4.5 Multiplexers

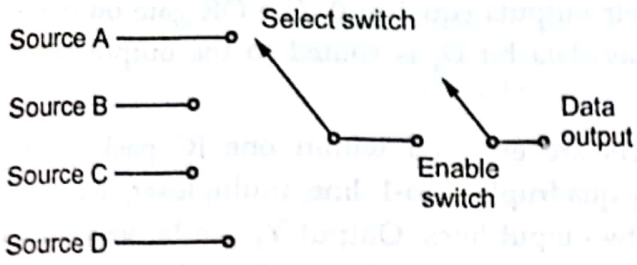
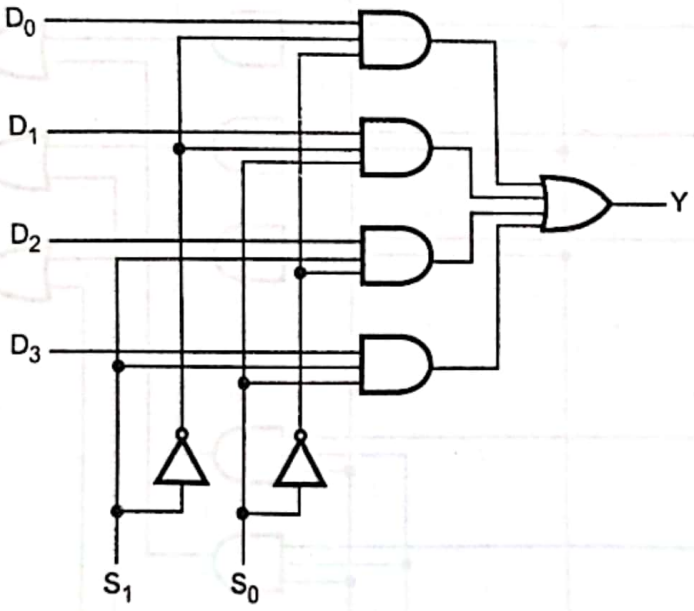


Fig. 4.46 Analog selector switch

Multiplexer is a digital switch. It allows digital information from several sources to be routed onto a single output line, as shown in the Fig. 4.46. The basic multiplexer has several data-input lines and a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected. Therefore, multiplexer is 'many into one' and it provides the digital equivalent of an analog selector switch.

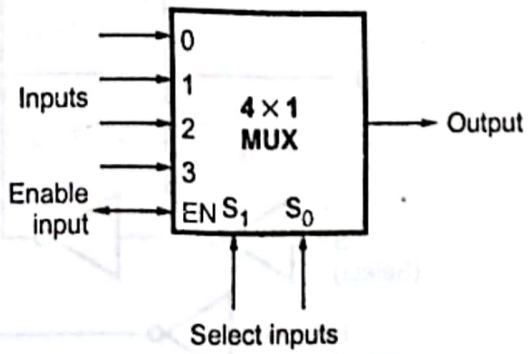
Fig. 4.47 (a) shows 4-to-1 line multiplexer. Each of the four lines, D_0 to D_3, is applied to one input of an AND gate. Selection lines are decoded to select a particular AND gate.



(a) Logic diagram

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

(b) Function table



(c) Logic symbol

Fig. 4.47 4-to-1 line multiplexer

For example, when S_1S_0 = 0 1, the AND gate associated with data input D_1 has two of its inputs equal to 1 and the third input connected to D_1. The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The OR gate output is now equal to the value of D_1, thus we can say data bit D_1 is routed to the output when S_1 S_0 = 0 1.

In some cases, two or more multiplexers are enclosed within one IC package, as shown in the Fig. 4.48. The Fig. 4.48 shows quadruple 2-to-1 line multiplexer, i.e. four multiplexers, each capable of selecting one of two input lines. Output Y_1 can be selected to be equal to either A_1 or B_1. Similarly output Y_2 may have the value of A_2 or B_2, and so on. The selection line S selects one of two lines in all four multiplexers. The control input E enables the multiplexers in the 0 state and disables them in the 1 state. When E = 1, outputs have all 0's, regardless of the value of S.

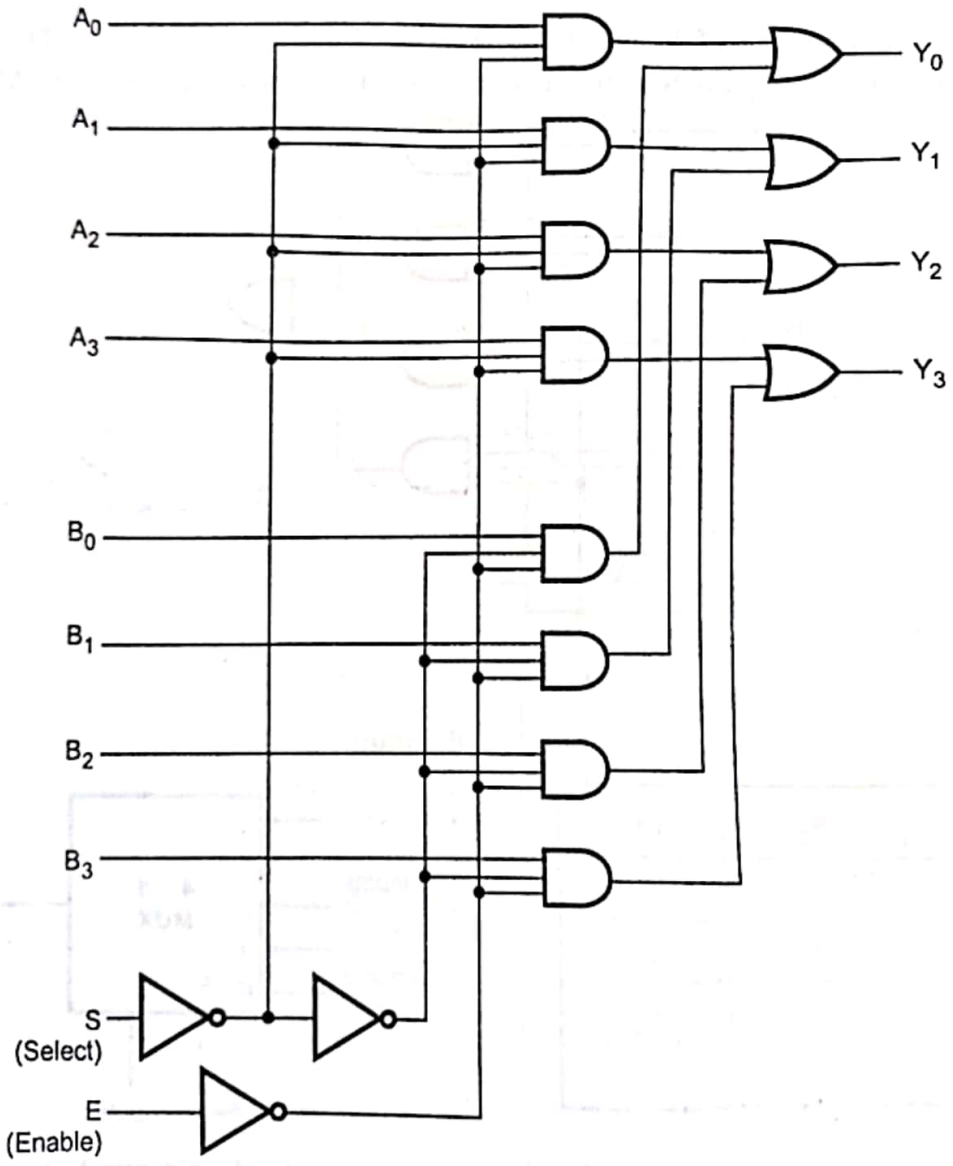


Fig. 4.48 Quadruple 2-to-1 line multiplexer

Function Table

E	S	Output Y
1	X	All 0s
0	0	Select A
0	1	Select B

4.5.1 The 74XX151 8 to 1 Multiplexer

The 74XX151 is a 8-to-1 multiplexer. It has eight inputs. It provides two outputs, one is active high, the other is active low. The Fig. 4.49 (b) shows the logic symbol for 74XX151. As shown in the logic symbol, there are three select inputs C, B and A which select one of the eight inputs. The 74XX151 is provided with active low enable input.

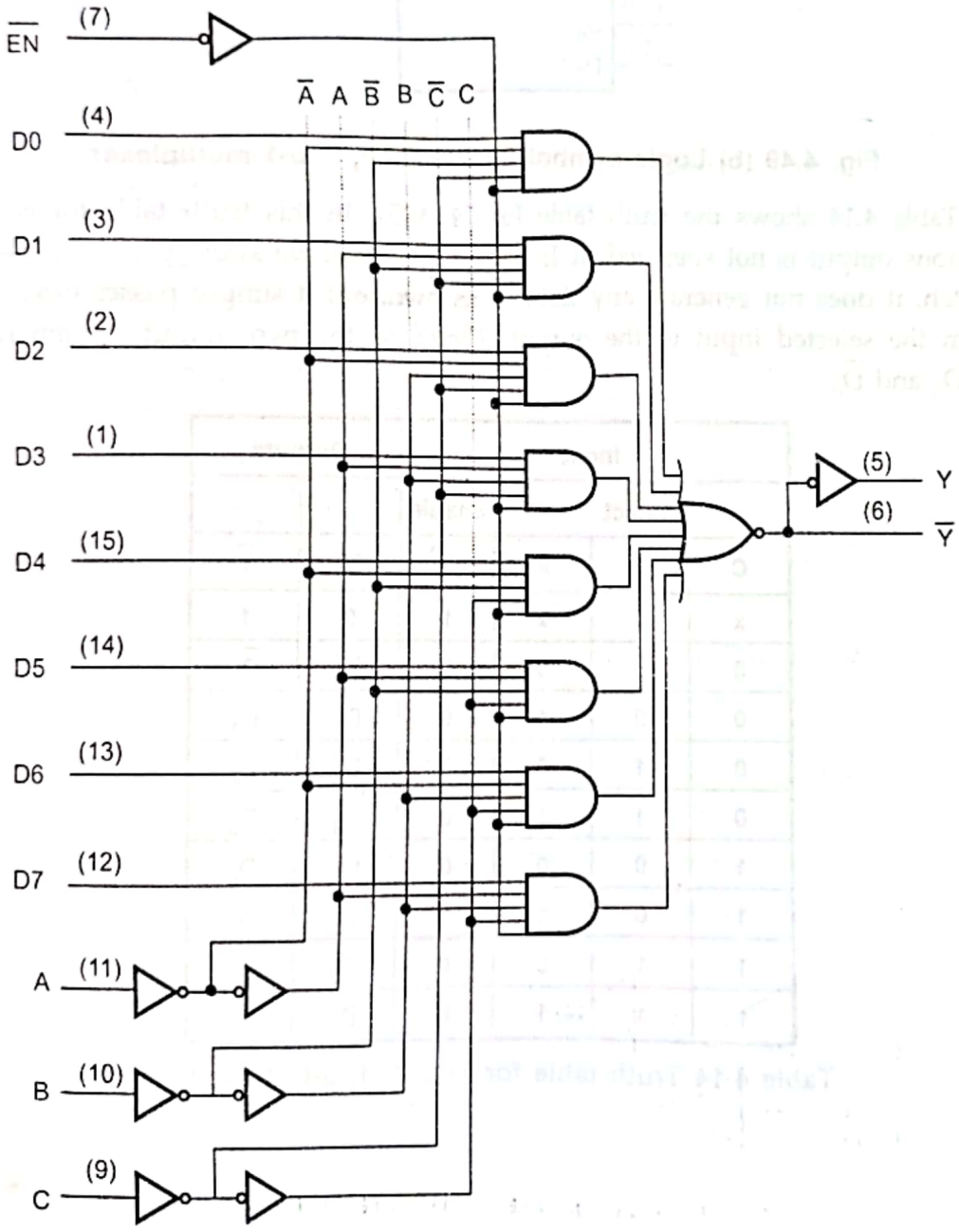


Fig. 4.49 (a) Logic diagram for IC74XX151

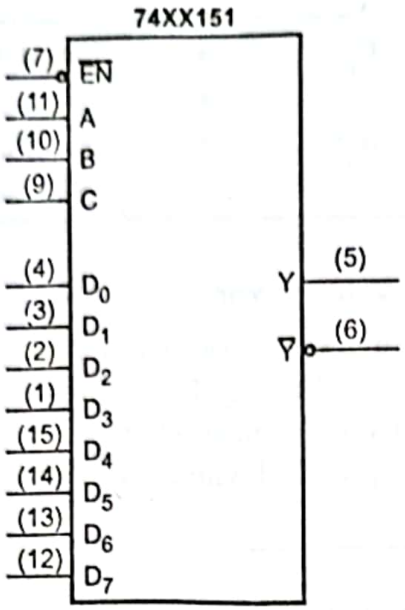


Fig. 4.49 (b) Logic symbol for 74XX151, 8-to-1 multiplexer

The Table 4.14 shows the truth table for 74XX151. In this truth table for each input combinations output is not specified in 1s and 0s. Because, we know that, multiplexer is a data switch, it does not generate any data of its own, but it simply passes external input data from the selected input to the output. Therefore, the two output column represent data by D_n and $\overline{D}_n$.

