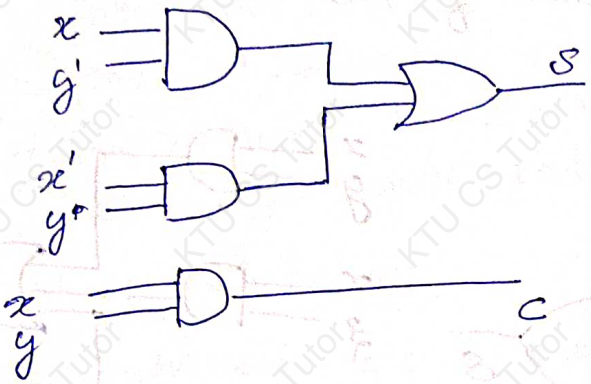
$S = x'y + xy'$

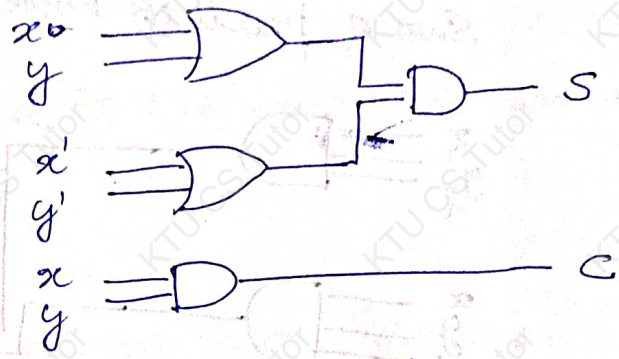
$C = xy$

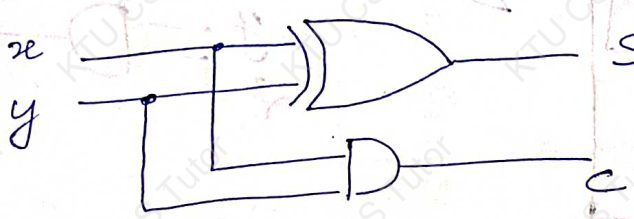
Product of Sum form.

$S = (x+y)(x'+y')$

$C = xy$







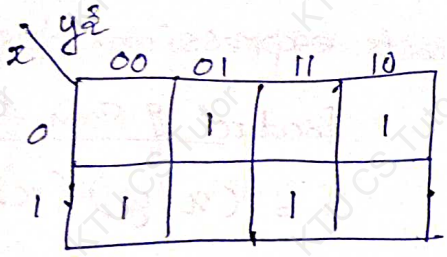
$S = x \oplus y$

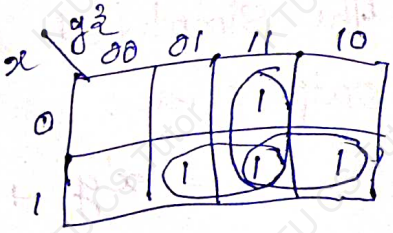
$C = xy$

Full Adder

- forms the sum of three input bits.
- three inputs and two outputs.
- two inputs x and y represent the two significant bits to be added. The third input, z represents the carry from the previous lower significant position.

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

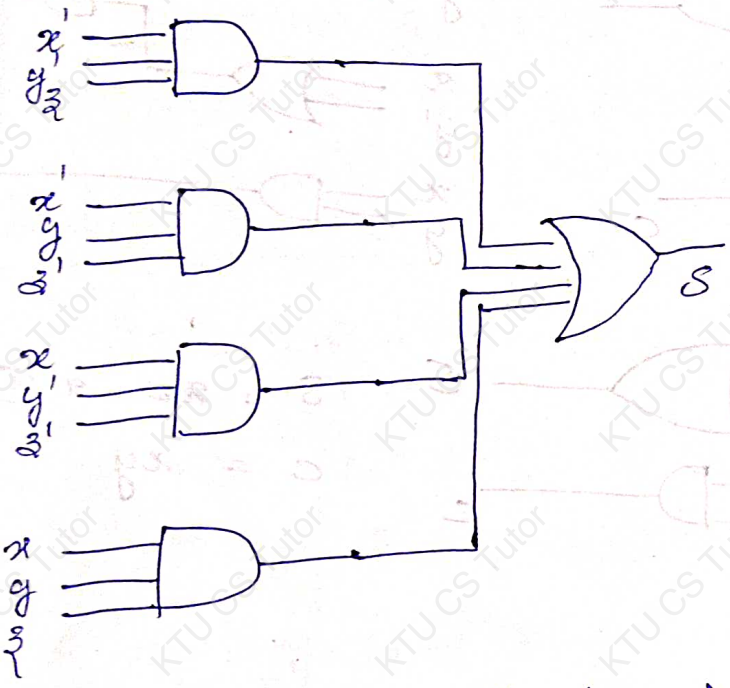


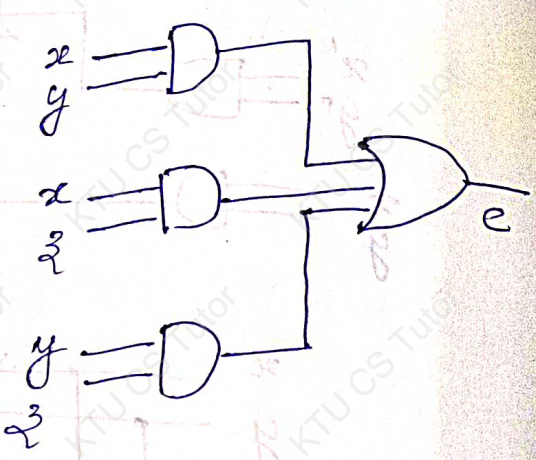


$C = xy + xz + yz$

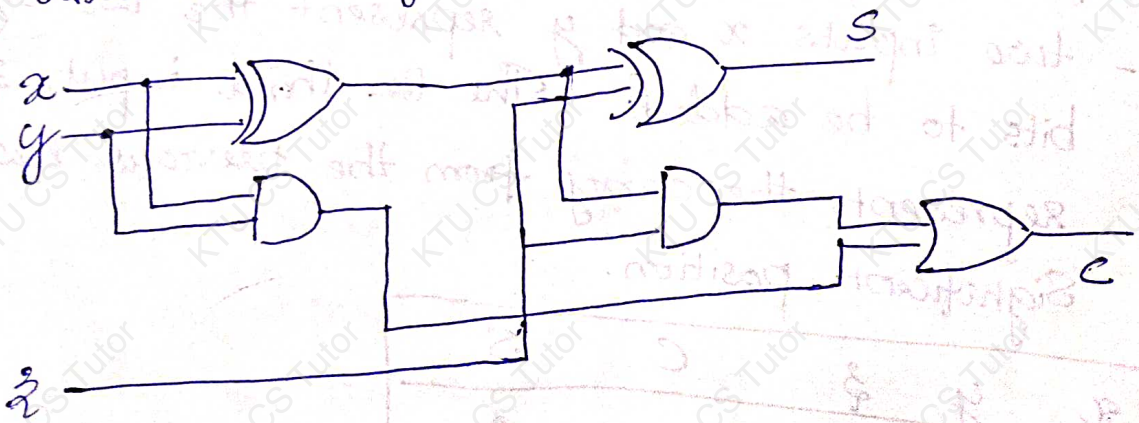
$S = x'y'z + x'yz' + xy'z' + xyz$

Sum of products implementation of full adder





Full adder can be implemented with two half adders and one OR gate



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

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

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

Subtractors

- Subtraction accomplished by taking the complement of the subtrahend and adding it to the minuend.
- can be implemented for direct method also.
- each subtrahend bit of the number is subtracted from its corresponding significant minuend bit to form a difference bit.
- If minuend bit is smaller, a 1 is borrowed from the next significant position.

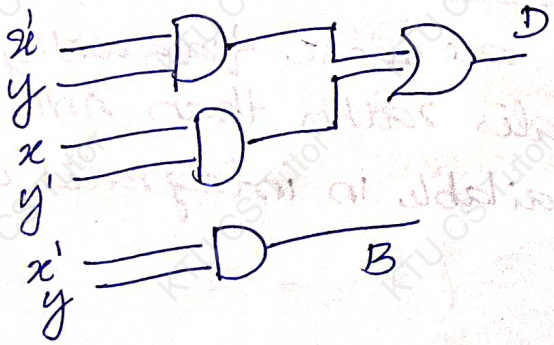
Half Subtractor

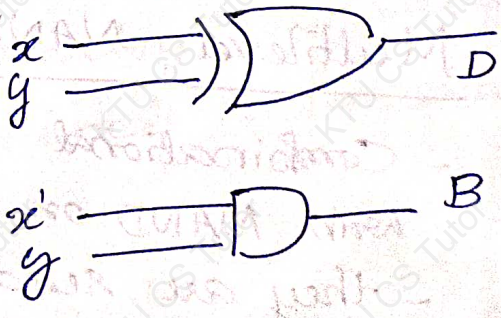
- Subtracts two bits and produces their difference.
- two input and two output (difference and Borrow)

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$D = x'y + xy' = x \oplus y$

$B = x'y$

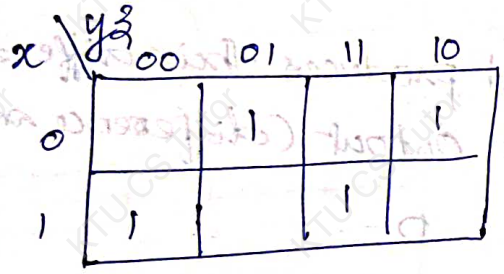


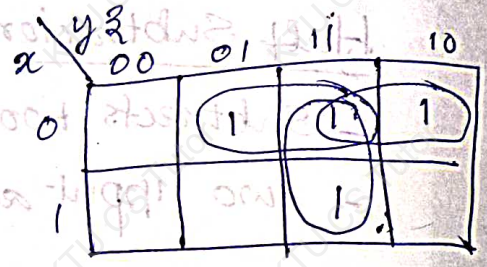


Full Subtractor

- also considers previous borrow for the subtraction.
- three input and two outputs.

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1





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

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

Output D is exactly same as output S in the full adder and output B resembles the function for C in the full adder except that the input variable x is complemented.

Multilevel NAND Circuits

- Combinational circuits are more frequently constructed with NAND or NOR gates rather than AND and OR.
- they are readily available in integrated circuit form

Universal Gate

- Any digital system can be implemented with it.
- Combinational circuits and sequential circuits can be constructed.
- AND, OR and NOT can be implemented with NAND gates

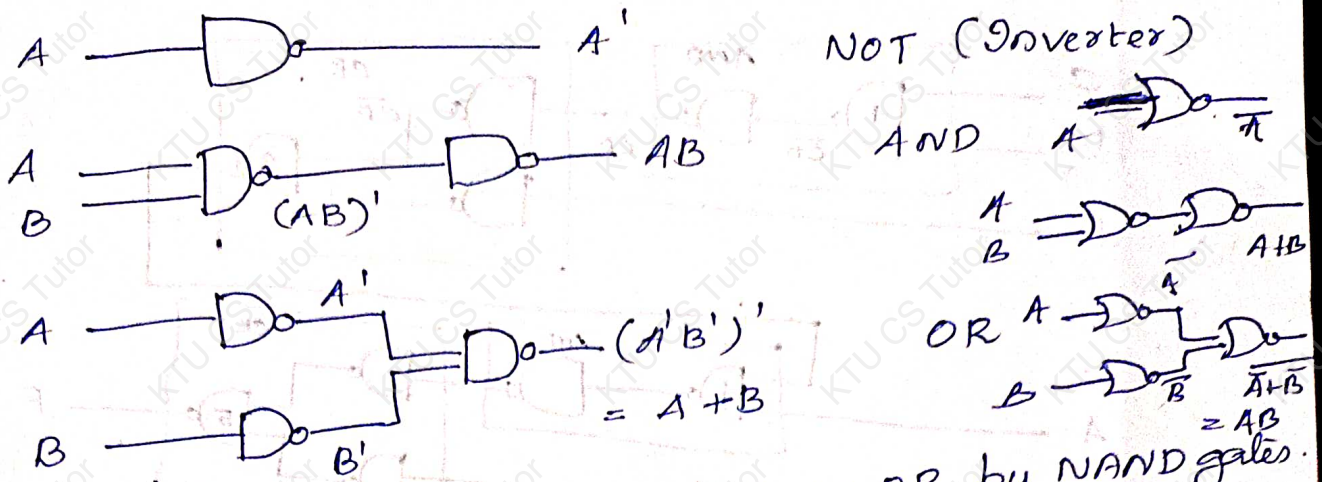


Fig. implementation of NOT, AND OR OR by NAND gates.

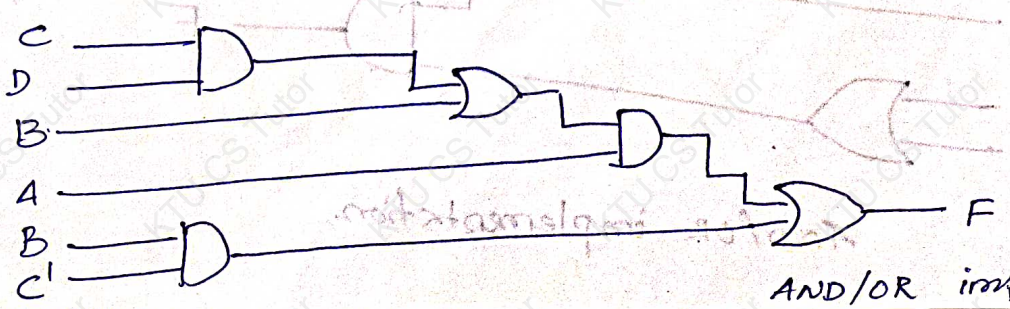
Boolean Function Implementation and Derivation

- two methods.
  1. Block diagram method for Boolean Function Implementation
  2. by Algebraic Manipulation for derivation of Boolean Function.

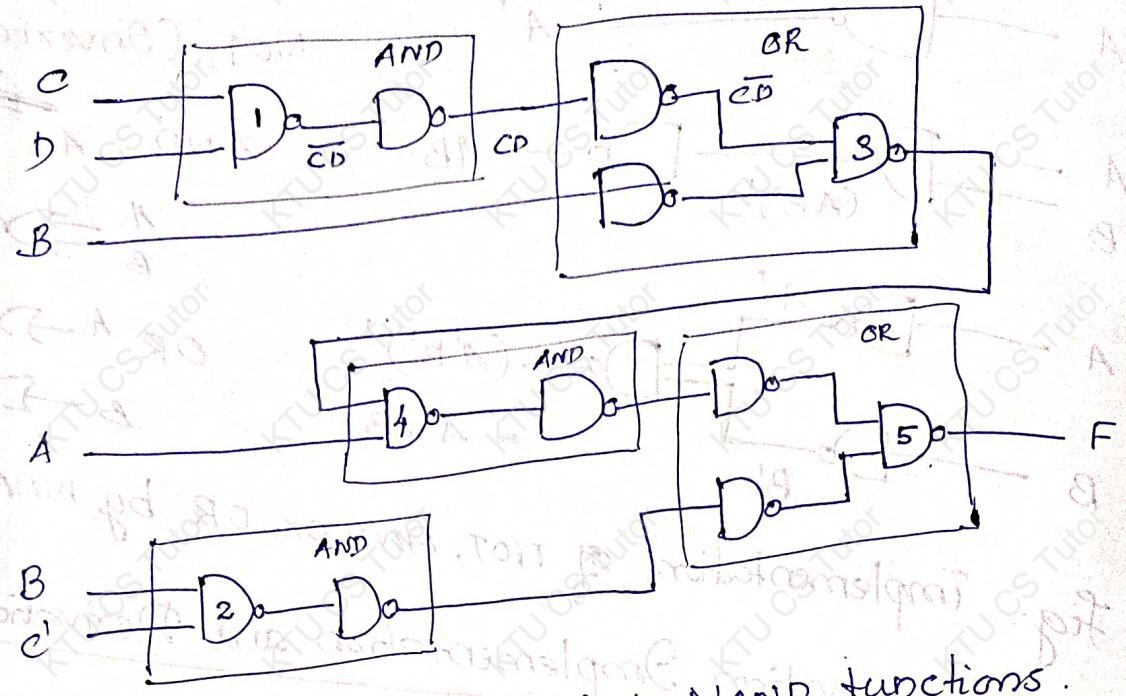
Block diagram Manipulation Method.

1. Draw the logic diagram for the given algebraic expression with AND, OR and NOT gates.
2. Draw a second logic diagram with the equivalent NAND logic substituted for each AND, OR and NOT gate
3. Remove any two cascaded inverters from the diagram. Remove inverters connected to single external inputs and complement the corresponding input variable. The new logic diagram obtained is the required NAND gate implementation.

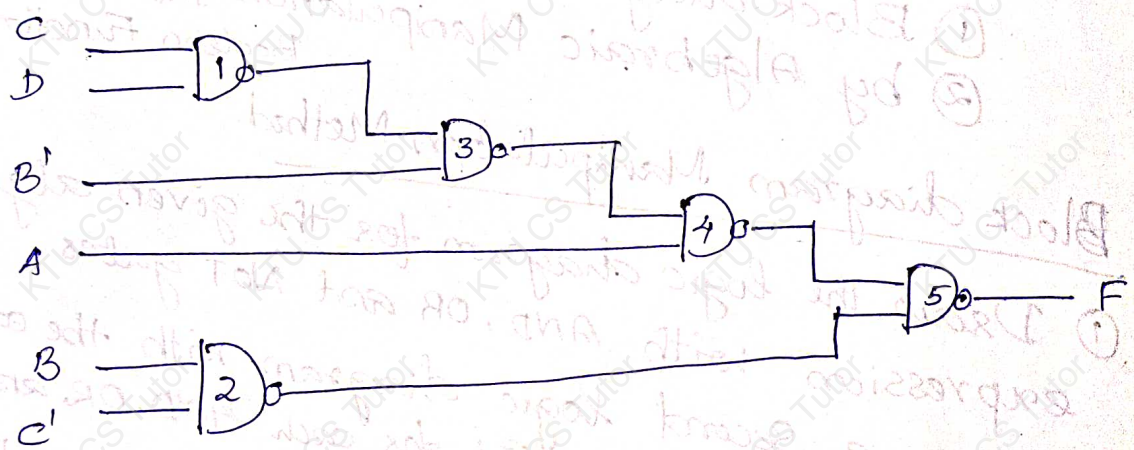
eg. $F = A(B + CD)' + BC'$.



AND/OR implementation.



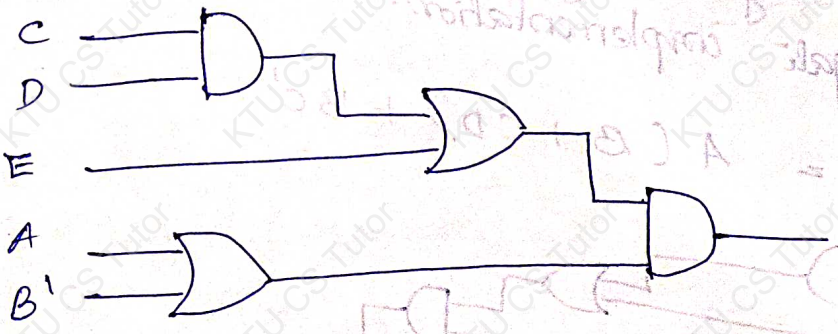
Substituting equivalent NAND functions.



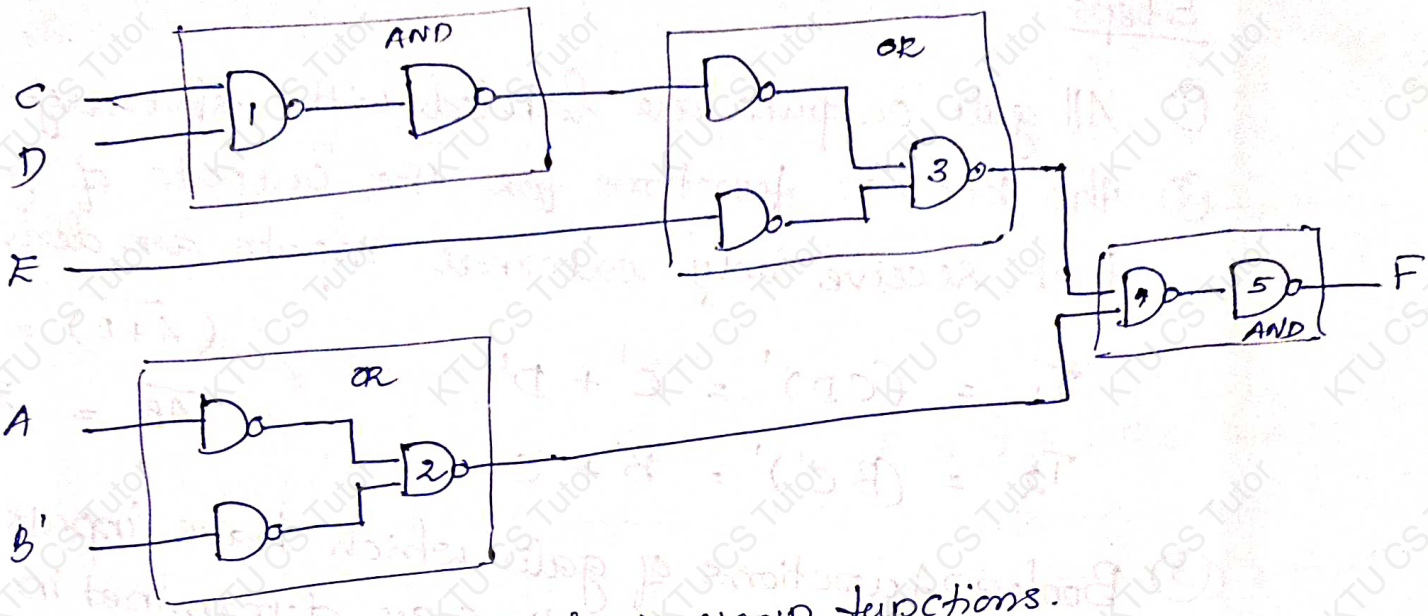
NAND implementation.

Example - 2

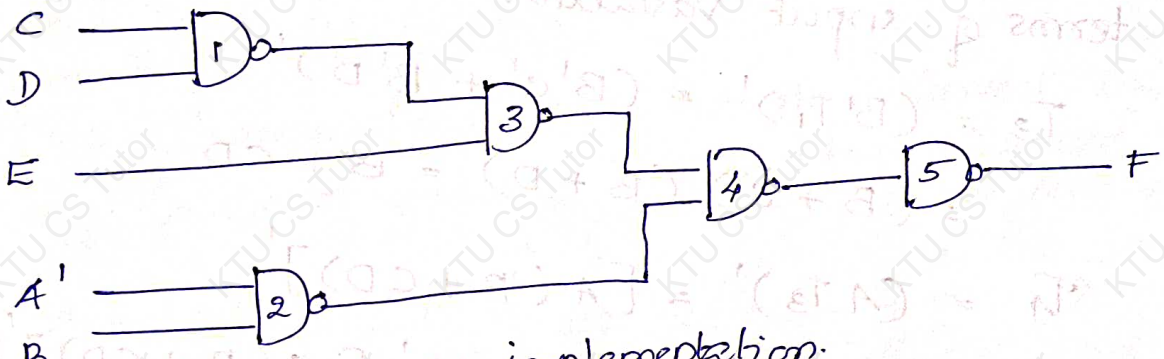
$F = (A + B')(CD + E)$



AND/OR implementation.



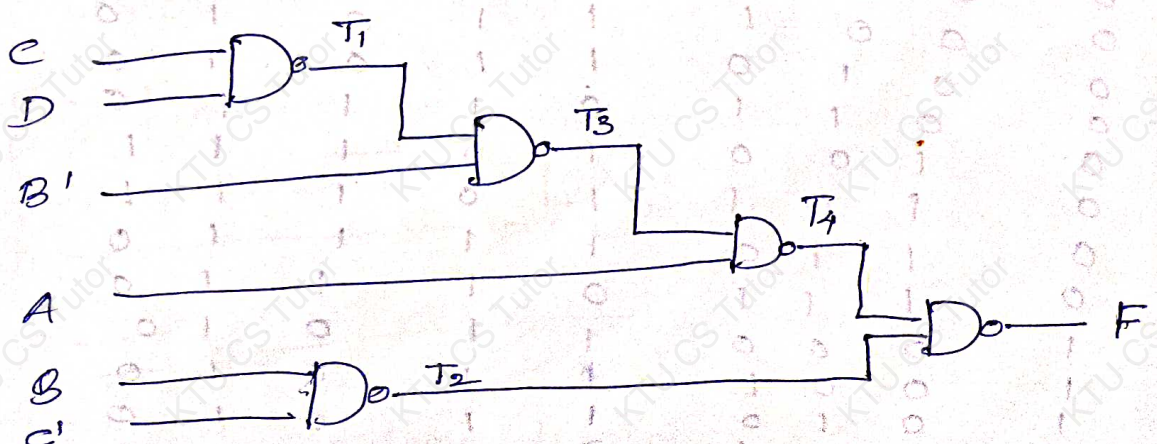
Substituting equivalent NAND functions.



NAND implementation.

Derivation of the Boolean Function by Algebraic Manipulation

Given the NAND logic diagram



Steps.

① All gate outputs are labeled with arbitrary symbols.

② The Boolean functions for the outputs of gates that receive only external inputs are derived.

$$T_1 = (CD)' = C' + D'$$

$$T_2 = (BC')' = B' + C$$

$$\overline{(A+B)} = \bar{A}.\bar{B} \qquad \overline{AB} = \bar{A}+\bar{B}$$

③ Boolean functions of gates which have inputs from previously derived functions are determined in consecutive order until the output is expressed in terms of input variables.

$$T_3 = (B'T_1)' = (B'C' + B'D')'$$

$$= (B+C)(B+D) = B + CD \qquad B + BD + BC + CD$$

$$T_4 = (AT_3)' = [A(B+CD)]'$$

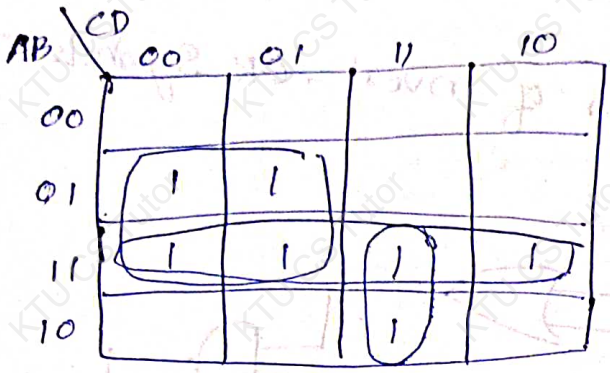
$$F = (T_2 T_4)' = \{(BC')'[A(B+CD)]'\}'$$

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

Derivation of Truth Table.

A	B	C	D	T_1	T_2	T_3	T_4	F
0	0	0	0	1	1	0	1	0
0	0	0	1	1	1	0	1	0
0	0	1	0	1	1	0	1	0
0	0	1	1	0	1	1	1	0
0	1	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1	1
0	1	1	0	1	1	1	1	0
0	1	1	1	0	1	1	1	0
1	0	0	0	1	1	0	1	0
1	0	0	1	1	1	0	1	0
1	0	1	0	1	1	0	1	0
1	0	1	1	0	1	0	1	0

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



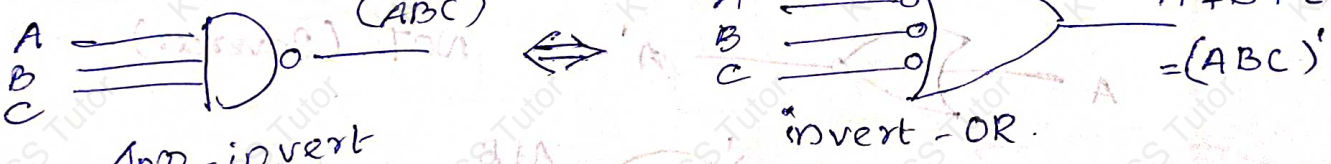
$F = AB + BC' + ACD$

$= AB + ACD + BC'$

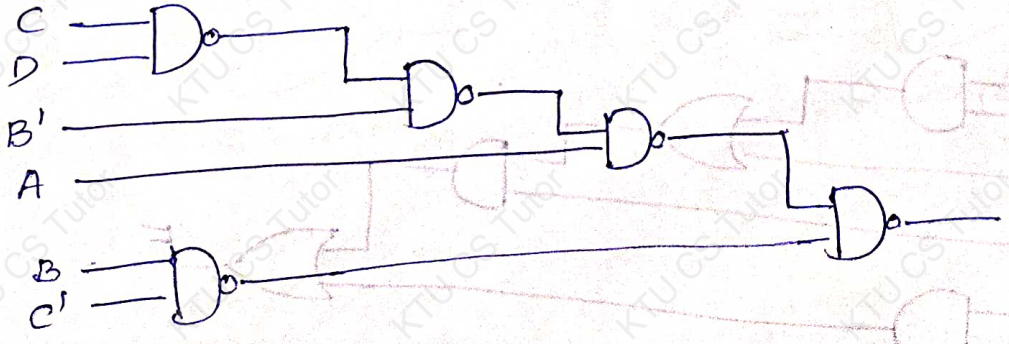
$= A'(B + CD) + BC'$

Conversion of NAND logic diagram to its equivalent AND-OR logic diagram

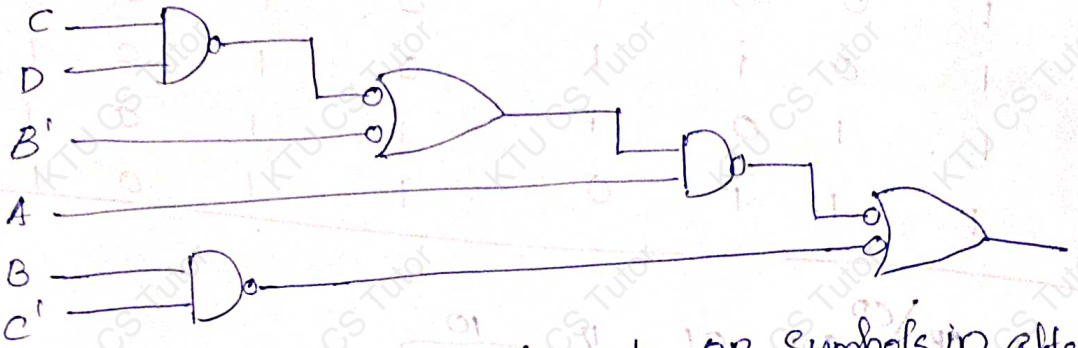
- Change from AND-invert to invert-OR in alternate levels of gates. First level to be changed to an invert-OR symbol should be the last level.



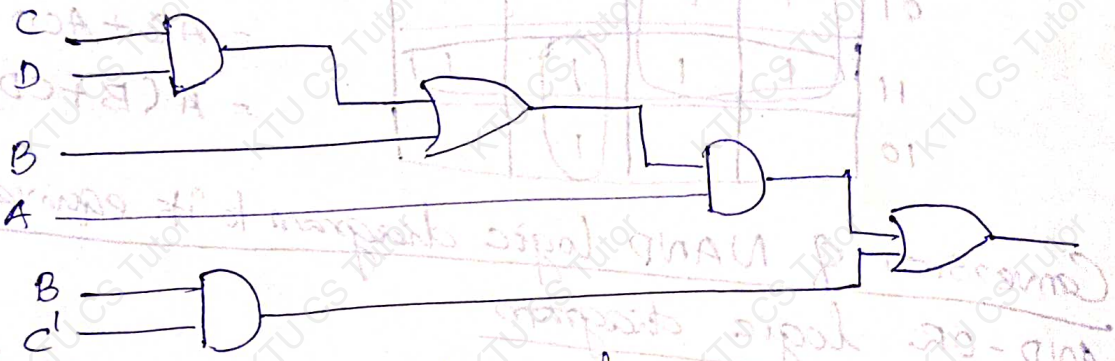
- pairs of circles along the same line can be removed, since they represent double complementation.
- One input AND or OR gate can be removed.
- One input AND or OR with a circle in the input or output is changed to an inverter circuit.



NAND logic diagram

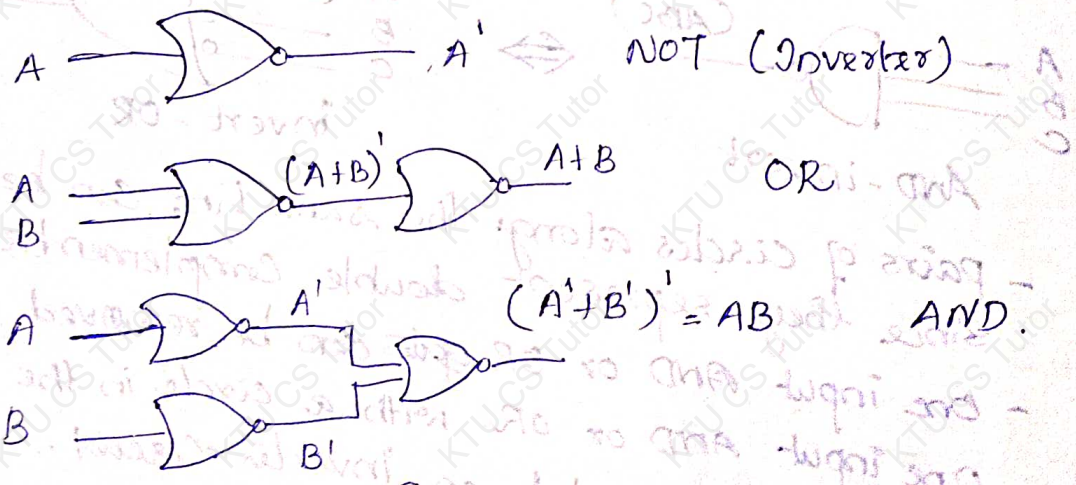


Substitution of invert-OR symbols in alternate levels.



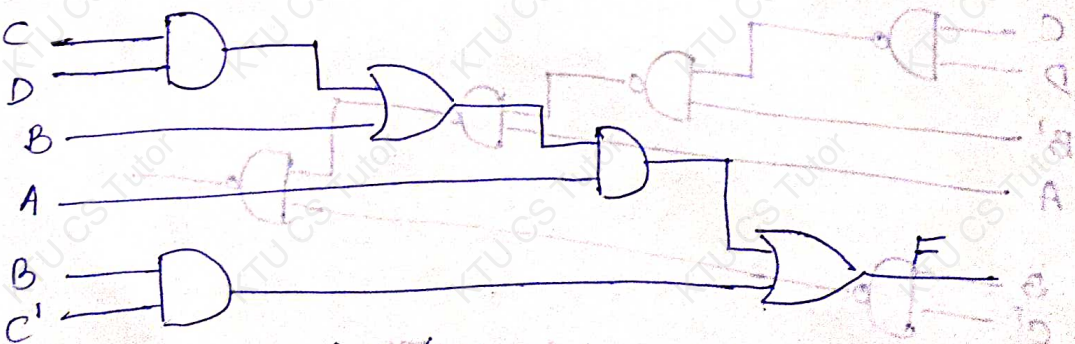
AND-OR logic diagram.

Multilevel NOR Gate

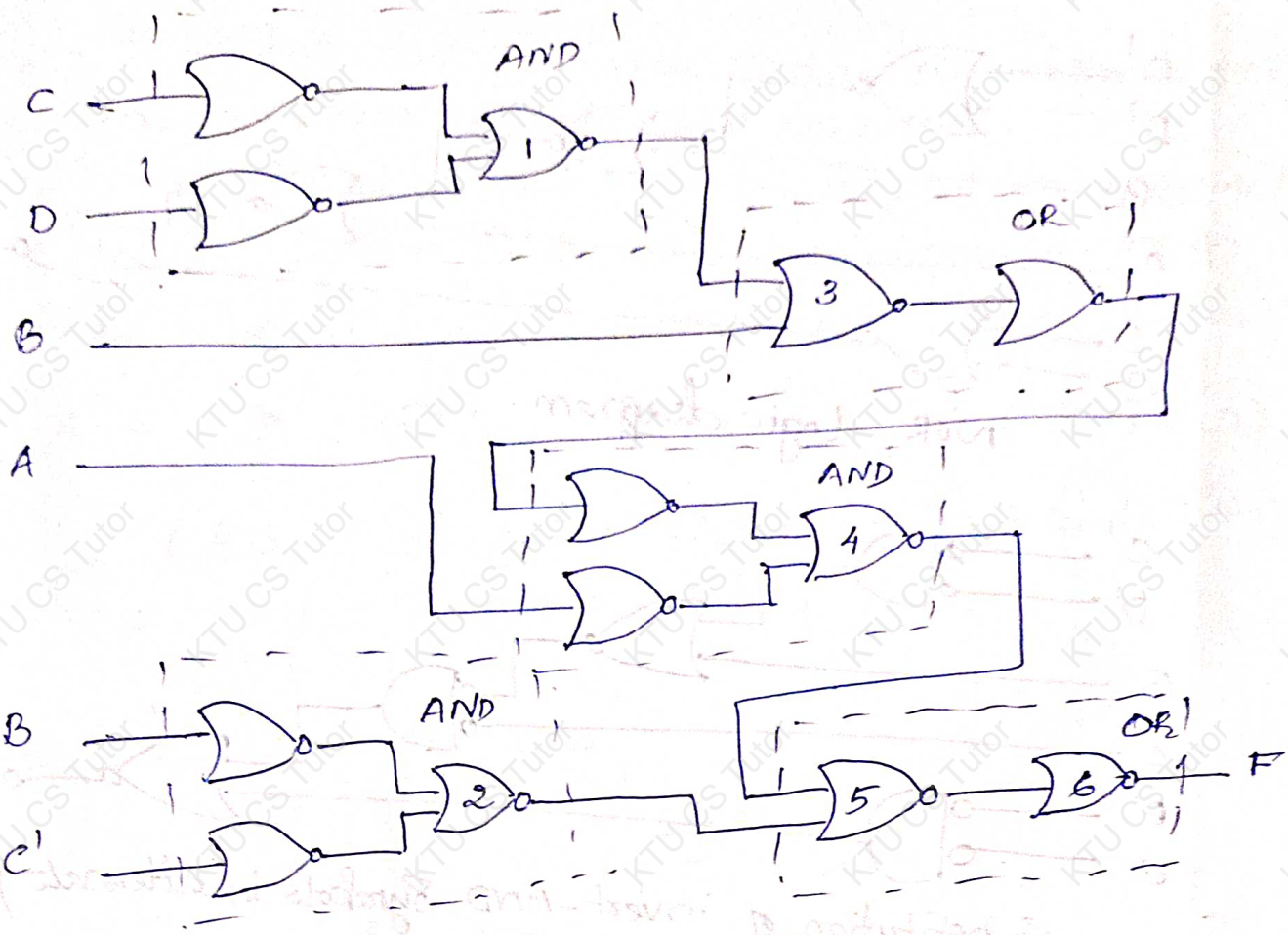


Boolean Function Implementation.

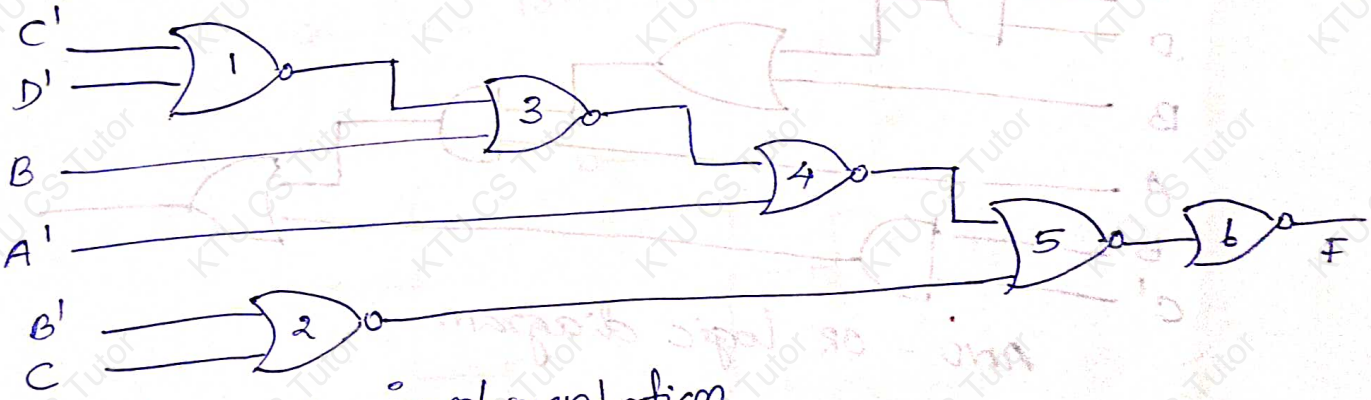
$F = A(B + CD) + BC'$



AND/OR implementation

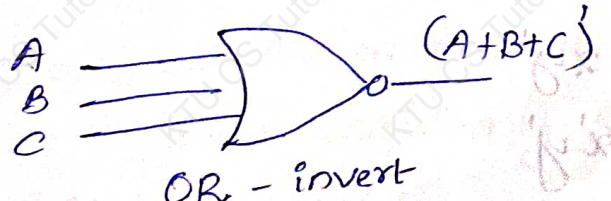


Substituting equivalent NOR.

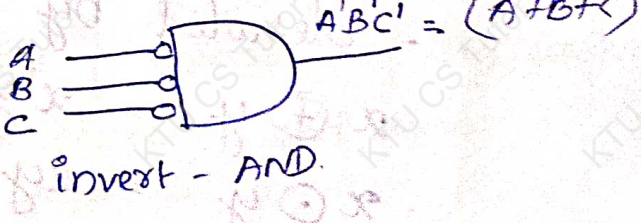


NOR implementation

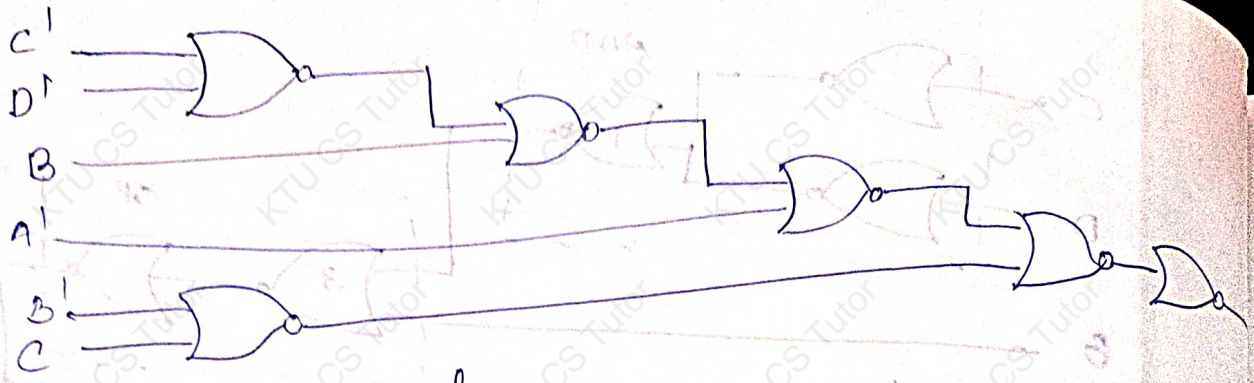
Conversion of NOR Logic diagram to AND-OR diagram



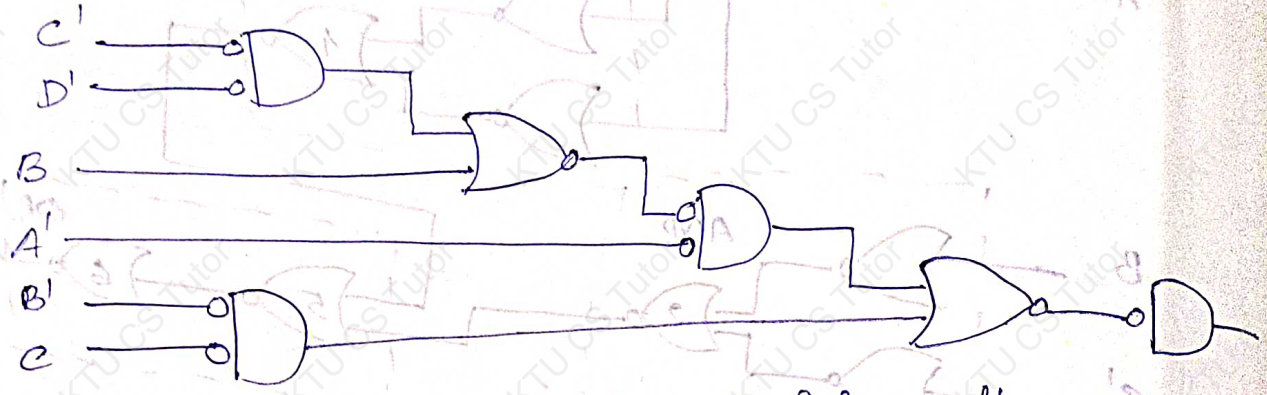
OR - invert



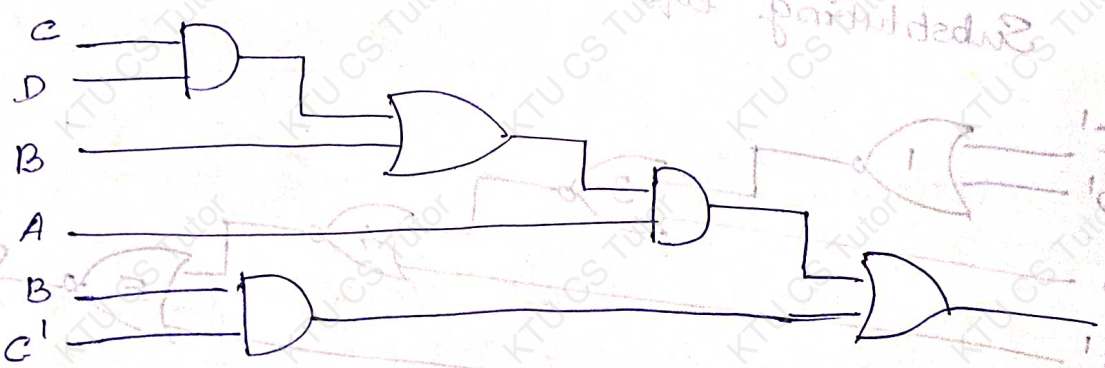
invert - AND



NOR logic diagram.



Substitution of invert-AND Symbols in alternate path.



AND - OR logic diagram

Exclusive - OR and Equivalence Functions

- denoted by $\oplus$ and $\odot$ respectively

$x \oplus y = x'y + xy'$

$x \odot y = xy + x'y'$.

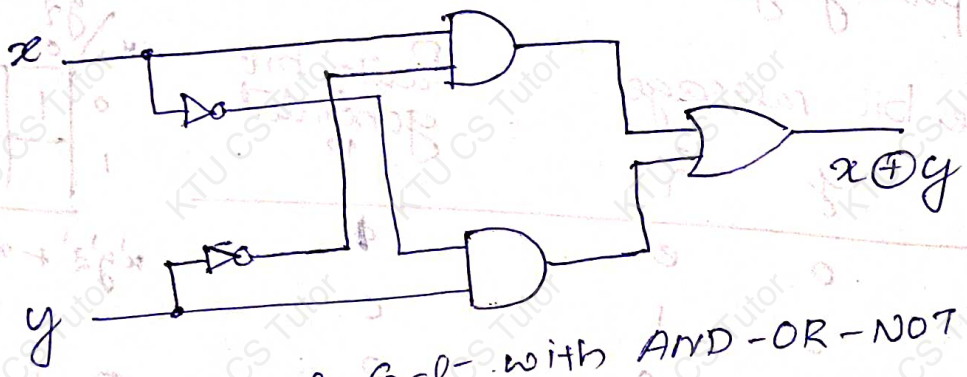
- two operations are the complements of each other
- Each is commutative and associative.

- these two functions are useful in arithmetic operations and in error detection and correction.
- an n-variable exclusive-OR expression is equal to the boolean function with $2^n/2$ minterms whose equivalent binary numbers have an odd number of 1's.

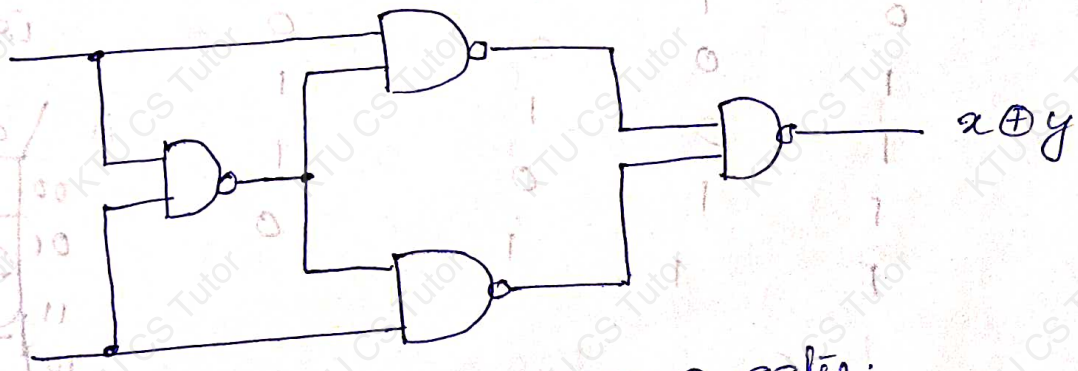
$$A \oplus B \oplus C \oplus D = (AB' + A'B) \oplus (CD' + C'D)$$

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

$$= \Sigma(1, 2, 4, 7, 8, 11, 13, 14).$$



EX-OR Gate with AND-OR-NOT gates.



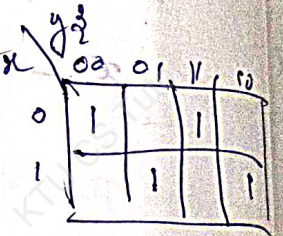
EX-OR Gate with NAND gates.

- Exclusive-OR and equivalence functions are very useful in systems requiring error detection and error correction codes.
- Parity bit is a scheme for detecting errors during transmission of binary information.
- Parity bit is an extra bit included with a binary msg

to make the number of 1's either odd or even.
- The msg including the parity bit, is transmitted and then checked at the receiving end for errors.
- An error is detected if the checked parity does not correspond to the one transmitted.
- The circuit that generates the parity bit in the transmitter is called a parity generator. The circuit that checks the parity in the receiver is called a parity checker.

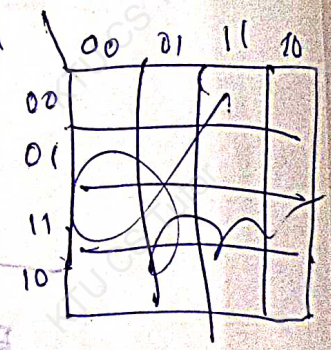
Odd parity generation for 3 bits.

Three bit message			Parity bit generated
x	y	z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

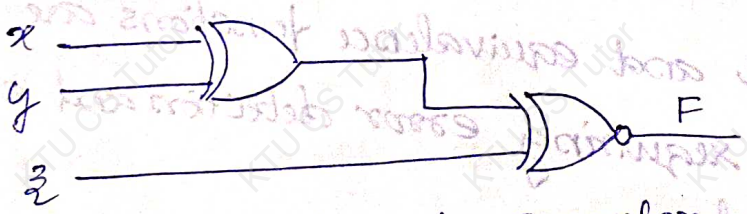


$x'y'z' + xy'z' + x'yz' + xyz'$

P = 1 when the number of 1's is even.



$P = x \oplus y \odot z$



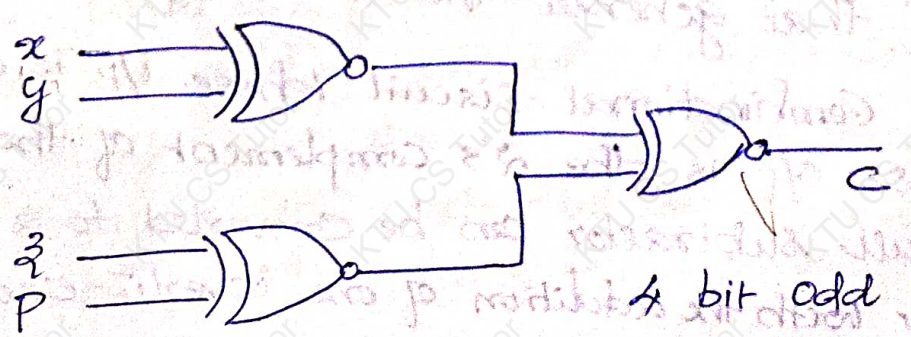
3 bit odd parity generator.

- The three bits and the parity bit are transmitted to destination and is applied to a parity checker circuit

- An error occurs during transmission if the parity of the four bits received is even, since the binary information transmitted was originally odd.
- The o/p of parity checker should be a 1 when an error occurs, i.e., when the number of 1's in the four inputs is even.

Four bits received				Parity error check
x	y	z	P	C
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

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



4 bit odd parity checker.

Qns.

(1) A combinational circuit has four inputs and one o/p. The o/p is equal to 1 when (1) all the inputs are equal to 1 or (2) none of the inputs are equal to 1 or (3) an odd number of inputs are equal to 1.

(a) Obtain the truth table
(b) Find the simplified o/p fn in sum of products.
(c) Find the simplified o/p function in product of sums.
(d) Draw the two logic diagrams.

(2) Design a combinational circuit that accepts a three bit number and generates an o/p binary number equal to the square of the i/p number.

(3) It is necessary to multiply two binary numbers, each two bits long, in order to form their product in binary. Let the two numbers be represented by a_1, a_0 and b_1, b_0 where subscript 0 denotes the least significant bit.

(a) Determine the number of o/p lines required.
(b) Find the simplified Boolean expressions for each o/p.

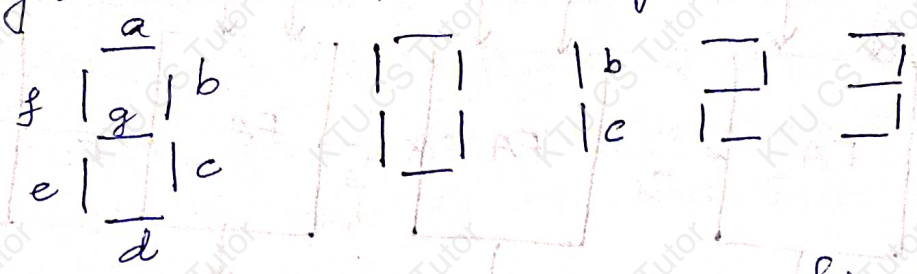
(4) Repeat the problem (3) to form the sum (instead of product) of the two binary numbers.

(5) Design a combinational circuit with four i/p lines that represent a decimal digit in BCD and four o/p lines that generate the 9's complement of the i/p digit.

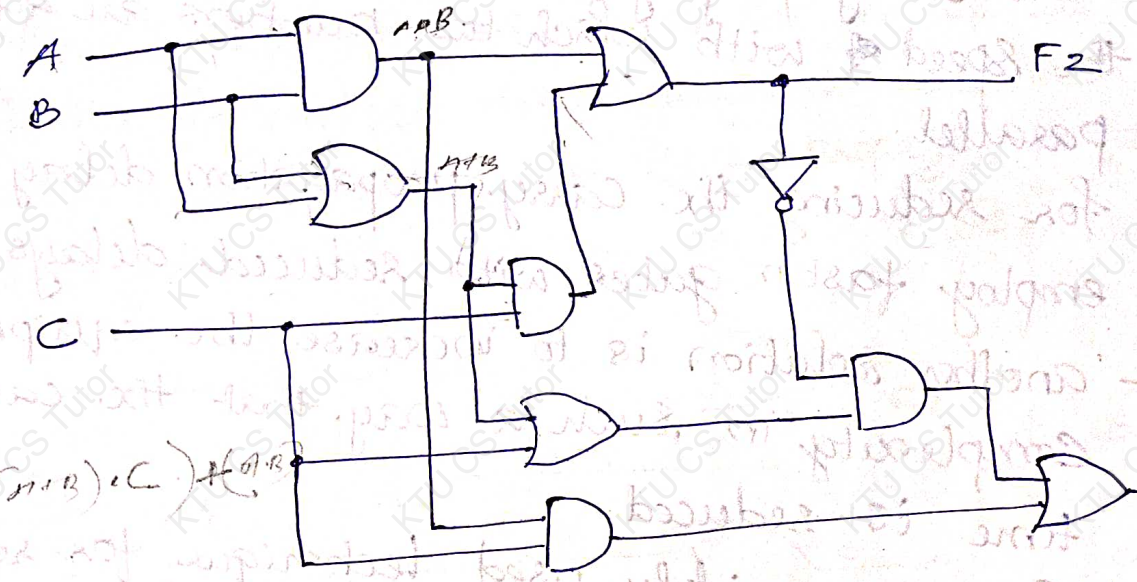
(6) Design a combinational circuit whose i/p is a 4 bit no. and whose o/p is the 2's complement of the i/p number.

(7) How a full subtractor can be converted to a full subtractor with the addition of one inverter circuit.

(8) Design the BCD to Seven Segment decoder circuit.



(9) Analyze the following two o/p combinational circuit Obtain the boolean functions for the two o/ps and explain the circuit operation and derive the truth table.



(10) Obtain the NAND logic diagram of a full adder from the boolean functions:

$$C = xy + xz + yz$$
$$S = C'(x+y+z) + xyz$$

Binary Parallel Adder

- a digital function or circuit that produces the arithmetic sum of two n-bit binary numbers in parallel.
- Full adders connected in cascade, with the o/p Carry from one full adder connected to the i/p Carry of the next full adder.
- n bit parallel adder requires n full adders.