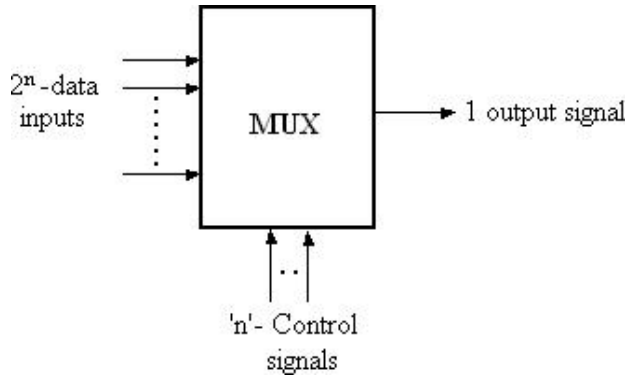
MULTIPLEXER: (Data Selector)

A ***Multiplexer*** or *MUX*, is a combinational circuit with more than one input line, one output line and more than one selection line. A multiplexer selects binary information present from one of many input lines, depending upon the logic status of the selection inputs, and routes it to the output line. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected. The multiplexer is often labeled as MUX in block diagrams.

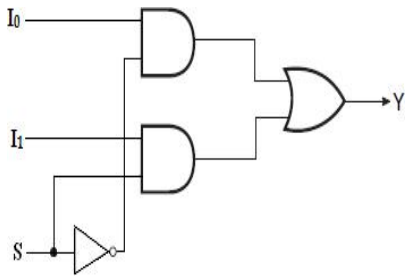
A multiplexer is also called a **data selector**, since it selects one of many inputs and steers the binary information to the output line.



Block diagram of Multiplexer

2-to-1- line Multiplexer:

The circuit has two data input lines, one output line and one selection line, S. When S= 0, the upper AND gate is enabled and I_0 has a path to the output.
When S=1, the lower AND gate is enabled and I_1 has a path to the output.



Logic diagram

The multiplexer acts like an electronic switch that selects one of the two sources.

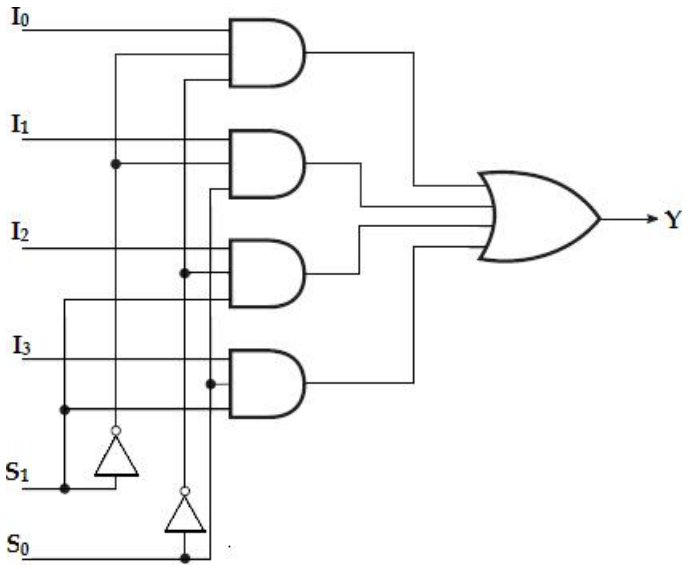
Truth table:

S	Y
0	I_0
1	I_1

4-to-1-line Multiplexer

A 4-to-1-line multiplexer has four (2^n) input lines, two (n) select lines and one output line. It is the multiplexer consisting of four input channels and information of one of the channels can be selected and transmitted to an output line according to the select inputs combinations. Selection of one of the four input channel is possible by two selection inputs.

Each of the four inputs I_0 through I_3, is applied to one input of AND gate. Selection lines S_1 and S_0 are decoded to select a particular AND gate. The outputs of the AND gate are applied to a single OR gate that provides the 1-line output.



4-to-1-Line Multiplexer

Function table:

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

To demonstrate the circuit operation, consider the case when S_1S_0= 10. The AND gate associated with input I_2 has two of its inputs equal to 1 and the third input connected to I_2. The other three AND gates have atleast one input equal to 0, which makes their outputs equal to 0. The OR output is now equal to the value of I_2, providing a path from the selected input to the output.

The data output is equal to I_0 only if S_1= 0 and S_0= 0; Y= $I_0S_1'S_0'$.

The data output is equal to I_1 only if S_1= 0 and S_0= 1; Y= $I_1S_1'S_0$.

The data output is equal to I_2 only if S_1= 1 and S_0= 0; Y= $I_2S_1S_0'$.

