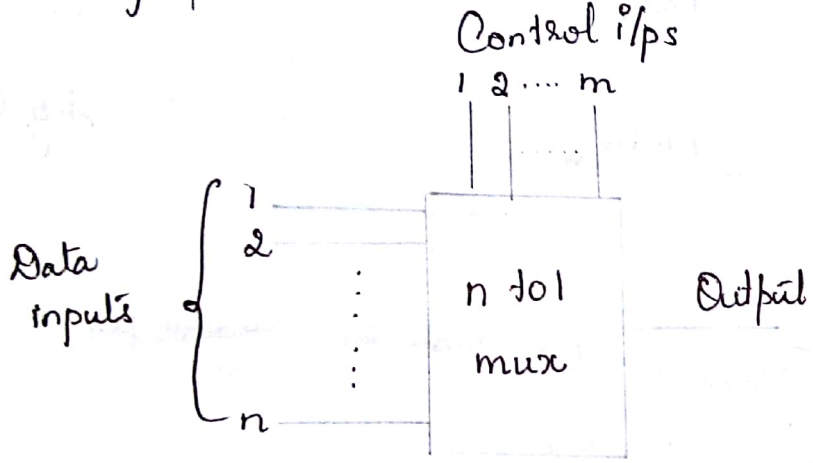
Module III

DATA PROCESSING CIRCUITS

Multiplexer :

Multiplex means many to one. A multiplexer is a circuit with many inputs but only one output. By applying control signals, we can steer any input to the output. Hence it is also called a data selector and control inputs are termed select inputs.



Multiplexer block diagram

The circuit has n input signals, m control signals and 1 o/p signal. Note that m control signal can select at the most 2^m inputs signals thus $n \leq 2^m$

The circuit diagram of a 4 to 1 multiplexer is shown in fig (a) and its truth table in fig (b). Depending on control inputs A, B one of 4 inputs D_0 to D_3 are steered to o/p Y.

Logic equation is

$$Y = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$$

if A = 0, B = 0 $Y = \bar{0}\bar{0}D_0 + \bar{0}0D_1 + 0\bar{0}D_2 + 00D_3$

$= 1 \cdot 1 D_0 + 1 \cdot 0 D_1 + 0 \cdot 1 D_2 + 0 \cdot 0 D_3$

$= D_0$

i.e when AB = 00 the 1st AND gate to which D_0 is connected remain active and equal to D_0 and all other AND gate are inactive with o/p held at logic 0. Thus multiplexer o/p y is D_0. If $D_0 = 0$ $y = 0$ and if $D_0 = 1$ $y = 1$.

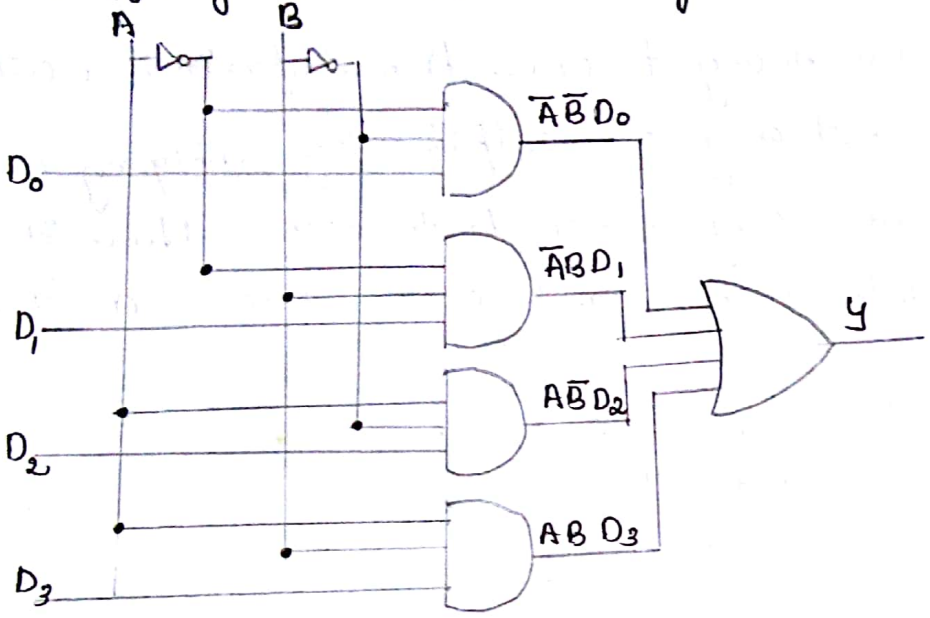


fig (a)

A	B	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

fig (b)

Commercial multiplexers IC's come in integer power of 2, eg. 2 to 1, 4 to 1, 8 to 1, 16 to 1 mux.

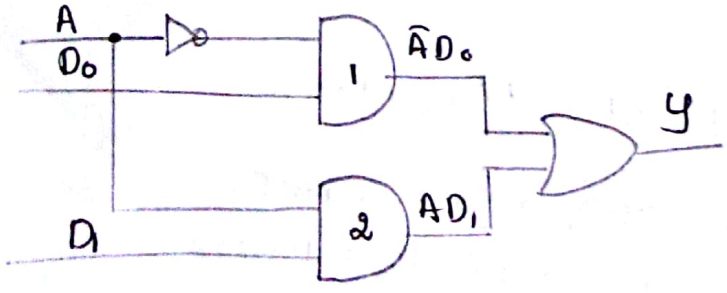
Explain 2:1 mux with logic diagram & truth table.

In 2:1 mux, depending on a control i/p A one of 2 i/ps D_0 or D_1 is steered to o/p y.

Logic equation is

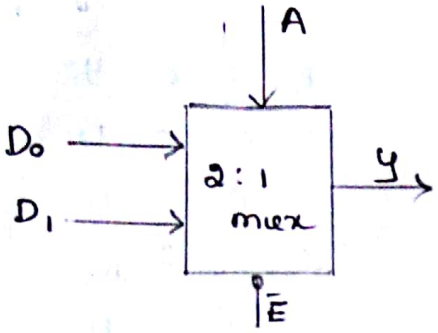
$$y = \bar{A}D_0 + AD_1$$

Logic diagram & truth table is shown below.

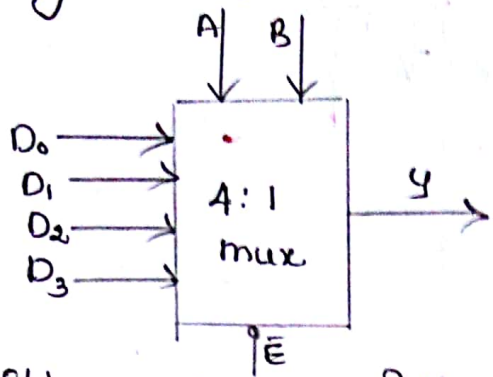


A	y
0	D_0
1	D_1

Logic Symbol 2:1 mux



Logic Symbol 4:1 mux

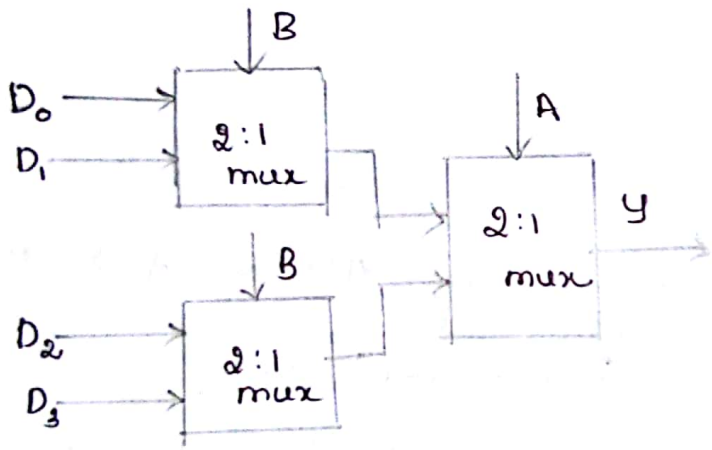


Problem: Construct 4:1 multiplexer using 2:1 mux.

Logic eqn for 2:1 mux is $Y = \bar{A}D_0 + AD_1$

Logic eqn for 4:1 mux is $Y = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$

$= \bar{A}(\bar{B}D_0 + BD_1) + A(\bar{B}D_2 + BD_3)$



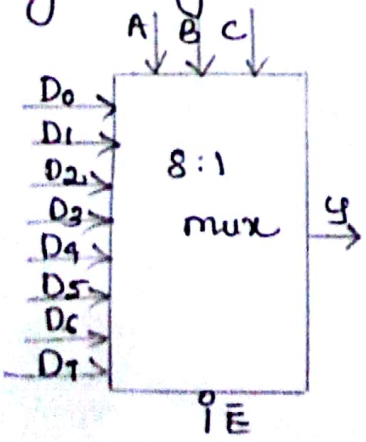
8 to 1 multiplexer:

In 8:1 mux, depending on control i/ps A, B, C, one of 8 i/ps ($D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$) is steered to o/p Y.

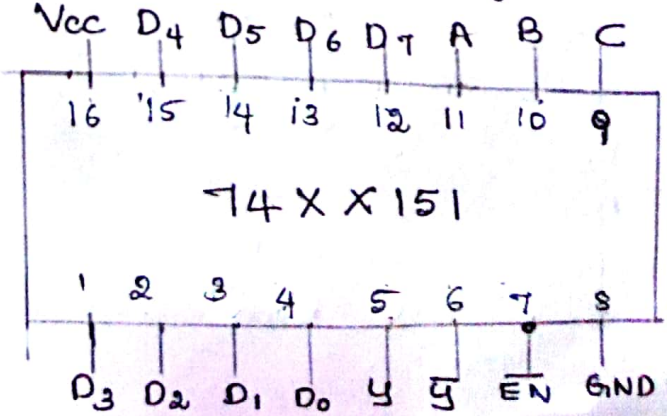
Logic equation is

$$Y = \bar{A}\bar{B}\bar{C}D_0 + \bar{A}\bar{B}CD_1 + \bar{A}B\bar{C}D_2 + \bar{A}BCD_3 + A\bar{B}\bar{C}D_4 + A\bar{B}CD_5 + AB\bar{C}D_6 + ABCD_7$$

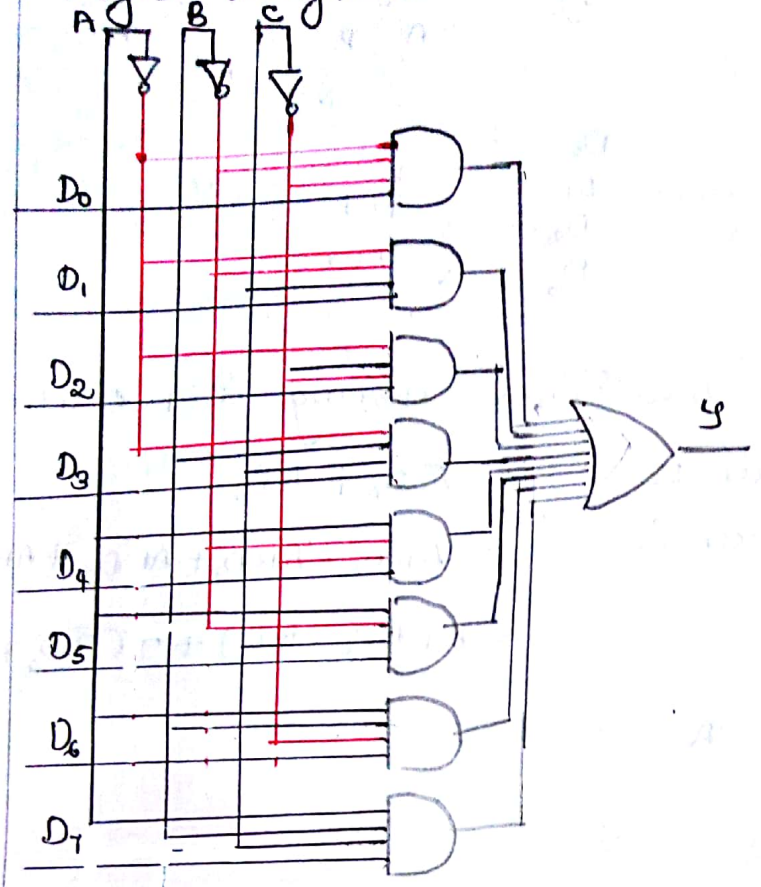
Logic symbol



Pin diagram 74x151 (IC for 8:1 mux)



