→ Encoders:-

→ An encoder is a digital circuit that performs the inverse operation of a decoder.

→ An encoder has 2^n (or fewer) input lines and n output lines.

→ The output lines generate the binary code corresponding to the input value.

→ **Octal-to-binary encoder:-**

It has eight inputs, one for each of the octal digits, and three outputs that generate the corresponding binary number.

→ It is assumed that only one input has a value of 1 at any given time; otherwise the circuit has no meaning.

→

inputs								outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

→ The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.

→ Output 'z' is equal to 1 when the input octal digit is 1 or 3 or 5 or 7.

→ output 'y' is 1 for octal digits 2, 3, 6, or 7, and output 'x' is 1 for digits 4, 5, 6, or 7.

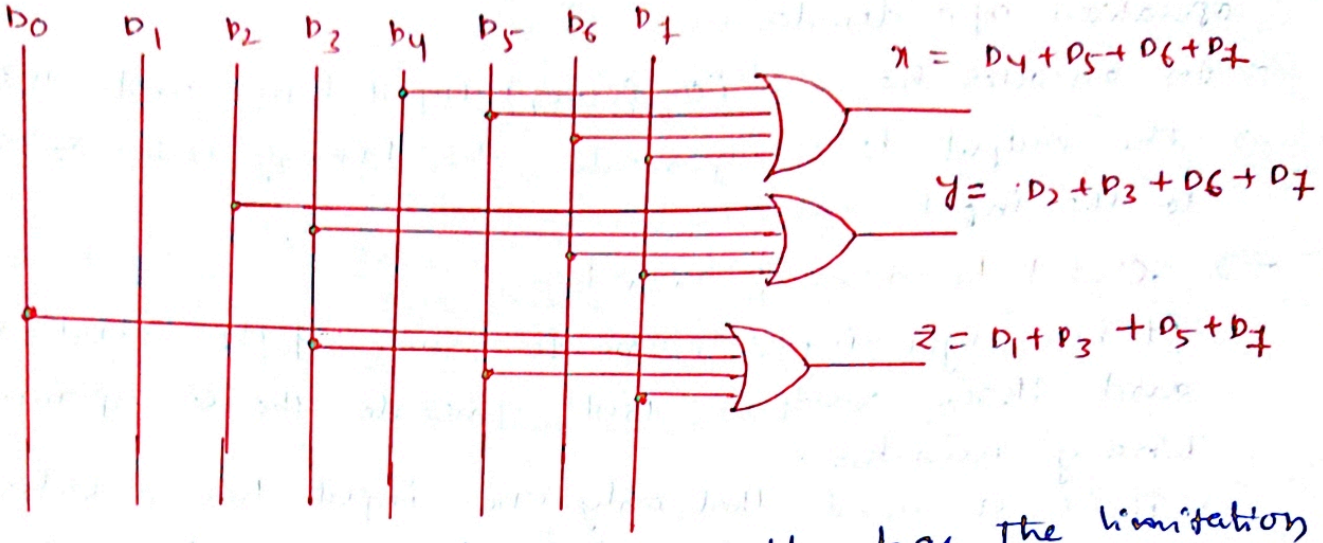
→ These conditions can be expressed by the following output Boolean functions.

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

→ The encoder is implemented with three OR gates as shown below.



→ The encoder defined in the table has the limitation that only one input can be active at any given time.

→ If two inputs are active simultaneously, the output produces an undefined ~~condition~~ combination.

→ For example, if D_3 and D_6 are 1 simultaneously, the output of the encoder will be 111 because all three outputs are equal to 1.

→ This does not represent binary 3 nor binary 6.

→ To resolve this ambiguity, encoder circuits must establish a priority to ensure that only one input is encoded.

→ If we establish a higher priority for inputs with higher subscript numbers, and if both D_3 and D_6 are 1 at the same time, the output will be 110 because D_6 has higher priority than D_3.

→ Another ambiguity in the octal-to-binary encoder is that an output with all 0's is generated, when all the inputs are '0'.

→ The problem is that an output with all 0's is also generated when D_0 is equal to 1.

→ This ambiguity can be resolved by providing an additional output that specifies the condition that none

of the inputs are active.

→ **Priority encoder:-**

→ A priority encoder is an encoder circuit that includes the priority function.

→ The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedance.

→ The truth table of a four-input priority encoder is given in table below.

→ The 'x' are don't-care conditions that designate the fact that the binary value may be equal either to 0 or 1.

→ Input D_3 has the highest priority; so regardless of the value of the other inputs, when this input is 1, the output for xy is 11 (binary 3). D_2 has the next priority level.

→ The output is 10 if $D_2 = 1$ provided that $D_3 = 0$, regardless of the values of the other two lower-priority inputs.

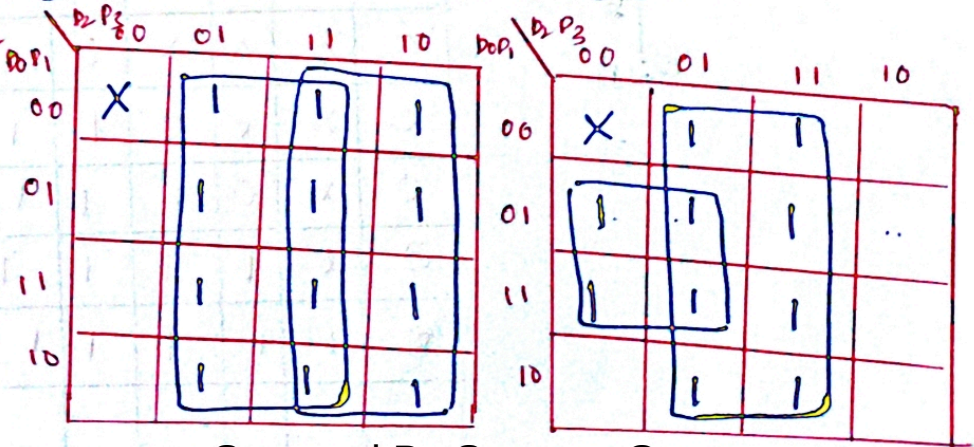
→ The outputs for D_1 is generated only if higher-priority inputs are 0, and so on down the priority level.

→ A valid-output indicator, designated by V, is set to 1 only when one or more of the inputs are equal to 1.

→ If all inputs are 0, V is equal to 0, and the other two outputs of the circuit are not used.

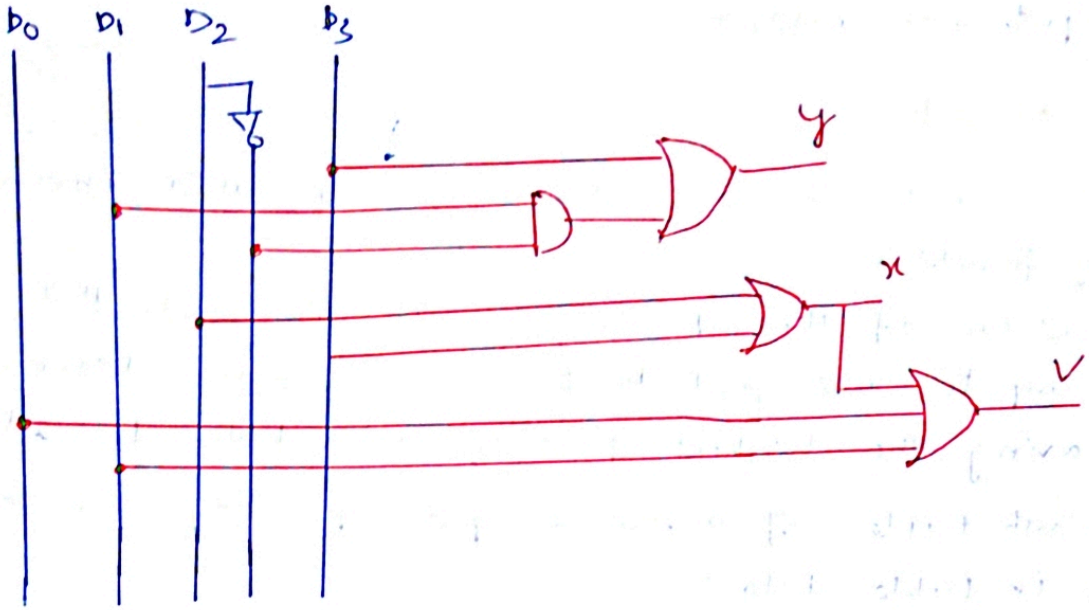
Truth Table of a priority encoder

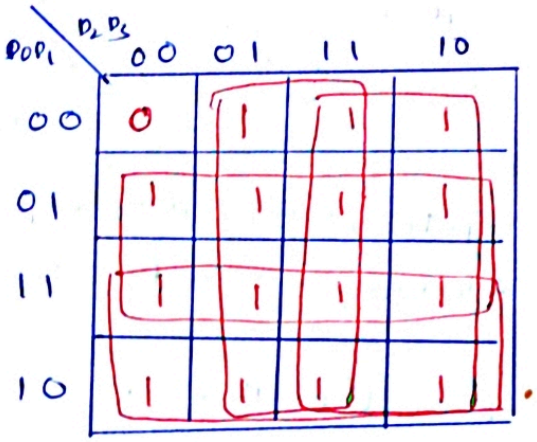
Inputs				outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1