Input				Outputs	
Select			Enable		
C	B	A	$\overline{EN}$	Y	$\overline{Y}$
x	x	x	1	0	1
0	0	0	0	D_0	$\overline{D}_0$
0	0	1	0	D_1	$\overline{D}_1$
0	1	0	0	D_2	$\overline{D}_2$
0	1	1	0	D_3	$\overline{D}_3$
1	0	0	0	D_4	$\overline{D}_4$
1	0	1	0	D_5	$\overline{D}_5$
1	1	0	0	D_6	$\overline{D}_6$
1	1	1	0	D_7	$\overline{D}_7$

Table 4.14 Truth table for 74XX151, 8-to-1 multiplexer

4.5.2 The 74XX157 Quad 2-Input Multiplexer

The IC 74XX157 is a quad 2-input multiplexer which selects four bits of data from two sources under the control of a common select input (S). The Enable input ($\overline{E}$) is active low. When $\overline{E}$ is high, all of the outputs (Y) are forced low regardless of all other input condititons.

Moving data from two groups of register to four common output buses is a common use of IC 74XX157. The state of the select input determines the particular register from which the data comes. Fig. 4.50 shows logic symbol for IC 74XX157.

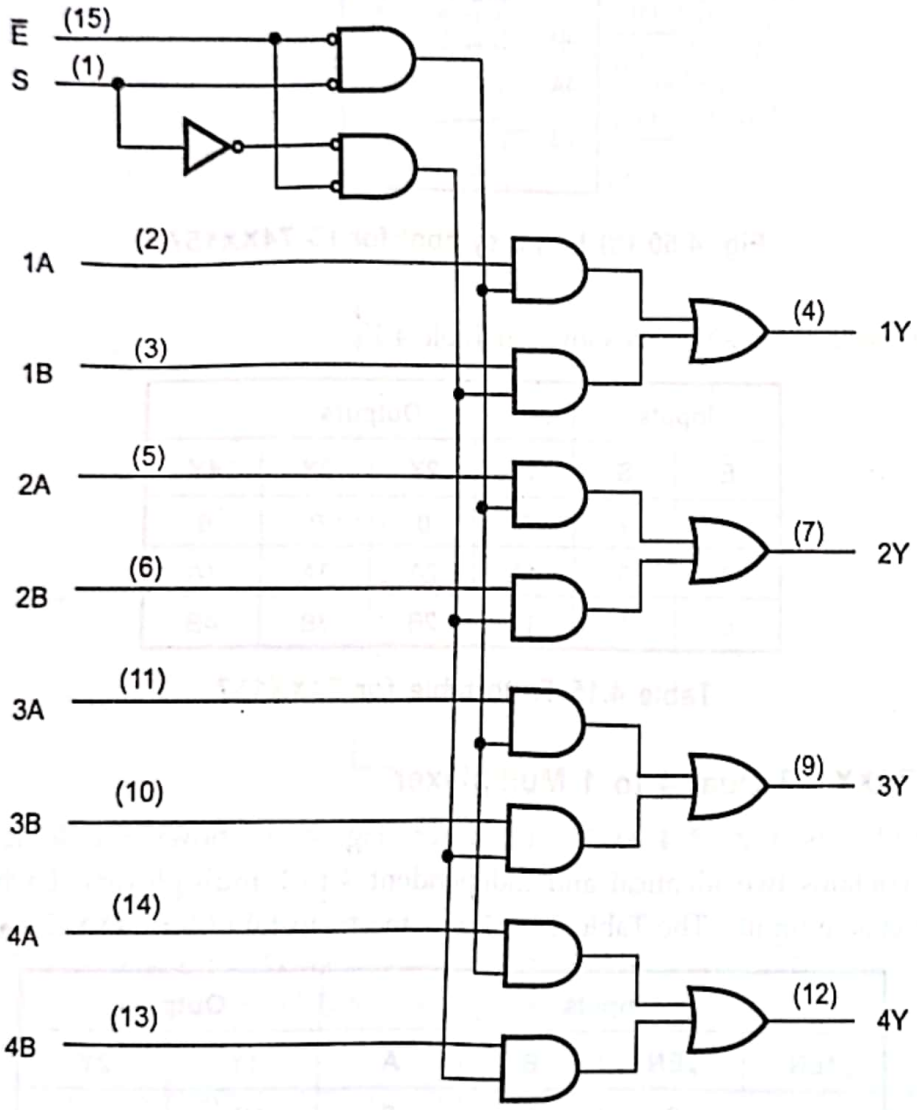


Fig. 4.50 (a) Logic diagram for IC 74XX157

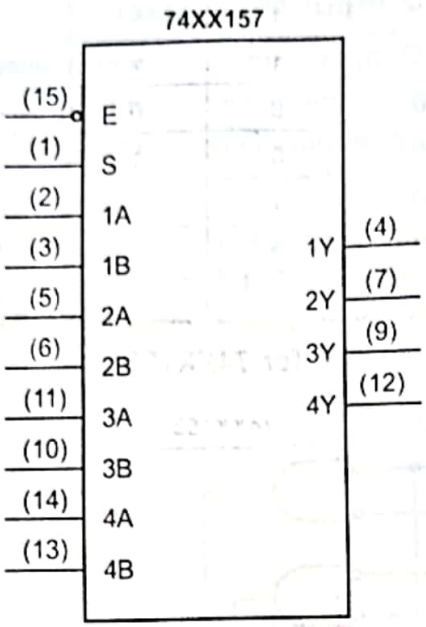


Fig. 4.50 (b) Logic symbol for IC 74XX157

The truth table for 74XX157 is shown in Table 4.15.

Inputs		Outputs			
$\overline{E}$	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

Table 4.15 Truth table for 74XX157

4.5.3 The 74XX153 Dual 4 to 1 Multiplexer

The 74XX153 is a dual 4 to 1 multiplexer. Fig. 4.51 shows the logic symbol for 74XX153. It contains two identical and independent 4-to-1 multiplexers. Each multiplexer has separate enable inputs. The Table 4.16 shows the truth table for 74XX153.

Inputs				Outputs	
1EN	2EN	B	A	1Y	2Y
0	0	0	0	$1D_0$	$2D_0$
0	0	0	1	$1D_1$	$2D_1$
0	0	1	0	$1D_2$	$2D_2$
0	0	1	1	$1D_3$	$2D_3$
0	1	0	0	$1D_0$	0
0	1	0	1	$1D_1$	0

0	1	1	0	$1D_2$	0
0	1	1	1	$1D_3$	0
1	0	0	0	0	$2D_0$
1	0	0	1	0	$2D_1$
1	0	1	0	0	$2D_2$
1	0	1	1	0	$2D_3$
1	1	x	x	0	0

Table 4.16 Truth table for 74XX153, dual 4-to-1 multiplexer

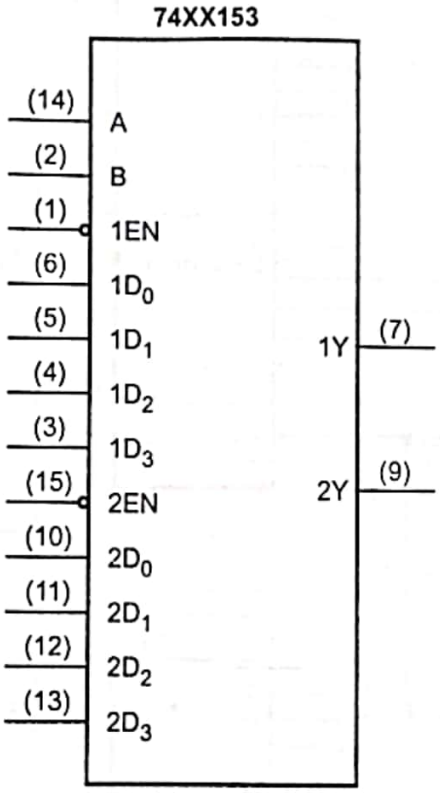


Fig. 4.51 Logic symbol for 74XX153

4.5.4 Expanding Multiplexers

Several digital multiplexer ICs are available such as 74150 (16-to-1), 74151 (8-to-1), 74157 (Dual 2 input) and 74153 (Dual 4 to 1) multiplexer. It is possible to expand the range of inputs for multiplexer beyond the available range in the integrated circuits. This can be accomplished by interconnecting several multiplexers. For example, two 74XX151, 8-to-1 multiplexers can be used together to form a 16-to-1 multiplexer, two 74XX150, 16-to-1 multiplexers can be used together to form a 32-to-1 multiplexer and so on.

Example 4.13 : *Design 32-to-1 multiplexer using two 74LS150.*

Solution : Fig. 4.52 shows a 32-to-1 multiplexer using two 74LS150 ICs.

