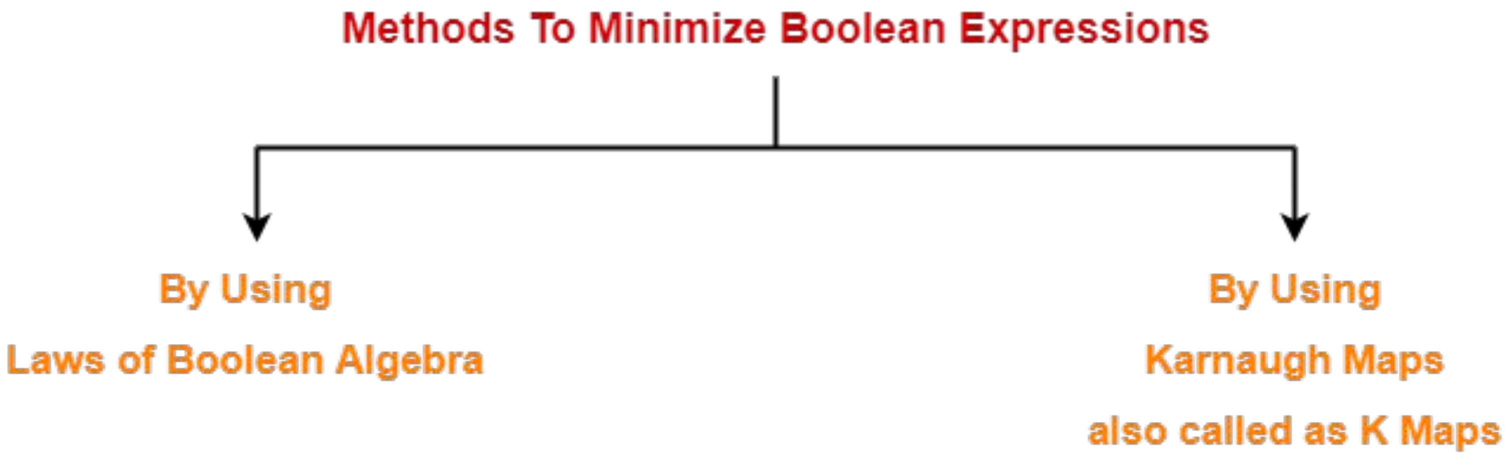
Karnaugh Map / K-map method

K-map is a technique to simplify Boolean expression.

It is a graphical chart which contains boxes called cells.



K-map method

K-map employs the use of two-dimensional tables to simplify the expressions, whose size increases at a very high rate with the increase in the number of variables.

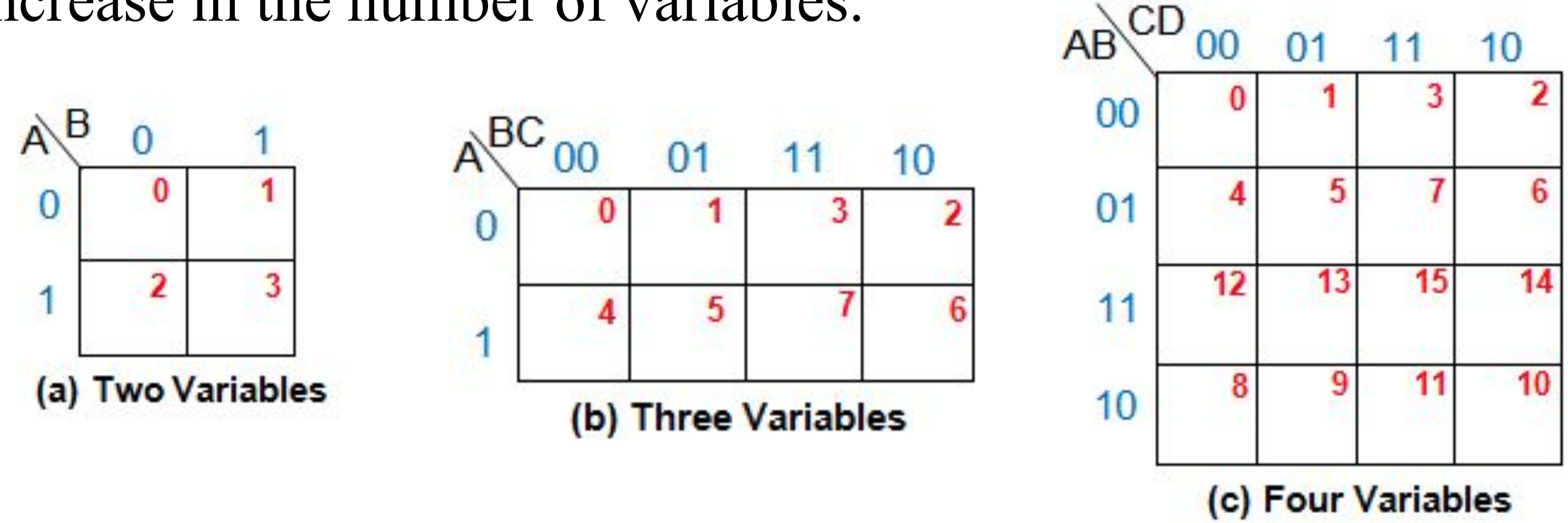


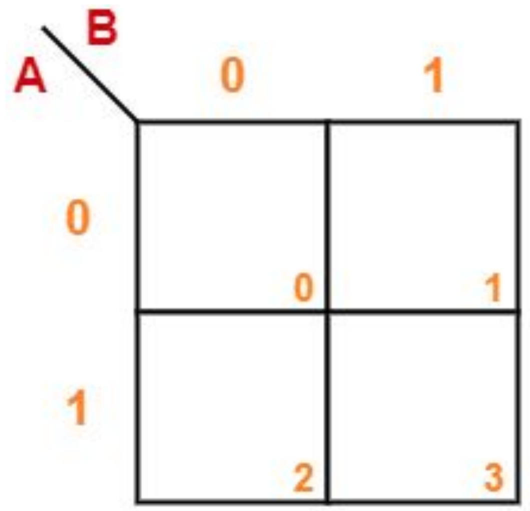
Figure:- Karnaugh Maps for (a) Two Variables (b) Three Variables (c) Four Variables

K-map method

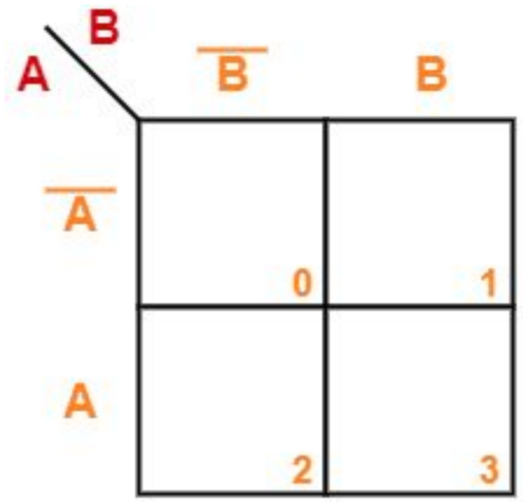
From the figure, it is evident that the number of cells in the K-map is a function of number of inputs. In general, if there are n inputs, then the corresponding K-map has to be of 2^n

cells. For example, if the number of input variables is 2, then we have to consider a K-map with 4 $(=2^2)$ cells, while if there are 3 input variables, then we require a 8 $(=2^3)$ cell K-map, and similarly for 4 inputs one gets 16 $(=2^4)$ cell K-map and so on.

K-map method : 2-variable



OR



K-map method : 3-variable

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

OR

A \ BC	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	0	1	3	2
A	4	5	7	6

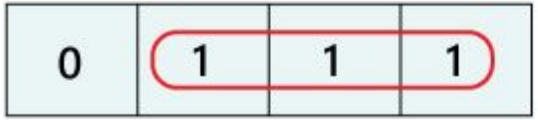
K-map method : 4-variable

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

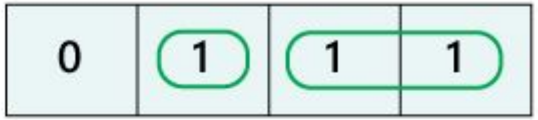
OR

AB \ CD	$\bar{C}\,\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\,\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

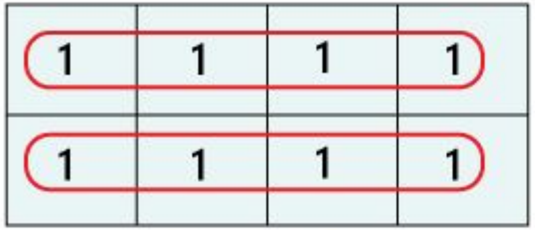
K-map method



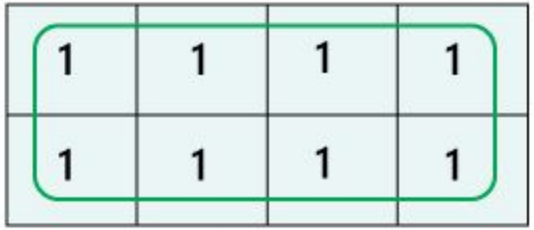
Incorrect



Correct

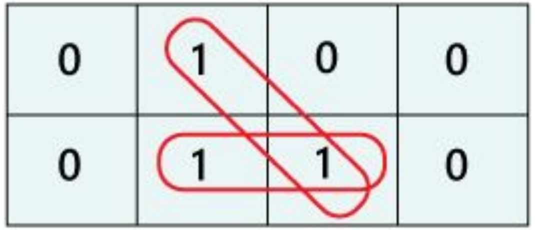


Incorrect

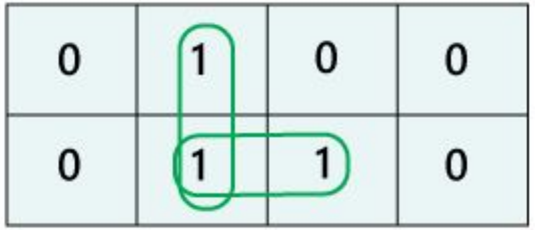


Correct

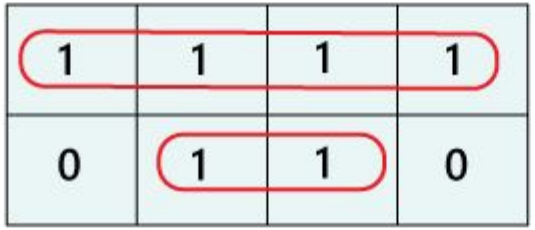
K-map method



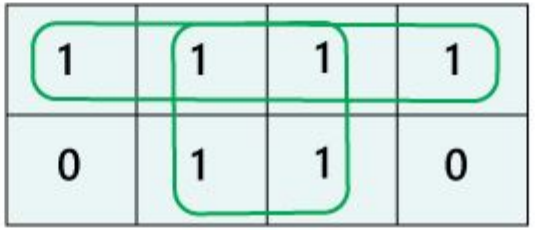
Incorrect



Correct



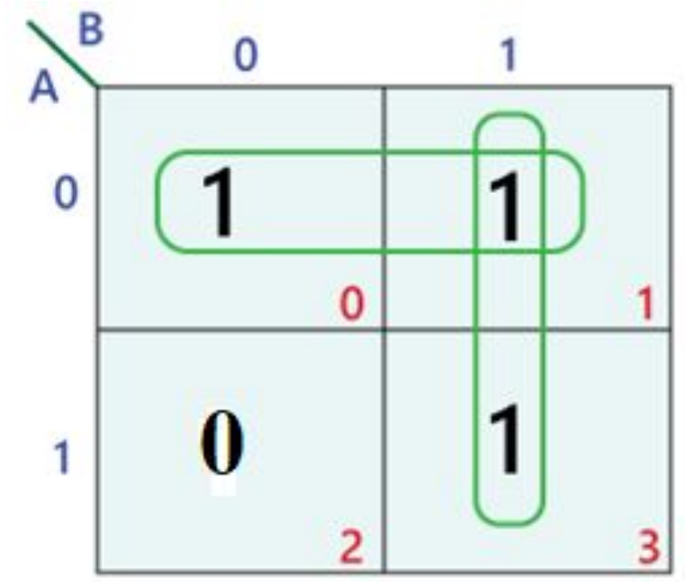
Incorrect



Correct

K-map method: SOP form

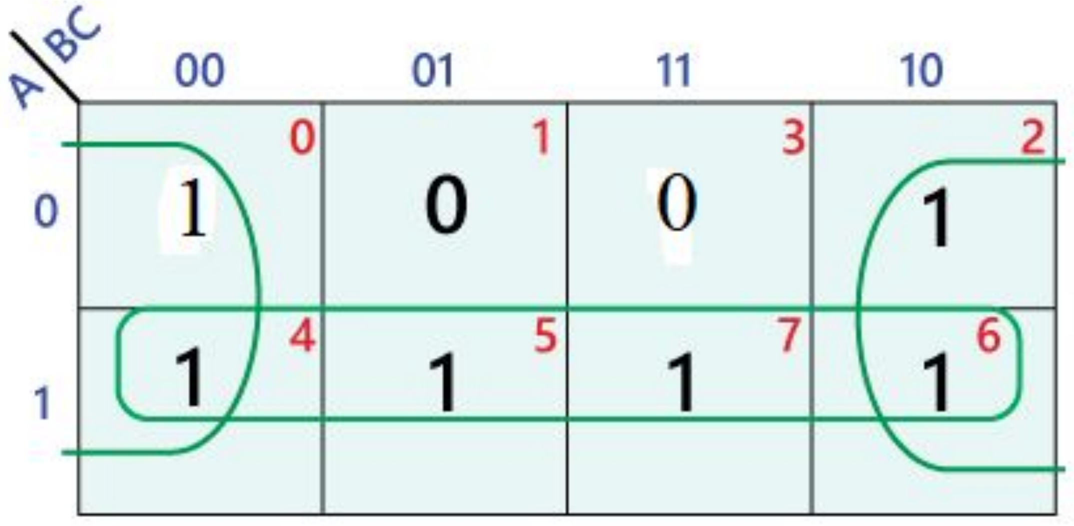
Plot Boolean expression Y=A'B' + A'B+AB on K-map



Simplified expression: Y=A'+B

K-map method for SOP

Plot Boolean expression Y=A'B'C'+A' BC'+AB' C'+AB' C+ABC'+ABC on K-map

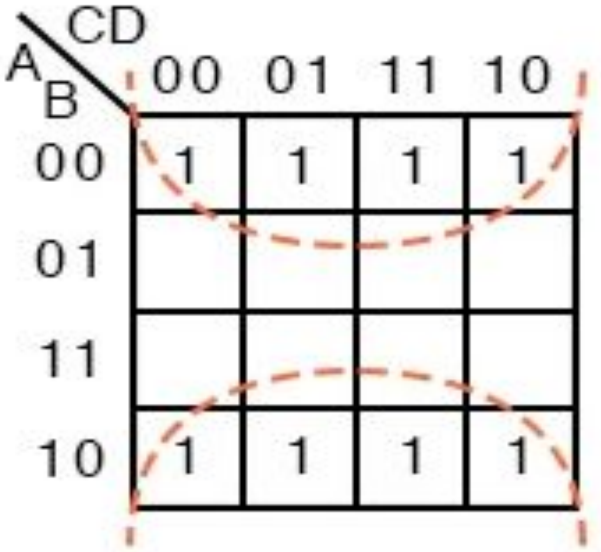


Simplified expression: Y=A+C'

K-map method: Example

Plot the following Boolean expression on K-map

$$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D}$$
$$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD + A\bar{B}C\bar{D}$$

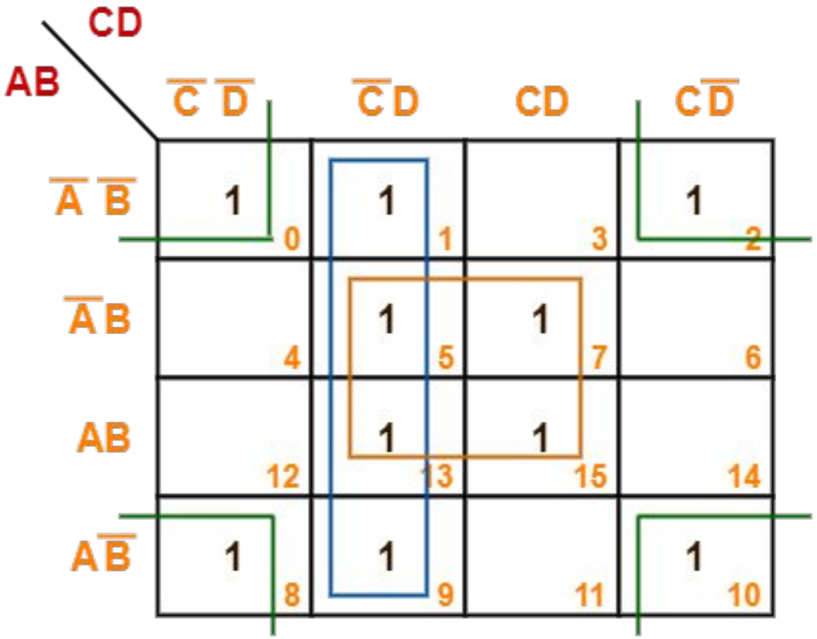


Simplified expression: Y=B'

K-map method: Example

Minimize the following boolean function-

$$F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 7, 8, 9, 10, 13, 15)$$



	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1 (0)	1 (1)	(3)	1 (2)
$\bar{A}B$	(4)	1 (5)	1 (7)	(6)
AB	(12)	1 (13)	1 (15)	(14)
$A\bar{B}$	1 (8)	1 (9)	(11)	1 (10)

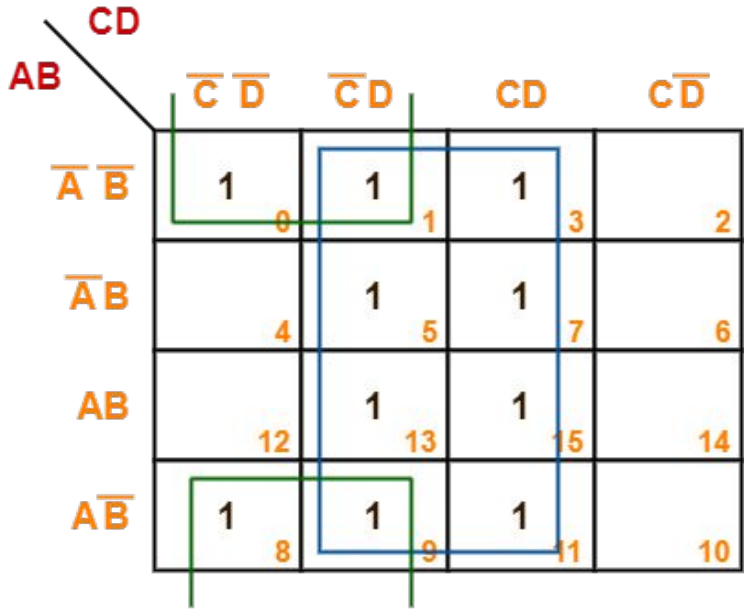
Thus, minimized boolean expression is-

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

K-map method - Examples

Minimize the following boolean function-

$$F(A, B, C, D) = \Sigma m(0, 1, 3, 5, 7, 8, 9, 11, 13, 15)$$



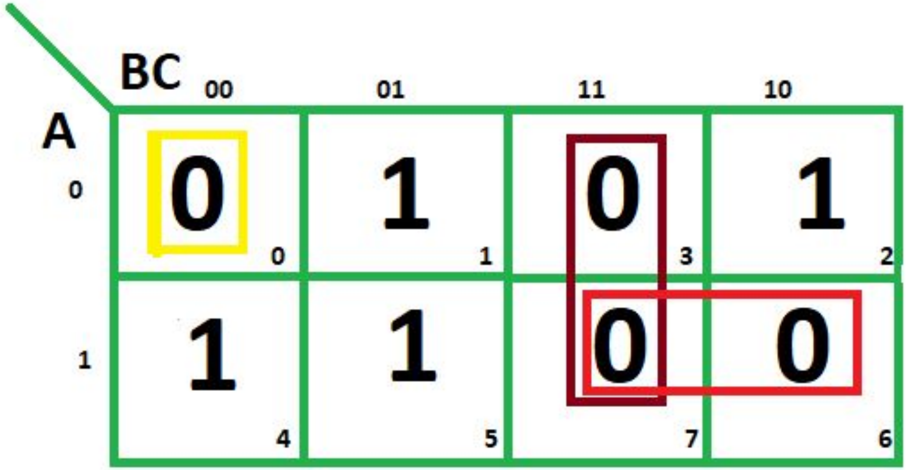
Thus,minimized boolean expression is

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

K-map method: POS form

Minimize the following boolean function-

$$F(A,B,C)=\pi(0,3,6,7)$$

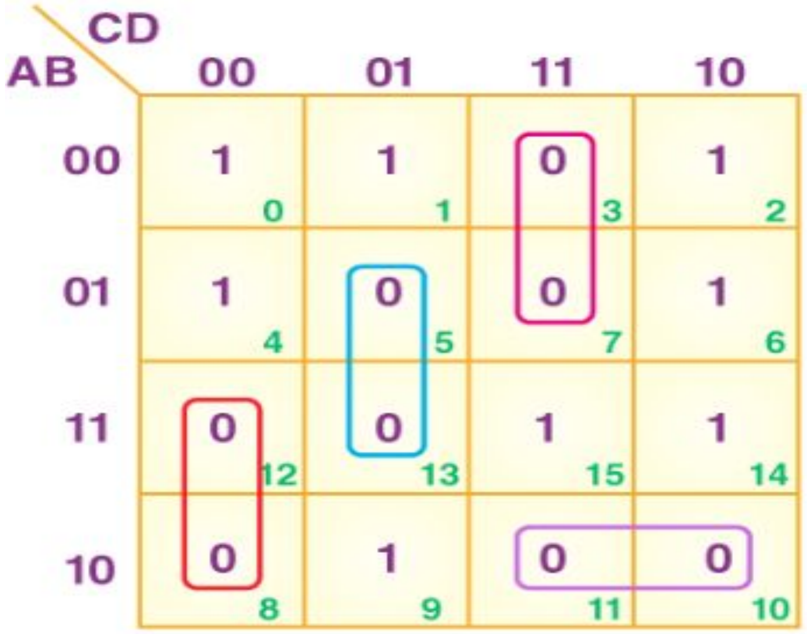


Final expression is (A' + B’) (B’ + C’) (A + B + C)

K-map method: POS form

Minimize the following boolean function-

F(A,B,C)= π (3, 5, 7, 8, 10, 11, 12, 13)



AB \ CD	00	01	11	10
00	1 0	1 1	0 3	1 2
01	1 4	0 5	0 7	1 6
11	0 12	0 13	1 15	1 14
10	0 8	1 9	0 11	0 10

Final expression is:

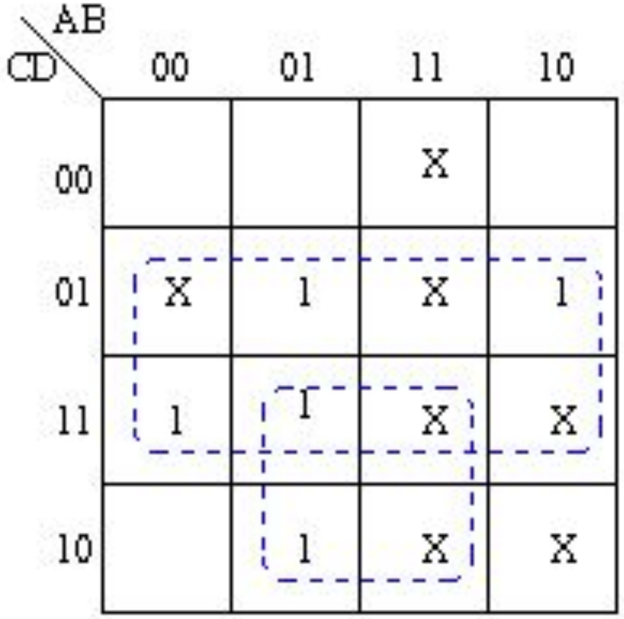
(A+C’+D’) (B’+C+D’)
(A’+C+D) (A’+B+C’)

Practice example

1. Minimize the following 4 variable POS function using K map: F(A,B,C,D,E)= πM (0,1,3,5,6,8,9,11,15)
2. Minimize the following 4 variable SOP function using K map: F(A,B,C,D,E)= $\sum m$(0,5,6,8,9,10,11,16)
3. Minimize the boolean function Y = (A'+B'+C+D) (A+B'+C+D) (A+B+C+D') (A+B+C'+D') (A'+B+C+D') (A+B+C'+D).
4. Minimize the boolean expression Y = ABC'D + ABC'D' + ABCD + A'BCD + ABCD' + A'BCD'.

Don’t care condition

- The "Don't care" condition says that we can use the blank cells of a K-map to make a group of the variables.
- To make a group of cells, we can use the "don't care" cells as either 0 or 1, and if required, we can also ignore that cell.
- We mainly use the "don't care" cell to make a large group of cells.

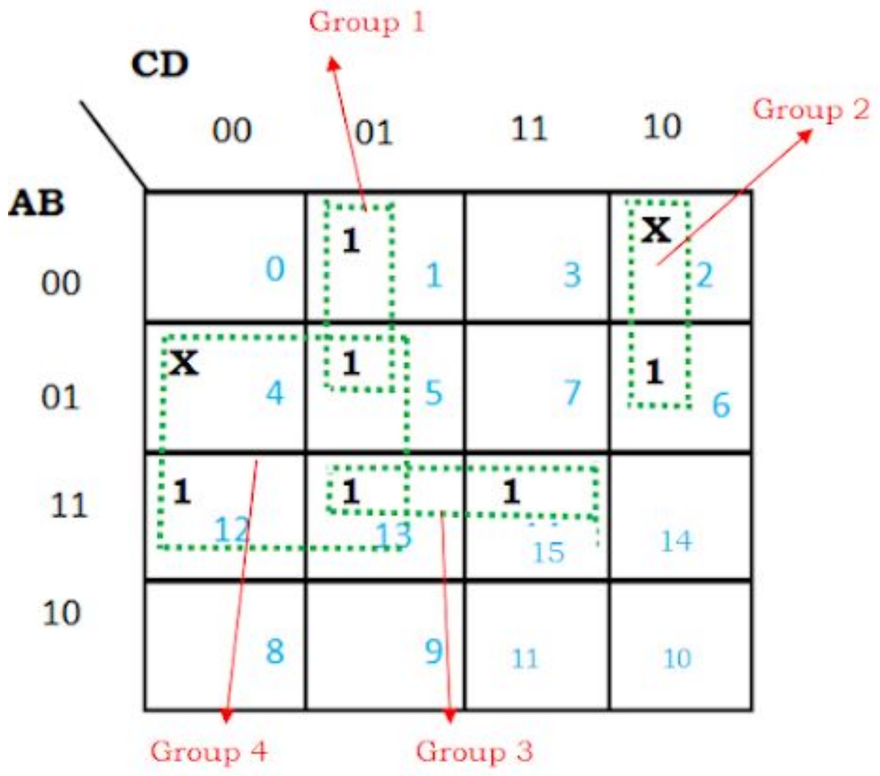


Don’t care condition-Example

Minimize the SOP expression using K map(with don’t care conditions):F(A,B,C,D)=∑m(1,5, 6,12,13,15)+ d(2,4)

=> So the minimized expression from K map is

F(A,B,C,D)= A'C'D+ A'CD'+ ABD+ BC'



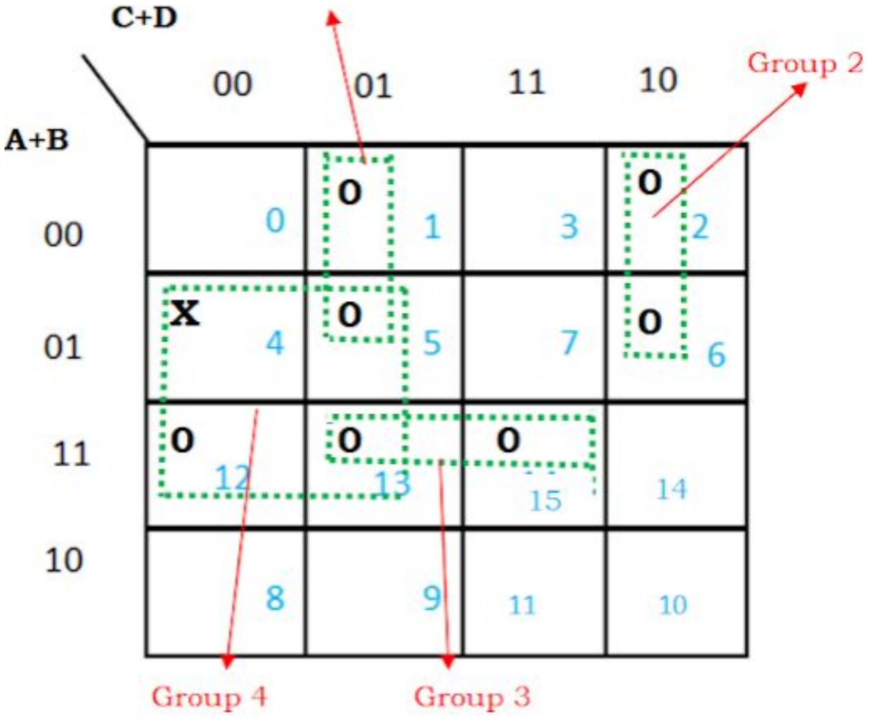
Don’t care condition-Example

Minimize the POS expression using K map(with don’t care conditions):

$f(A,B,C,D)=\pi M(1,5,6,12,13,15)+d(2,4)$

=> So the minimized expression from K map is

$f(A,B,C,D)=(A+C+D').(A+C'+D).(A'+B'+D').(B'+C)$



Don’t care condition

Practice Example1: Minimize F=m(1,5,6,12,13,14)+**d(4)** in SOP minimal form

Practice Example2: Minimize F(A, B, C, D) = M(6, 7, 8, 9) + d(12, 13, 14, 15) in POS form.

Subtractor

Subtractor circuits take two binary numbers as input and subtract one binary number input from the other binary number input. Similar to adders, it gives out two outputs, difference and borrow (carry-in the case of Adder).

There are two types of subtractors.

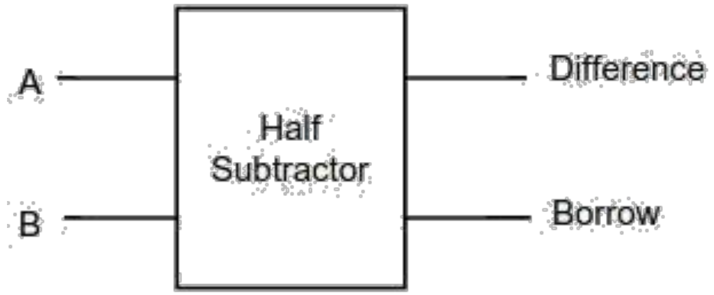
1. Half Subtractor
2. Full Subtractor

Half Subtractor

The half subtractor is also a building block for subtracting two binary numbers.

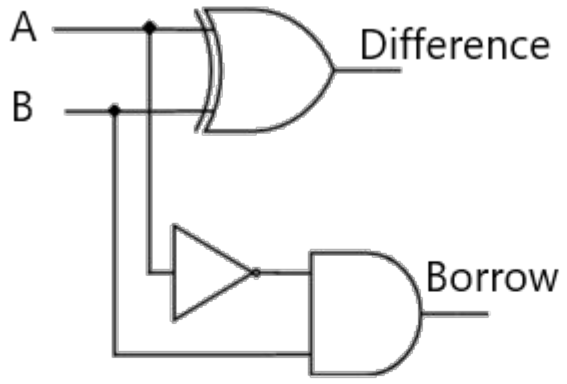
It has two inputs and two outputs.

This circuit is used to subtract two single bit binary numbers A and B. The 'diff' and 'borrow' are two output states of the half subtractor.



Half Subtractor

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

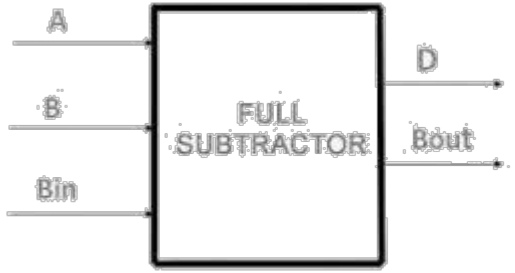


$$\text{Difference} = A \oplus B$$

$$\text{Borrow} = A' B$$

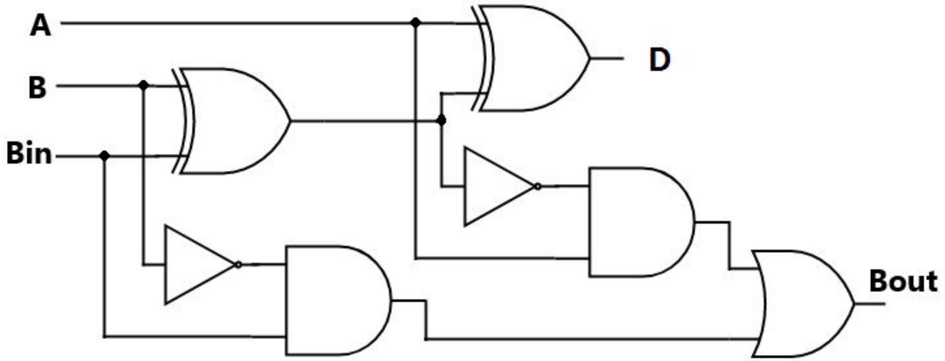
Full Subtractor

- A full subtractor is a combinational circuit that performs subtraction involving three bits, namely A (minuend), B (subtrahend), and Bin (borrow-in) .
- It accepts three inputs: A (minuend), B (subtrahend) and a Bin (borrow bit) and it produces two outputs: D (difference) and Bout (borrow out).



Full Subtractor

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



$$D = A \oplus B \oplus Bin$$

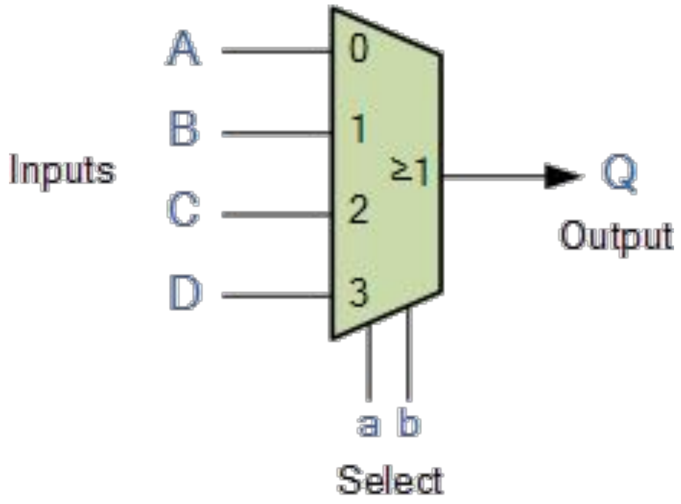
$$Bout = A'\,Bin + A'\,B + B\,Bin$$

Multiplexer

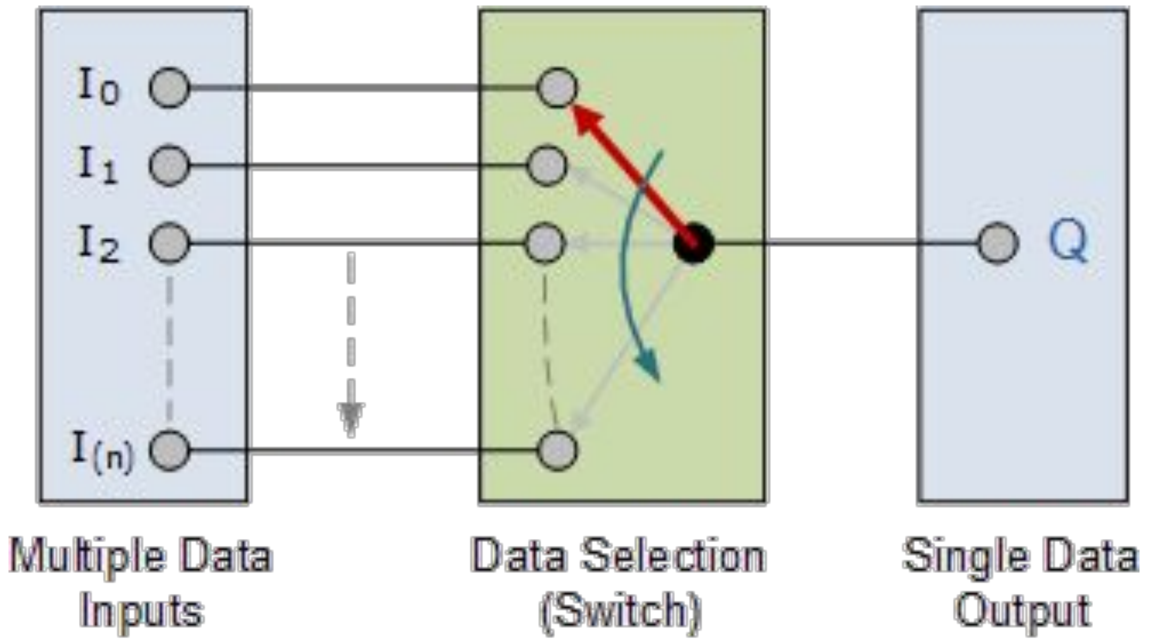
- A multiplexer is a combinational circuit that has 2^n input lines and a single output line.
- Simply, the multiplexer is a multi-input and single-output combinational circuit.
- The binary information is received from the input lines and directed to the output line. On the basis of the values of the selection lines, one of these data inputs will be connected to the output. A multiplexer is also treated as **Mux**.

Multiplexer

Multiplexer symbol-

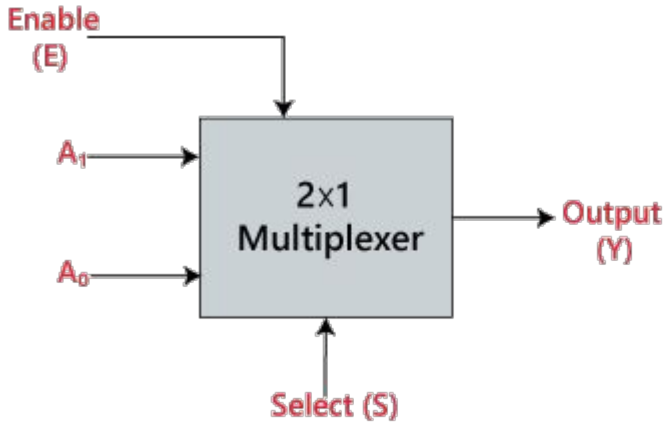


Basic Multiplexing switch



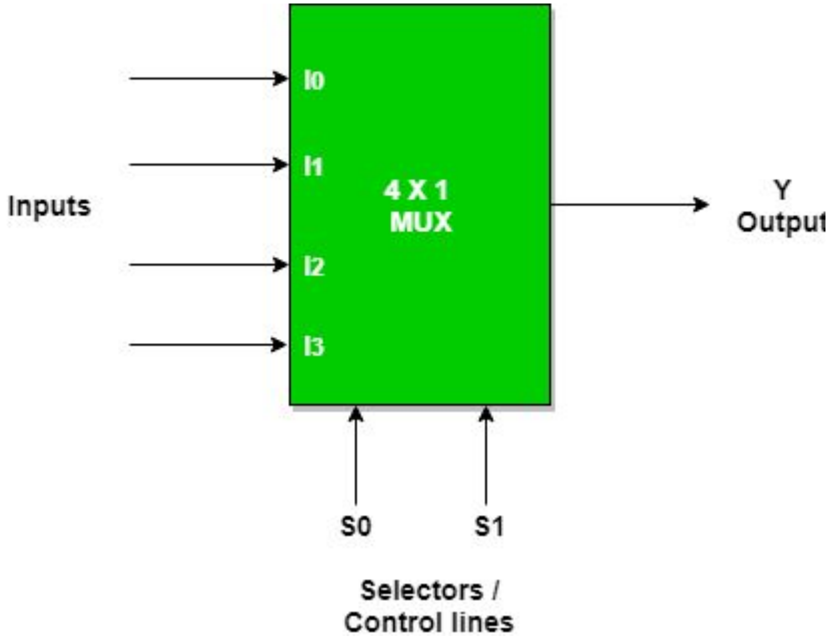
2X1 Multiplexer

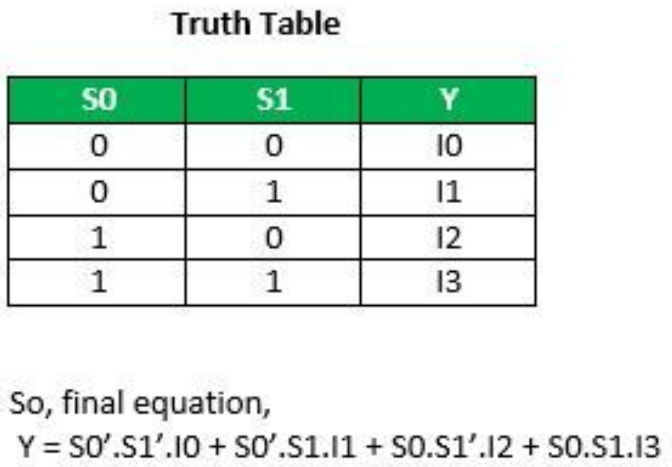
In 2×1 multiplexer, there are only two inputs, i.e., A_0 and A_1, 1 selection line, i.e., S_0 and single outputs, i.e., Y.



4X1 Multiplexer

In 4×1 multiplexer, there are only 4 inputs, 2 selection line and single outputs.





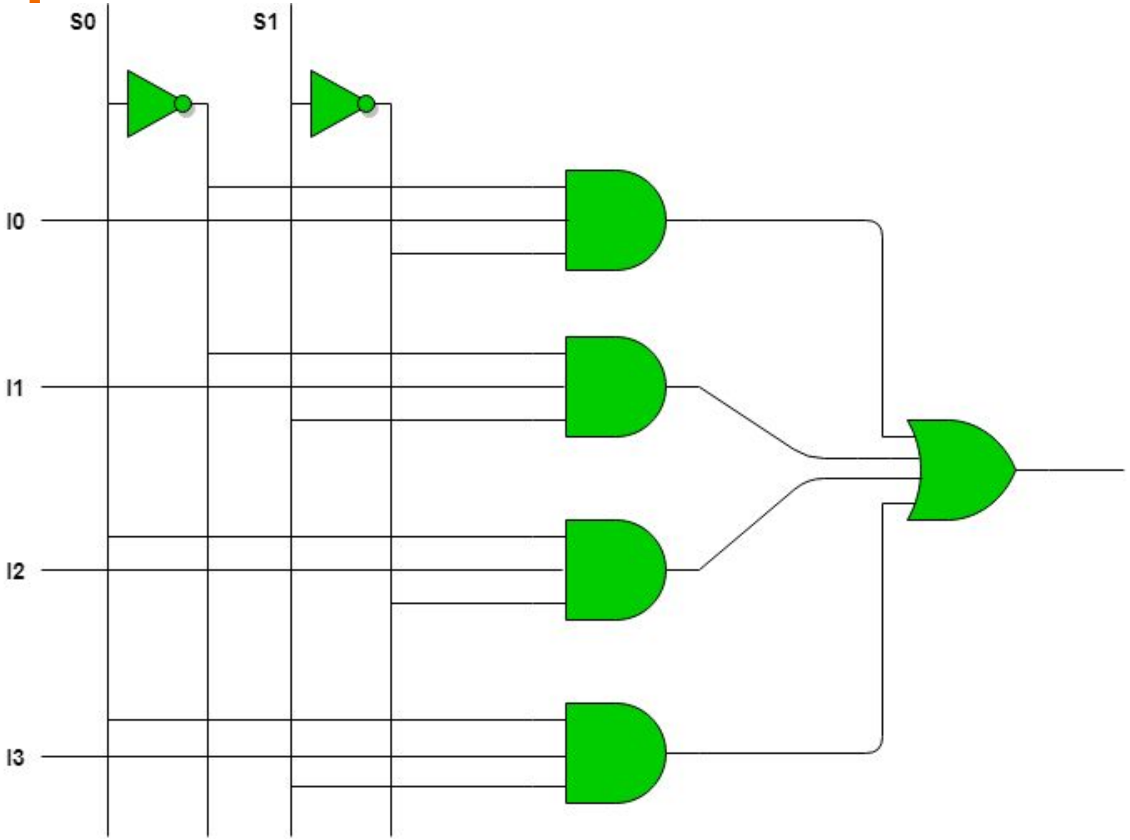
Truth Table

S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

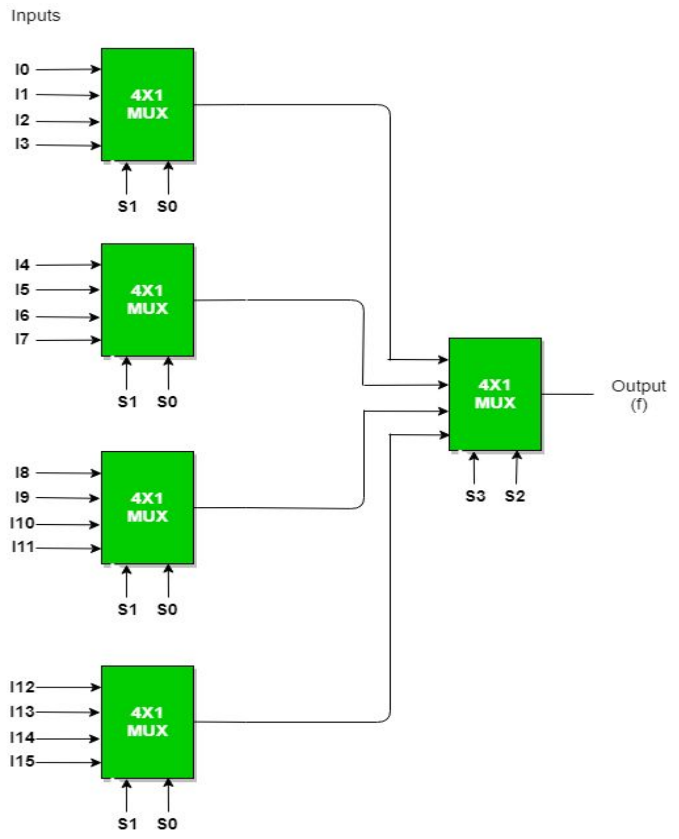
So, final equation,

$Y = S0'.S1'.I0 + S0'.S1.I1 + S0.S1'.I2 + S0.S1.I3$

4X1 Multiplexer



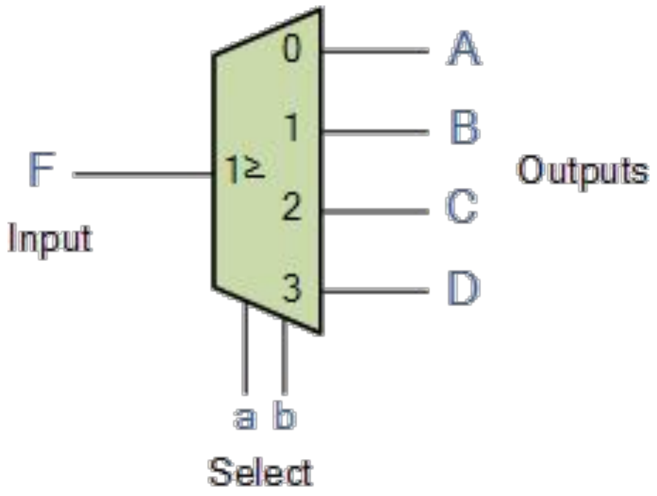
16:1 using 4:1 Multiplexer

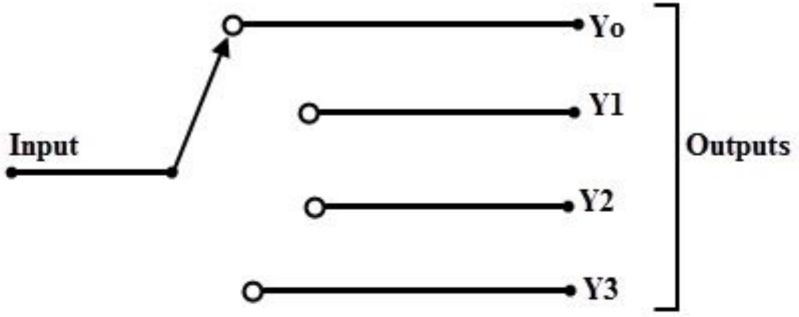


Demultiplexer

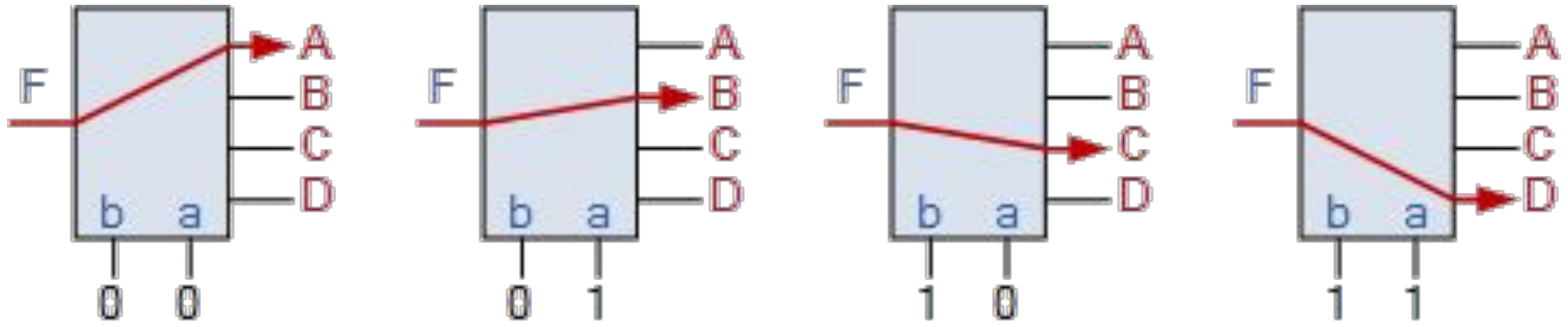
- De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer.
- It has single input, ‘n’ selection lines and maximum of 2n outputs.
- The input will be connected to one of these outputs based on the values of selection lines.
- Demultiplexer is also called as De-Mux.

1:4 Demultiplexer



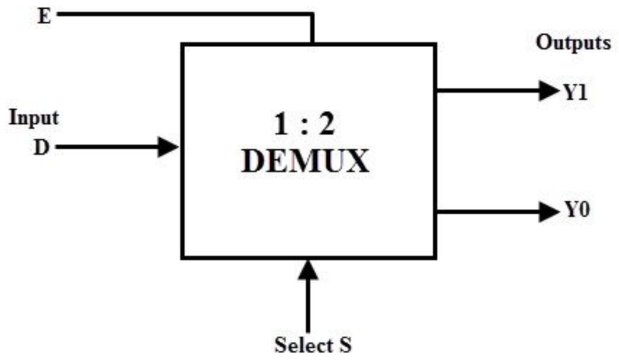


Demultiplexer output line selection



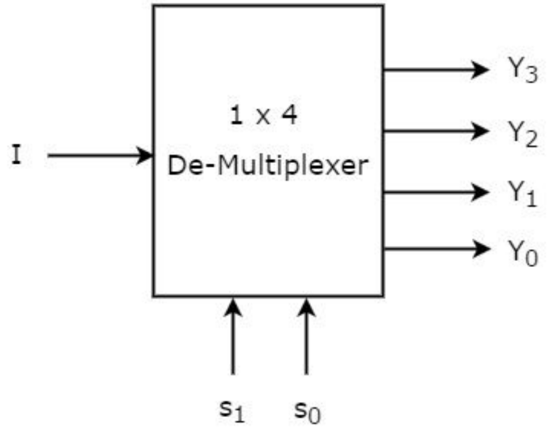
1:2 Demultiplexer

- A 1:2 demux consists of one input line, two output lines and one select line.
- The signal on the select line helps to switch the input to one of the two outputs.
- The figure below shows the block diagram of a 1-to-2 demultiplexer with additional enable input.

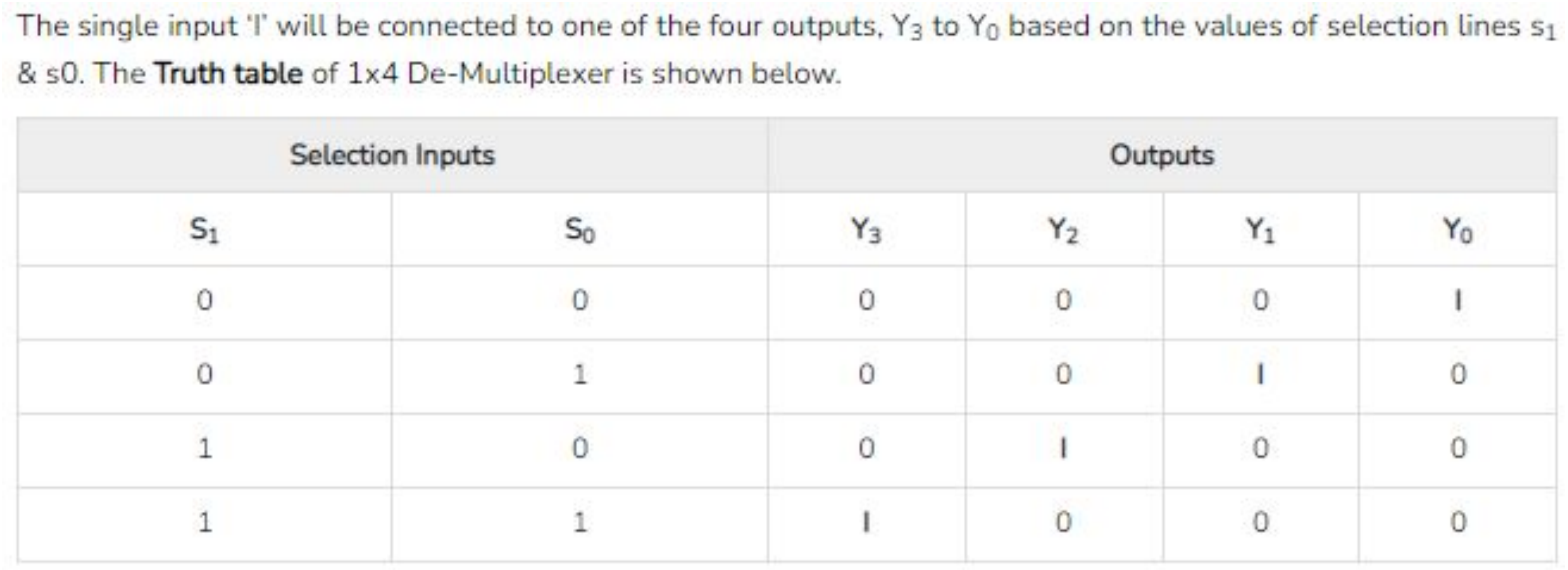


1:4 Demultiplexer

- 1x4 De-Multiplexer has one input I, two selection lines, s1 & s0 and four outputs Y3, Y2, Y1 &Y0.
- The block diagram of 1x4 De-Multiplexer is shown in the following figure.



1:4 Demultiplexer



The single input 'I' will be connected to one of the four outputs, Y_3 to Y_0 based on the values of selection lines s_1 & s0. The **Truth table** of 1x4 De-Multiplexer is shown below.

Selection Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

1:4 Demultiplexer

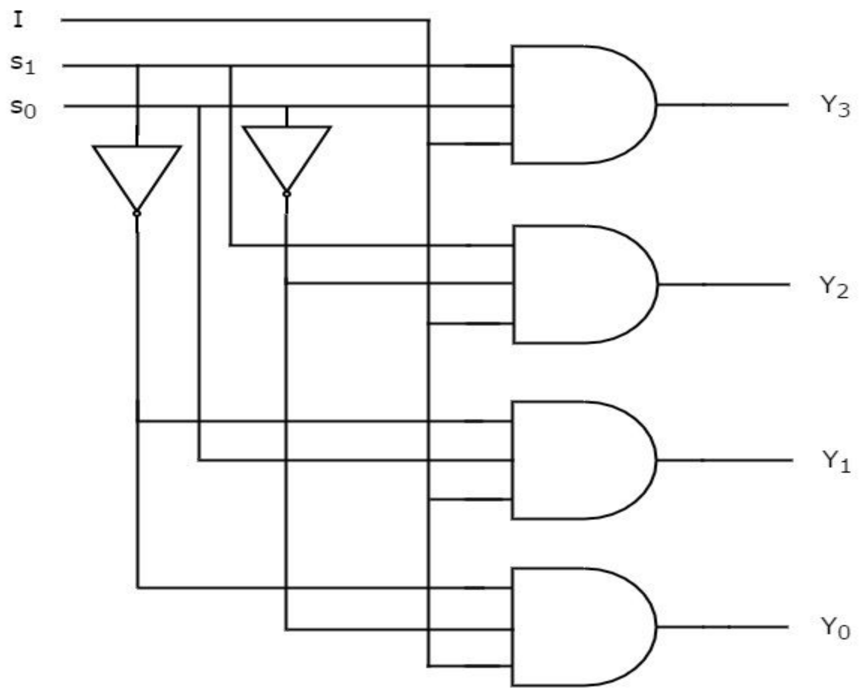
From the above truth table, the Boolean Expressions for the outputs as follows:

$Y0 = S_1'S_0'I$

$Y1 = S_1'S_0I$

$Y2 = S_1S_0'I$

$Y3 = S_1S_0I$



I
s_1
s_0
Y_3
Y_2
Y_1
Y_0