$x = D_3 + D_2$ $y = D_3 + D_1\overline{D_2}$

V =

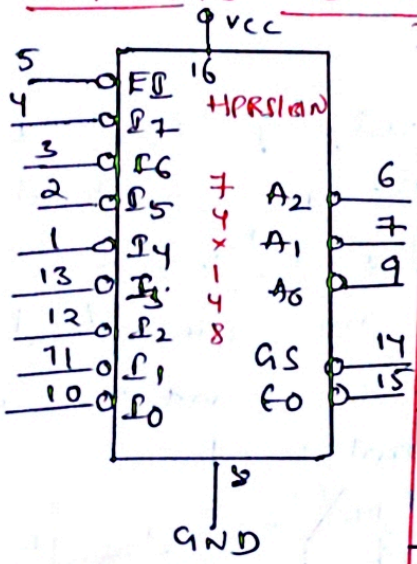




$V = D_1 + D_0 + D_3 + D_2$

→ 74LS148 8-line-to-3-line encoder

Truth-table for 74x148



Inputs									Outputs				
$\overline{EI}$_L	I_0_L	I_1_L	I_2_L	I_3_L	I_4_L	I_5_L	I_6_L	I_7_L	A_2_L	A_1_L	A_0_L	$\overline{GS}$	$\overline{EO}$
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	X	X	X	X	X	X	X	0	0	0	0	0	0
0	X	X	X	X	X	X	0	1	0	0	1	0	0
0	X	X	X	X	X	0	1	1	0	1	0	0	0
0	X	X	X	X	0	1	1	1	0	1	1	0	0
0	X	X	X	0	1	1	1	1	1	0	0	0	0
0	X	X	0	1	1	1	1	1	1	0	1	0	0
0	X	0	1	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0

→ The 74LS148 is a priority encoder that has eight active-Low inputs and three active-Low binary outputs as shown in fig.

→ This device can be used for converting octal inputs to a 3-bit binary code.

→ To enable the device, the $\overline{EI}$ (enable input) must be LOW.

→ It also has the $\overline{EO}$ (enable output) and GS output for expansion purposes.

→ The $\overline{EO}$ is LOW when the $\overline{EI}$ is LOW and none of the inputs (0 through 7) is active.

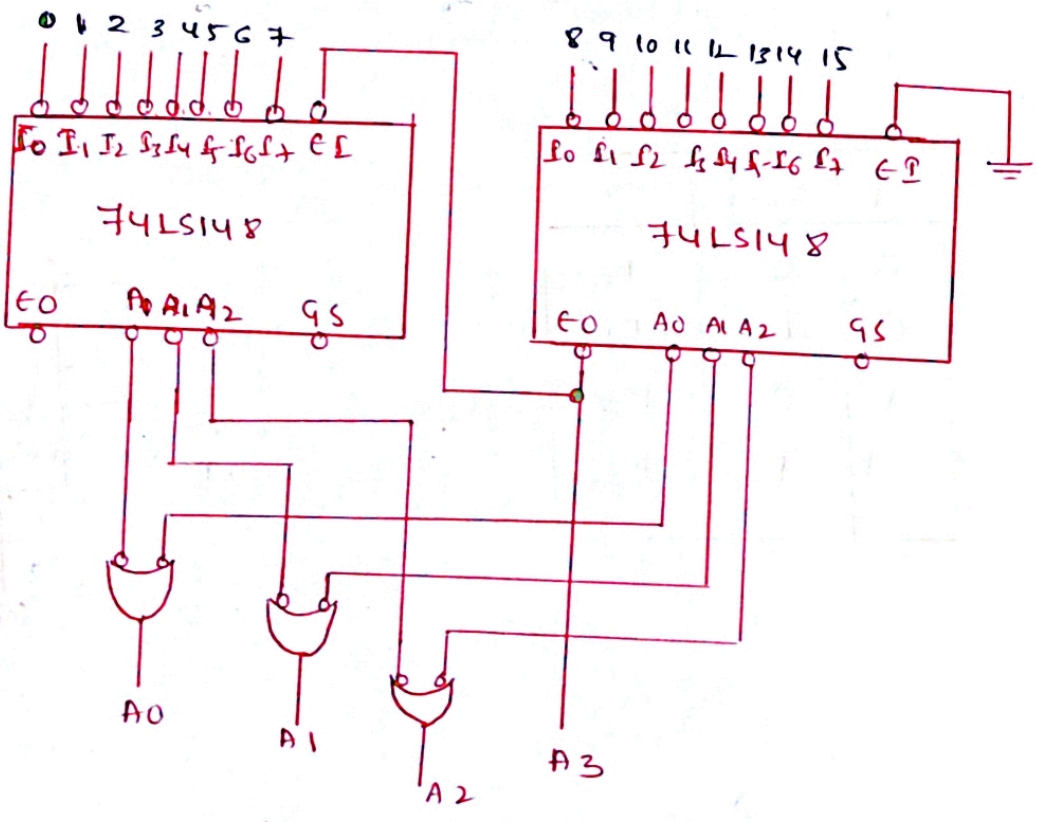
→ GS is LOW when $\overline{EI}$ is LOW and any of the inputs is active.

→ The 74LS148 can be expanded to a 16-line-to-4-line encoder by connecting the $\overline{EO}$ of the higher-order encoder to the $\overline{EI}$ of the lower-order encoder and negative-ORing the corresponding binary outputs as shown in below fig.

→ The EO is used as the fourth and most-significant bit.

→ This particular configuration produces active-HIGH outputs for the 4-bit binary number.

→



Binary outputs

→ Multiplexers (Data selectors) :-

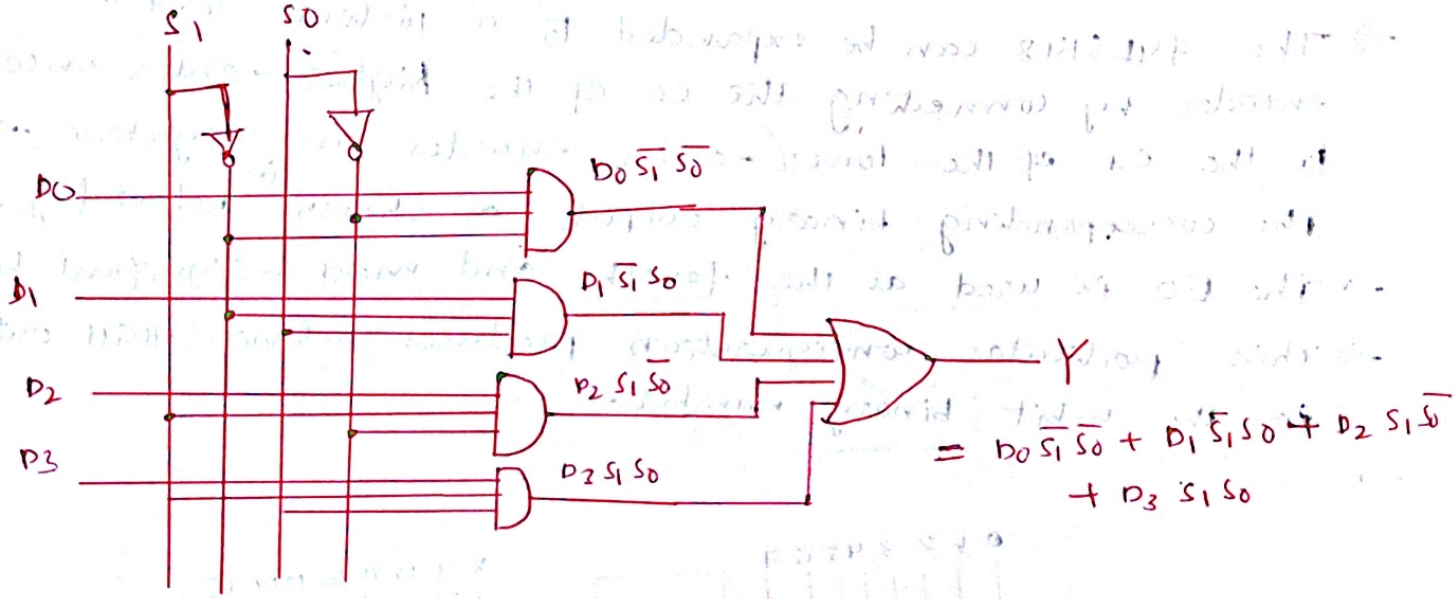
→ A multiplexer (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.

→ The basic multiplexer has several data-input lines and a single output line.