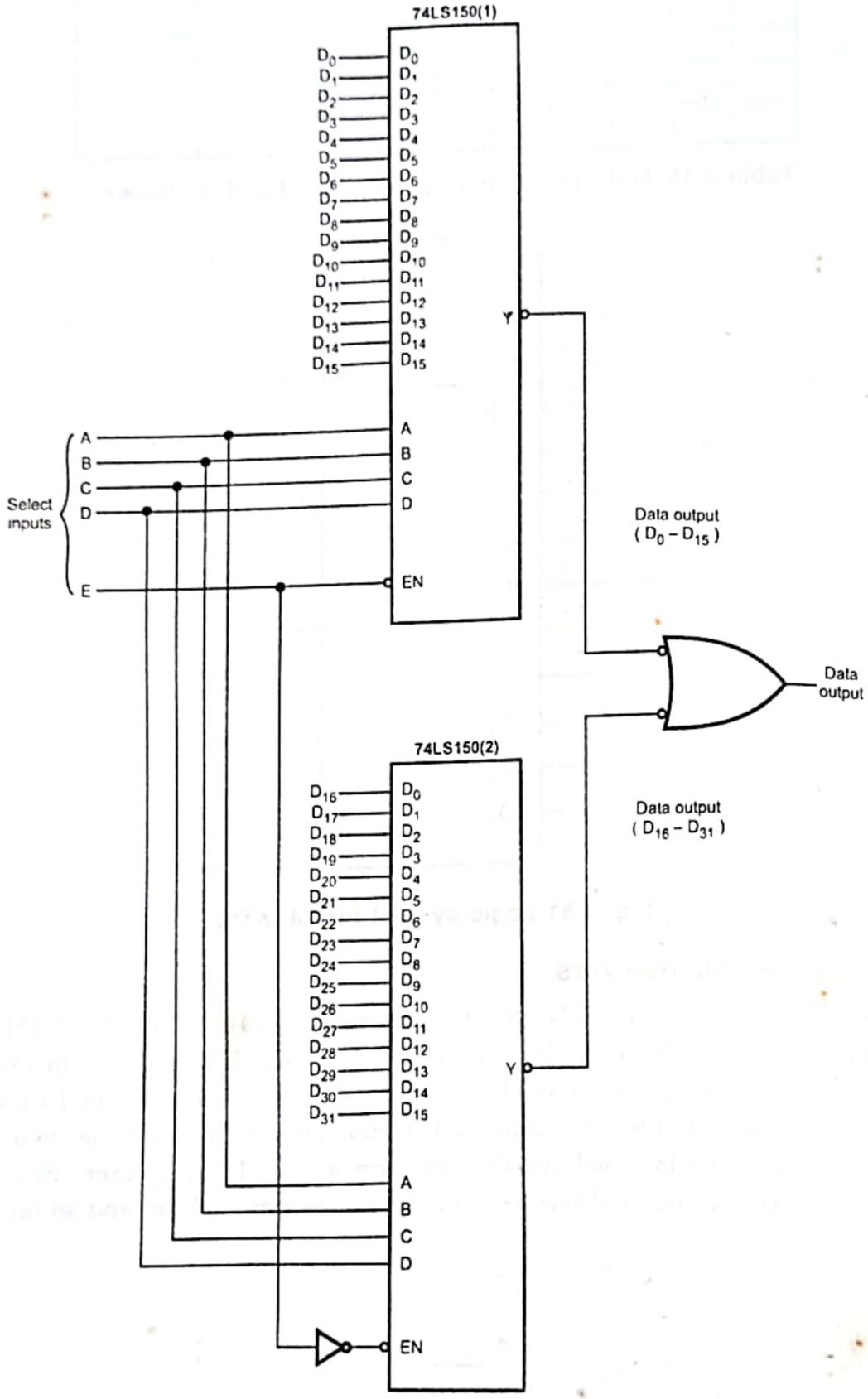


Fig. 4.52 32-to-1 multiplexer using two 74LS150 ICs

Example 4.14 : *Design 32-to-1 multiplexer using four 8-to-1 multiplexers and 2-to-4 decoder.*

Solution : Fig. 4.53 shows the 32-to-1 multiplexer using four 8-to-1 multiplexer and one 2-to-4 decoder.

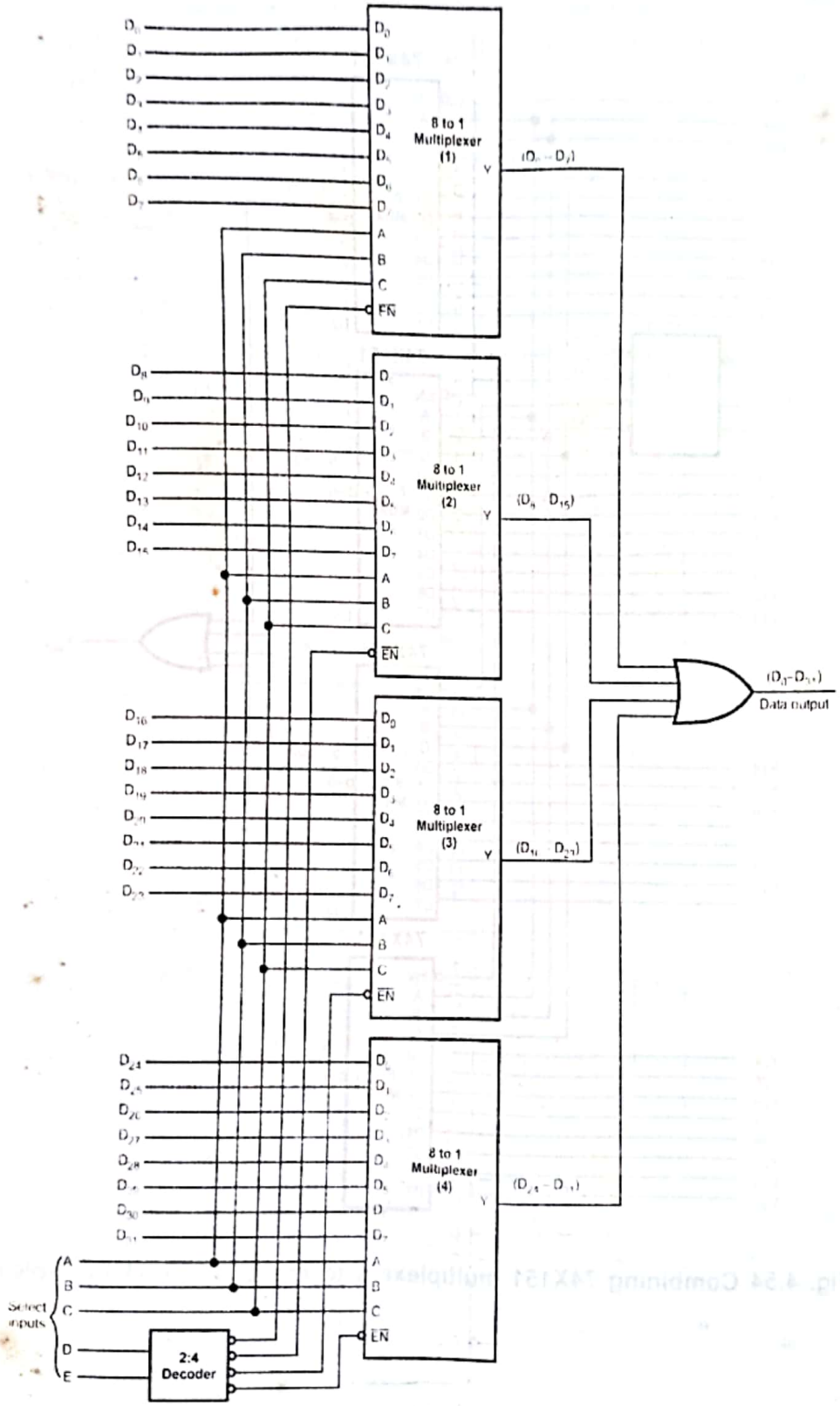


Fig. 4.53

Example 4.15 : *Design a 32-to-1 multiplexer using four 74X151 multiplexers and one 74X139 decoder.*

Solution : The output of multiplexer (74X151) is given to OR gate to obtain the final result. The circuit diagram of a 32-to-1 MUX using four 74X151 and one 74X139 is as shown in figure below.

