



Sri  
**SAI RAM**  
ENGINEERING COLLEGE  
INSTITUTE OF TECHNOLOGY  
West Tambaram, Chennai - 44



**SAIRAM**  
DIGITAL RESOURCES



**CS8351**

**DIGITAL PRINCIPLES AND SYSTEM DESIGN**  
(Common to CSE & IT)

## UNIT NO 2

### COMBINATIONAL LOGIC

#### 2.10 Multiplexers

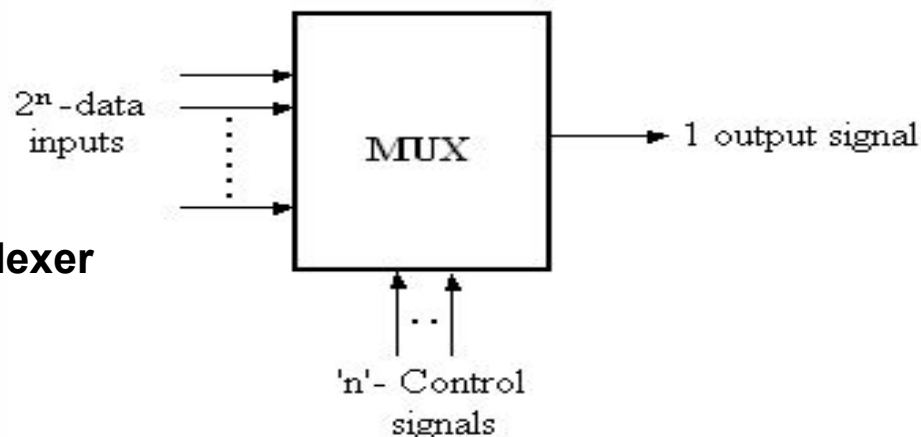
## INFORMATION TECHNOLOGY



## MULTIPLEXER

- 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.

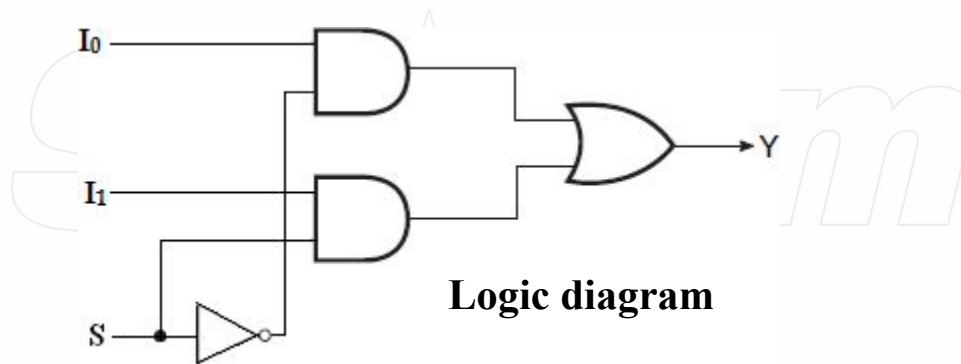


Block diagram of Multiplexer

## 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.



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

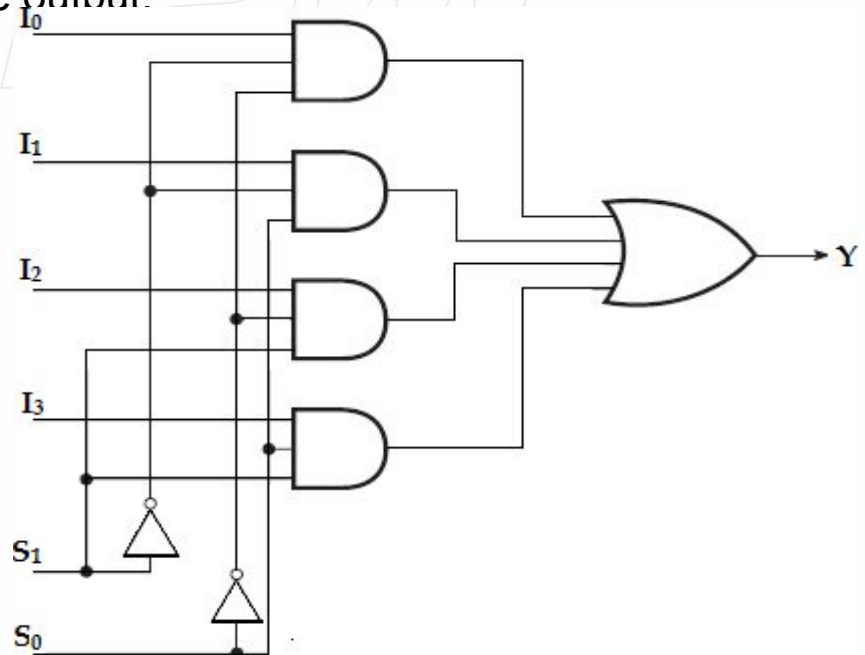
### Truth table

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

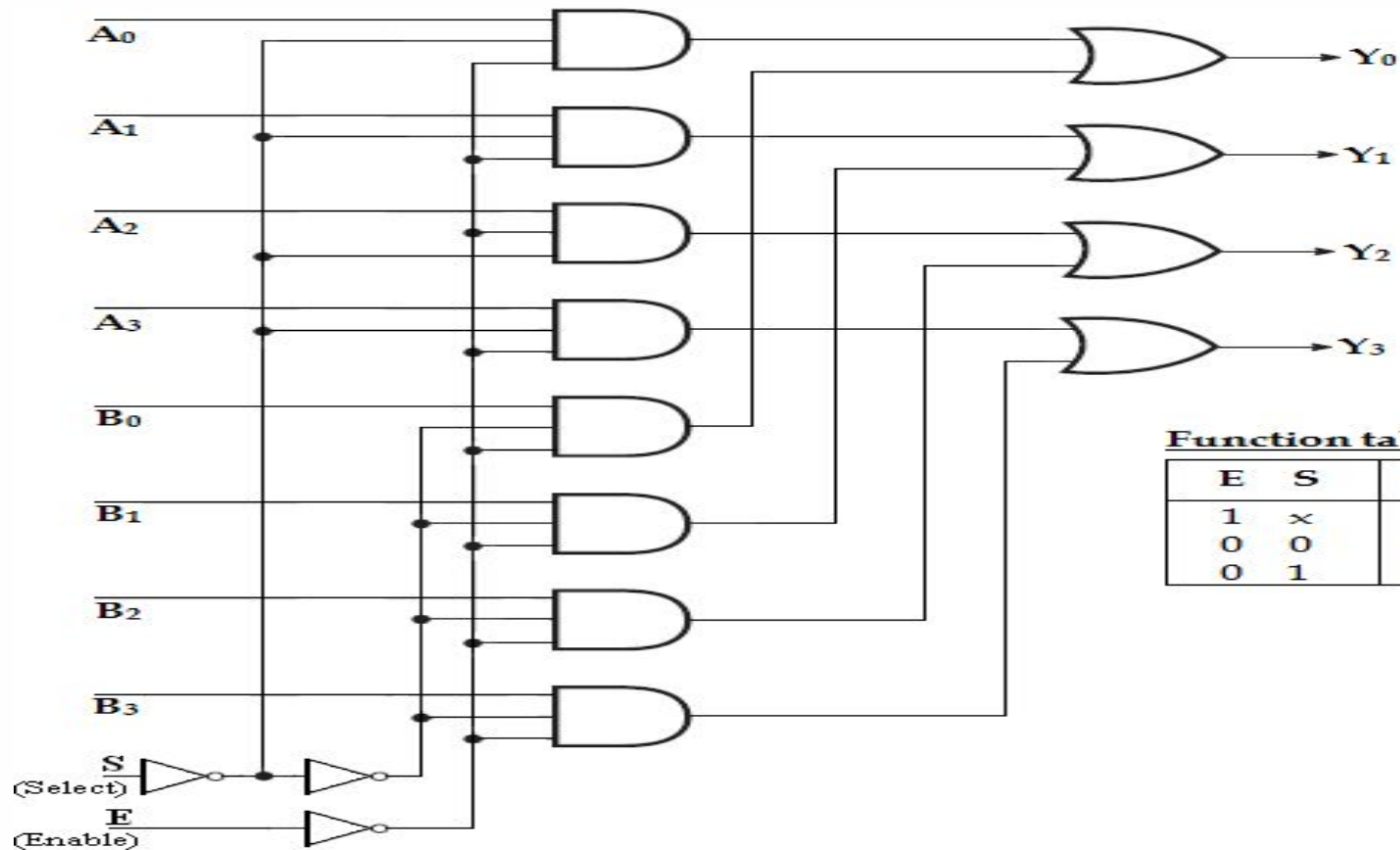
S <sub>1</sub>	S <sub>0</sub>	Y
0	0	I <sub>0</sub>
0	1	I <sub>1</sub>
1	0	I <sub>2</sub>
1	1	I <sub>3</sub>