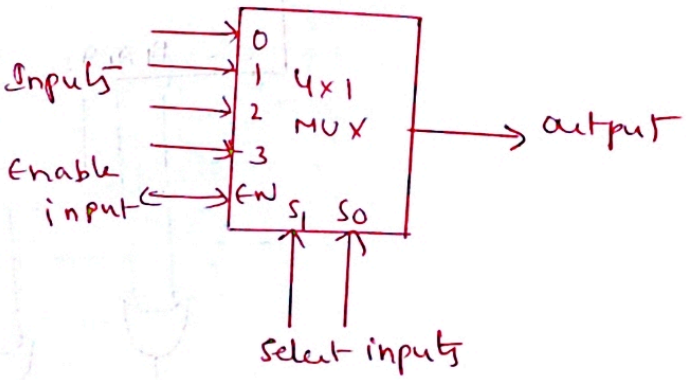
→ It also has data-select inputs, which permit digital data on any one of the inputs to be switched to the output line.

→ Multiplexers are also known as data selectors.

→ **4×1 MUX** - (4-to-1 line multiplexer)



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



→ Standard MSI multiplexer

→ The 74x151 8 to 1 multiplexer.

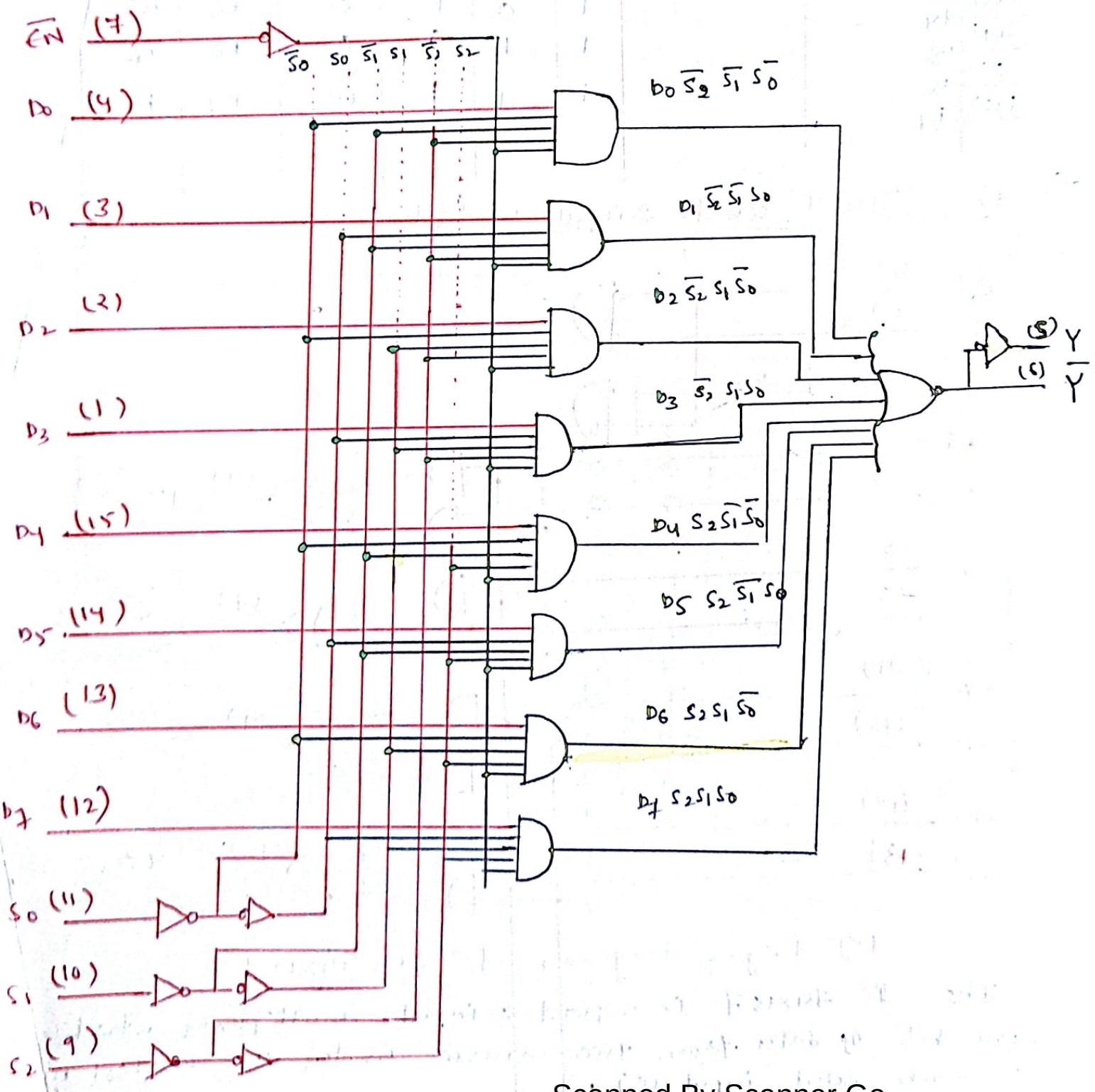
→ The 74x151 is a 8-to-1 multiplexer.

→ It has eight inputs.

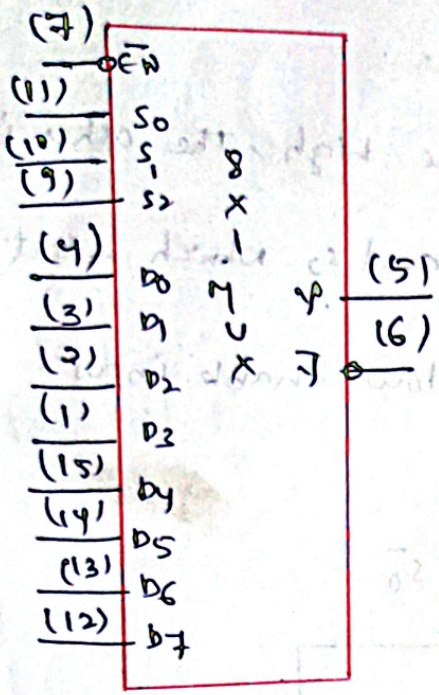
→ It provides two outputs, one is active high, the other is active low.

→ There are three select inputs S_0, S_1, and S_2 which select one of the eight inputs.

→ The 74x151 is provided with active low enable input.



74X151



Input-				outputs	
Select			Enable		
S_2	S_1	S_0	$\overline{EN}$	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	$\overline{D_7}$

→ The 74X157 Quad-2-input Multiplexer:-

→

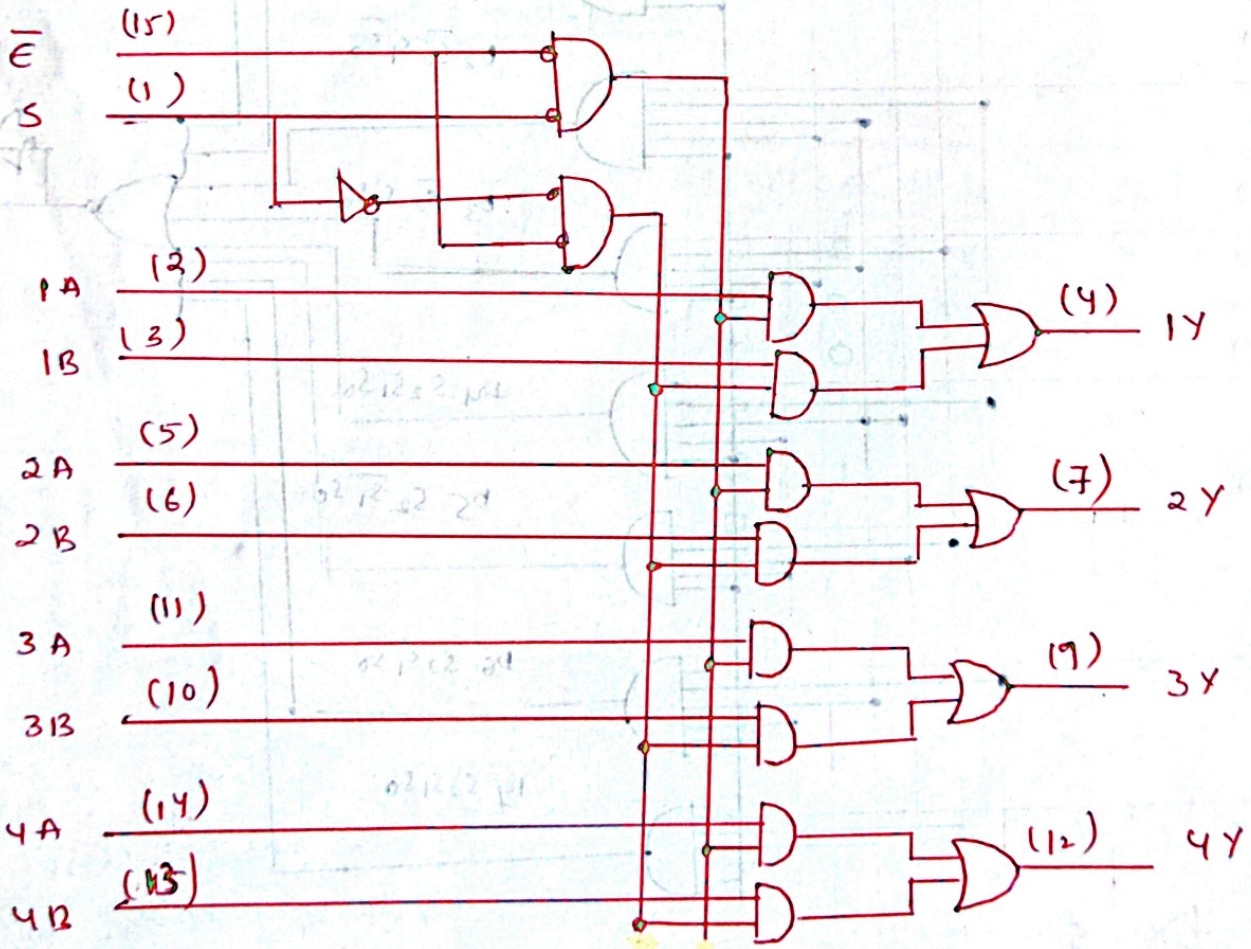


fig: Logic diagram for IC 74X157

→ The IC 74X157 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 ($\bar{E}$) is active low.
→ when $\bar{E}$ is high, all of the outputs (Y) are forced low regardless of all other input conditions.

→

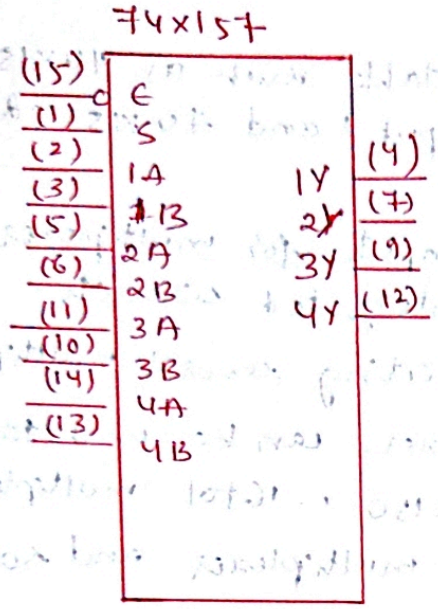


fig: Logic symbol for 74x157

Truth table for 74x157

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

→ The 74x153 dual 4to1 multiplexer.

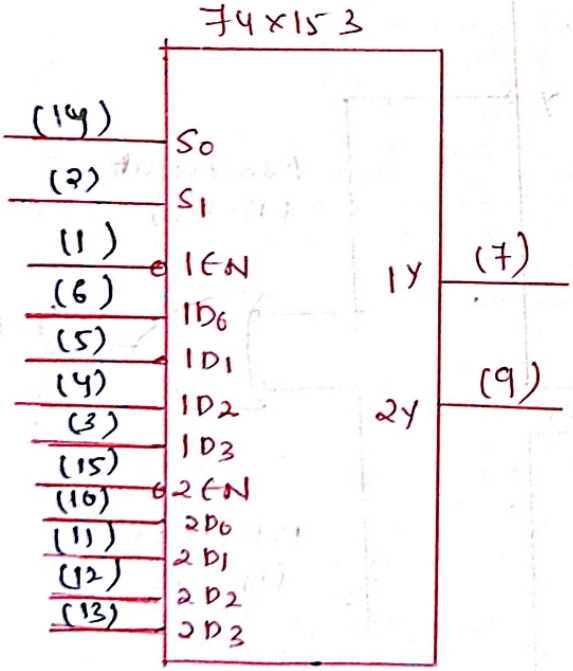


Fig. Logic symbol for 74x153

Inputs				outputs	
1EN	2EN	S_1	S_0	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

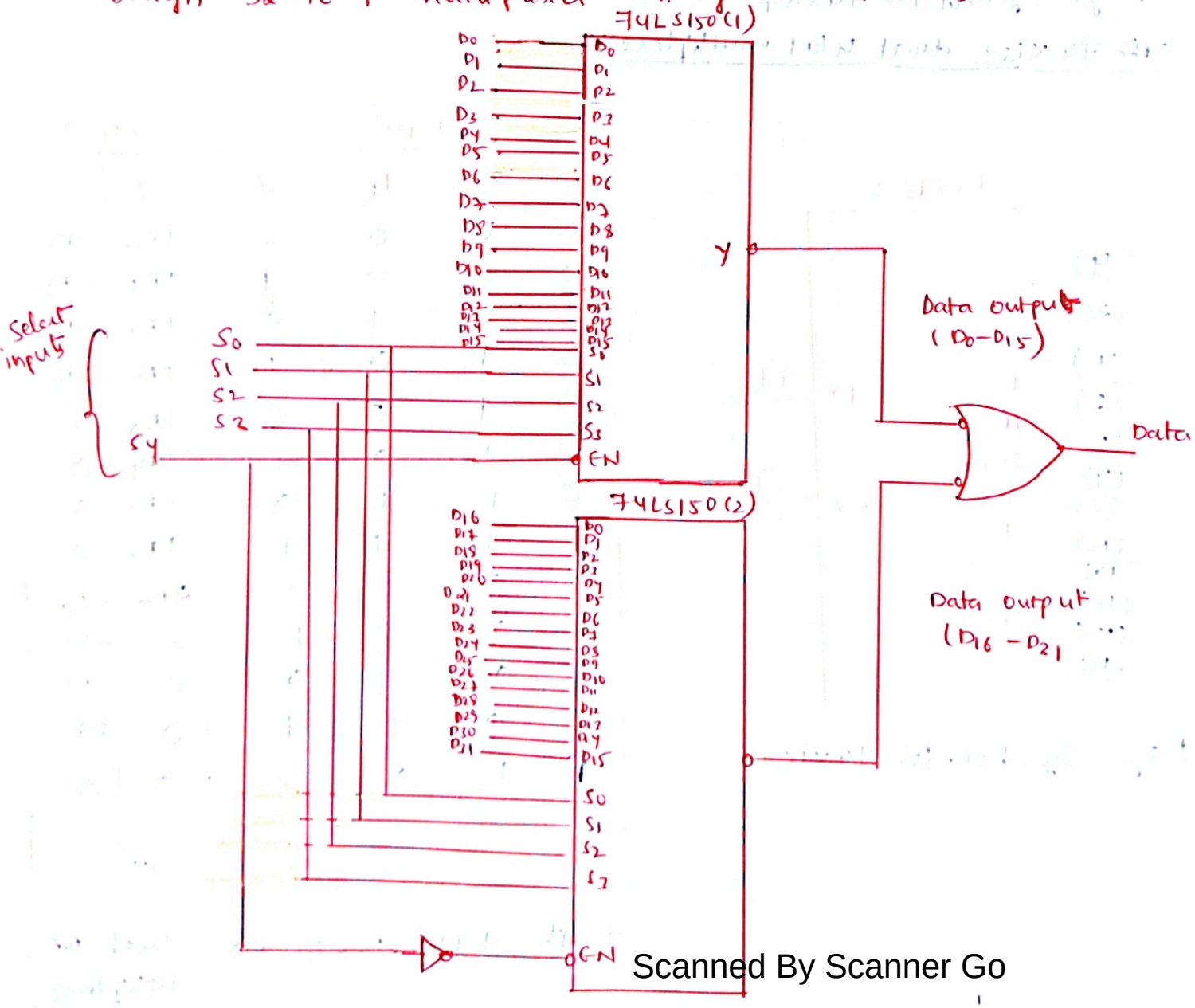
Truth table for 74x153, dual 4to1 multiplexer

→ The 74X153 is a dual 4 to 1 multiplexer.
→ It contains two identical and Independent 4to1 multiplexers.
→ Each multiplexer has separate enable inputs.

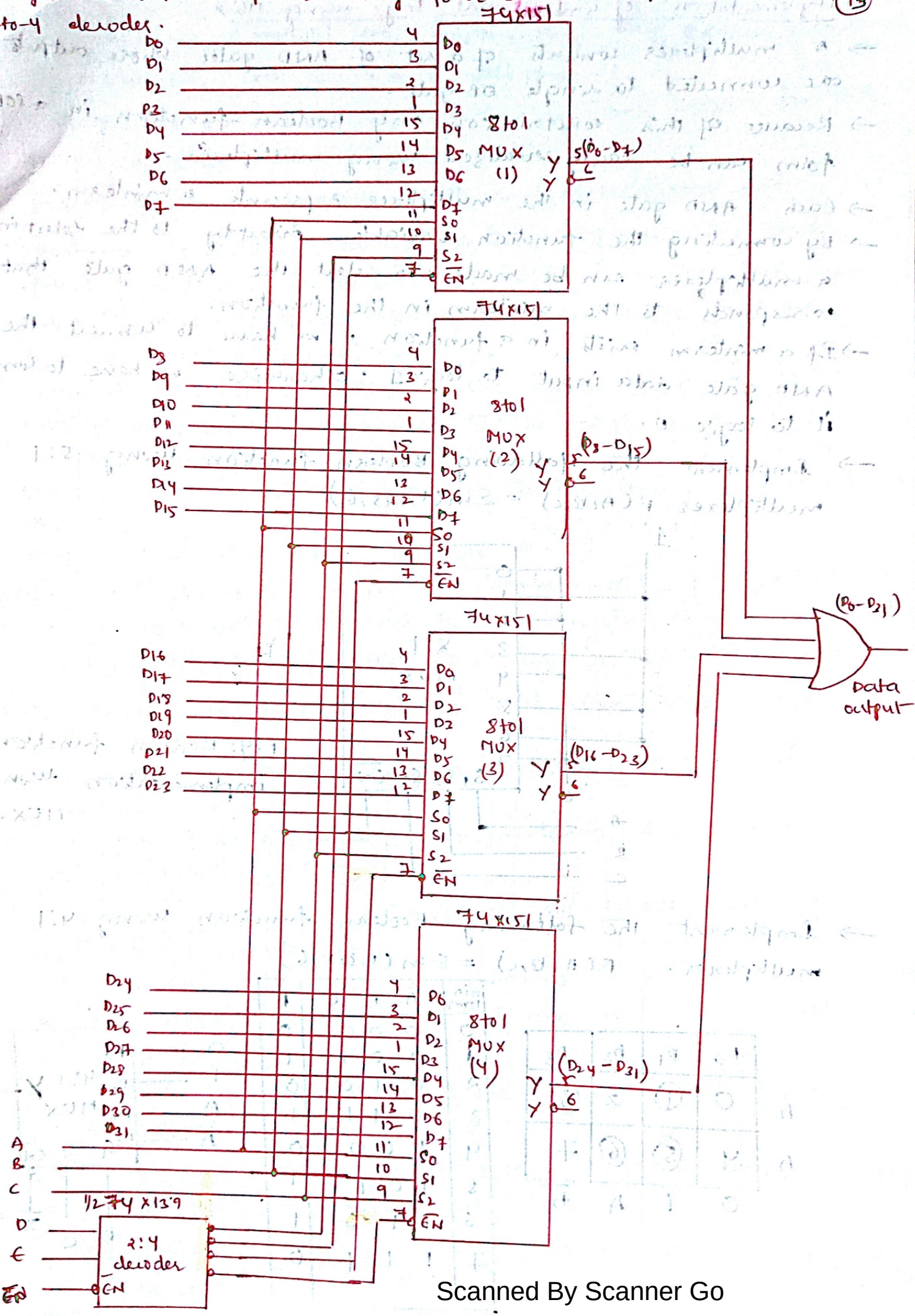
→ **Expanding multiplexers :-**

→ Several digital multiplexer ICs are available such as 74X150 (16 to 1), 74151 (8-to-1), 74157 (dual 2 input) and 74X153 (dual 4to1) 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 74X151, 8to1 multiplexers can be used together to form a 16-to-1 multiplexer, two 74X150, 16to1 multiplexers can be used together to form a 32-to-1 multiplexer and so on.

→ Design 32-to-1 multiplexer using two 74LS150



→ Design 32-to-1 multiplexer using four 8-to-1 multiplexers and 2-to-4 decoder.



→ 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.

→ 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.

→ Implement the following Boolean function using 8:1 multiplexer $F(A,B,C) = \Sigma m(1,3,5,6)$

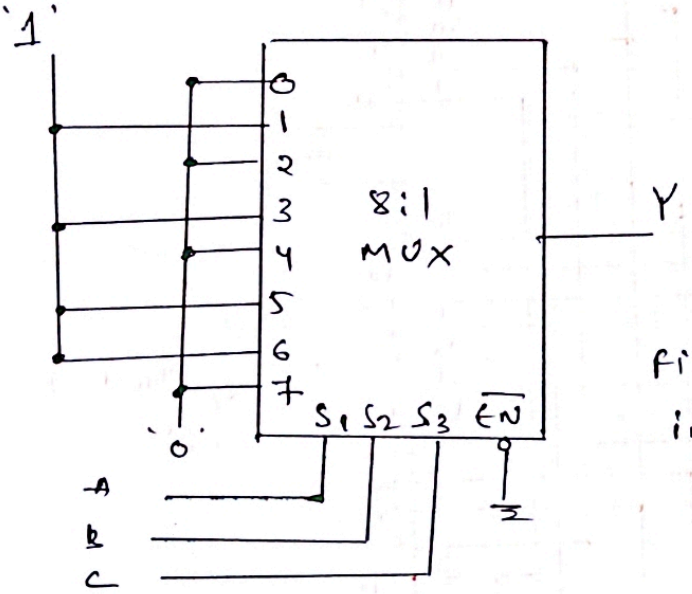
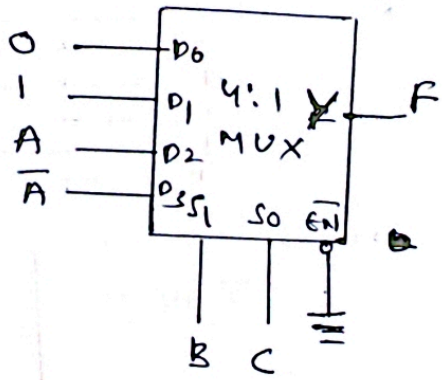


Fig: Boolean function implementation using MUX.

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

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

min term	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



→ [This gives a method for implementing any Boolean function of n variables with a 2^n to-1 multiplexer.]

→ A general procedure for implementing any boolean function of n variables with a 2^{n-1} to-1 multiplexer.

→ First, express the function in its sum of minterms form.

→ Assume that the ordered sequence of variables chosen for the minterms in ABCD ...., where A is the leftmost variable in the ordered sequence of n variables and BCD ... are the remaining n-1 variables.

→ Connect the n-1 variables to the selection lines of the multiplexer, with B connected to the high-order selection line, C to the next lower selection line, and so on down to the last variable, which is connected to the lowest-order selection line so.

→ Consider now the single variable A.

→ Since this variable is in the highest-order position in the sequence of variables, it will be complemented in minterms 0 to $(\frac{2^n}{2})-1$, which comprise the first half in the list of minterms.

→ The second half of the minterms will have their A variable uncomplemented.

→ For a three-variable function, A, B, C, we have eight minterms.

→ Variable A is complemented in minterms 0 to 3 and uncomplemented in minterms 4 to 7.

→ List the inputs of the multiplexer and under them list all the minterms in two rows

→ The first row lists all those minterms where 'A' is complemented, and the second row all the minterms with A uncomplemented, as shown in fig.

→ Circle all the minterms of the function and inspect each column separately.

→ If the two minterms in a column are not circled, apply 0 to the corresponding multiplexer input.

→ If the two minterms are circled, apply 1 to the corresponding multiplexer input.

→ If the bottom minterm is circled and the top is not circled, apply A to the corresponding multiplexer input.

→ If the top minterm is circled and the bottom is not circled, apply 'A' to the corresponding multiplexer input.

→ The function can be implemented with a 4-to-1 multiplexer.

→ Two of the variables, B and C are applied to the selection lines in that order, i.e., B is connected to S_1 and C to S_0.

→ The input of the multiplexer are 0, 1, A and $\bar{A}$.

→ When BC = 00, output = 0 since $D_0 = 0$.
Therefore, both minterms $m_0 = A'B'C'$ and $m_4 = AB'C'$ produce a 0 output, since the output is 0 when BC = 00 regardless of the value of 'A'.

→ When BC = 01, output F = 1, since $D_1 = 1$.
Therefore, both minterm $m_1 = A'B'C$ and $m_5 = AB'C$ produce a 1 output, since the output is 1 when BC = 01 regardless of the value of A.

→ When BC = 10, input D_2 is selected.
Since A is connected to this input, the output will be equal to 1 only for minterm $m_6 = ABC'$, but not for minterm $m_2 = A'BC'$, because when $A' = 1$, then $A = 0$, since $D_2 = 0$, we have F = 0.

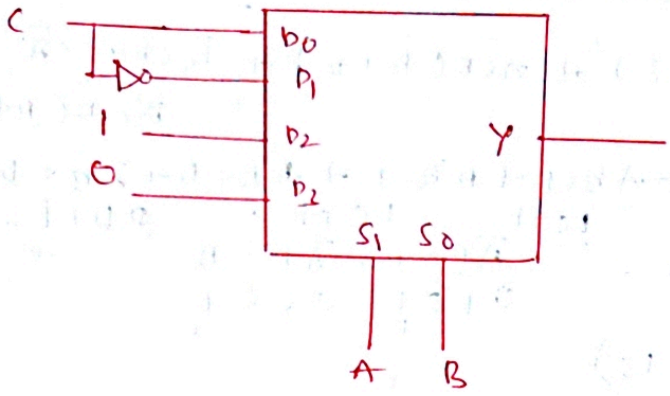
→ Finally, when BC = 11, input D_3 is selected.
Since A' is connected to this input, the output will be equal to 1 only for minterm $m_3 = A'BC$, but not for $m_7 = ABC$

→ It is not necessary to choose the left-most variable in the ordered sequence of a variable list for the data inputs of the multiplexer.

→ $F(A, B, C) = \Sigma m(1, 2, 4, 5)$

	D_0	D_1	D_2	D_3
C'	0	(2)	(4)	6
$= C$	(1)	3	(5)	7
	C	C'	1	0

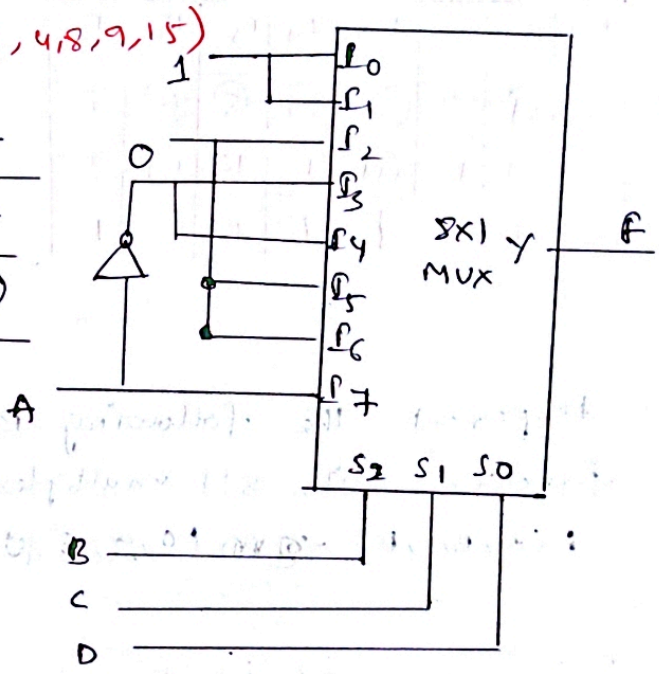
A	B	C	F	
0	0	0	0	$F = C$
0	0	1	1	
0	1	0	1	$F = C'$
0	1	1	0	
1	0	0	1	$F = 1$
1	0	1	1	
1	1	0	0	$F = 0$
1	1	1	0	



→ Implement the following Boolean function using 8:1 MUX.

$F(A, B, C, D) = \Sigma m(0, 1, 3, 4, 8, 9, 15)$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	A'	A'	0	0	A



→ Implement the following Boolean function using 4:1 MUX

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

	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
	$\bar{A}\cdot 1$		$\bar{A}$	0	1	0	1	0

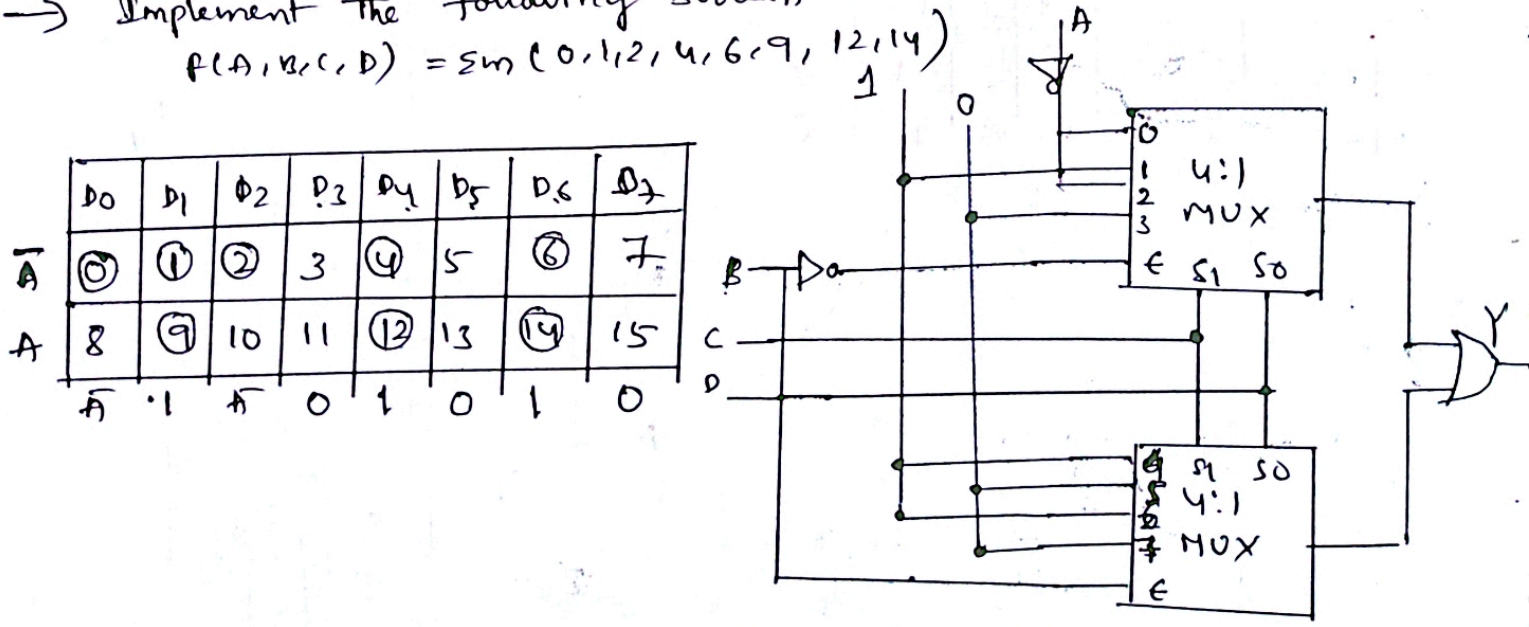


fig: Implementation using two 4:1 multiplexer.

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

$F(A,B,C,D) = \bar{A}B\bar{D} + ACD + \bar{B}CD + \bar{A}\bar{C}D$

Sol:-

$F(A,B,C,D) = \bar{A}B\bar{D}(C+\bar{C}) + ACD(B+\bar{B}) + \bar{B}CD(A+\bar{A}) + \bar{A}\bar{C}D(B+\bar{B})$

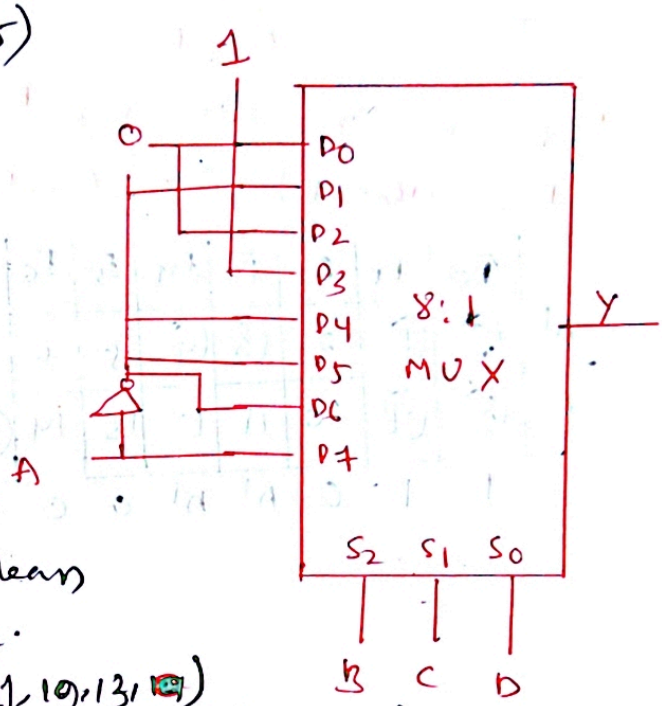
$= \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + ABCD + A\bar{B}CD + A\bar{B}CD + \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D$

0110 0100 1111 1011 1011 0011 $\bar{A}B\bar{C}D$ $\bar{A}\bar{B}\bar{C}D$

0101 0001

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

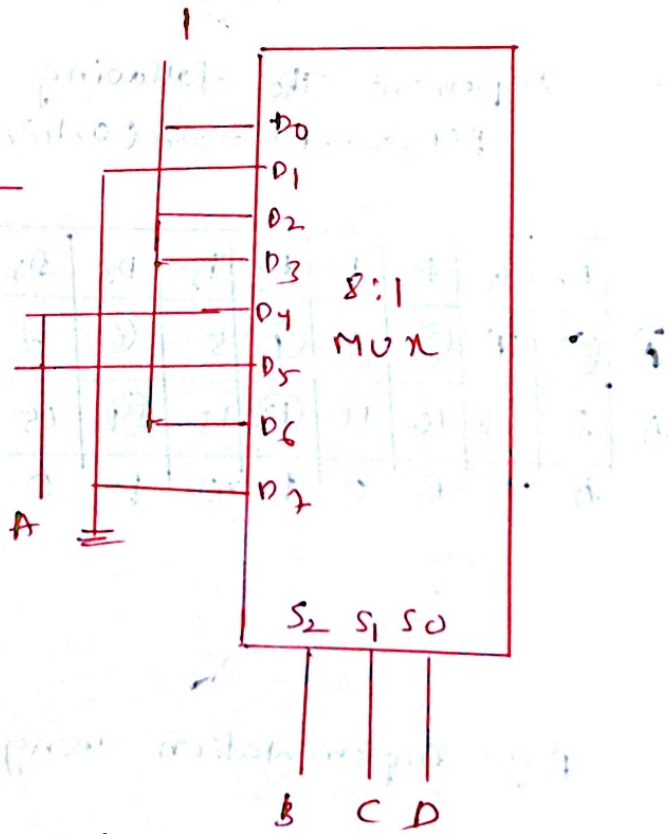
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	(1)	2	(3)	(4)	(5)	(6)	7
A	8	9	10	(11)	12	13	14	(15)
	0	$\bar{A}$	0	1	$\bar{A}$	$\bar{A}$	$\bar{A}$	A



→ Implement the following Boolean function with 8:1 multiplexer.

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

	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
	1	0	1	1	A	A	1	0



→ Demultiplexers :-

→ A decoder with an enable input can function as a demultiplexer. (data distributer)

→ A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.

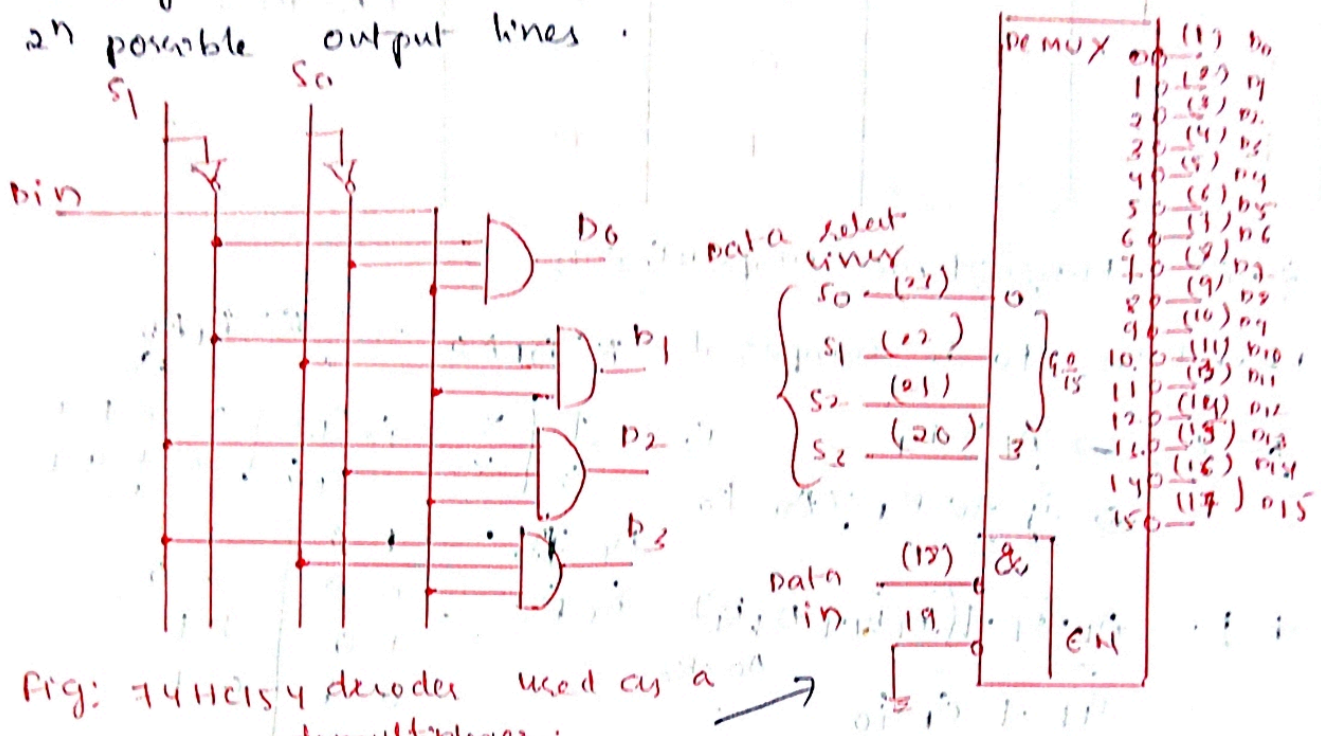


Fig: 74HC154 decoder used as a demultiplexer.

→ 74HC154 decoder - 4 line to 16 line decoder.

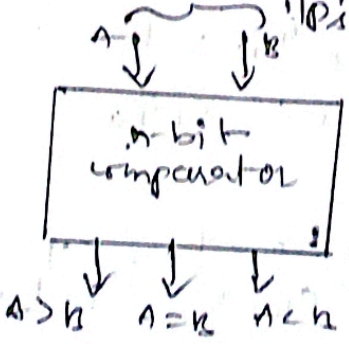
→ This device and other decoders, can also be used in demultiplexing applications.

→ In demultiplexer applications, the input lines are used as the data-select lines.

→ One of the chip select inputs is used as the data-input line, with the other chip select input held LOW to enable the internal negative-AND gate at the bottom of the diagram.

→ Magnitude Comparator :-

→ A comparator is a combinational circuit designed to compare the relative magnitude of two binary numbers.



→ It receives two n-bit numbers A & B as inputs and the outputs are A > B, A = B and A < B.

→ Depending upon the relative magnitudes of the two numbers, one of the outputs will be high.

→ 1-bit magnitude comparison

Two nos of 1-bit.

A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$(A>B) : G_1 = A.\bar{B}$

$(A=B) : \epsilon = \bar{A}.\bar{B} + A.B$
$= (A \odot B)$

$(A<B) : L = \bar{A}.B$

→ 2-bit Magnitude comparision

Numbers to be compared $A : A_1 A_0$ and $B : B_1 B_0$

① $A > B$: → $A_1 > B_1$ i.e, $\begin{array}{c|c} 1 & X \\ 0 & X \end{array}$

→ if $A_1 = B_1$ & $A_0 > B_0$ i.e, $\begin{array}{c|cccc} A_1 & 1 & & & \\ B_1 & 0 & & & \\ \hline A_0 & 0 & 0 & 1 & 1 \\ B_0 & 0 & 1 & 0 & 1 \end{array}$

A_1 A_0 / B_1 B_0

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

$(A>B) = A_1 \cdot \bar{B}_1 + (\bar{A}_1 \bar{B}_1 + A_1 B_1)(A_0 \cdot \bar{B}_0)$

$= G_1 + \epsilon_1 G_0$

② $A = B$: → if $A_1 = B_1$ and $A_0 = B_0$

$(A=B) = (\bar{A}_1 \bar{B}_1 + A_1 B_1) \cdot (\bar{A}_0 \bar{B}_0 + A_0 B_0) = \epsilon_1 \cdot \epsilon_0$

③ $A < B$: if $A_1 < B_1$

if $A_1 = B_1$ and $A_0 < B_0$

$(A<B) = \bar{A}_1 B_1 + (\bar{A}_1 \bar{B}_1 + A_1 B_1)(\bar{A}_0 B_0)$

$= L_1 + \epsilon_1 L_0$

→ n-bit magnitude comparison

Numbers to be compared ; $A : A_{n-1} A_{n-2} \cdots A_1 A_0$ and $B : B_{n-1} B_{n-2} \cdots B_1 B_0$

$(A>B) = G_{n-1} + \epsilon_{n-1} G_{n-2} + \epsilon_{n-1} \epsilon_{n-2} G_{n-3} + \cdots + \epsilon_{n-1} \epsilon_{n-2} \cdots \epsilon_1 G_0$

$(A=B) = \epsilon_{n-1} \epsilon_{n-2} \cdots \epsilon_1 \epsilon_0$

$(A<B) = L_{n-1} + \epsilon_{n-1} L_{n-2} + \epsilon_{n-1} \epsilon_{n-2} L_{n-3} + \cdots + \epsilon_{n-1} \epsilon_{n-2} \cdots \epsilon_1 L_0$

→ With 'cascading inputs' coming from another block with lower significant bits:

$$(A>B)_{out} = G_{n-1} + E_{n-1} G_{n-2} + E_{n-1} E_{n-2} G_{n-3} + \cdots + E_{n-1} E_{n-2} \cdots E_1 G_0 + E_{n-1} E_{n-2} \cdots E_1 E_0 (A>B)_{in}$$

$$(A=B)_{out} = E_{n-1} E_{n-2} \cdots E_1 E_0 \cdot (A=B)_{in}$$

$$(A<B)_{out} = L_{n-1} + E_{n-1} L_{n-2} + E_{n-1} E_{n-2} L_{n-3} + \cdots + E_{n-1} E_{n-2} \cdots E_1 L_0 + E_{n-1} E_{n-2} \cdots E_1 E_0 (A<B)_{in}$$

→

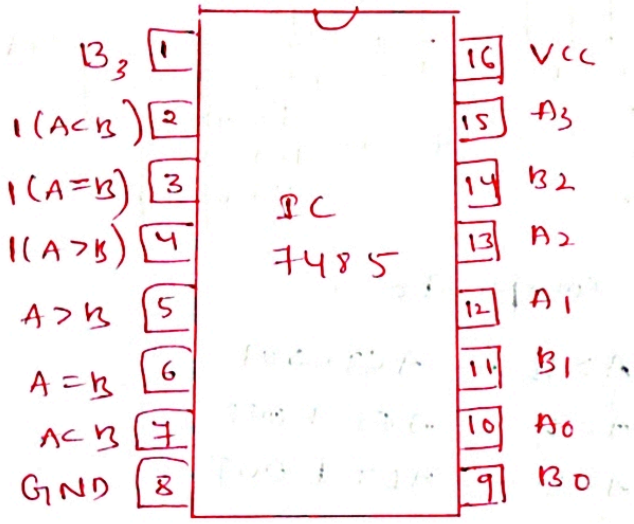
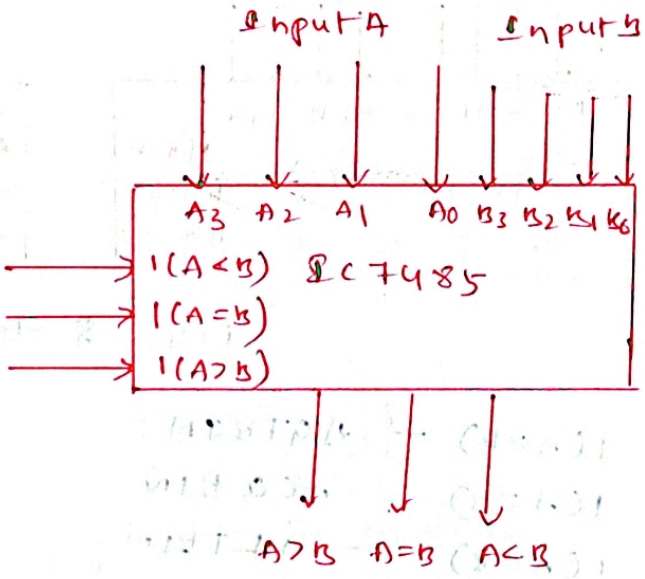


Fig: pin diagram (IC 7485)



(b) Logic diagram (IC 7485)

Comparing inputs A B	Cascading inputs I(A>B)	I(A=B)	I(A<B)	outputs A>B	A=B	A<B
A > B	X	X	X	1	0	0
A = B	1	0	0	1	0	0
	X	1	X	0	1	0
	0	0	1	0	0	1
	0	0	0	1	0	1
	1	0	1	0	0	0
A < B	X	X	X	0	0	1

(c) Function table for

→ $(A>B) = A_3 \cdot \bar{B}_3 + (A_3 \odot B_3) A_2 \cdot \bar{B}_2 + (A_3 \odot B_3)(A_2 \odot B_2) A_1 \bar{B}_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) A_0 \bar{B}_0$

$(A=B) = (A_3 \odot B_3) \cdot (A_2 \odot B_2) \cdot (A_1 \odot B_1) \cdot (A_0 \odot B_0)$

$(A<B) = \bar{A}_3 \cdot B_3 + (A_3 \odot B_3) \bar{A}_2 \cdot B_2 + (A_3 \odot B_3)(A_2 \odot B_2) \bar{A}_1 B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(\bar{A}_0 B_0)$

→. Design an 8-bit comparator using two 7485 ICs.

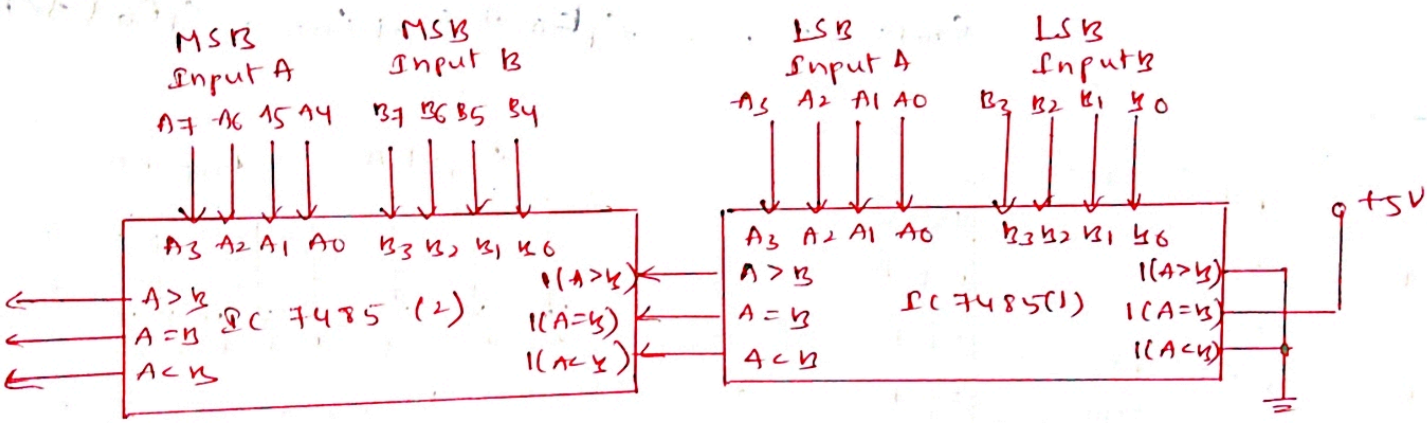


Fig: 8-bit comparator.

I(A>B) — AGTBIN
I(A=B) — AEQBIN
I(A<B) — ALTBIN

A>B — AGTBOUT
A=B — AEQBOUT
A<B — ALTBOUT

→ Code converters :-

Binary code				BCD code				
D	C	B	A	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1

Binary code				BCD code.				
D	C	B	A	B_4	B_3	B_2	B_1	B_0
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	0	1	1

$B_0 = A$; $B_1 = DC\bar{B} + \bar{D}B$; $B_2 = \bar{D}C + CB$; $B_3 = D\bar{C}\bar{B}$

$B_4 = DC + DB$

→ BCD to binary converter

B_4	B_3	B_2	B_1	B_0	E	D	C	B	A
0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0	1	1

→ for $A = B_0$; $C = \bar{B_4}B_2 + B_2\bar{B_1} + B_4\bar{B_2}B_1$

$B = B_1 \oplus B_4$; $D = \bar{B_4}B_3 + B_4\bar{B_3}\bar{B_2} + B_4\bar{B_3}\bar{B_1}$

$E = B_4B_5 + B_4B_2B_1$

→ BCD to excess-3

decimal	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
9	1	0	0	1	1	1	0	0

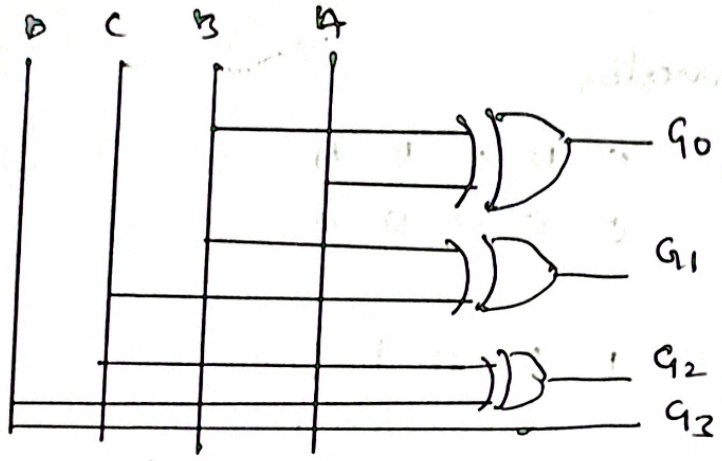
for $E_3 = B_3 + B_2(B_0 + B_1)$; $E_2 = B_2\bar{B_1}\bar{B_0} + \bar{B_2}(B_0 + B_1)$

$E_1 = B_1 \odot B_0$; $E_0 = \bar{B_0}$

→ Binary to gray code converter

decimal	Binary code				Gray code			
	D	C	B	A	G_3	G_2	G_1	G_A
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4								
5								
6								
7								

for $g_0 = B \oplus A$ $\quad g_1 = C \oplus B$ $\quad g_2 = D \oplus C$ $\quad g_3 = D$



→ gray code to binary code converter

g_3	g_2	g_1	g_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0

→ for $A = (g_3 \oplus g_2) \oplus (g_1 \oplus g_0)$

$B = g_3 \oplus g_2 \oplus g_1$

$C = g_3 \oplus g_2$