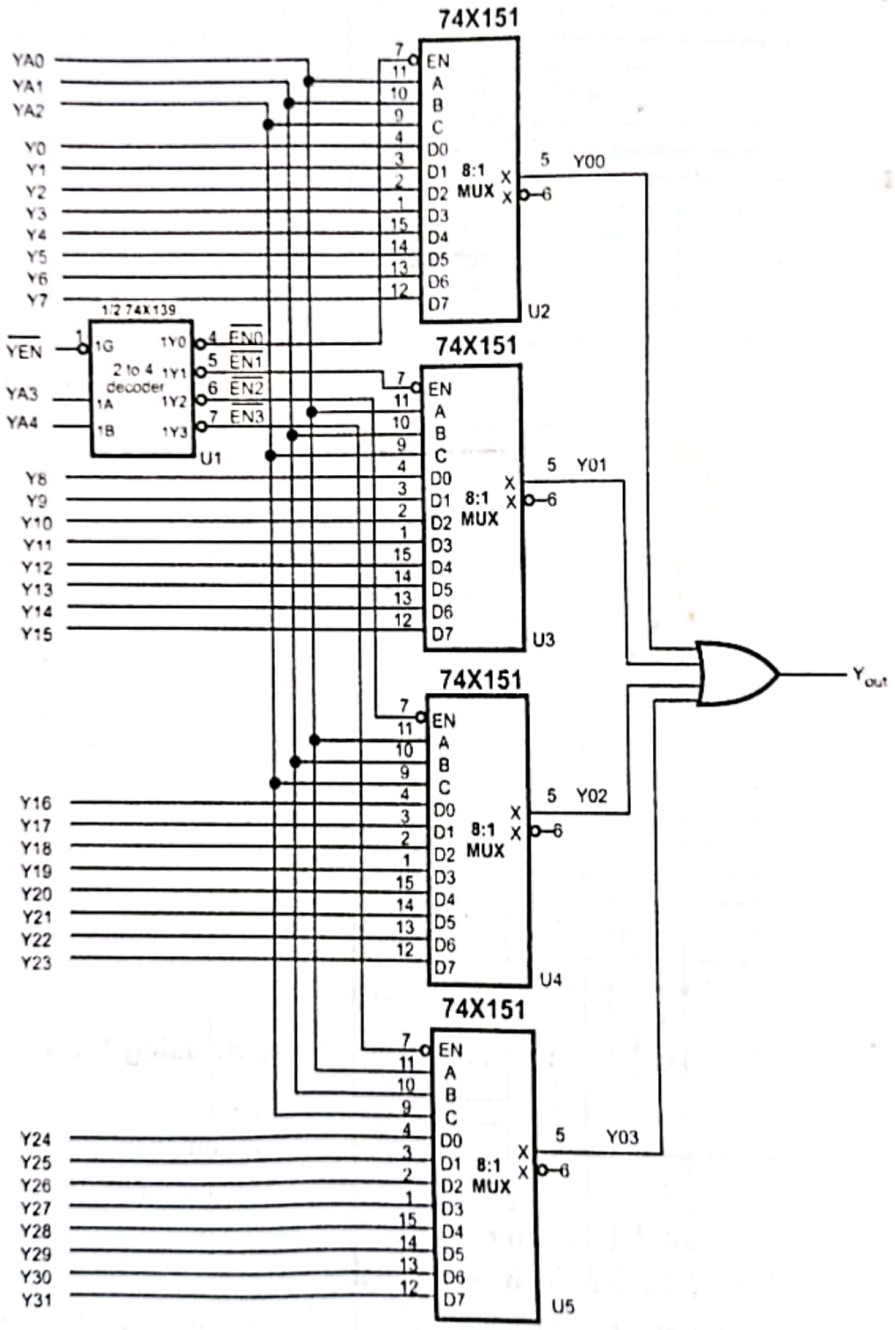


Fig. 4.54 Combining 74X151 multiplexers to make a 32-to-1 multiplexer

4.5.5 Implementation of Combinational Logic using MUX

A multiplexer consists of a set of AND gates whose outputs are connected to single OR gate. Because of this construction any Boolean function in a SOP form can be easily realized using multiplexer. Each AND gate in the multiplexer represents a minterm. In 8-to-1 multiplexer, there are 3 select inputs and 23 minterms. By connecting the function variables directly to the select inputs, a multiplexer can be made to select the AND gate that corresponds to the minterm in the function. If a minterm exists in a function, we have to connect the AND gate data input to logic 1; otherwise we have to connect it to logic 0. This is illustrated in the following example.

Example 4.16 : *Implement the following Boolean function using 8 : 1 multiplexer.*
$F(A, B, C) = \sum m(1, 3, 5, 6)$

Solution : The function can be implemented with a 8-to-1 multiplexer, as shown in Fig. 4.55. Three variables A, B and C are applied to the select lines. The minterms to be included (1, 3, 5 and 6) are chosen by making their corresponding input lines equal to 1. Minterms 0, 2, 4 and 7 are not included by making their input lines equal to 0.

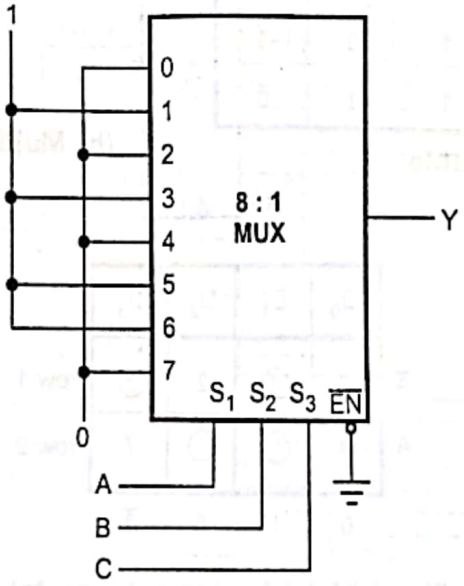


Fig. 4.55 Boolean function implementation using MUX

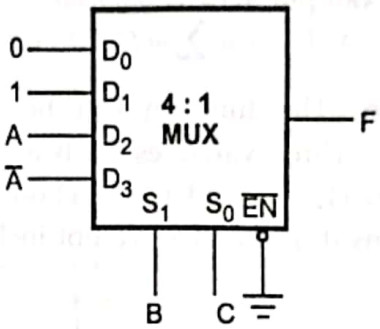
In the above example we have seen the method for implementing Boolean function of 3 variables with 2^3 (8) - to -1 multiplexer. Similarly, we can implement any Boolean function of n variables with 2^n-to-1 multiplexer. However, it is possible to do better than this. If we have Boolean function of n + 1 variables, we take n of these variables and connect them to the selection lines of a multiplexer. The remaining single variable of the function is used for the inputs of the multiplexer. In this way we can implement any Boolean function of n variables with 2^{n-1}- to-1 multiplexer. Let us see one example.

Example 4.17 : *Implement the following Boolean function using 4 : 1 multiplexer.*
$F(A, B, C) = \sum m(1, 3, 5, 6)$

Solution : Fig. 4.56 shows the implementation of function with 4-to-1 multiplexer. As mentioned earlier, here, two of the variables, B and C, are applied to the selection lines. B is connected to S_1 and C is connected to S_0. The inputs for multiplexer are derived from the implementation table.