- ❖ To demonstrate the circuit operation, consider the case when S<sub>1</sub>S<sub>0</sub>= 10. The AND gate associated with input I<sub>2</sub> has two of its inputs equal to 1 and the third input connected to I<sub>2</sub>. The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The OR output is now equal to the value of I<sub>2</sub>, providing a path from the selected input to the output.
- The data output is equal to I<sub>0</sub> only if S<sub>1</sub>= 0 and S<sub>0</sub>= 0;  $Y = I_0S_1'S_0$ .
- The data output is equal to I<sub>1</sub> only if S<sub>1</sub>= 0 and S<sub>0</sub>= 1;  $Y = I_1S_1'S_0$ .
- The data output is equal to I<sub>2</sub> only if S<sub>1</sub>= 1 and S<sub>0</sub>= 0;  $Y = I_2S_1S_0'$ .
- The data output is equal to I<sub>3</sub> only if S<sub>1</sub>= 1 and S<sub>0</sub>= 1;  $Y = I_3S_1S_0$ .

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

$$Y = I_0S_1'S_0' + I_1S_1'S_0 + I_2S_1S_0' + I_3S_1S_0.$$

## 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 Y0 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

The multiplexer is a very useful MSI function and has various ranges of applications in data communication. Signal routing and data communication are the important applications of a multiplexer. It is used for connecting two or more sources to guide to a single destination among computer units and it is useful for constructing a common bus system. One of the general properties of a multiplexer is that Boolean functions can be implemented by this device.



## Implementation of Boolean Function using MUX

- Any Boolean or logical expression can be easily implemented using a multiplexer. If a Boolean expression has  $(n+1)$  variables, then  $n$  of these variables can be connected to the select lines of the multiplexer.
- The remaining single variable along with constants 1 and 0 is used as the input of the multiplexer. For example, if C is the single variable, then the inputs of the multiplexers are C, C', 1 and 0. By this method any logical expression can be implemented.
- In general, a Boolean expression of  $(n+1)$  variables can be implemented using a multiplexer with  $2^n$  inputs.

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 =  $n - 1 = 2$  (**S1, S0**)

$2^{n-1}$  to MUX i.e.,  $2^2$  to 1 = 4 to 1 MUX Input lines =  $2^{n-1} = 2^2 = 4$  (**D0, D1, D2, D3**)



## Implementation table

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

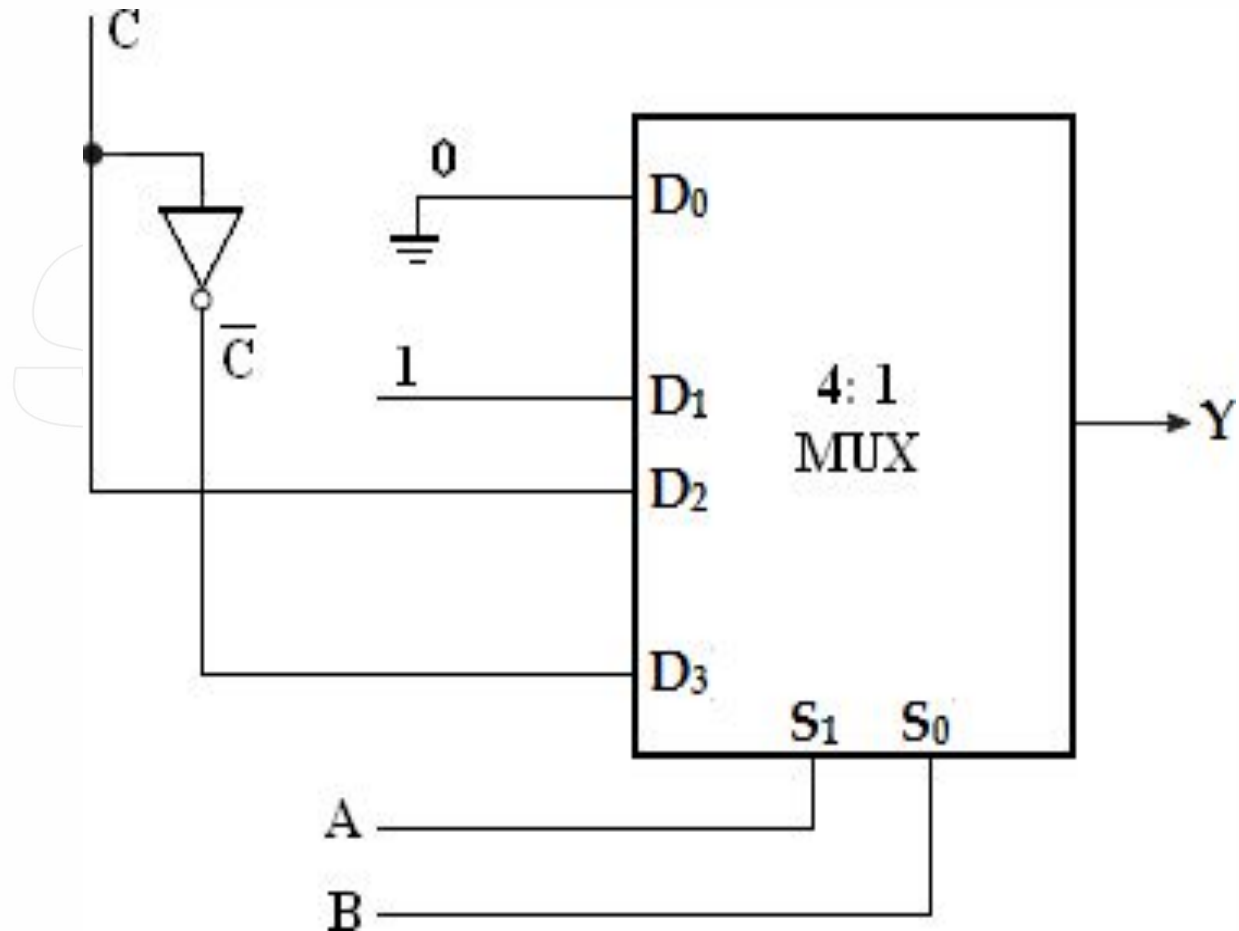
1. List the input of the multiplexer
2. List under them all the minterms in two rows as shown below.

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.