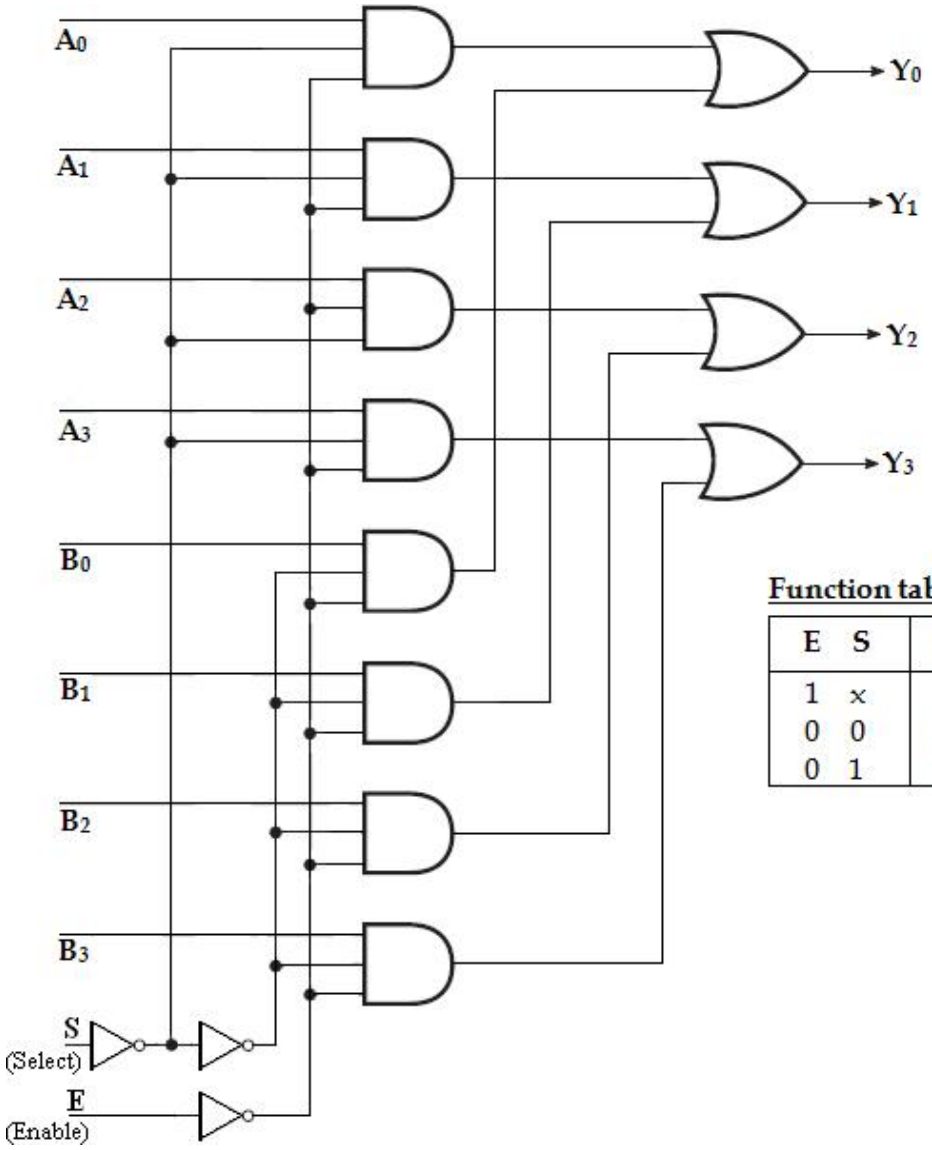
The data output is equal to I_3 only if S_1= 1 and S_0= 1; Y= $I_3S_1S_0$.

When these terms are ORed, the total expression for the data output is,

$$\mathbf{Y= I_0S_1'S_0'+ I_1S_1'S_0 +I_2S_1S_0'+ I_3S_1S_0}.$$

As in decoder, multiplexers may have an enable input to control the operation of the unit. When the enable input is in the inactive state, the outputs are disabled, and when it is in the active state, the circuit functions as a normal multiplexer.

Quadruple 2-to-1 Line Multiplexer



Function table:

E	S	Output Y
1	x	All 0's
0	0	Select A
0	1	Select B

This circuit has four multiplexers, each capable of selecting one of two input lines. Output Y_0 can be selected to come from either A0 or B0. Similarly, output Y1 may have the value of A1 or B1, and so on. Input selection line, S selects one of the lines in each of the four multiplexers. The enable input E must be active for normal operation.