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(a) Truth table



(b) Multiplexer implementation

Fig. 4.56

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

Fig. 4.56 (c) Implementation table

As shown in the Fig. 4.56 (c) the implementation table is nothing but the list of the inputs of the multiplexer and under them list of all the minterms in two rows. The first row lists all those minterms where A is complemented, and the second row lists all the minterms with A uncomplemented. The minterms given in the function are circled and then each column is inspected separately as follows :

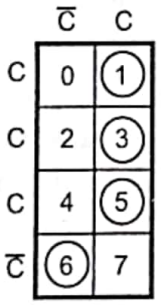
- If the two minterms in a column are not circled, 0 is applied to the corresponding multiplexer input (see column 0).
- If the two minterms in a column are circled, 1 is applied to the corresponding multiplexer input (see column 1).

- If the minterm in the second row is circled and minterm in the first row is not circled, A is applied to the corresponding multiplexer input (see column 2).
- If the minterm in the first row is circled and minterm in the second row is not circled, $\overline{A}$ is applied to the corresponding multiplexer input (see column 3).

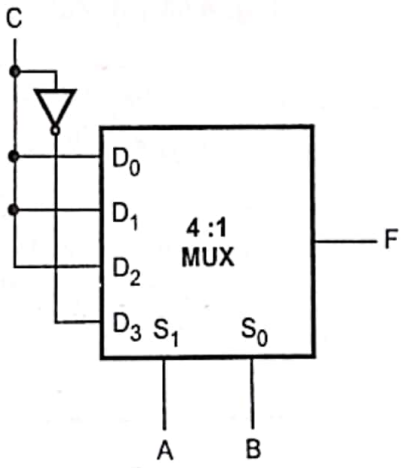
In the previous multiplexer implementation two least significant variables, i.e. B and C are connected to the select lines of the multiplexer. The multiplexer implementation is also possible by connecting most significant variables i.e. A and B (in above case) to the select lines of the multiplexer. The procedure for same is explained below :

Here, implementation table consists of two columns. The first column lists all the where least significant variable is complemented ($\overline{C}$ in this case), and the second column lists all the minterms with least significant variable is uncomplemented (C in this case). The minterms given in the function are circled and then each row is inspected separately as follows :

- If the two in a row are not circled, 0 is applied to corresponding multiplexer input.
- If the two in a row are circled, 1 is applied to corresponding multiplexer input.
- If the minterm in the column 1 is circled, least significant variable is complemented ($\overline{C}$ in this case) and applied to the corresponding multiplexer input.
- If the minterm in the column 2 is circled, least significant variable (C in this case) is applied to the corresponding multiplexer input.



(a) Implementation table



(b) Multiplexer Implementation

Fig. 4.57

Example 4.18 : *Implement the following Boolean function using 8 :1 MUX.*

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

Solution : Fig. 4.58 shows the implementation of given Boolean function with 8 : 1 multiplexer.

	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

Fig. 4.58 (a) Implementation table

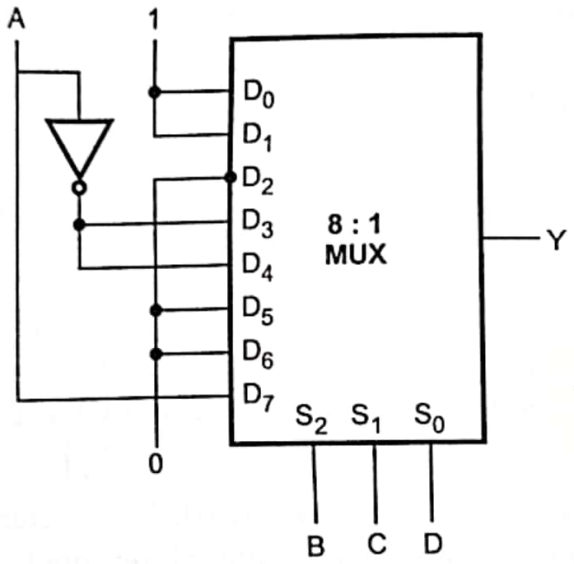


Fig. 4.58 (b) Multiplexer Implementation

Example 4.19 : *Implement the following Boolean function using 4 : 1 MUX*
$F(A, B, C, D) = \sum m\ (0, 1, 2, 4, 6, 9, 12, 14)$

Solution : The function has four variables. To implement this function we require 8 : 1 multiplexer. i.e., two 4 : 1 multiplexers. We have already seen how to construct 8 : 1 multiplexer using two 4 : 1 multiplexers. The same concept is used here to implement given Boolean function.

	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

Fig. 4.59 (a) Implementation table

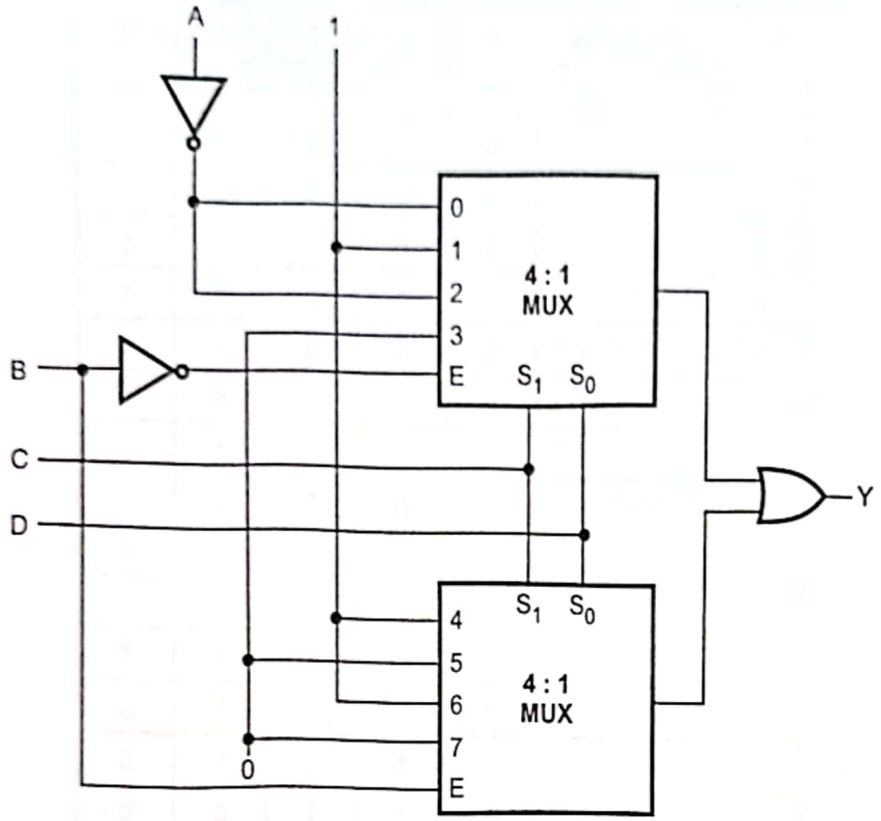


Fig. 4.59 (b) Implementation using two 4 : 1 multiplexer

➡ **Example 4.20 :** *Implement the following Boolean function using 8 : 1 multiplexer*
$F(A, B, C, D) = \overline{A}\ B\ \overline{D} + A\,C\,D + \overline{B}\ C\ D + \overline{A}\ \overline{C}\ D$

Solution : The given Boolean expression is not in standard SOP form. Let us first convert this in standard SOP form

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

The truth table for this standard SOP form can be given as

No.	Minterms	A	B	C	D	Y
0		0	0	0	0	0
1	$\bar{A}\bar{B}\bar{C}D$	0	0	0	1	1
2		0	0	1	0	0
3	$\bar{A}\bar{B}CD$	0	0	1	1	1
4	$\bar{A}B\bar{C}\bar{D}$	0	1	0	0	1
5	$\bar{A}B\bar{C}D$	0	1	0	1	1
6	$\bar{A}BC\bar{D}$	0	1	1	0	1
7		0	1	1	1	0
8		1	0	0	0	0
9		1	0	0	1	0
10		1	0	1	0	0
11	$A\bar{B}CD$	1	0	1	1	1
12		1	1	0	0	0
13		1	1	0	1	0
14		1	1	1	0	0
15	$ABCD$	1	1	1	1	1

Table 4.17 Truth table

From the truth table Boolean function can be implemented using 8 : 1 multiplexer as follows :

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

(a) Implementation table

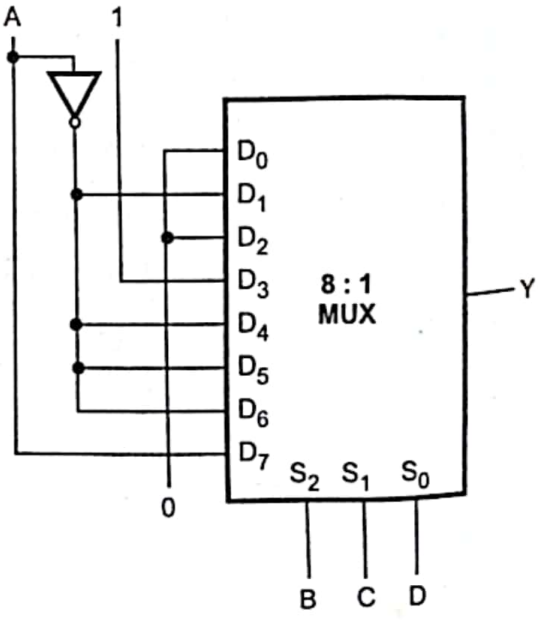


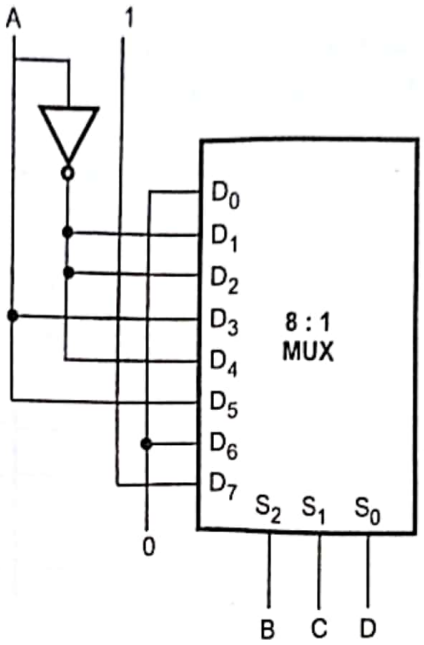
Fig. 4.60

(b) Multiplexer implementation

Example 4.21 : *Implement the following Boolean function with 8 : 1 multiplexer*
$F(A, B, C, D) = \pi M (0, 3, 5, 8, 9, 10, 12, 14)$

Solution : Here, instead of minterms, maxterms are specified. Thus, we have to circle maxterms which are not included in the Boolean function. Fig. 4.61 shows the implementation of Boolean function with 8 :1 multiplexer.

	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)
	0	$\overline{A}$	$\overline{A}$	A	$\overline{A}$	A	0	1



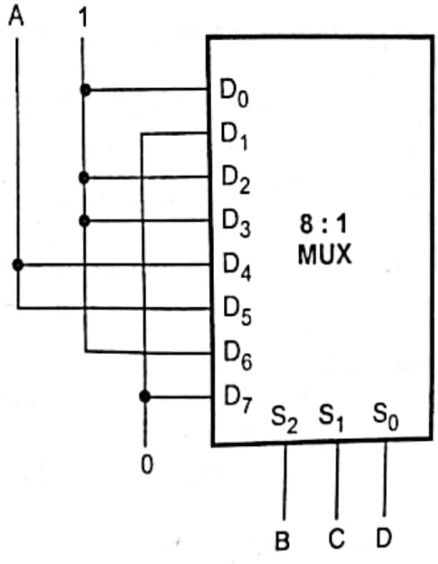
(a) Implementation table Fig. 4.61 (b) Multiplexer implementation

Example 4.22 : *Implement the following Boolean function with 8 : 1 multiplexer.*
$F(A, B, C, D) = \Sigma\, m(0, 2, 6, 10, 11, 12, 13) + d\,(3, 8, 14)$

Solution : In the given Boolean function three don't care conditions are also specified. We know that don't care conditions can be treated as either 0s or 1s. Fig. 4.62 shows the implementation of given Boolean function using 8 : 1 multiplexer.

	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	0	1	1	A	A	1	0

Here, don't cares are treated as 1s



(a) Implementation table Fig. 4.62 (b) Implementation