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

	$D_0$	$D_1$	$D_2$	$D_3$
$\bar{C}$	0	1	2	3
$C$	4	5	6	7
	0	1	$C$	$\bar{C}$

## Multiplexer Implementation



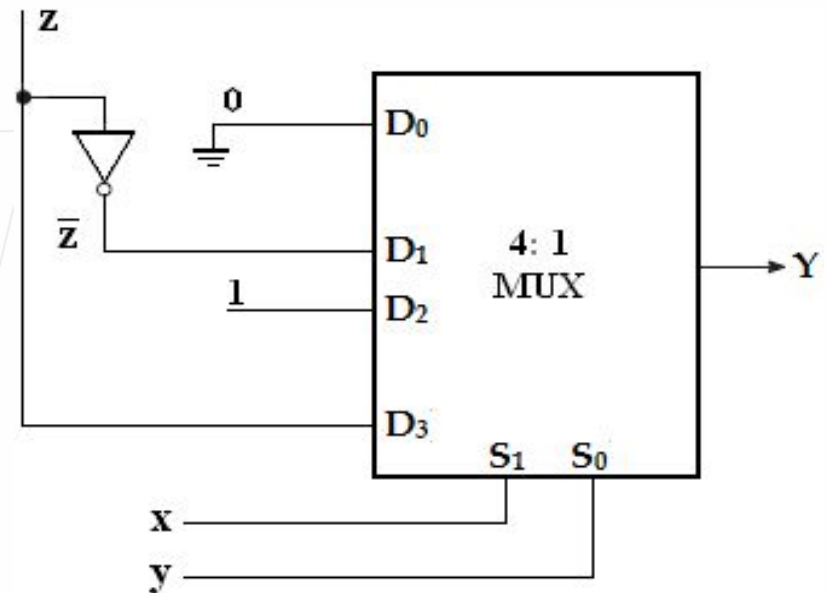
2.  $F(x, y, z) = \sum m(1, 2, 6, 7)$

Solution:

Implementation table

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
$\bar{z}$	0	1	2	3
z	4	5	6	7
	0	$\bar{z}$	1	z

Multiplexer Implementation



3.  $F(A, B, C) = \sum m(1, 2, 4, 5)$

Solution:

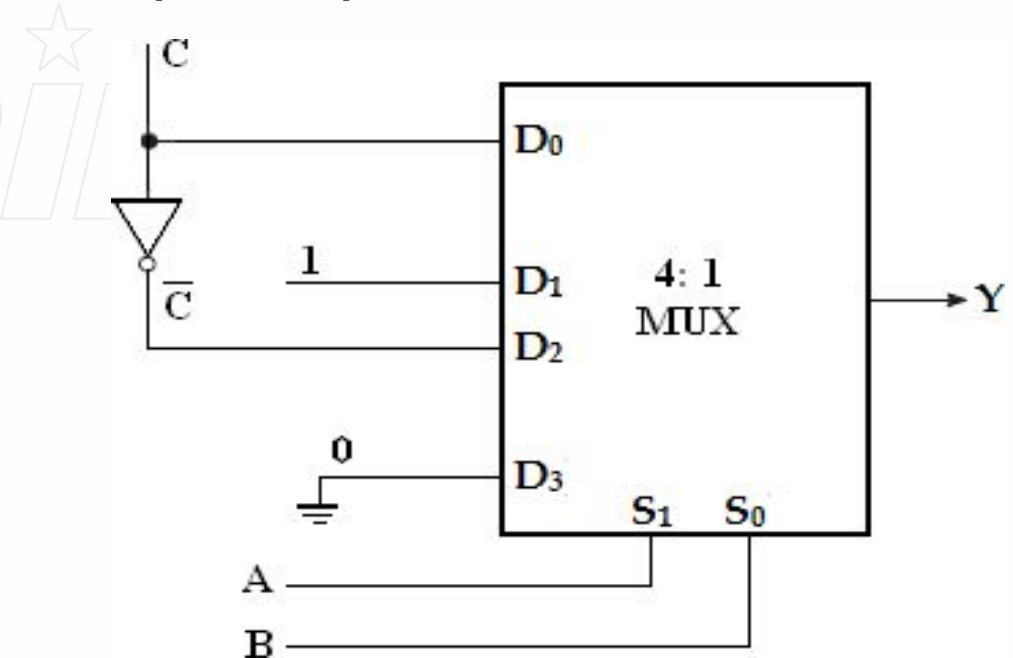
Variables,  $n = 3$  (A, B, C) Select lines =  $n - 1 = 2$  (**S1, S0**)

$2^{n-1}$  to MUX i.e.,  $2^2$  to 1 = 4 to 1 MUX Input lines =  $2^{n-1} = 2^2 = 4$  (**D0, D1, D2, D3**)

Implementation table

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
$\bar{C}$	0	1	2	3
C	4	5	6	7
	C	1	$\bar{C}$	0

Multiplexer Implementation



4.  $F(P, Q, R, S) = \sum m(0, 1, 3, 4, 8, 9, 15)$

**Solution:**

Variables,  $n = 4$  ( $P, Q, R, S$ ) Select lines =  $n - 1 = 3$  ( $S_2, S_1, S_0$ )

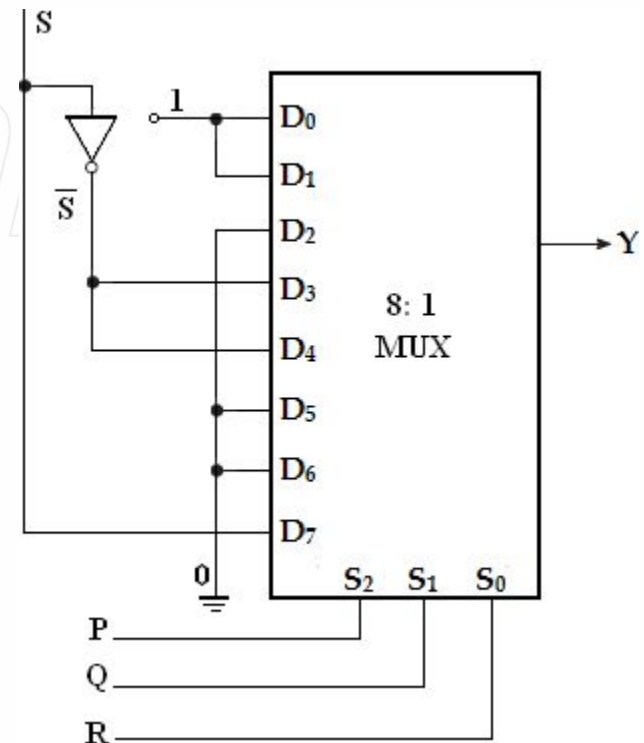
$2^{n-1}$  to MUX i.e.,  $2^3$  to  $1 = 8$  to  $1$  MUX

Input lines =  $2^n - 1 = 2^4 - 1 = 15 = 8$  ( $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$ )

**Implementation table**

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

**Multiplexer Implementation**



5. Implement the Boolean function using 8: 1 and also using 4:1 multiplexer  
 $F(A, B, C, D) = \sum m(0, 1, 2, 4, 6, 9, 12, 14)$

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{S}$	0	1	2	3	4	5	6	7
S	8	9	10	11	12	13	14	15
	1	1	0	$\bar{S}$	$\bar{S}$	0	0	S

**Solution:**

Variables,  $n = 4$  (A, B, C, D) Select lines =  $n - 1 = 3$  (**S<sub>2</sub>, S<sub>1</sub>, S<sub>0</sub>**)

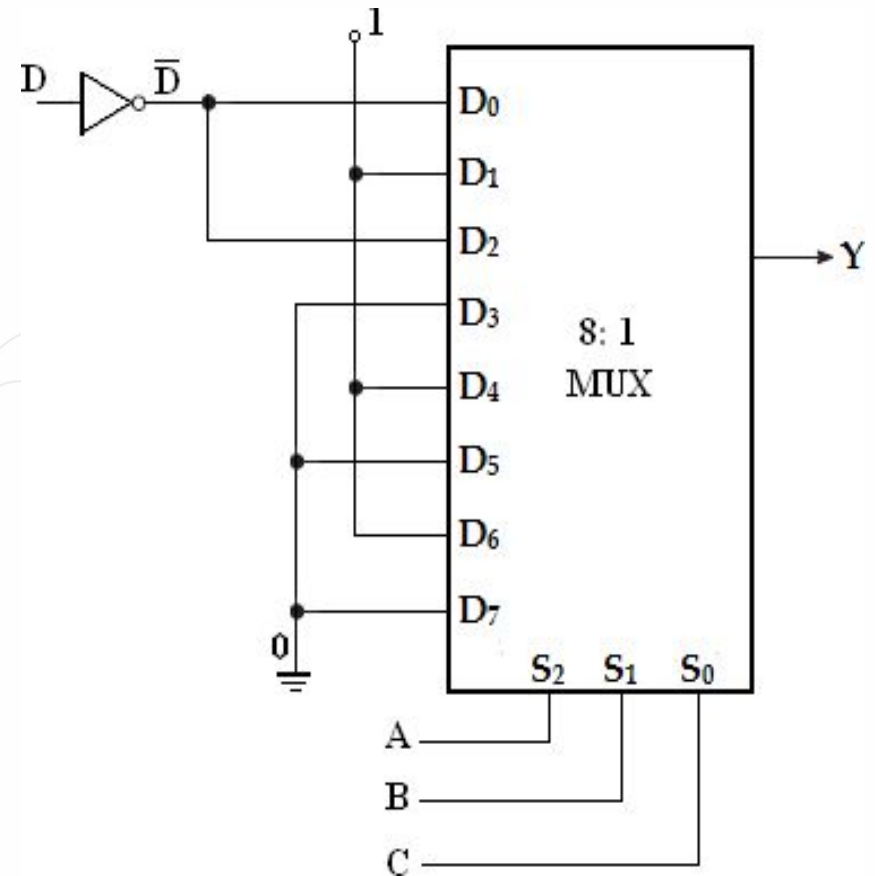
$2^{n-1}$  to MUX i.e., 23 to 1 = 8 to 1 MUX

Input lines =  $2^{n-1} = 2^3 = 8$  (**D<sub>0</sub>, D<sub>1</sub>, D<sub>2</sub>, D<sub>3</sub>, D<sub>4</sub>, D<sub>5</sub>, D<sub>6</sub>, D<sub>7</sub>**)

## Multiplexer Implementation

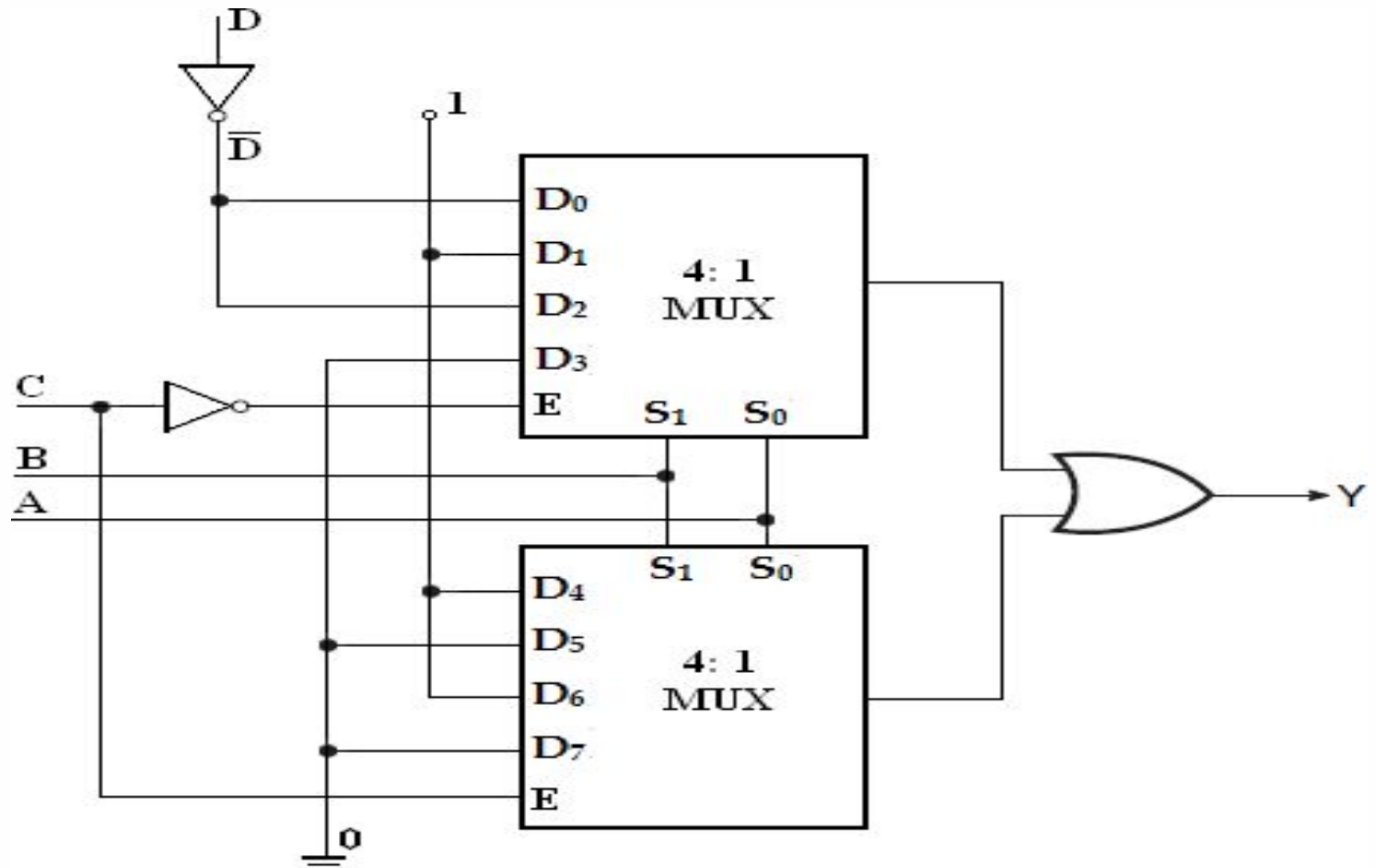
	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	$D_6$	$D_7$
$\bar{D}$	0	1	2	3	4	5	6	7
$D$	8	9	10	11	12	13	14	15
	$\bar{D}$	1	$\bar{D}$	0	1	0	1	0

## Multiplexer Implementation (Using 8: 1 MUX)





Using 4: 1 MUX



6.  $F(A, B, C, D) = \sum m(1, 3, 4, 11, 12, 13, 14, 15)$

**Solution:**

Variables,  $n = 4$  (A, B, C, D) Select lines =  $n - 1 = 3$  (**S2, S1, S0**)

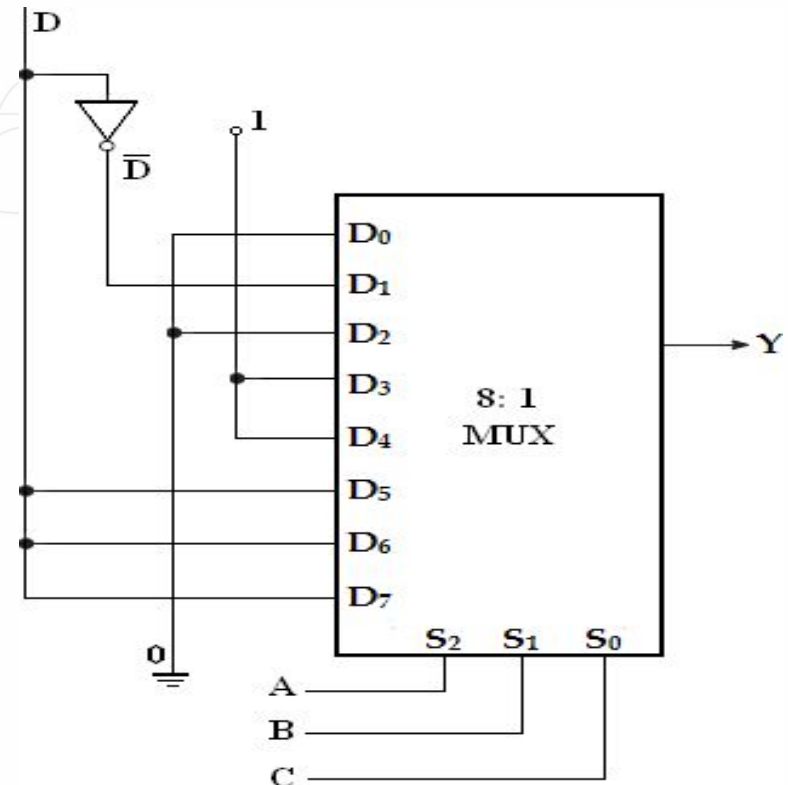
$2^{n-1}$  to MUX i.e., 23 to 1 = 8 to 1 MUX