Although the circuit contains four 2-to-1-Line multiplexers, it is viewed as a circuit that selects one of two 4-bit sets of data lines. The unit is enabled when E= 0. Then if S= 0, the four A inputs have a path to the four outputs. On the other hand, if S=1, the four B inputs are applied to the outputs. The outputs have all 0's when E= 1, regardless of the value of S.

Application:

1. They are used as a data selector to select out of many data inputs.
2. They can be used to implement combinational logic circuit.
3. They are used in time multiplexing systems.
4. They are used in frequency multiplexing systems.
5. They are used in A/D and D/A converter.
6. They are used in data acquisition systems.

IMPLEMENTATION OF BOOLEAN FUNCTION USING MUX:

1. **Implement the following boolean function using 4: 1 multiplexer,**

   $F(A, B, C) = \sum m(1, 3, 5, 6)$.

Solution:

Variables, n= 3 (A, B, C)

Select lines= n-1 = 2 (S_1, S_0)

2^{n-1} to MUX i.e., 2^2 to 1 = 4 to 1 MUX

Input lines= $2^{n-1} = 2^2 = 4$ (D_0, D_1, D_2, D_3)

Implementation table:

Apply variables A and B to the select lines. The procedures for implementing the function are:

i. List the input of the multiplexer

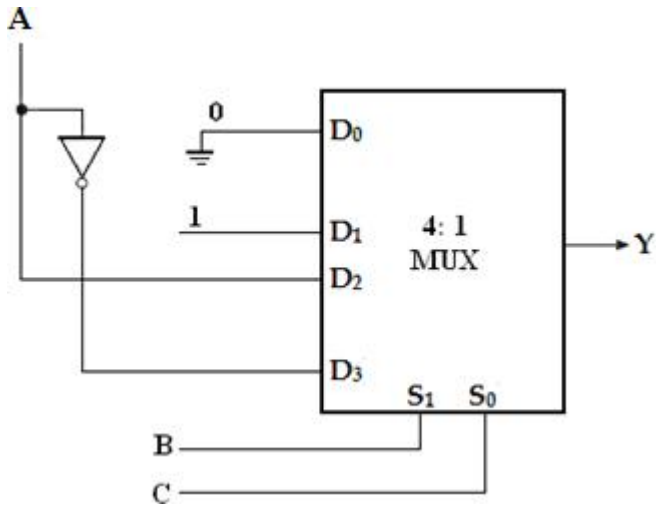
ii. List under them all the minterms in two rows as shown below.

The first half of the minterms is associated with A' and the second half with A. The given function is implemented by circling the minterms of the function and applying the following rules to find the values for the inputs of the multiplexer.

1. If both the minterms in the column are not circled, apply 0 to the corresponding input.
2. If both the minterms in the column are circled, apply 1 to the corresponding input.
3. If the bottom minterm is circled and the top is not circled, apply C to the input.
4. If the top minterm is circled and the bottom is not circled, apply C' to the input.

	D_0	D_1	D_2	D_3
$\overline{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	**1**	**A**	$\overline{A}$

Multiplexer Implementation:

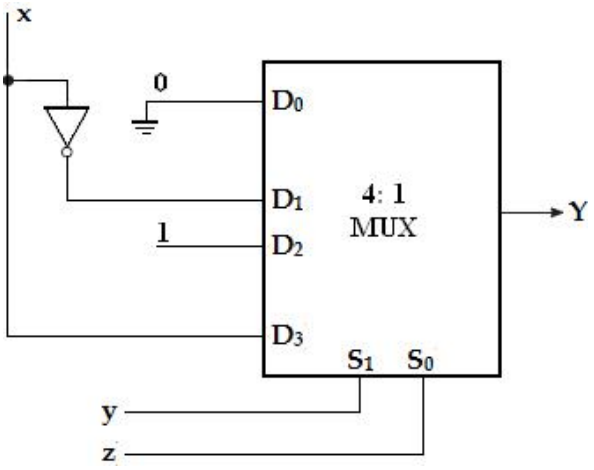


2. **F (x, y, z) = ∑m (1, 2, 6, 7)**

Implementation table:

	D_0	D_1	D_2	D_3
$\overline{x}$	0	(1)	(2)	3
x	4	5	(6)	(7)
	0	$\overline{x}$	**1**	**x**

Multiplexer Implementation:



1. **F (A, B, C) = ∑m (1, 2, 4, 5)**

Solution:

Variables, n= 3 (A, B, C)

Select lines= n-1 = 2 (**S_1, S_0**)