Logic Diagram



Truth Table

A	B	C	y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Problem: Construct 8:1 mux using 4:1 mux & 2:1 mux

$y = \bar{A}D_0 + AD_1$

2:1 mux $\Rightarrow$

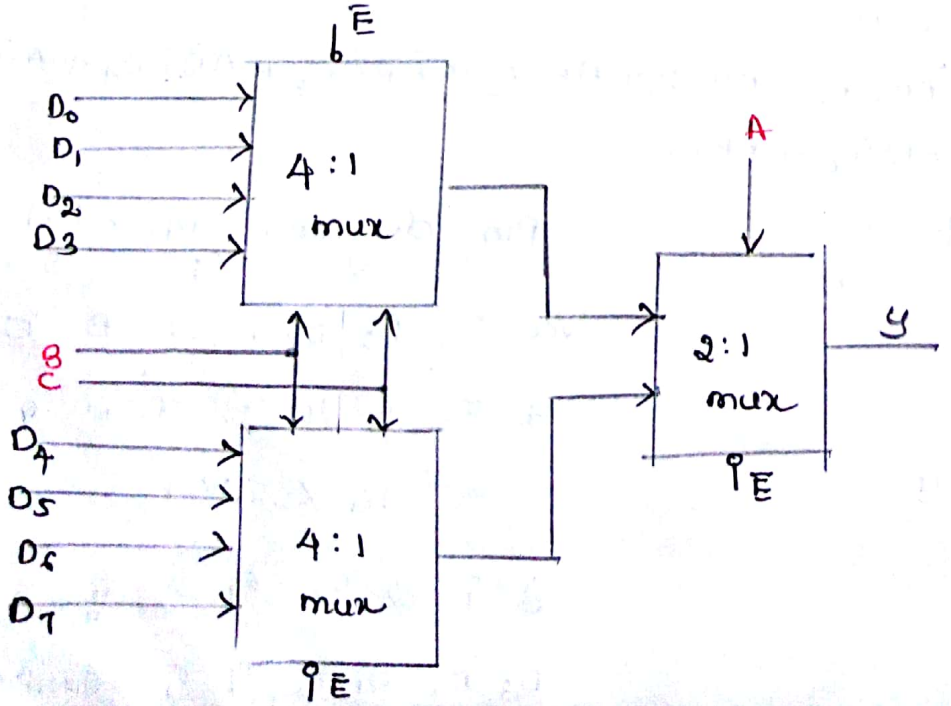
4:1 mux $\Rightarrow$ $y = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$

8:1 mux $\Rightarrow$ $y = \bar{A}\bar{B}\bar{C}D_0 + \bar{A}\bar{B}CD_1 + \bar{A}B\bar{C}D_2 + \bar{A}BCD_3 + A\bar{B}\bar{C}D_4$

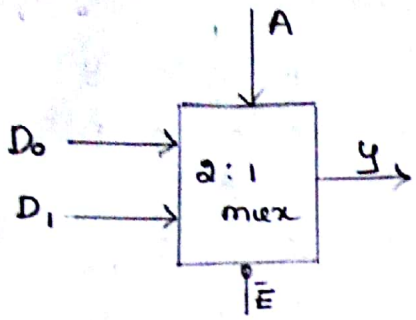
$A\bar{B}CD_5 + AB\bar{C}D_6 + ABCD_7$

$= \bar{A}(\bar{B}\bar{C}D_0 + \bar{B}CD_1 + B\bar{C}D_2 + BCD_3) +$

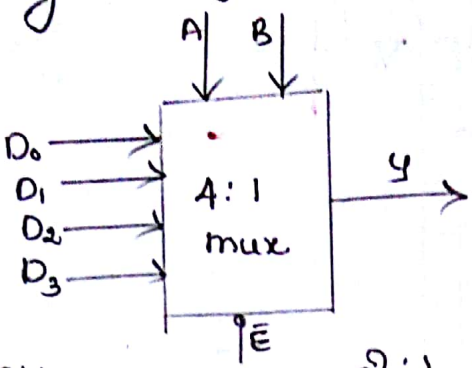
$A(C\bar{B}\bar{C}D_4 + \bar{B}CD_5 + B\bar{C}D_6 + BCD_7)$



Logic Symbol 2:1 mux



Logic Symbol 4:1 mux

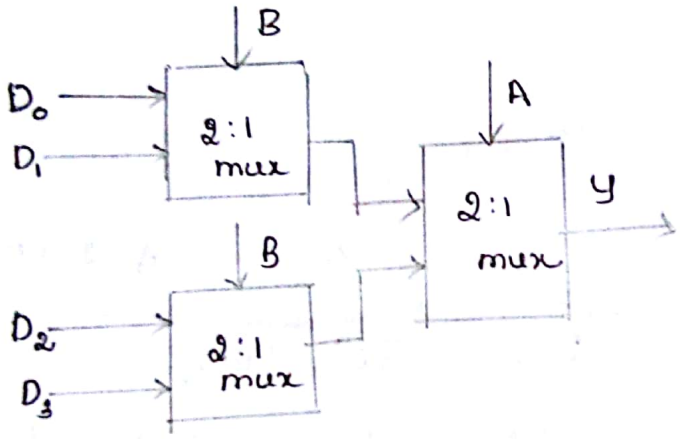


Problem: Construct 4:1 multiplexer using 2:1 mux.

Logic eqn for 2:1 mux is $y = \bar{A}D_0 + AD_1$

Logic eqn for 4:1 mux is $y = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$

$= \bar{A}(\bar{B}D_0 + BD_1) + A(\bar{B}D_2 + BD_3)$



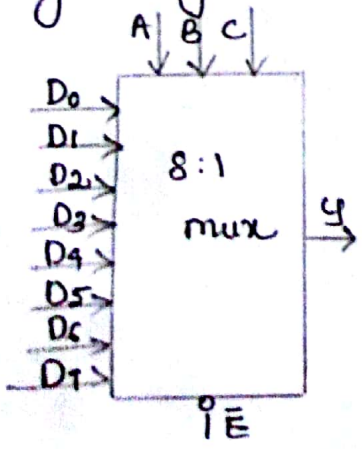
8 to 1 multiplexer:

In 8:1 mux, depending on control i/ps A, B, C, one of 8 i/ps ($D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$) is steered to o/p y.

Logic equation is

$$y = \bar{A}\bar{B}\bar{C}D_0 + \bar{A}\bar{B}CD_1 + \bar{A}B\bar{C}D_2 + \bar{A}BCD_3 + A\bar{B}\bar{C}D_4 + A\bar{B}CD_5 + AB\bar{C}D_6 + ABCD_7$$

Logic symbol



Pin diagram 74x151 (IC for 8:1 mux)

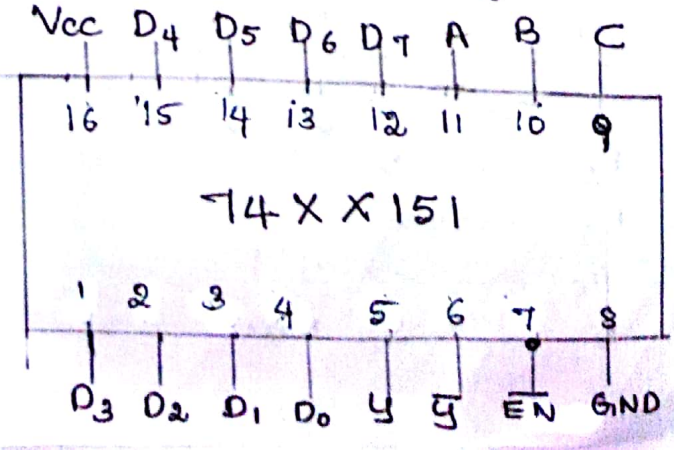
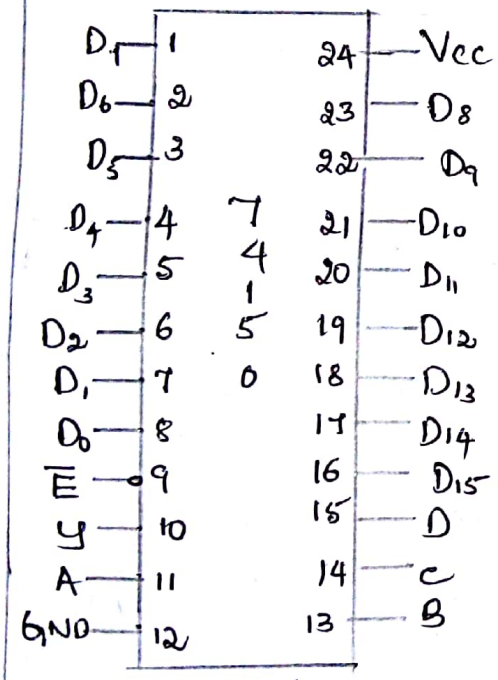


Figure (a) shows a 16 to 1 mux. The i/p bits are labeled D_0 to D_{15}. ABCD are the control inputs. Only one of D_0 to D_{15} is transmitted to the o/p.

Logic Equation for 16 : 1 mux is

$$Y = \bar{A}\bar{B}\bar{C}\bar{D}D_0 + \bar{A}\bar{B}\bar{C}DD_1 + \bar{A}\bar{B}C\bar{D}D_2 + \bar{A}\bar{B}CDD_3 + \bar{A}B\bar{C}\bar{D}D_4 + \bar{A}B\bar{C}DD_5 + \bar{A}BC\bar{D}D_6$$
$$\bar{A}BCDD_7 + A\bar{B}\bar{C}\bar{D}D_8 + A\bar{B}\bar{C}DD_9 + A\bar{B}C\bar{D}D_{10} + A\bar{B}CDD_{11} + AB\bar{C}\bar{D}D_{12}$$
$$AB\bar{C}DD_{13} + ABC\bar{D}D_{14} + ABCDD_{15}$$

The 74150 is a 16 to 1 mux TTL and it Active low IC. 24 pins and 9th pin is the STROBE which is active low.



pin diagram

A	B	C	D	Y
0	0	0	0	$\bar{D}_0$
0	0	0	1	$\bar{D}_1$
0	0	1	0	$\bar{D}_2$
0	0	1	1	$\bar{D}_3$
0	1	0	0	$\bar{D}_4$
0	1	0	1	$\bar{D}_5$
0	1	1	0	$\bar{D}_6$
0	1	1	1	$\bar{D}_7$
1	0	0	0	$\bar{D}_8$
1	0	0	1	$\bar{D}_9$
1	0	1	0	$\bar{D}_{10}$
1	0	1	1	$\bar{D}_{11}$
1	1	0	0	$\bar{D}_{12}$
1	1	0	1	$\bar{D}_{13}$
1	1	1	0	$\bar{D}_{14}$
1	1	1	1	$\bar{D}_{15}$

truth table

Multiplexer Logic:

Third method of implementing truth table is using multiplexer which is called multiplexer solution.

Bubbles on Signal Line:

Bubble on signal line indicate it is active low signal.