Input lines =  $2^{n-1} = 2^3 = 8$  (**D0, D1, D2, D3, D4, D5, D6, D7**)

**Implementation table**

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{D}$	0	1	2	3	4	5	6	7
D	8	9	10	11	12	13	14	15
	0	$\bar{D}$	0	1	1	D	D	D

**Multiplexer Implementation**



7. Implement the Boolean function using 8: 1 multiplexer.

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

**Solution:**

Convert into standard SOP form,

$$= A'BD' (C'+C) + ACD (B'+B) + B'CD (A'+A) + A'C'D (B'+B)$$

$$= A'BC'D' + A'BCD' + \underline{AB'CD} + ABCD + A'B'CD + \underline{AB'CD} + A'B'C'D + A'BC'D$$

$$= A'BC'D' + A'BCD' + AB'CD + ABCD + A'B'CD + A'B'C'D + A'BC'D$$

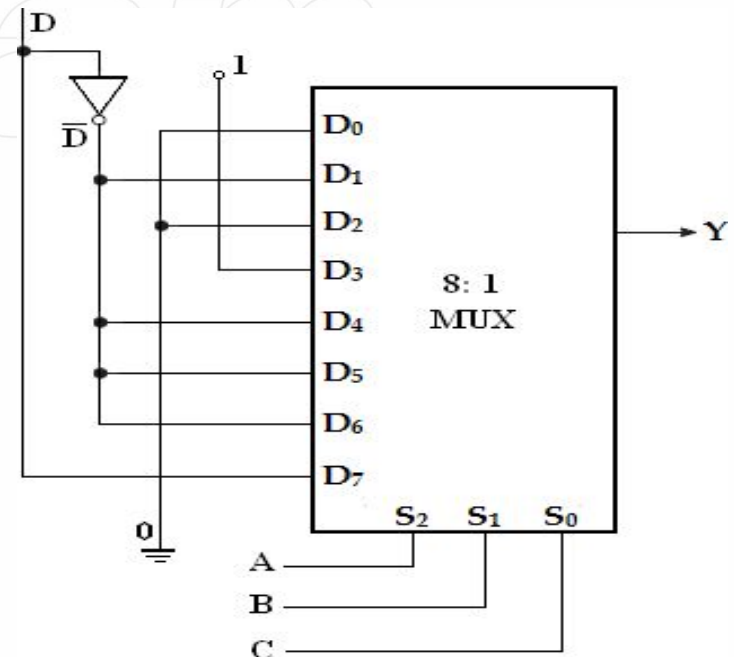
$$= m_4 + m_6 + m_{11} + m_{15} + m_3 + m_1 + m_5$$

$$= \sum m(1, 3, 4, 5, 6, 11, 15)$$

**Implementation table**

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{D}$	0	1	2	3	4	5	6	7
D	8	9	10	11	12	13	14	15
	0	$\bar{D}$	0	1	$\bar{D}$	$\bar{D}$	$\bar{D}$	D

**Multiplexer Implementation**



8. Implement the Boolean function using 8: 1 multiplexer.

$$F(A, B, C, D) = AB'D + A'C'D + B'CD' + AC'D.$$

**Solution:**

Convert into standard SOP form,

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

$$= \underline{AB'C'D} + AB'CD + A'B'C'D + A'BC'D + A'B'CD' + AB'CD' + \underline{AB'C'D} + ABC'D$$

$$= AB'C'D + AB'CD + A'B'C'D + A'BC'D + A'B'CD' + AB'CD' + ABC'D$$

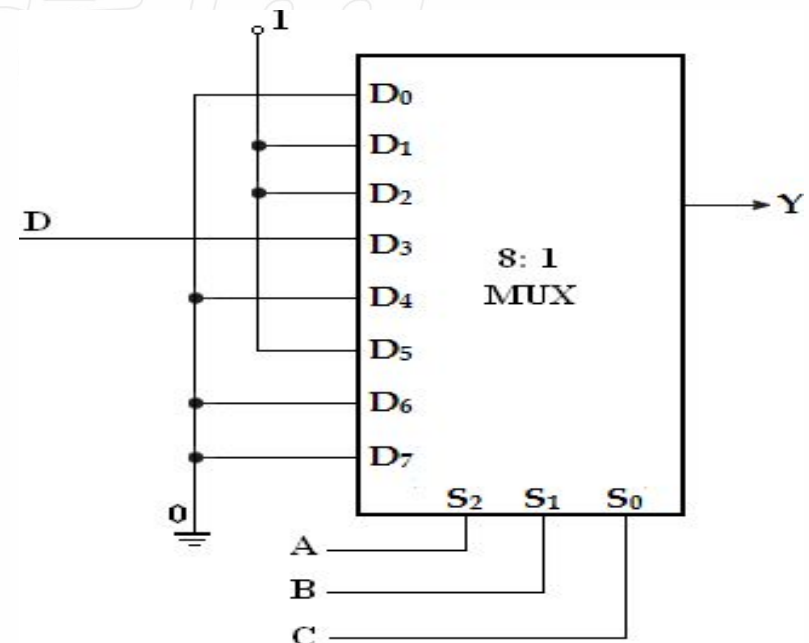
$$= m_9 + m_{11} + m_1 + m_5 + m_2 + m_{10} + m_{13}$$

$$= \sum m(1, 2, 5, 9, 10, 11, 13).$$

**Implementation table**

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{D}$	0	1	2	3	4	5	6	7
D	8	9	10	11	12	13	14	15
	0	1	1	D	0	1	0	0

**Multiplexer Implementation**



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

$$F(w, x, y, z) = \sum m(1, 2, 3, 6, 7, 8, 11, 12, 14)$$

**Solution:**

Variables,  $n = 4$  ( $w, x, y, z$ ) Select lines =  $n - 1 = 3$  (**S2, S1, S0**)

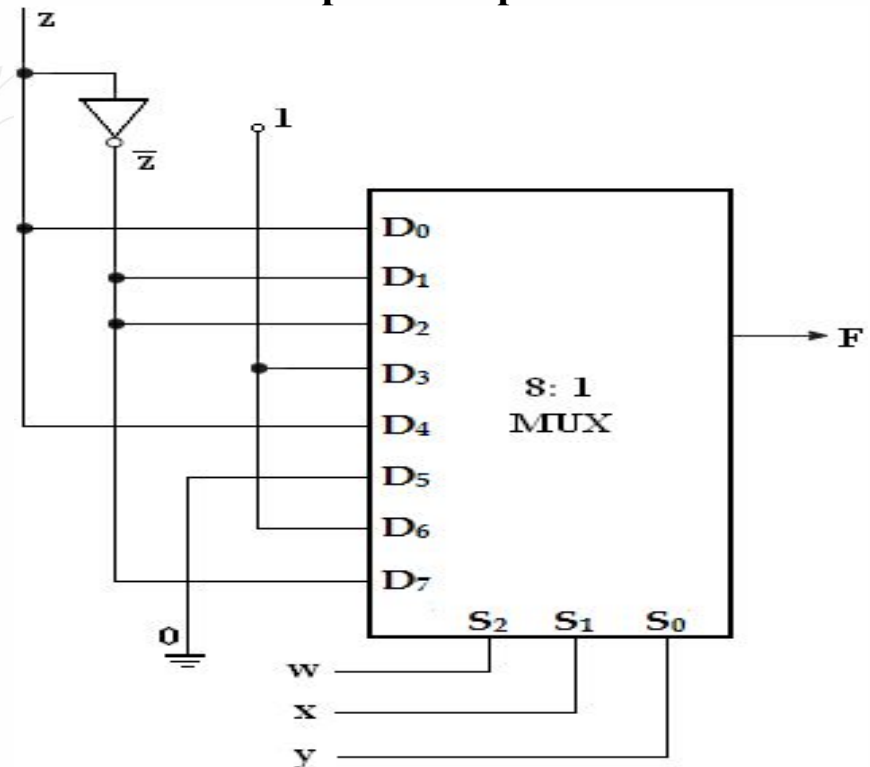
$2^n - 1$  to MUX i.e.,  $2^3 - 1 = 8 - 1 = 7$  to 1 MUX

Input lines =  $2^n - 1 = 2^3 - 1 = 7$  to 1 MUX

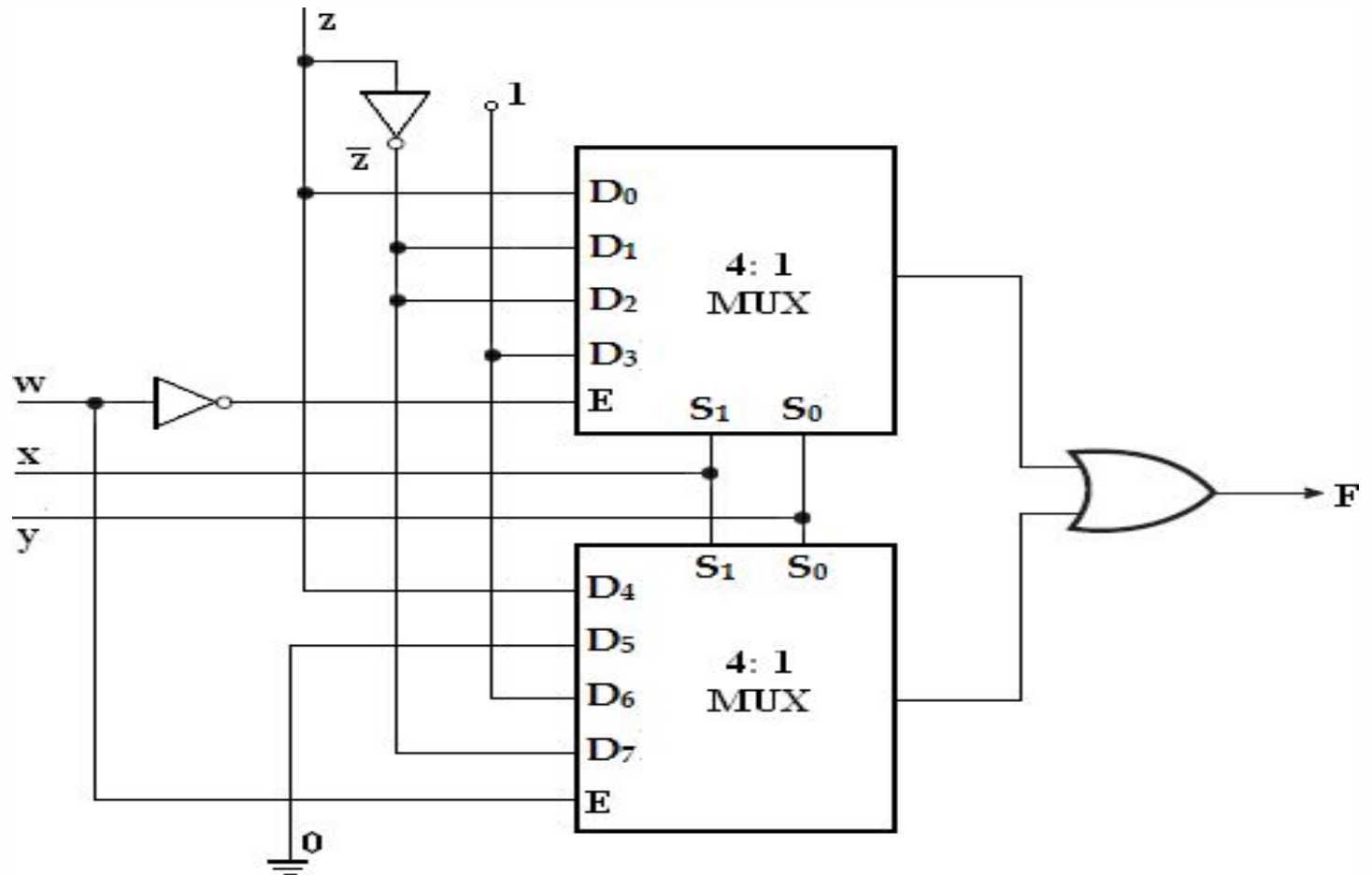
**Implementation table**

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{z}$	0	1	2	3	4	5	6	7
$z$	8	9	10	11	12	13	14	15
	$z$	$\bar{z}$	$\bar{z}$	1	$z$	0	1	$\bar{z}$

**Multiplexer Implementation**



(Using 4:1 MUX)



10. Implement the Boolean function using 8: 1 multiplexer

$$F(A, B, C, D) = \sum m(0, 3, 5, 8, 9, 10, 12, 14)$$

**Solution:**

Variables,  $n = 4$  (A, B, C, D) Select lines =  $n - 1 = 3$  (**S2, S1, S0**)

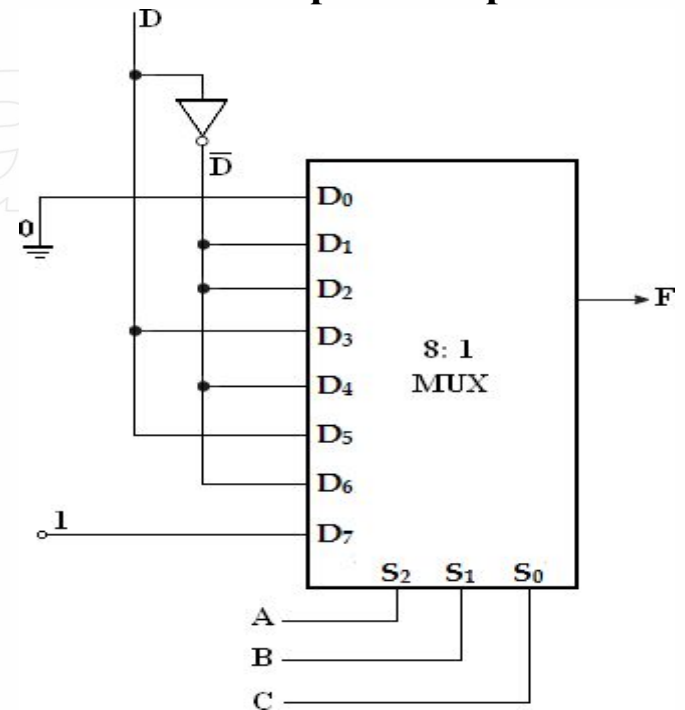
$2^n - 1$  to MUX i.e.,  $2^4 - 1 = 15$  to  $1 = 8$  to  $1$  MUX

Input lines =  $2^n - 1 = 15 = 8$  (**D0, D1, D2, D3, D4, D5, D6, D7**)

**Implementation table**

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{D}$	0	1	2	3	4	5	6	7
D	8	9	10	11	12	13	14	15
	0	$\bar{D}$	$\bar{D}$	D	$\bar{D}$	D	$\bar{D}$	1

**Multiplexer Implementation**





11. Implement the Boolean function using 8: 1 multiplexer

$$F(A, B, C, D) = \sum m(0, 2, 6, 10, 11, 12, 13) + d(3, 8, 14)$$

**Solution:**

Variables,  $n = 4$  (A, B, C, D) Select lines =  $n - 1 = 3$  (**S2, S1, S0**)

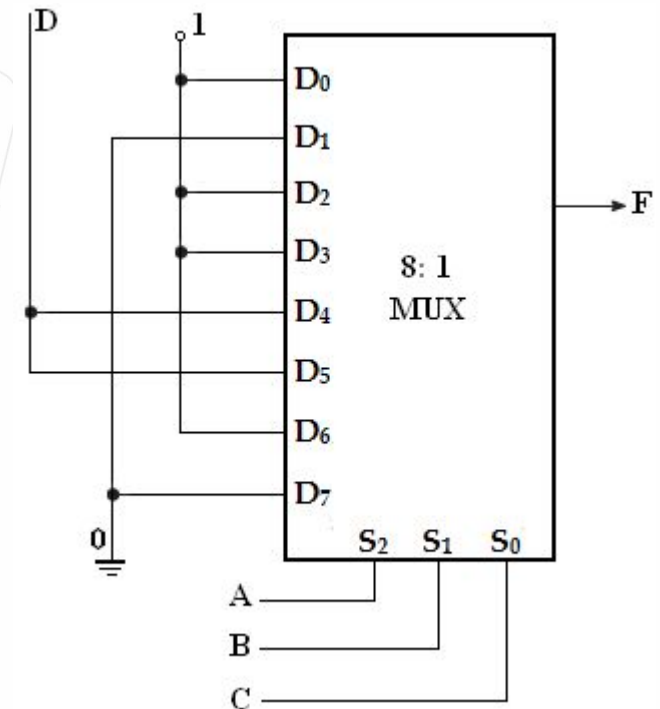
$2n - 1$  to MUX i.e., 23 to 1 = 8 to 1 MUX

Input lines =  $2n - 1 = 23 = 8$  (**D0, D1, D2, D3, D4, D5, D6, D7**)

**Implementation table**

	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$\bar{D}$	0	1	2	3	4	5	6	7
D	8	9	10	11	12	13	14	15
	1	0	1	1	D	D	1	0

**Multiplexer Implementation**

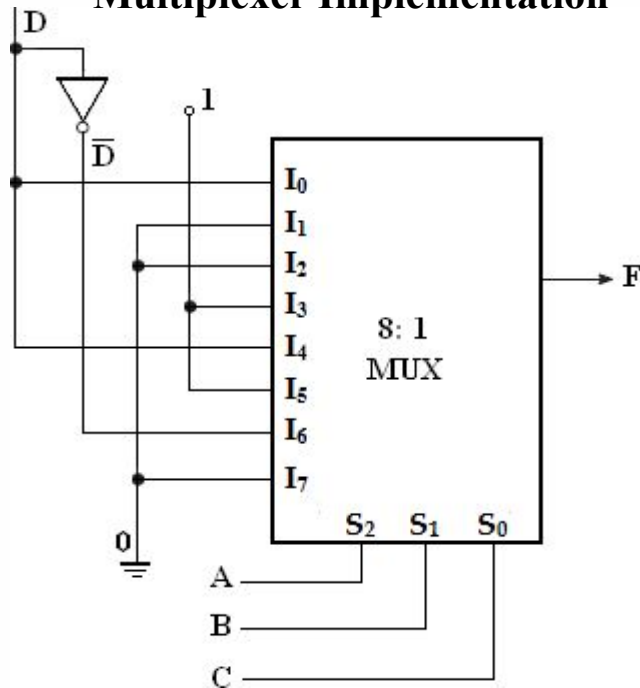


12. An 8×1 multiplexer has inputs A, B and C connected to the selection inputs S2, S1, and S0 respectively. The data inputs I0 to I7 are as follows

**I1=I2=I7= 0; I3=I5= 1; I0=I4= D and I6= D'.**

Determine the Boolean function that the multiplexer implements.

**Multiplexer Implementation**



**Implementation table**

	I <sub>0</sub>	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>
$\bar{D}$	0	1	2	3	4	5	6	7
D	8	9	10	11	12	13	14	15
	D	0	0	1	D	1	$\bar{D}$	0

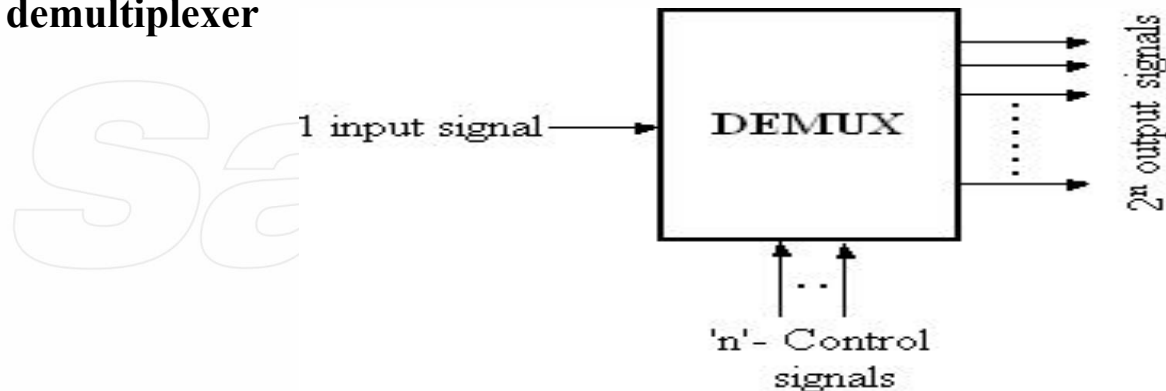
$$F(A, B, C, D) = \sum m(3, 5, 6, 8, 11, 12, 13).$$

## Demultiplexing

Demultiplexing means one into many. Demultiplexing is the process of taking information from one input and transmitting the same over one of several outputs.

A demultiplexer is a combinational logic circuit that receives information on a single input and transmits the same information over one of several ( $2^n$ ) output lines.

### Block diagram of demultiplexer

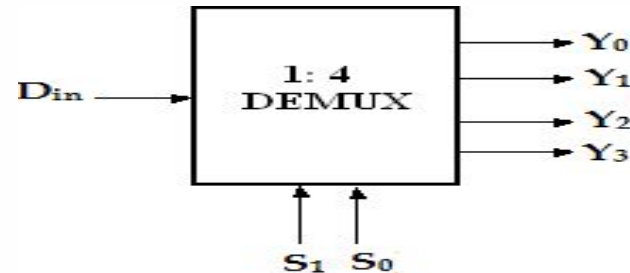


The block diagram of a demultiplexer which is opposite to a multiplexer in its operation is shown above. The circuit has one input signal,  $n$  select signals and  $2^n$  output signals. The select inputs determine to which output the data input will be connected. As the serial data is changed to parallel data, i.e., the input caused to appear on one of the  $n$  output lines, the demultiplexer is also called a —**data distributor** or a —**serial-to-parallel converter**.

## 1-to-4 Demultiplexer

A 1-to-4 demultiplexer has a single input, **D<sub>in</sub>**, four outputs (**Y<sub>0</sub> to Y<sub>3</sub>**) and two select inputs (**S<sub>1</sub> and S<sub>0</sub>**).

Logic Symbol



The input variable  $D_{in}$  has a path to all four outputs, but the input information is directed to only one of the output lines.

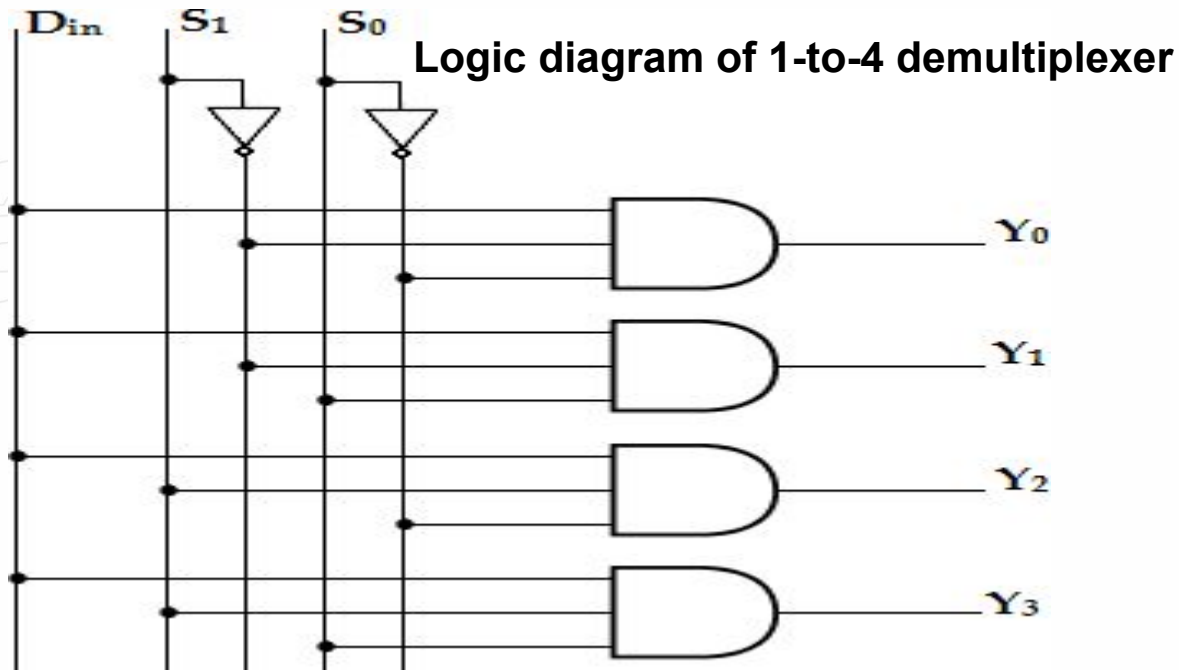
The truth table of the 1-to-4 demultiplexer is shown below.

Enable	S <sub>1</sub>	S <sub>0</sub>	D <sub>in</sub>	Y <sub>0</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>
0	x	x	x	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

Truth table of 1-to-4 demultiplexer

From the truth table, it is clear that the data input,  $D_{in}$  is connected to the output  $Y_0$ , when  $S_1 = 0$  and  $S_0 = 0$  and the data input is connected to output  $Y_1$  when  $S_1 = 0$  and  $S_0 = 1$ . Similarly, the data input is connected to output  $Y_2$  and  $Y_3$  when  $S_1 = 1$  and  $S_0 = 0$  and when  $S_1 = 1$  and  $S_0 = 1$ , respectively. Also, from the truth table, the expression for outputs can be written as follows,

- $Y_0 = S_1' S_0' D_{in}$
- $Y_1 = S_1' S_0 D_{in}$
- $Y_2 = S_1 S_0' D_{in}$
- $Y_3 = S_1 S_0 D_{in}$



Now, using the above expressions, a 1-to-4 demultiplexer can be implemented using four 3-input AND gates and two NOT gates. Here, the input data line  $D_{in}$ , is connected to all the AND gates. The two select lines  $S_1$ ,  $S_0$  enable only one gate at a time and the data that appears on the input line passes through the selected gate to the associated output line.

## 1-to-8 Demultiplexer

A 1-to-8 demultiplexer has a single input,  $D_{in}$ , eight outputs ( $Y_0$  to  $Y_7$ ) and three select inputs ( $S_2$ ,  $S_1$  and  $S_0$ ). It distributes one input line to eight output lines based on the select inputs. The truth table of 1-to-8 demultiplexer is shown below.

$D_{in}$	$S_2$	$S_1$	$S_0$	$Y_7$	$Y_6$	$Y_5$	$Y_4$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

From the above truth table, it is clear that the data input is connected with one of the eight outputs based on the select inputs. Now from this truth table, the expression for eight outputs can be written as follows:

$$Y_0 = S_2'S_1'S_0'D_{in}$$

$$Y_4 = S_2 S_1'S_0'D_{in}$$

$$Y_1 = S_2'S_1'S_0D_{in}$$

$$Y_5 = S_2 S_1'S_0D_{in}$$

$$Y_2 = S_2'S_1S_0'D_{in}$$

$$Y_6 = S_2 S_1S_0'D_{in}$$

$$Y_3 = S_2'S_1S_0D_{in}$$

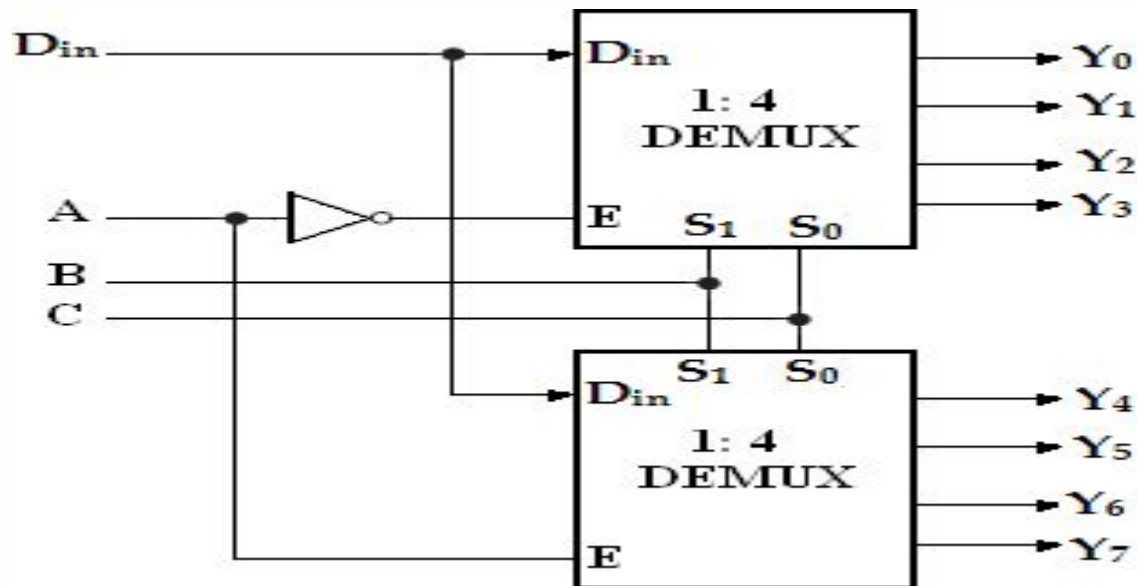
$$Y_7 = S_2S_1S_0D_{in}$$

- Now using the above expressions, the logic diagram of a 1-to-8 demultiplexer can be drawn as shown below.
- Here, the single data line,  $D_{in}$  is connected to all the eight AND gates, but only one of the eight AND gates will be enabled by the select input lines. For example, if  $S_2S_1S_0 = 000$ , then only AND gate-0 will be enabled and thereby the data input,  $D_{in}$  will appear at  $Y_0$ .
- Similarly, the different combinations of the select inputs, the input  $D_{in}$  will appear at the respective output.





1. Design 1:8 demultiplexer using two 1:4 DEMUX.



## 2. Implement full subtractor using demultiplexer.

Inputs			Outputs	
A	B	B <sub>in</sub>	Difference(D)	Borrow(B <sub>out</sub> )
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