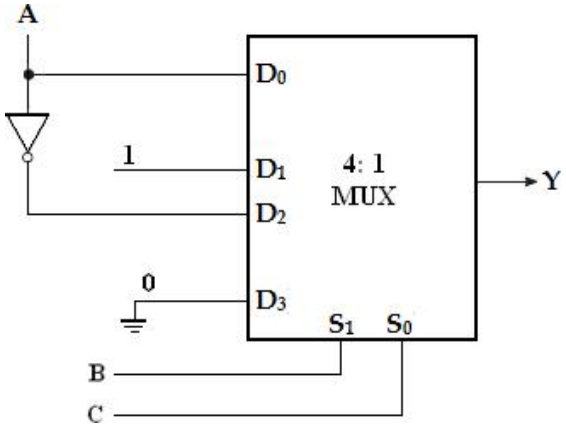
2^{n-1} to MUX i.e., 2^2 to 1 = 4 to 1 MUX

Input lines= $2^{n-1} = 2^2 = 4$ (**D_0, D_1, D_2, D_3**)

Implementation table:

	D_0	D_1	D_2	D_3
$\overline{A}$	0	(1)	(2)	3
A	(4)	(5)	6	7
	A	1	$\overline{A}$	0

Multiplexer Implementation



2. **F(A, B, C, D)= ∑m (0, 1, 3, 4, 8, 9, 15)**

Solution:

Variables, n= 4 (A, B, C, D)

Select lines= n-1 = 3 (**S_2, S_1, S_0**)

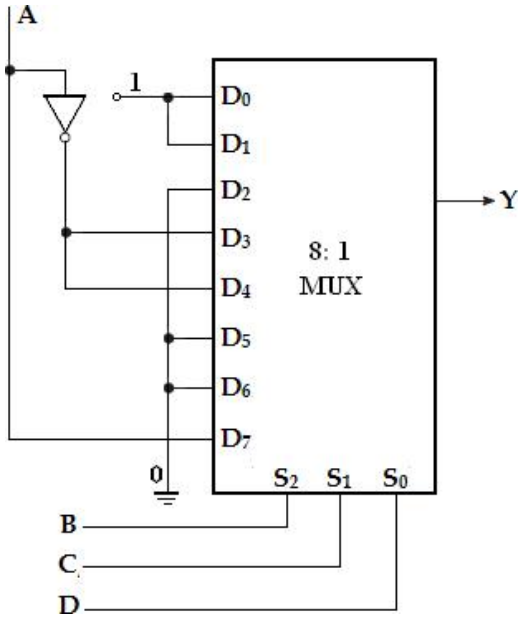
2^{n-1} to MUX i.e., 2^3 to 1 = 8 to 1 MUX

Input lines= $2^{n-1} = 2^3 = 8$ (**D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7**)

Implementation table:

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	**1**	**0**	$\overline{A}$	$\overline{A}$	**0**	**0**	**A**

Multiplexer Implementation:



3. **Implement the Boolean function using 8: 1 and also using 4:1 multiplexer**

   F (A, B, C, D) = ∑m (0, 1, 2, 4, 6, 9, 12, 14)

Solution:

Variables, n= 4 (A, B, C, D)

Select lines= n-1 = 3 (**S_2, S_1, S_0**)

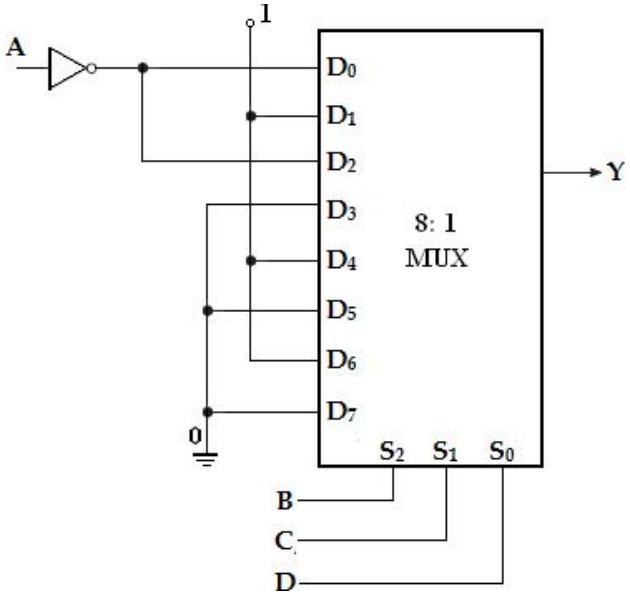
2^{n-1} to MUX i.e., 2^3 to 1 = 8 to 1 MUX

Input lines= $2^{n-1} = 2^3 = 8$ (**D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7**)

Implementation table:

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	$\overline{A}$	**1**	$\overline{A}$	**0**	**1**	**0**	**1**	**0**

Multiplexer Implementation (Using 8: 1 MUX)



Using 4: 1 MUX: