Karnaugh Map POS Minimization

In this section, we will focus on POS expressions instead of SOP. The approaches are much the same except that with POS expressions, 0's representing the standard sum terms are placed on the karnaugh map instead of 1's.

Example: Map the following POS expression on a karnaugh map.

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$

1100 1011 0010 1111 0011

	AB \ CD	00 $\overline{C}\overline{D}$	01 $\overline{C}D$	11 CD	10 $C\overline{D}$
00	$\overline{A}\overline{B}$			**0**	**0**
01	$\overline{A}B$				
11	AB	**0**		**0**	
10	$A\overline{B}$			**0**	

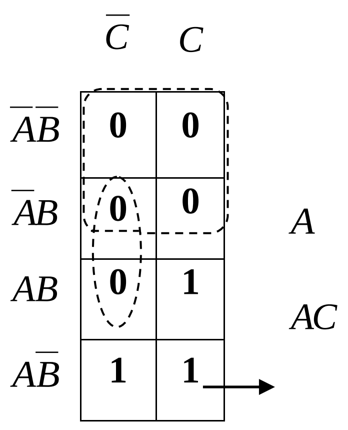
$\overline{A}+\overline{B}+\overline{C}+\overline{D}$

$\overline{A}+B+\overline{C}+\overline{D}$

Example: Use a karnaugh map to minimize the POS expression:

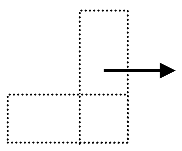
$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$

$Out=A(\overline{B}+C)$



keep in mind that this minimum POS expression is equivalent to the original standard POS expression grouping the 1's as shown yields a SOP expression that is equivalent to grouping the 0's .

$$\boxed{AC + A\overline{B} = A(\overline{B}+C)}$$



Review Questions:

1. Use a Karnaugh map to minimize the POS expression:

$$(B+C+D)(A+B+\overline{C}+D)(\overline{A}+B+C+\overline{D})(A+\overline{B}+C+D)(\overline{A}+\overline{B}+C+D)$$

2. Map the following SOP expression on a karnaugh map:

$$\overline{B}\overline{C}+A\overline{B}+AB\overline{C}+A\overline{B}C\overline{D}+\overline{A}\overline{B}\overline{C}D+A\overline{B}CD$$

3. What is the difference in mapping a POS expression and an SOP expression?
4. What is the standard sum term for a 0 in cell 1011?
5. What is the standard product term for a 1 in cell 0010?
6. Determine the minimum expression for each K-map in figure below:

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	**1**	**1**	**1**	**1**
$\overline{A}B$	**1**	**1**	**0**	**0**
AB	**0**	**0**	**0**	**1**
$A\overline{B}$	**0**	**1**	**1**	**0**

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	**1**	**0**	**1**	**1**
$\overline{A}B$	**1**	**0**	**0**	**1**
AB	**0**	**0**	**0**	**0**
$A\overline{B}$	**1**	**0**	**1**	**1**

7. Use a K-map to simplify each expression to a minimum SOP form:

(a) $\overline{A}\overline{B}\overline{C}+A\overline{B}C+\overline{A}BC+AB\overline{C}$ $\quad$ $Ans: No\ Simplification$

(b) $AC[\overline{B}+B(B+\overline{C})]$ $\quad$ $Ans: AC$

(c) $DE\overline{F}+\overline{D}E\overline{F}+\overline{D}\overline{E}\overline{F}$ $\quad$ $Ans: \overline{D}\overline{F}+E\overline{F}$

8. Reduce the function specified in the truth table in figure below to its minimum SOP form by using K-map.

A	**B**	**C**	**X**
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Ans: $\overline{B}+C$

9. Use a k-map to simplify each expression to a minimum POS form:

   (a) $(A+\overline{B}+C+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})$

   (b) $(X+\overline{Y})(W+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(W+X+Y+Z)$

10. Simplify the following equation: $f(A,B,C,D)=\sum(1,3,7,11,15)$.

11. Simplify the following equation:

    $f(A,B,C,D)=\sum(1,5,6,11,14,15)$

    $dm(2,3,7)$ *where dm is denoted for don't care.*

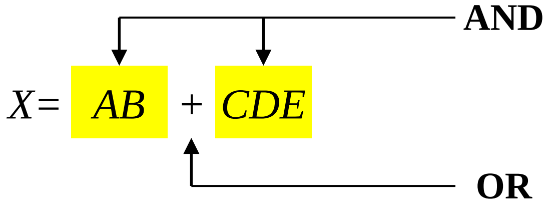
Implementation of Logic Circuit:

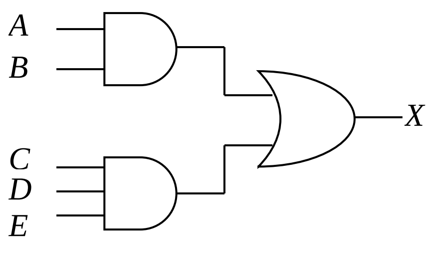
In this section, several general examples are used to illustrate how to implement a logic circuit from a Boolean expression or a truth table.

1- *From Boolean Expression to a Circuit:*

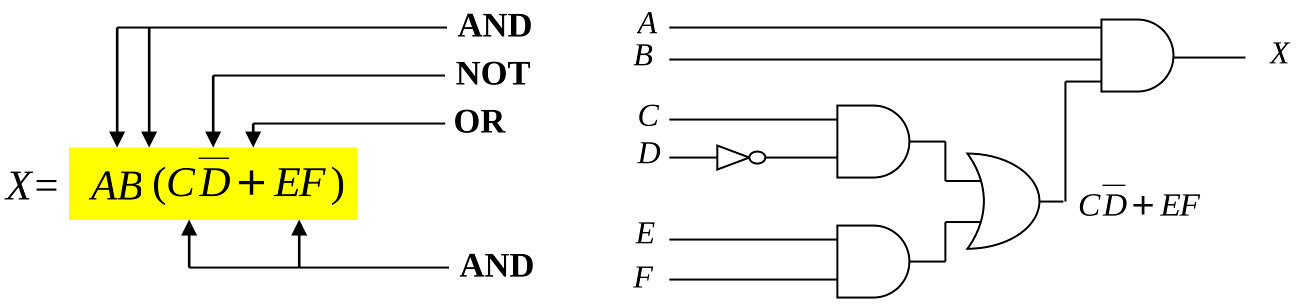
The following expression is the first in the series of examples that will be considered:- *X=AB+CDE*

This expression is composed of two terms, *AB* and *CDE*, with a total of five variables. The first term is formed by ANDing *A* with *B*, and the second term is formed by ANDing *C*,*D*, and *E*. The two terms are then ORed to form the output *X*. These operations are indicated in the structure of the expression as follows:

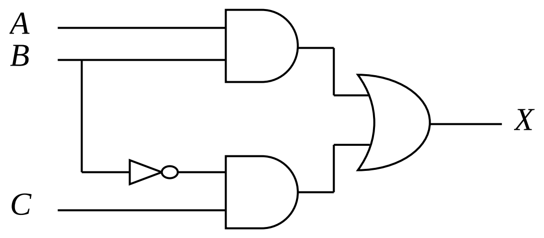




Example: Implement the following expression: $X = AB(C\overline{D} + EF)$



Example: Draw the circuit diagram that implements the expression: $X = AB + \overline{B}C$



2- *From Truth Table to a Circuit:*

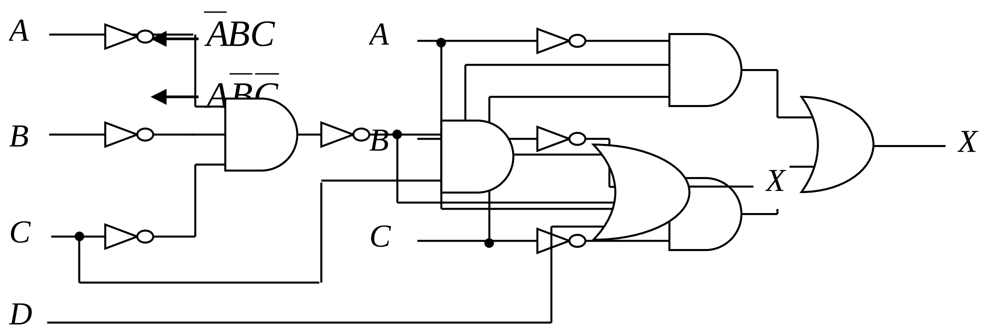
Beginning with a truth table, we will write the SOP expression from the truth table and then implement the logic circuit. The table below specifies a logic function. The expression is obtained from a truth table by ORing the product terms for all occurrences of *X*=1. The expression is: $X = \overline{A}BC + A\overline{B}\overline{C}$

Input			Output
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	**1**

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

The logic gates required to implement this expression are as follows: Three inverters to form the $\bar{A}, \bar{B}$ *and* $\bar{C}$ variables; two 3-input AND gates to form the term $\bar{A}BC$ and $A\bar{B}\bar{C}$; and one 2-input OR gate to form the final output.

Example: Reduce the combinationa am form:



The expression for the output of the circuit is: $X = (\overline{\bar{A}\bar{B}\bar{C}})C + \overline{\bar{A}\bar{B}\bar{C}} + D$

Applying DeMorgan's theorem and Boolean algebra:

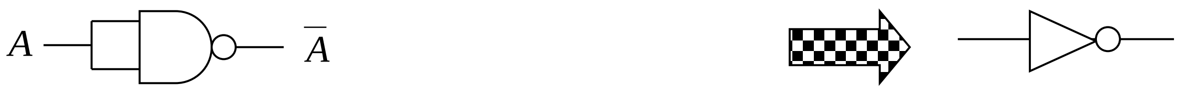
$$X = (\bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}})C + \bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}} + D$$
$$= AC + BC + CC + A + B + C + D = AC + BC + C + A + B + C + D$$
$$= C(A + B + 1 + 1) + A + B + D = A + B + C + D$$

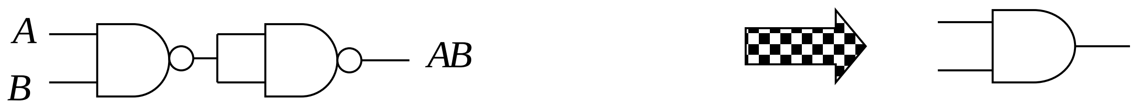
Hence, the simplified circuit is a **4-input OR gate**.

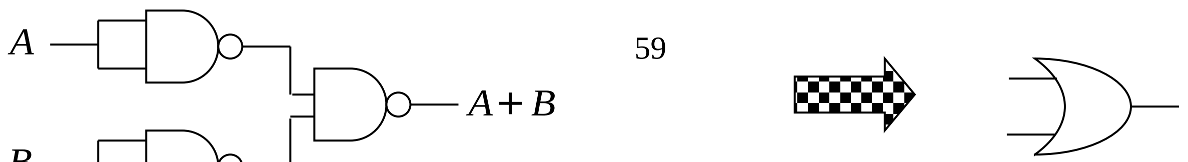
Universal Property of NAND and NOR Gates:

Up to this point, combinational circuits implemented with AND gates, OR gates and inverters have been studied. In this section, the universal property of the NAND gate and the NOR gate is introduced. the universality of the NAND gate means that it can be used as an inverter and that combinations of NAND gates can be used to implement the AND, OR and NOR operations. Similarly, the NOR gate can be used to implement the inverter, AND, OR and NAND operations.

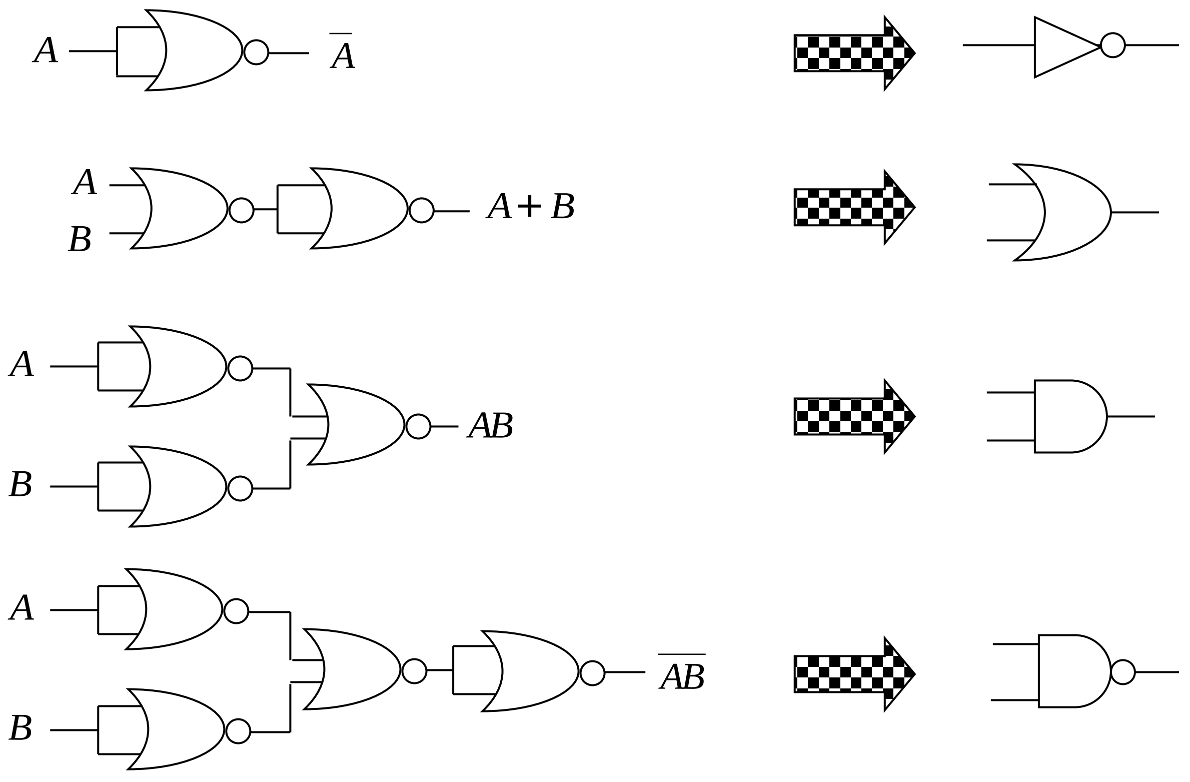
1- ***Using NAND Gate:***



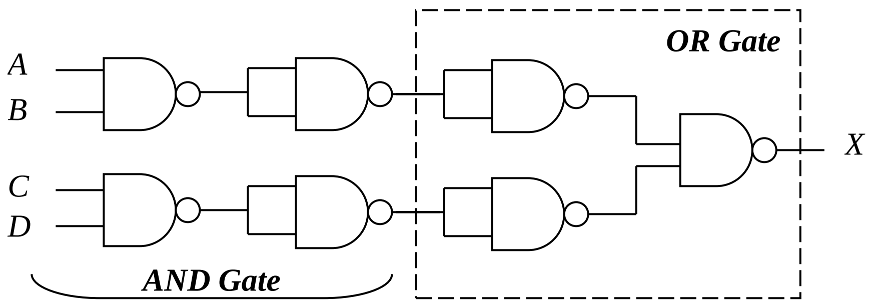




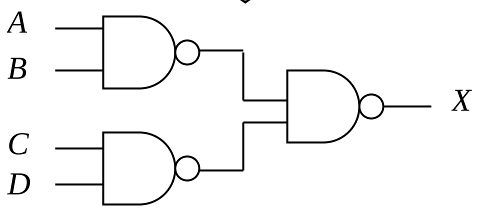
2- ***Using NOR Gate:***



Example: Design using NAND gates the following expression: $X = AB + CD$

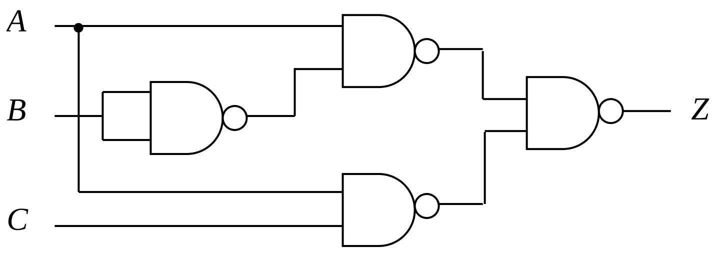


After eliminating double inversion



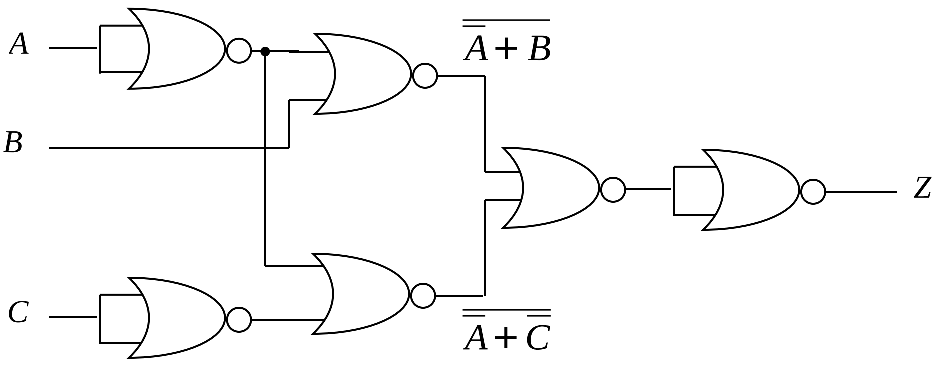
Example: Design $Z = AC + A\overline{B}$ using NAND gates only:

$Z = \overline{\overline{AC + A\overline{B}}} = \overline{\overline{AC}.\overline{A\overline{B}}}$ (Conversion to Product form)

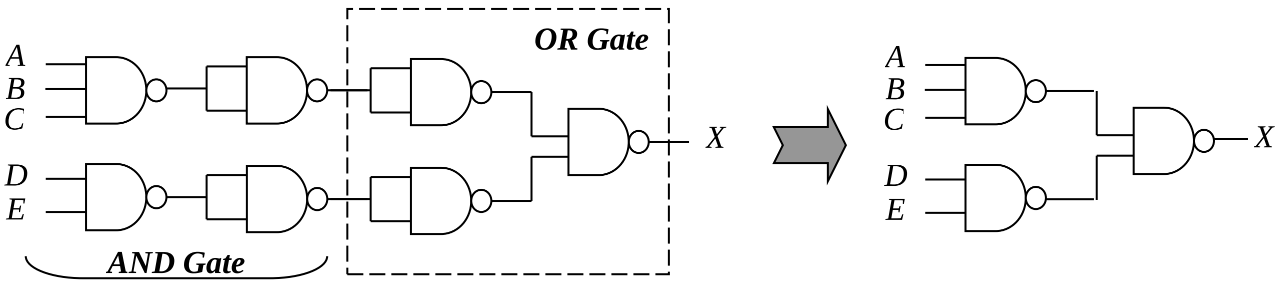


Example: Design $Z = AC + A\overline{B}$ using NOR gates only:

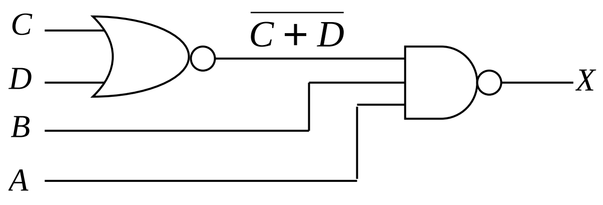
$Z = \overline{\overline{AC}} + \overline{\overline{A\overline{B}}} = \overline{(\overline{A} + \overline{C})} + \overline{(\overline{A} + B)}$ (Conversion to Sum form)


A
B
C
Z
$\overline{\overline{A} + B}$
$\overline{\overline{A} + \overline{C}}$

<u>Example:</u> Implement the following expression with NAND gates only: $ABC + DE$



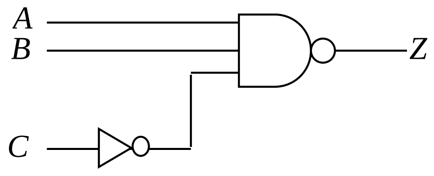
<u>Example:</u> Implement the logic circuit that has the expression $X = \overline{AB.(\overline{C+D})}$ using only one NAND and NOR gate.



<u>Example:</u> Implement a circuit having the output expression $Z = \overline{A} + \overline{B} + C$ using only one NAND and an inverter gate.

$Z = \overline{A} + \overline{B} + C$ *can be written as*:

$Z = \overline{\overline{\overline{A}}.\overline{\overline{B}}.\overline{C}} = \overline{A.B.\overline{C}}$



<u>Review Questions:</u>

8. Implement the following logic expressions:

(a) $X = ABC + AB + AC$ (b) $X = AB(C + DE)$

(c) $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C)$ (d) $X = \overline{ABC} + B(EF + \overline{G})$

9. Use NAND gates to implement::

(a) $X = \overline{A} + B$ (b) $X = A\overline{B}$

10. Implement the expression $X = \overline{(\overline{A} + \overline{B} + \overline{C})DE}$ by using NAND logic.

11. Implement the expression $X = \overline{\overline{A}\,\overline{B}\,\overline{C} + (D + E)}$ with NOR logic.

12. Design a logic circuit to implement the operation specified in the truth table indicated by Table (a):
13. Minimize the combinational logic circuit shown in Fig. (a):
14. Implement the logic circuit in Fig. (b) using only NAND gates. Repeat the design using only NOR gates.

Table (a)

Input			**Output**
A	***B***	***C***	***X***
0	0	0	0
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

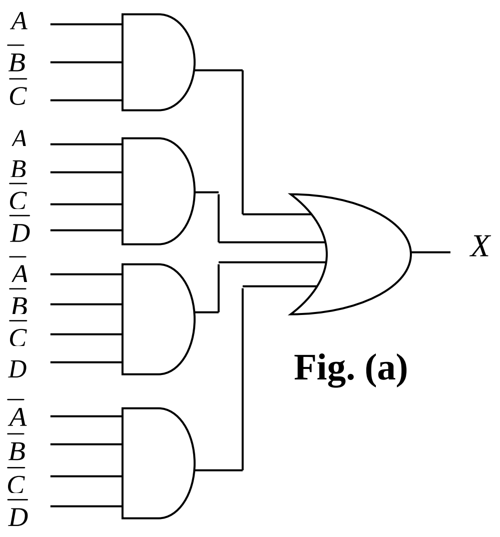


Fig. (a)

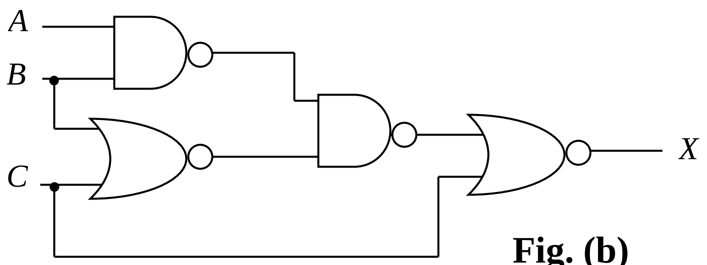


Fig. (b)

15. Show how the following expression can be implemented as stated using only NAND gates:

(a) $X = ABC$ (b) $X = \overline{ABC}$

(c) $X = A + B$ (d) $X = A + B + \overline{C}$

(e) $X = \overline{AB} + \overline{CD}$ (f) $X = (A + B)(C + D)$

(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

16. Implement each function by using only NAND gates:

(a) $X = A\overline{B} + AB$ (b) $X = A[BC(A + B + C + D)]$

17. Minimize the gates required to implement the function in SOP form:

    (a) $X = A\overline{B} + AB$ (b) $X = \overline{ABC} + B(EF + \overline{G})$

    (c) $X = A[BC(A + B + C + D)]$ (d) $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C)$

18. Design a minimal circuit to detect 31-day months, coded in 8-4-2-1 binary.

19. Obtain a minimal SOP for the function below then implement using NAND gates only: $f(A,B,C,D) = \sum(0,7,8,10,12) + d(2,6,11)$.

20. Using a karnaugh map, convert the following standard POS expression into a minimum POS expression, a standard SOP expression and a minimum SOP expression:

    $f(A,B,C,D) = \Pi(1,2,3,4,9,12)$ $\;\; where\, \Pi\, is\ denoted\ for\, POS$

<u>Basic Adders:</u>

Adders are important not only in computers, but in many types of digital systems in which numerical data are processed. An understanding of the basic adder operation is fundamental to the study of digital systems. In this section, the half-adder and the full-adder are introduced.

3- <u>The Half-Adder (H.A):</u>

Recall the basic rules for binary addition as stated in the previous lectures:

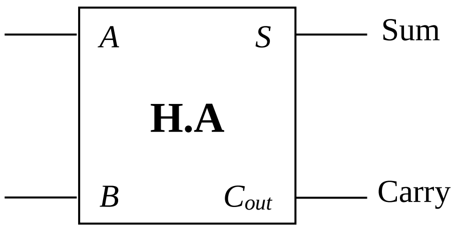
$0+0=0$

$0+1=1$

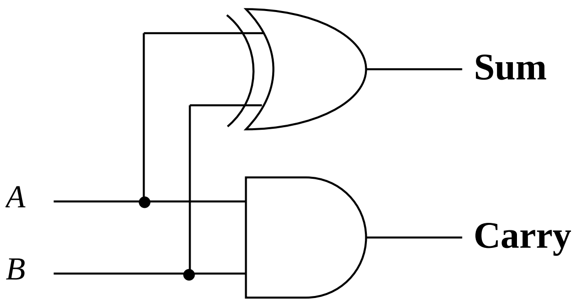
$1+0=1$

$1+1=10$

These operations are performed by a logic circuit called a half-adder. The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs, a **Sum** bit and a **Carry** bit.



A	B	C_{out}	S
0	**0**	**0**	**0**
0	**1**	**0**	**1**
1	**0**	**0**	**1**
1	**1**	**1**	**0**

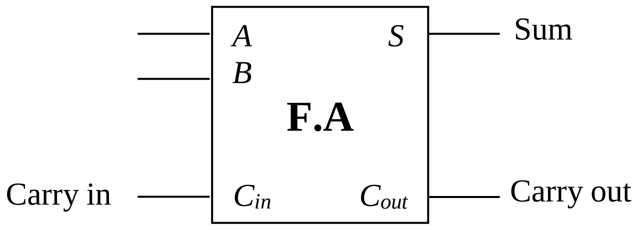


$$\boxed{\begin{aligned} S &= A \oplus B \\ C_{out} &= AB \end{aligned}}$$

4- ***The Full-Adder(F.A):***

The second basic category of adder is the Full-adder. The full-adder accepts three inputs including an input carry and generates a **Sum** output and output **carry.**

A	B	C_{in}	C_{out}	S
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**0**

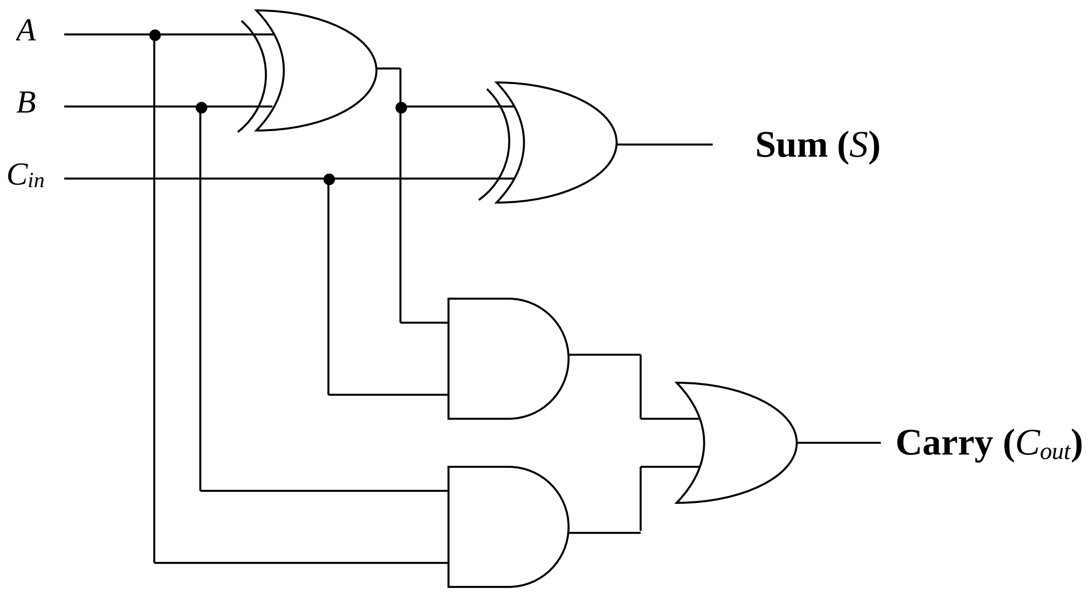


1	**0**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**
1	**1**	**0**	**1**	**0**
1	**1**	**1**	**1**	**1**

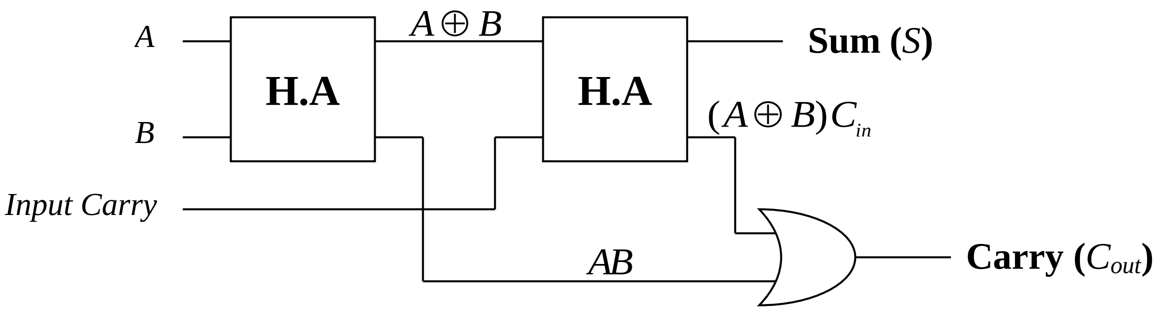
From the truth table:-

$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\overline{C_{in}} + ABC_{in}$$
$$= (\bar{A}B + A\bar{B})C_{in} + AB$$
$$C_{out} = (A \oplus B)C_{in} + AB \qquad \text{...........}Eq.(1)$$

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\overline{C_{in}} + A\bar{B}\overline{C_{in}} + ABC_{in}$$
$$= \bar{A}\bar{B}C_{in} + (\bar{A}B + A\bar{B})\overline{C_{in}} + ABC_{in} \qquad \text{.........}Eq.(2)$$
$$But\ \ \bar{A}\bar{B}C_{in} + ABC_{in} = \overline{\overline{(\bar{A}\bar{B} + AB)}}C_{in}$$
$$= \overline{(A+B)(\bar{A}+\bar{B})}C_{in}$$
$$= \overline{(A\bar{B} + B\bar{A})}C_{in} \qquad \text{.........}Eq.(3)$$
$$Substituting\ Eq.3\ in\ Eq.2$$
$$S = (\bar{A}B + A\bar{B})\overline{C_{in}} + \overline{(\bar{A}B + A\bar{B})}C_{in} = A \oplus B \oplus C_{in}$$



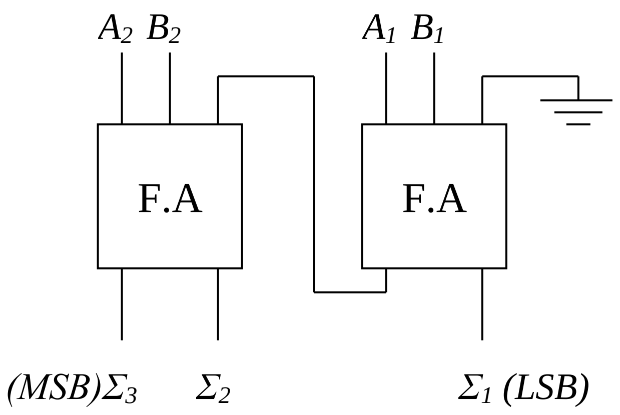
Example: Arrange two half-adders to form full-adder.



Parallel Binary Adders:

Adders that are available in integrated circuits form are parallel binary adders. As you saw in the previous section, a single full-adder is capable of adding two 1-bit numbers and an input carry. To add binary numbers with more than one bit, additional full-adders must be employed. To implement the addition of binary numbers, a full-adder is required for each bit in the numbers. So, for 2-bit numbers, two adders are needed; for 4-bit numbers, four adders are used; and so on.

$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline \Sigma_3\,\Sigma_2\,\Sigma_1 \end{array}$$


A_2 B_2
A_1 B_1
F.A
F.A
(MSB)Σ_3
Σ_2
Σ_1 (LSB)