Universal Logic Circuit-

Mux sometimes called universal logic circuit- because a 2^n to 1 mux can be used as a design soln for any n

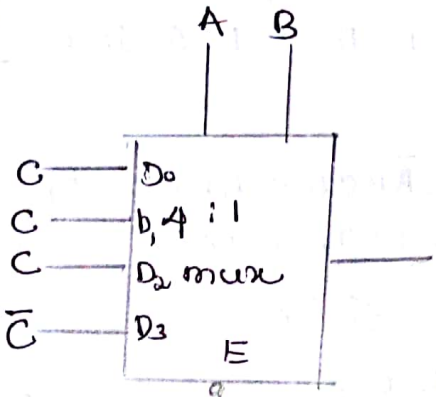
variable truth table. Eg: 4 variable truth table can be realized by 16:1 mux.

Problems:

1) Implement the following Boolean function using 4:1 mux.

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

m_i	A	B	C	y	8:1 mux data i/p
0	0	0	0	0	$D_0 = C$
1	0	0	1	1	
2	0	1	0	0	$D_1 = \bar{C}$
3	0	1	1	1	
4	1	0	0	0	$D_2 = C$
5	1	0	1	1	
6	1	1	0	1	$D_3 = \bar{C}$
7	1	1	1	0	



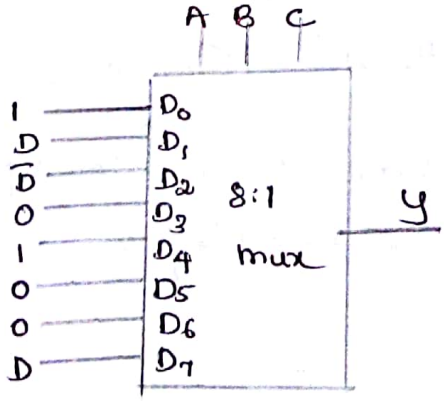
(A, B are control i/p)

2) Implement the following boolean function using 8:1 Mux

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

A, B, C are control i/p

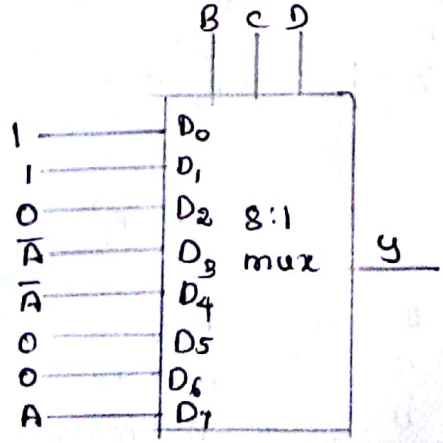
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0 $\bar{D}$	1_0	0_2	1_4	0_6	1_8	0_{10}	0_{12}	0_{14}
1 D	1_1	1_3	0_5	0_7	1_9	0_{11}	0_{13}	1_{15}
	1	D	$\bar{D}$	0	1	0	0	D



(or)

ii) B C D are control i/p

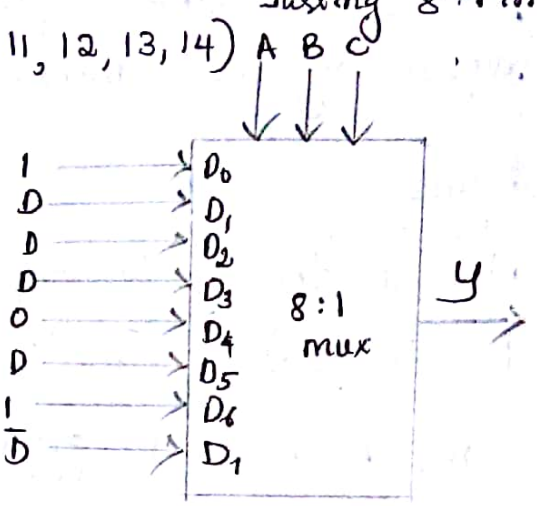
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	1_0	1_1	0_2	1_3	1_4	0_5	0_6	0_7
A	1_8	1_9	0_{10}	0_{11}	0_{12}	0_{13}	0_{14}	1_{15}
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A



3) $f(A,B,C,D) = \Sigma(0,1,3,5,7,11,12,13,14)$ using 8:1 mux

if A, B, C are control i/ps.

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
I_0	0_2	0_4	0_6	0_8	0_{10}	I_{12}	I_{14}
I_1	I_3	I_5	I_7	0_9	I_{11}	I_{13}	0_{15}
1	D	D	D	0	D	1	$\bar{D}$



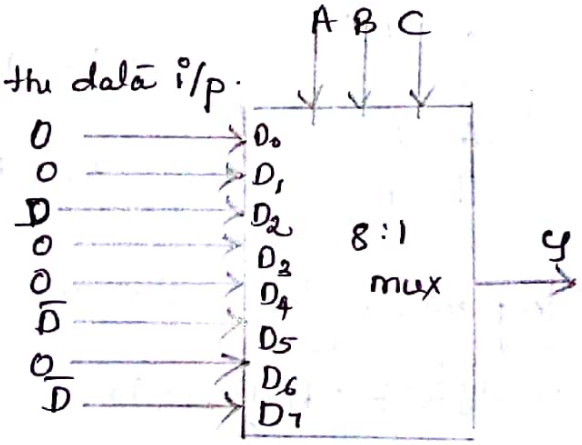
4) $Y = \bar{A}B\bar{C}D + A\bar{B}C\bar{D} + ABC\bar{D}$ using 8:1 mux

$= 0101 + 1010 + 1110$

$= \Sigma(5, 10, 14)$

if A, B, C are control i/p & D is the data i/p.

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0_0	0_2	0_4	0_6	0_8	I_{10}	0_{12}	I_{14}
0	0_3	I_5	0_7	0_9	0_{11}	0_{13}	0_{15}
0	0	D	0	0	$\bar{D}$	0	$\bar{D}$

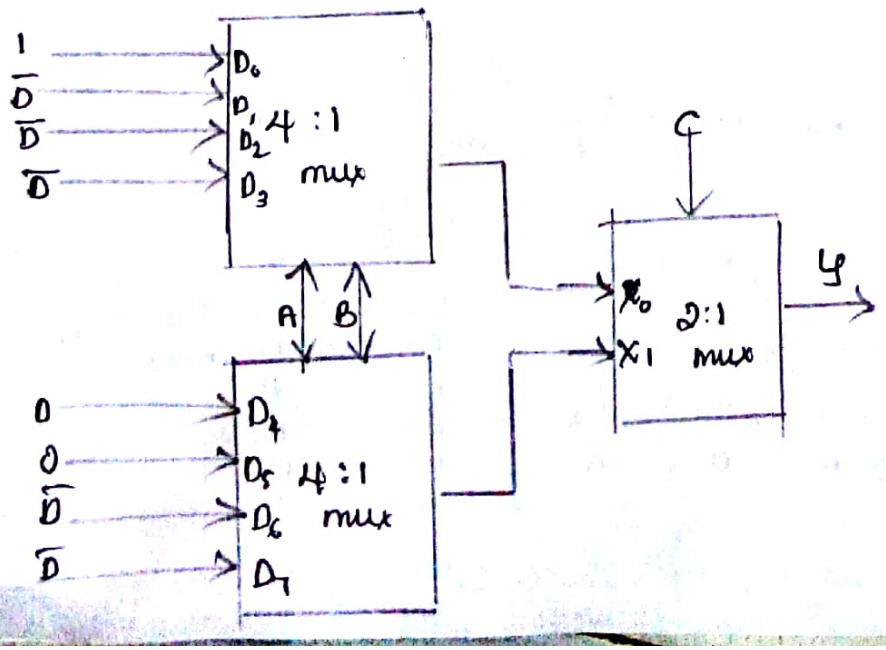


5) $F(A,B,C,D) = \Sigma m(0,1,2,4,6,9,12,14)$ using 4:1 mux

A, B, C are the control i/ps

D is the Data input.

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
I_0	I_2	I_4	I_6	0_8	0_{10}	I_{12}	I_{14}
I_1	0_3	0_5	0_7	I_9	0_{11}	0_{13}	0_{15}
1	$\bar{D}$	$\bar{D}$	$\bar{D}$	D	0	$\bar{D}$	$\bar{D}$



6. For given 4 Bit Binary to Gray Code converter and realize the same using 4 8:1 mux.

A	B	C	D	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Implementation for y_3

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0_0	0_2	0_4	0_6	1_8	1_{10}	1_{12}	1_{14}
0_1	0_3	0_5	0_7	1_9	1_{11}	1_{13}	1_{15}
0	0	0	0	1	1	1	1

A, B, C → Control i/p
D → Data i/p

Implementation for y_2

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0

A, B, C → control i/ps
D → Data i/p

Implementation for y_1

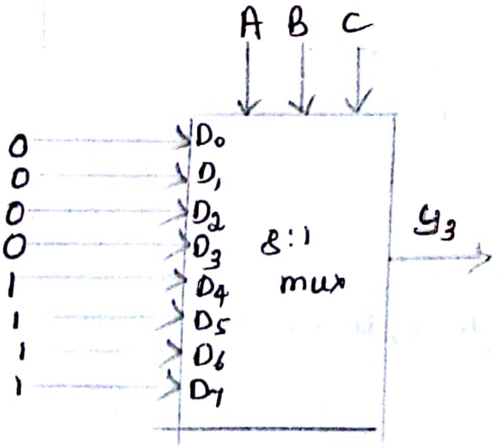
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0

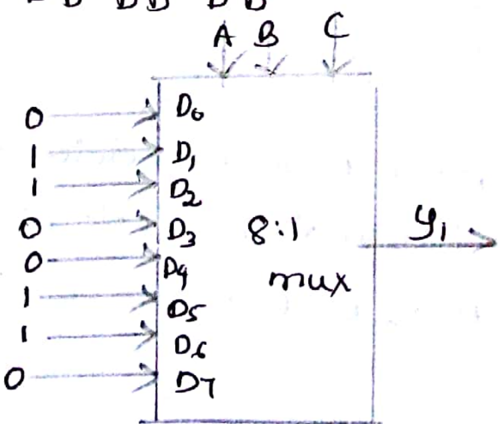
A, B, C → control i/ps
D → data i/p

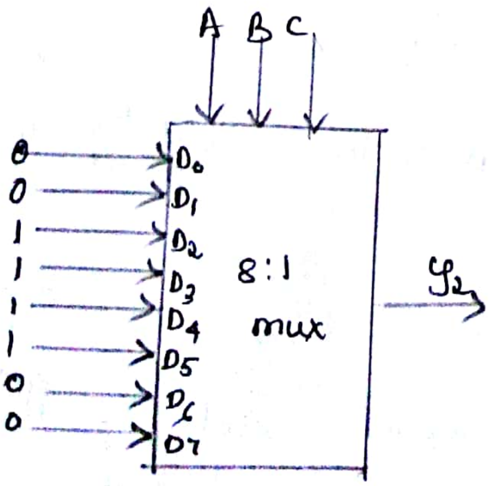
Implementation for y_0

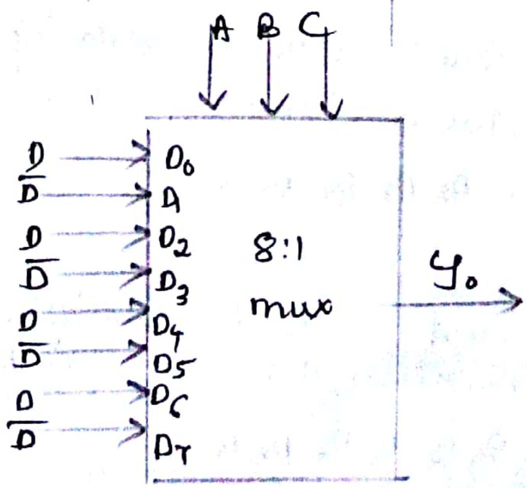
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
D	$\bar{D}$	D	$\bar{D}$	D	$\bar{D}$	D	$\bar{D}$

A, B, C → control i/ps
D → data i/p







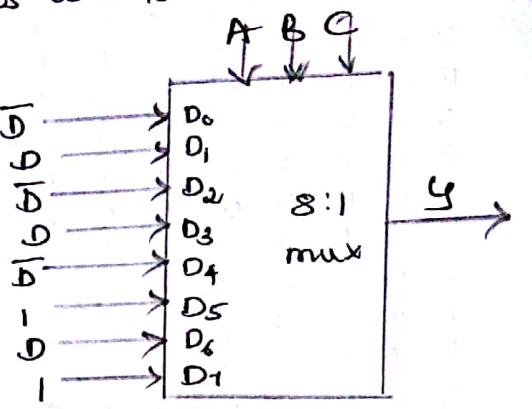


7. Implement the Boolean function expressed by POS $f(A,B,C,D) = \pi M(1,2,5,6,9,12)$ using 8:1 Mux

The only change is instead of putting 1 in place of (1,2,5,6,9,12) put 0. All the remaining steps are same.

A, B, C - control i/ps & D is data i/p

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
1.	0	1	0	1	1	0	1
0	1	0	1	0	1	1	0
$\bar{D}$	D	$\bar{D}$	D	$\bar{D}$	1	D	1



Note:

For a 4 variable expression simplification using 8:1 mux if we want A to be data i/p and B, C, D the control i/ps then implementation table would be.

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

if we want B to be data i/p and A, C, D the control i/p then implementation table would be.

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

if we want C to be data i/p & A, B, D the control i/p then implementation table would be.

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

if we want D to be data i/p & A, B, C the control i/p then implementation table would be.

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

Implement 8:1 mux using 2:1 mux only.

$Y = \bar{A}\bar{B}\bar{C}D_0 + \bar{A}\bar{B}CD_1 + \bar{A}B\bar{C}D_2 + \bar{A}BCD_3 + A\bar{B}\bar{C}D_4 + A\bar{B}CD_5 + AB\bar{C}D_6 + ABCD_7$

$= \bar{A}\bar{B}(\bar{C}D_0 + CD_1) + \bar{A}B(\bar{C}D_2 + CD_3) + A\bar{B}(\bar{C}D_4 + CD_5) + AB(\bar{C}D_6 + CD_7)$

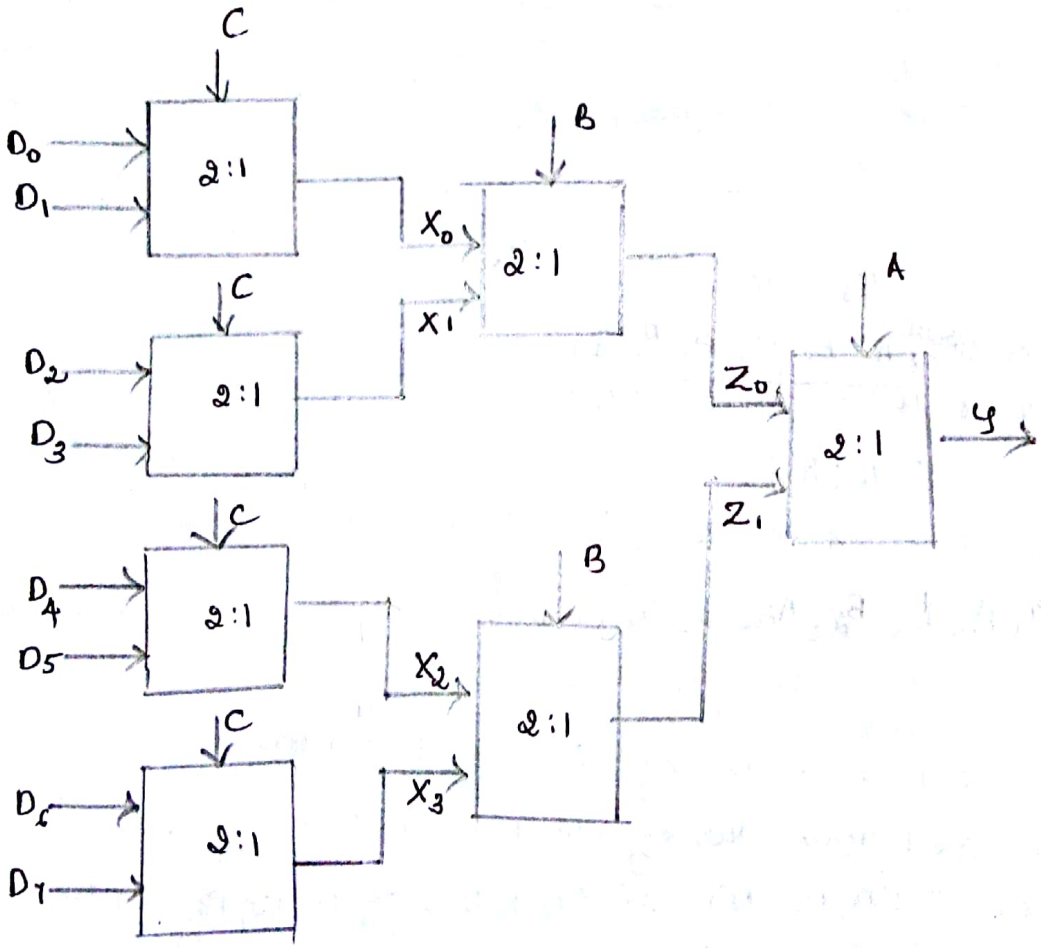
(2:1) (2:1) (2:1) (2:1)

$= \bar{A}\bar{B}X_0 + \bar{A}BX_1 + A\bar{B}X_2 + ABX_3$

$= \bar{A}(\bar{B}X_0 + BX_1) + A(\bar{B}X_2 + BX_3)$

(2:1) (2:1)

$= \bar{A}Z_0 + AZ_1$



Nibble Multiplexer:

Sometimes we want to select one of 2 i/p nibbles. In this case, we can use a nibble multiplexer. The i/p nibble on the left is $A_3A_2A_1A_0$ and one on the right is $B_3B_2B_1B_0$. The control signal labeled SELECT determine

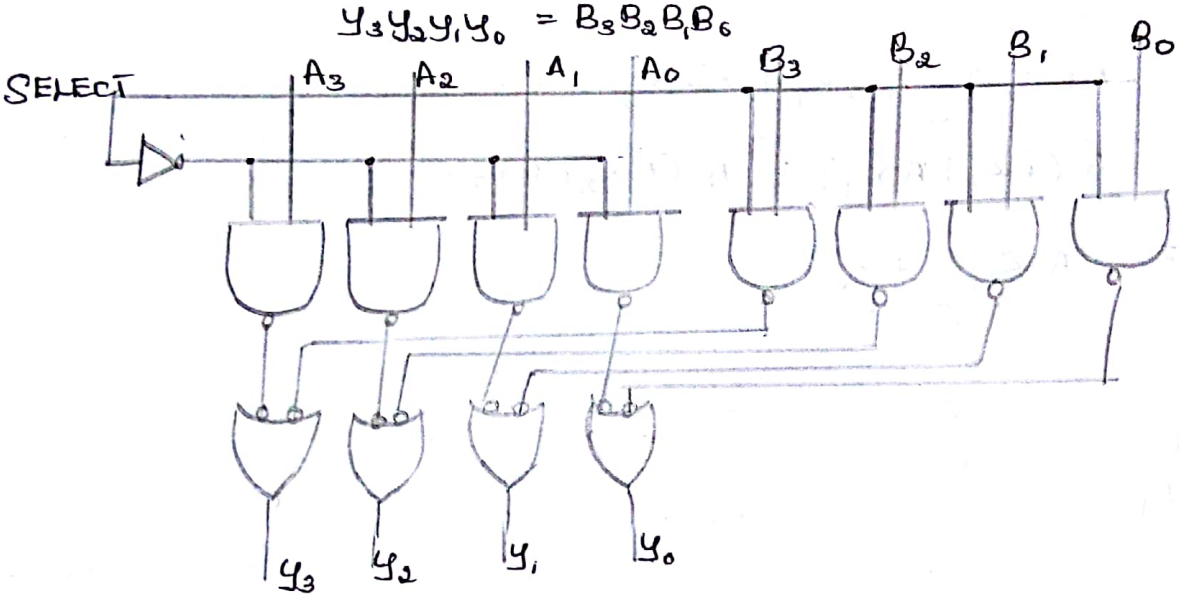
which i/p nibble is transmitted to the o/p.

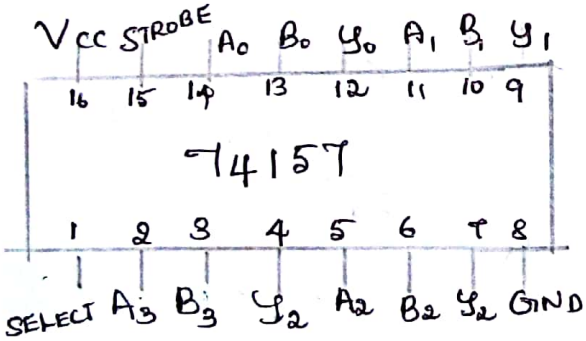
When SELECT is low, the 4 NAND gates on the left are activated

$$Y_3Y_2Y_1Y_0 = A_3A_2A_1A_0$$

When SELECT is high, the 4 NAND gates on the right are activated

$$Y_3Y_2Y_1Y_0 = B_3B_2B_1B_0$$



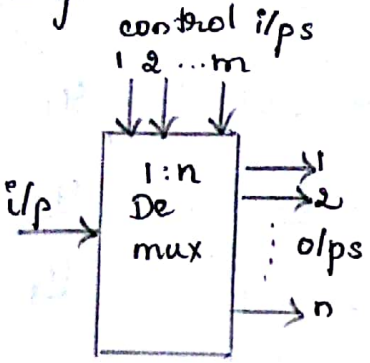


Problems:

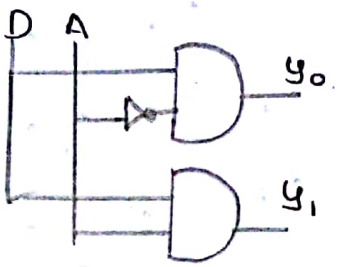
1. Construct 16:1 mux using 4:1 and 2:1 mux
2. Construct 32:1 mux using 16:1 & 2:1 mux
3. Implement $F(A,B,C,D) = \Sigma(0,1,3,5,7,11,12,13,14)$ using 8:1 mux.
4. Implement $F(A,B,C,D) = \Sigma(0,1,5,6,8,10,12,15)$ using 8:1 mux.
5. Implement $F(A,B,C,D) = \pi(1,2,5,6,9,12)$ using 8:1 mux
6. Implement 16:1 mux using 8:1 mux & 2:1 mux
7. Implement 32:1 mux using 8:1 mux & 4:1 mux
8. Write the truth table for Gray Code (binary to Gray code) and realize the same using 2 8:1 mux

Demultiplexer:

Demultiplex means one to many. A demultiplexer is a logic circuit with one input & many outputs. The circuit has 1 input signal, m control/select signals and n output signals where $n \leq 2^m$



De-mux block diagram



Logic circuit of 1 to 2 de-mux

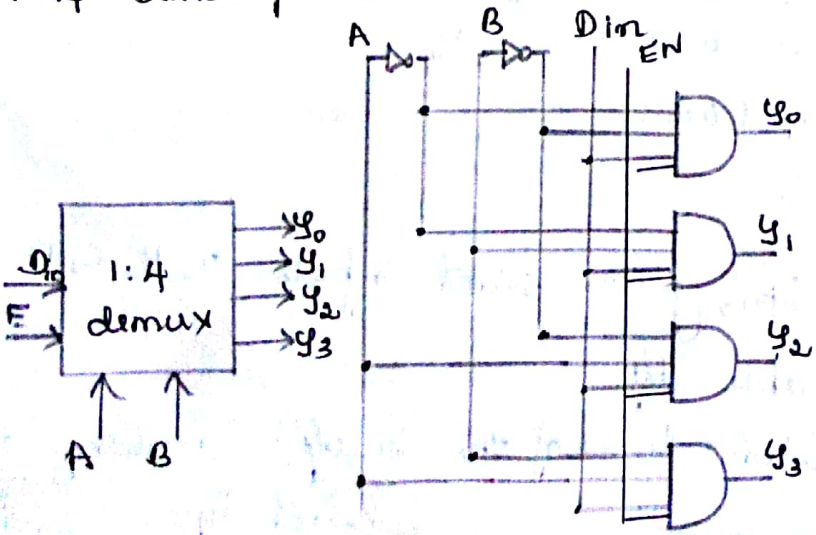
IC No.	DEMUX type	Decoder type
74154	1 to 16	4 to 16
74138	1 to 8	3 to 8
74155	1 to 4	2 to 4

Commercially available ICs.

Difference b/w Mux & Demux

Parameter	Mux	Demux
No. of data i/ps	2^n	1
No. of data o/ps	1	2^n
Relationship of i/p & o/p	Many to one	One to many
Applications	→ Used as a data selector → In time division multiplexing at transmitting end	→ Used as a data distributor → In time division multiplexing at receiving end.

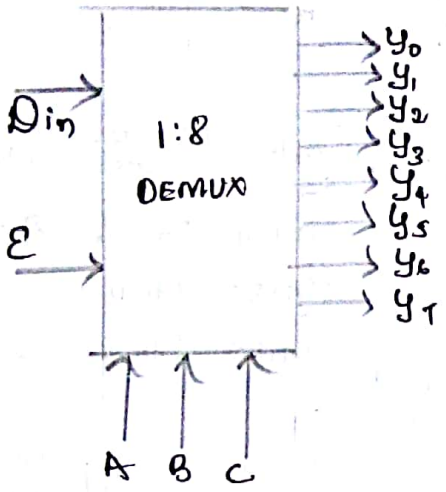
1:4 Demultiplexer:

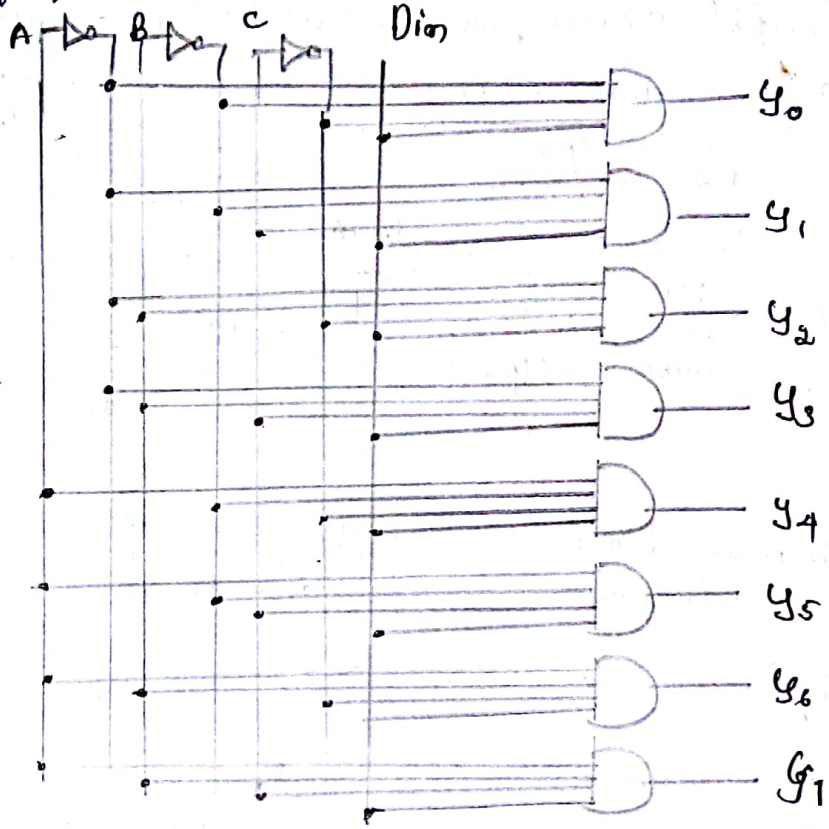


A	B	Din	y_0	y_1	y_2	y_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

1:8 Demultiplexer:

The single i/p data Din has a path to all 8 o/ps, but the i/p information is directed to only one of the o/p line depending on select line.





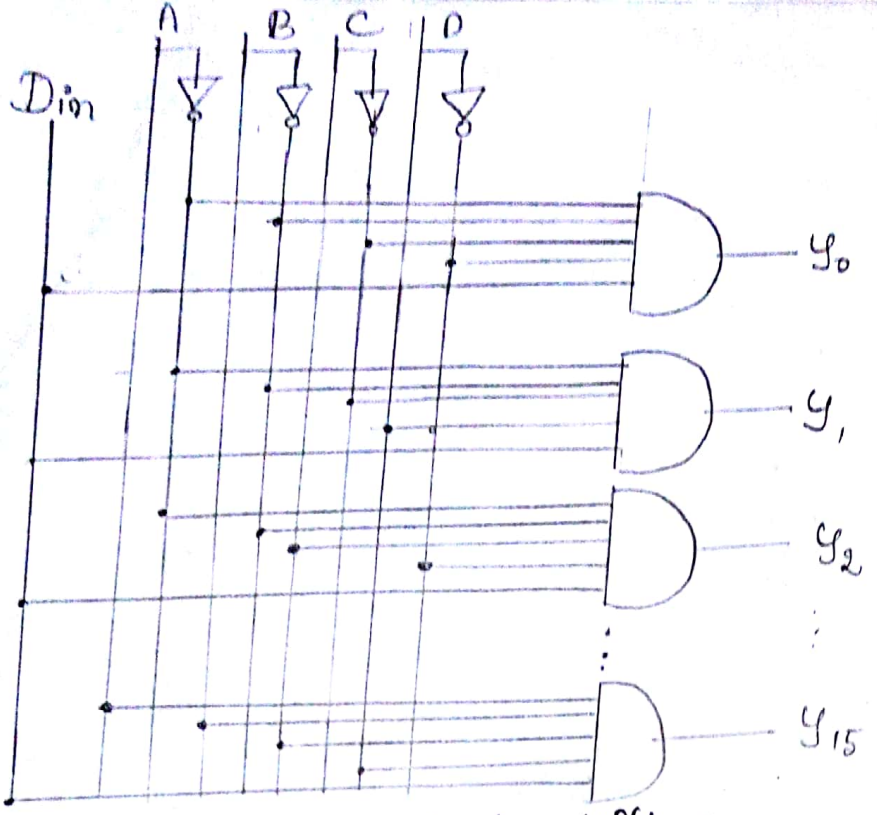
E	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	D_{in}	0	0	0	0	0	0	0
1	0	0	1	0	D_{in}	0	0	0	0	0	0
1	0	1	0	0	0	D_{in}	0	0	0	0	0
1	0	1	1	0	0	0	D_{in}	0	0	0	0
1	1	0	0	0	0	0	0	D_{in}	0	0	0
1	1	0	1	0	0	0	0	0	D_{in}	0	0
1	1	1	0	0	0	0	0	0	0	D_{in}	0
1	1	1	1	0	0	0	0	0	0	0	D_{in}

D_{in} may be high or low (0 or 1)

1:16 Demultiplexer:

It has one data input Din, four select inputs A, B, C, D 16 outputs and 1 enable i/p

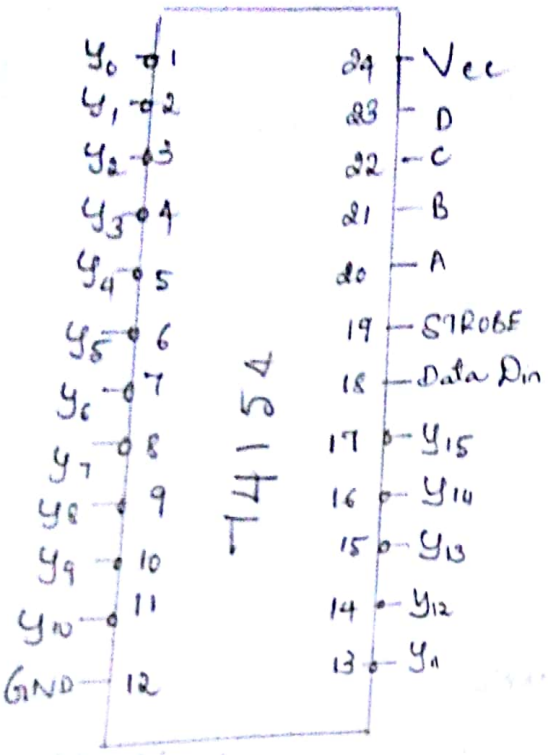
The i/p Din is routed to 1 of the 16 o/ps according to select i/ps.



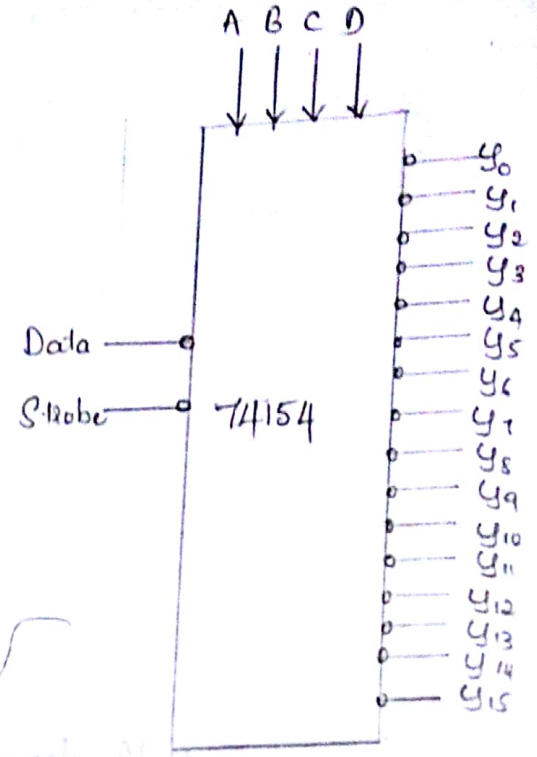
1 to 16 demultiplexer

S	A	B	C	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
	0	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	Din	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	0	0	0	0	Din	0	0	0	0	0	0	0	0	0	0
	0	1	1	0	0	0	0	0	0	0	Din	0	0	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	0	0	0	Din	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0	0	0	0	0
	1	0	0	1	0	0	0	0	0	0	0	0	0	Din	0	0	0	0	0	0
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0	0	0
	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0	0
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0
	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Din	0	0
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Din	0
	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Din
	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Truth Table for 16 1 to 16 demux.

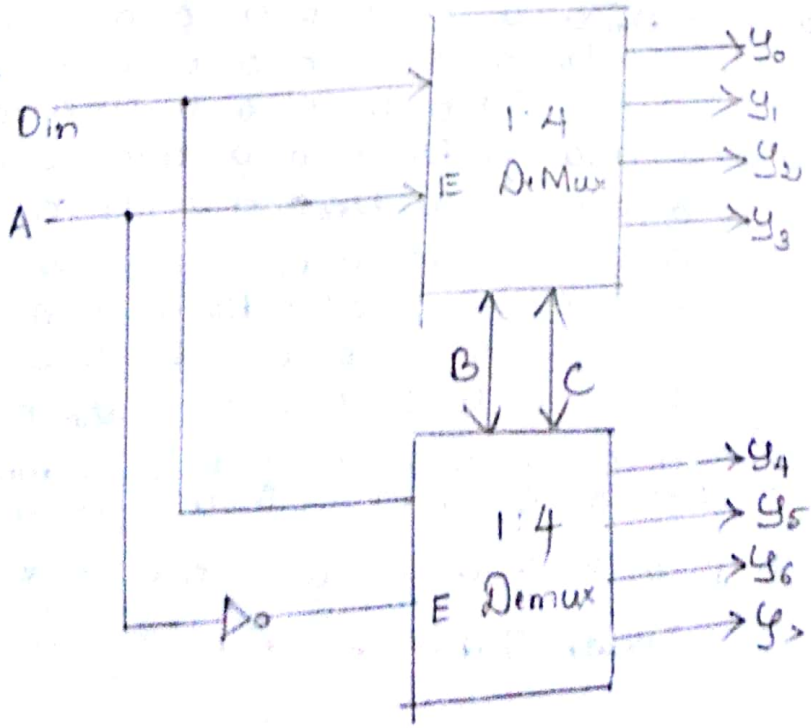


Pinout Diagram.



Logic Diagram of 74154

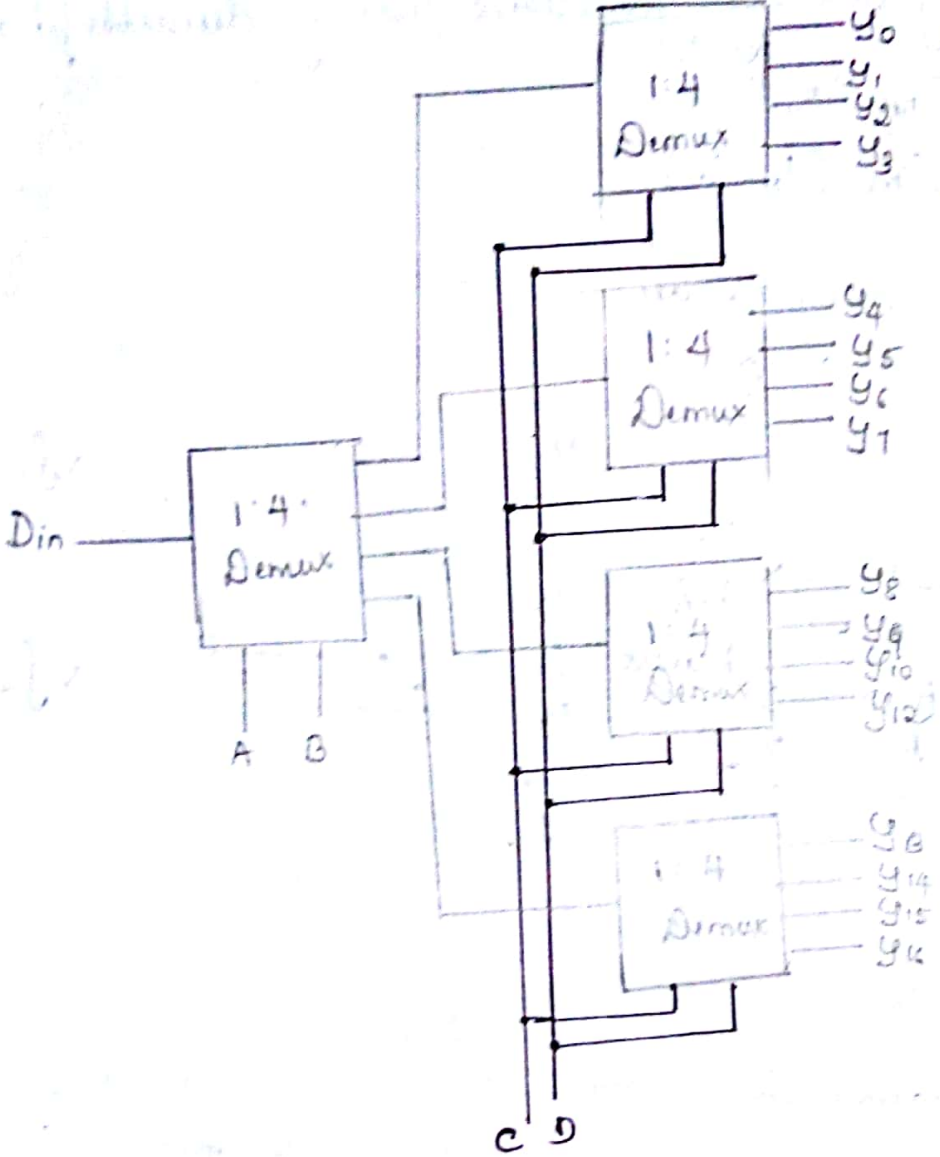
Design 1:8 Demultiplexer using 2 1:4 demultiplexers.



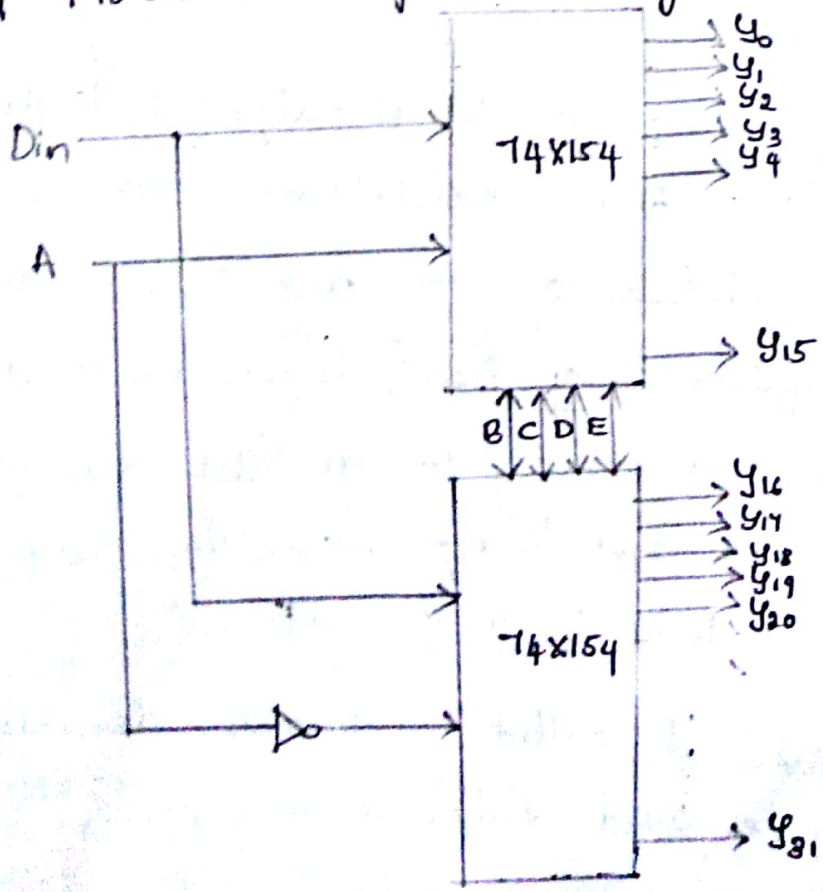
A=0 upper Demux is activated

A=1 lower demux is activated

Design 1:16 multiplexer using 1:4 demultiplexers.



Construct 1 to 32 Demultiplexer using 2 74x154 ics.

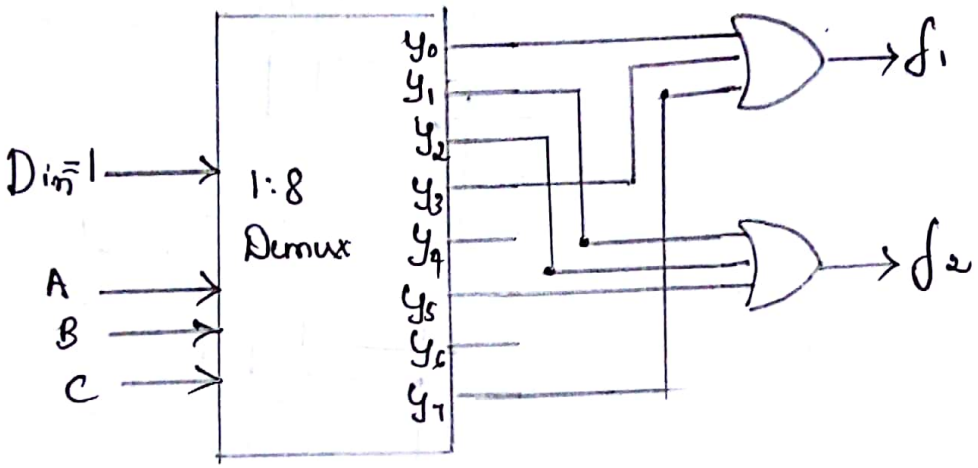


Implement the following functions using demultiplexer

$f_1(A, B, C) = \Sigma m\ (0, 3, 7)$

$f_2(A, B, C) = \Sigma m\ (1, 2, 5)$

Implementation using 1:8 demux



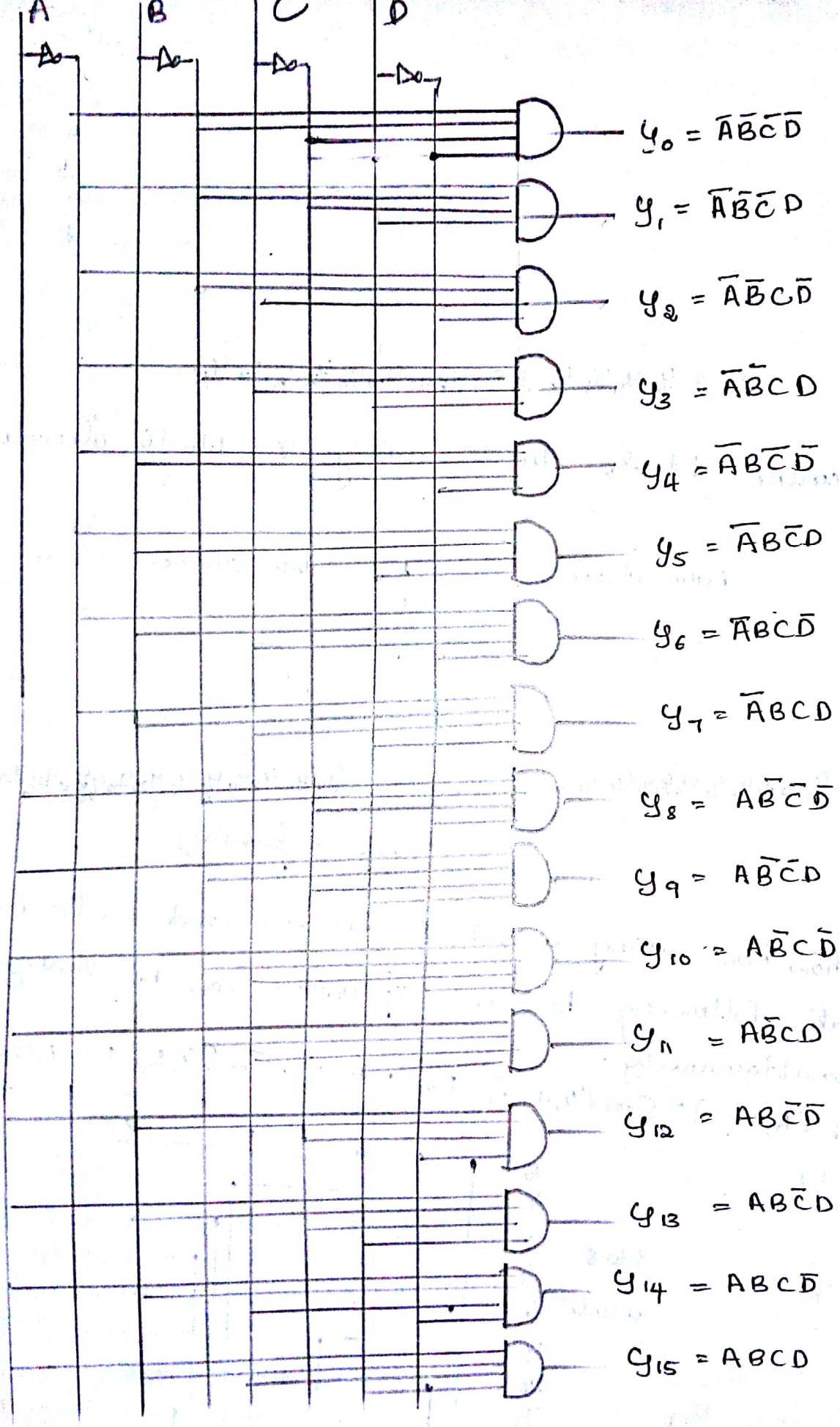
1 to 16 Decoder:

Decoder is identical to a demultiplexer without any data input. The only inputs are the control bits.

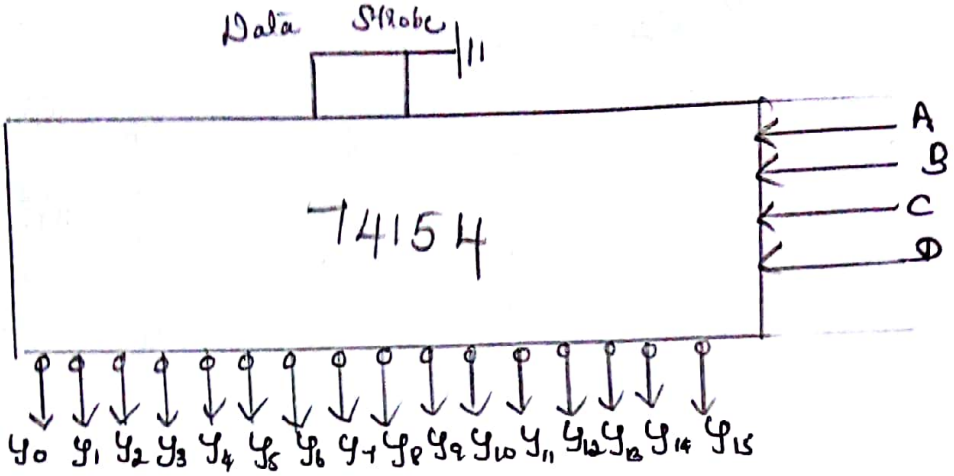
In 1 to 16 decoder one of the 16 output line is high. For instance when ABCD is 0001, only Y1 is output is high.

If we check the ABCD possibilities 0000 to 1111 we will find that the subscript of the high output always equals the decimal equivalent of ABCD. Hence 1 to 16 decoder is sometimes called a binary to decimal decoder. Beoz it has 4 i/p lines and 16 o/p lines, the circuit is also known as a 4 line to 16 line decoder.

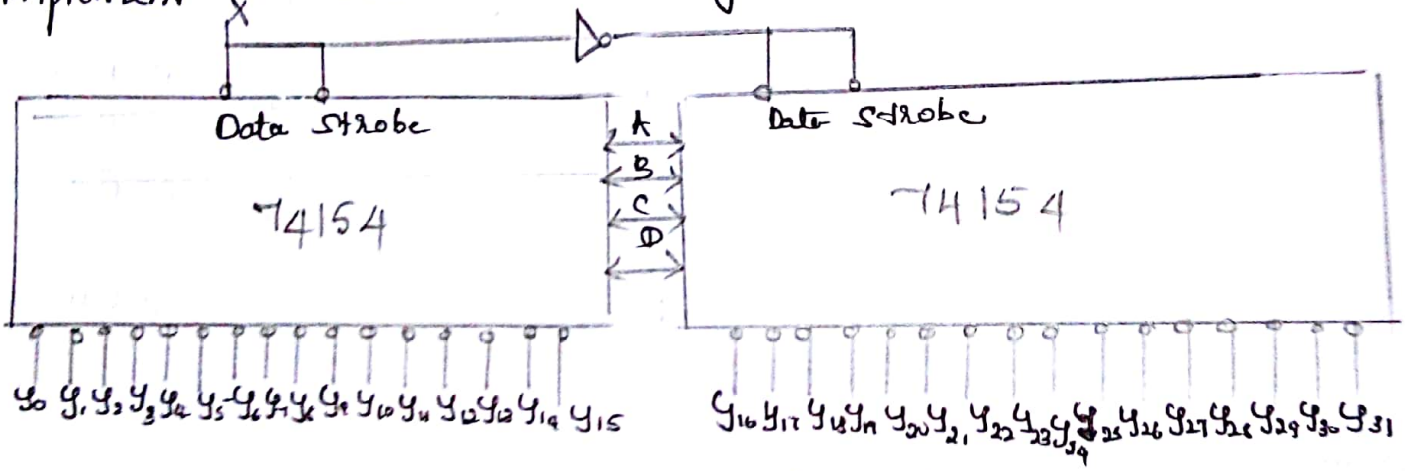
The 74154 IC is called a decoder - demultiplexer beoz it can be used either as a decoder or a demux.



To use 74154 IC as a decoder, ground the DATA & STROBE inputs



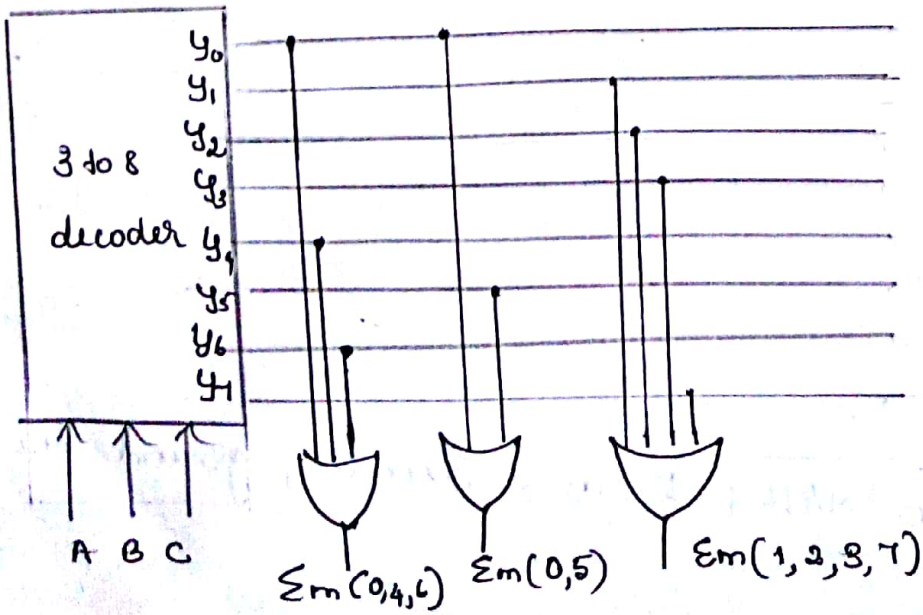
Implement 1 to 32 decoder using 2 1-to 16 decoder.



(Chip expansion)

Show how using a 3 to 8 decoder and multi input OR gate following Boolean expressions can be realized simultaneously.

$F1(A,B,C) = \Sigma m(0,4,6)$ $F2(A,B,C) = \Sigma m(0,5)$ $F3(A,B,C) = \Sigma m(1,2,3,7)$



BCD to Decimal Decoder:

BCD - Binary coded decimal

Convert decimal number 429 to BCD

4 2 9

0100 0010 1101

Convat 010101111000 BCD to decimal number.

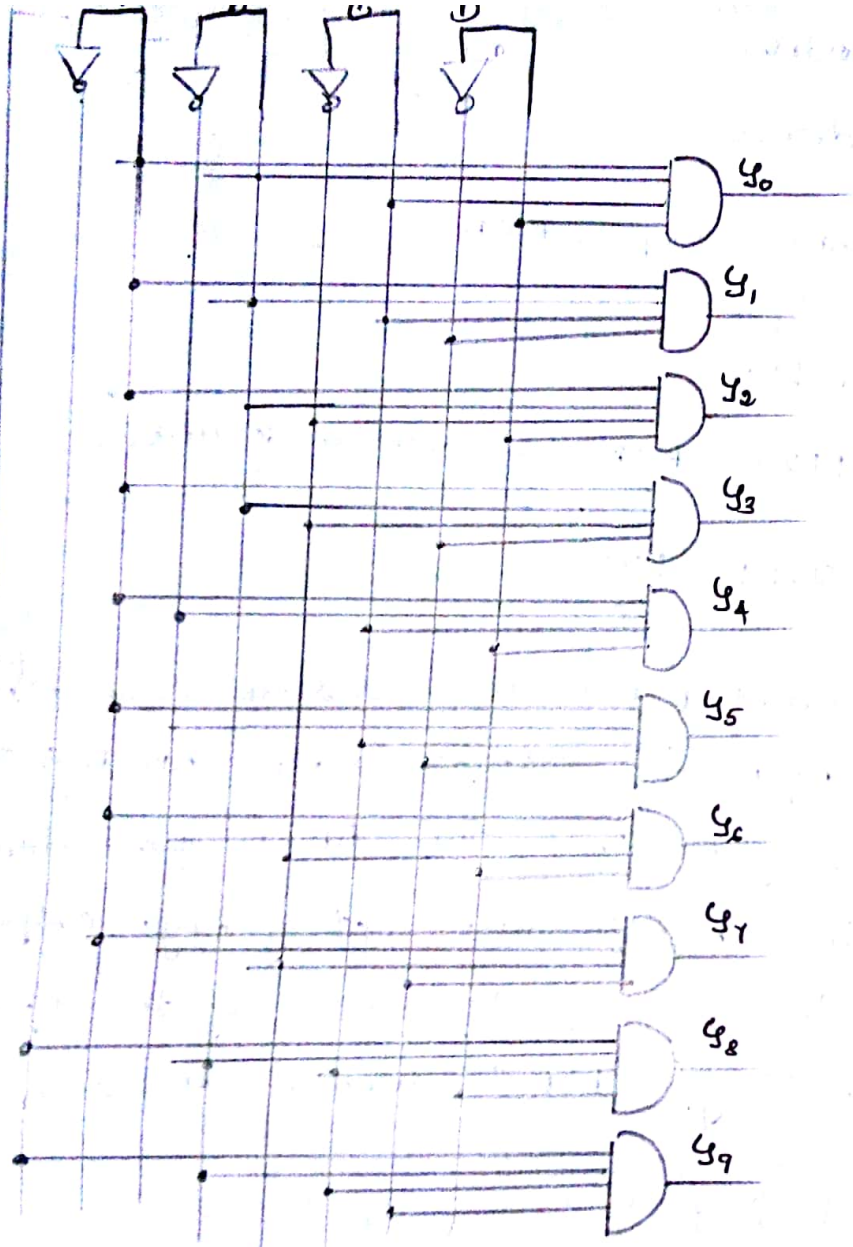
0101 0111 1000

5 7 8

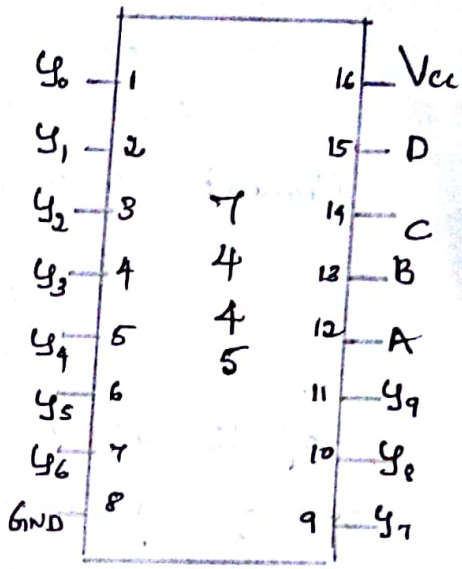
1 of 10 decoder is called BCD to decimal Decoder in which one of o/p lines is high. For example when ABCD is 0011 Y_3 output is high. If we check the ABCD poribilities (0000 to 1001) we find that the subscript of high output is always equals the decimal equivalent of the input BCD digit. 7445 is the IC for BCD to decimal decoder.

	inputs				output									
No.	A	B	C	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

Truth Table for 7445



1 of 10 decoder

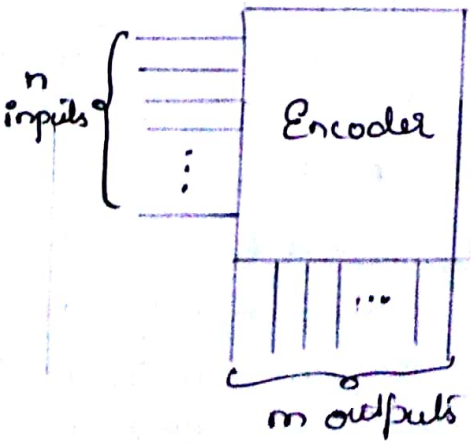


Pinout diagram of 7445

Encoders:

An encoder converts an active input signal into a coded output signal.

There are 'n' input lines, only one of which is active. Internal logic within the encoder converts this active input to a coded binary output with m bits. General idea is how in the fig.

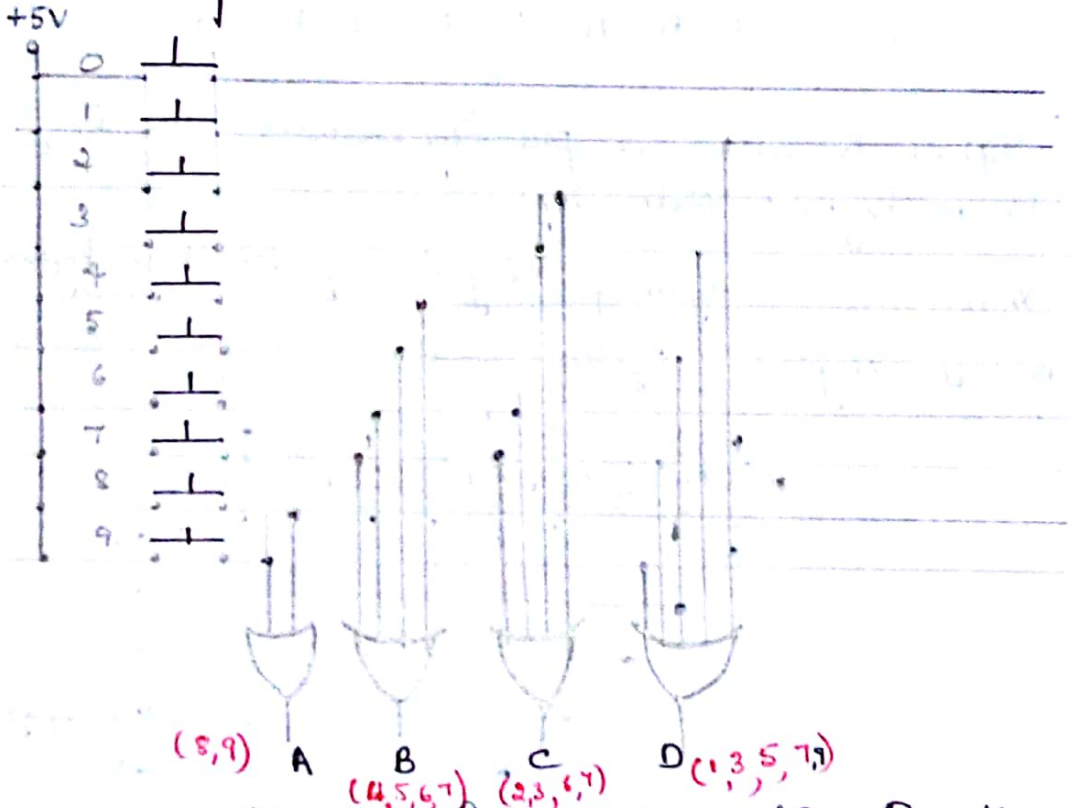


Encoder block diagram

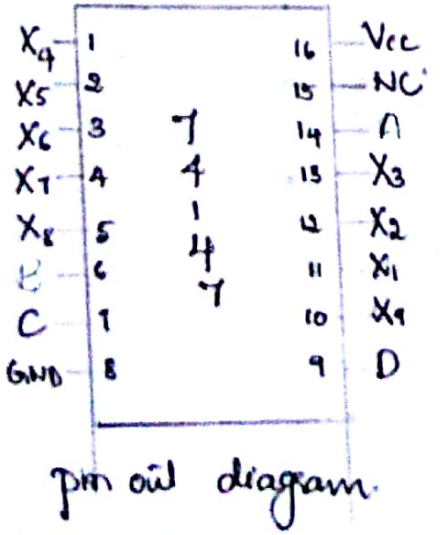
Decimal to BCD Encoder: It is a common type of encoder. In the decimal to BCD encoder the switches are push button switches like those of pocket calculator. When button 3 is pressed, the C and D OR gates have high inputs. therefore o/p is ABCD = 0011

if button 5 is pressed the o/p becomes ABCD = 0101

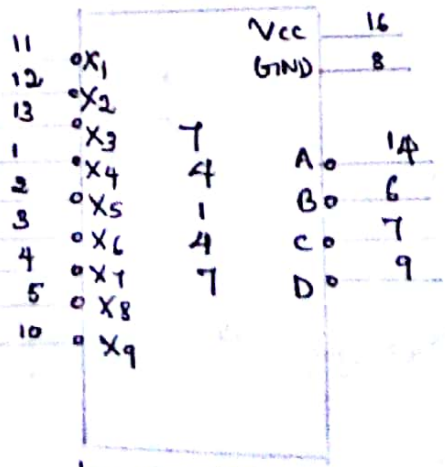
when switch 9 is pressed ABCD = 1001



74147 is the IC for decimal to BCD encoder Pin 16 is for the supply voltage & pin 8 is grounded, pin 15 is not used / no connection. Bubble indicate active low signal (i/p or o/p) and we get o/p in complemented form.



pin out diagram.



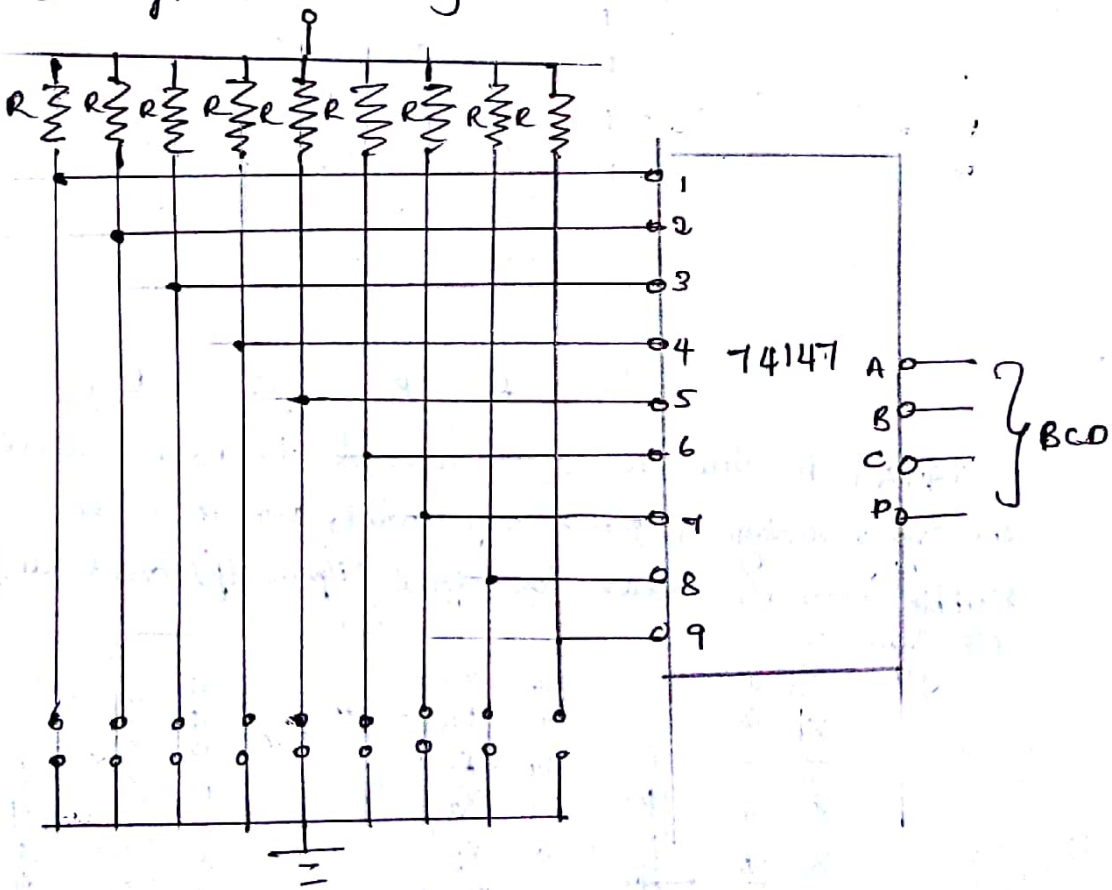
Logic diagram

Truth Table for 74147.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	A	B	C	D
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

74147 is called a priority encoder bcoz it gives priority to the highest order i/p.

Draw the interfacing diagram of ten key keypad interface to a digital system using decimal to BCD encoder



Design a 3 bit priority encoder.

Input				Output	
S	X_1	X_2	X_3	A	B
0	x	x	x	0	0
1	1	x	x	0	1
1	0	1	x	1	0
1	0	0	1	1	1
1	0	0	0	0	0

A	$\bar{S}\bar{X}_1$	$\bar{S}X_1$	SX_1	$S\bar{X}_1$
$\bar{X}_2\bar{X}_3$	0	0	0	0
$\bar{X}_2X_3$	0	0	0	1
X_2X_3	0	0	0	1
$X_2\bar{X}_3$	0	0	0	1

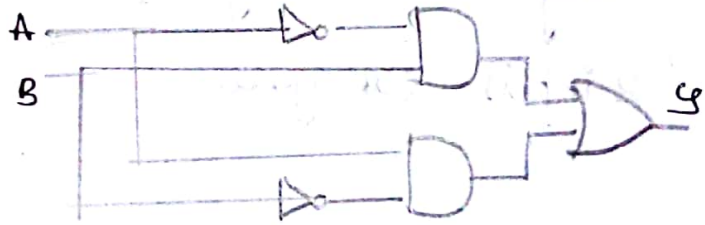
B	$\bar{S}\bar{X}_1$	$\bar{S}X_1$	SX_1	$S\bar{X}_1$
$\bar{X}_2\bar{X}_3$	0	0	1	0
$\bar{X}_2X_3$	0	0	1	1
X_2X_3	0	0	1	0
$X_2\bar{X}_3$	0	0	1	0

$A = S\bar{X}_1X_3 + S\bar{X}_1X_2$ .. $B = SX_1 + S\bar{X}_2X_3$

Exclusive OR gates:

The XOR gate has a high output only when an odd number of inputs is high.

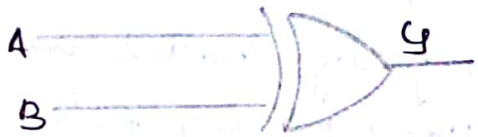
$$Y = A \oplus B$$
$$= \bar{A}B + A\bar{B}$$



XOR circuit using basic gate

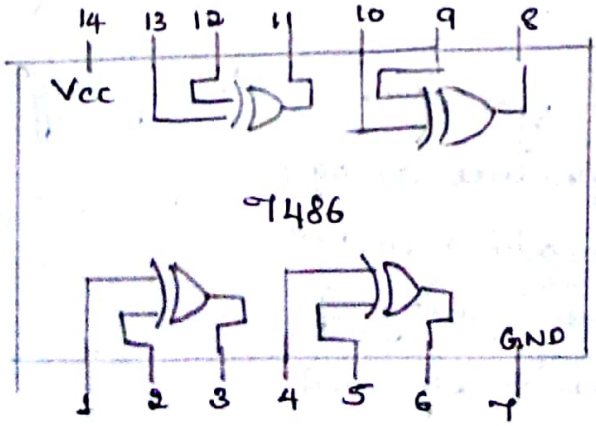
Truth table for 2 i/p XOR gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

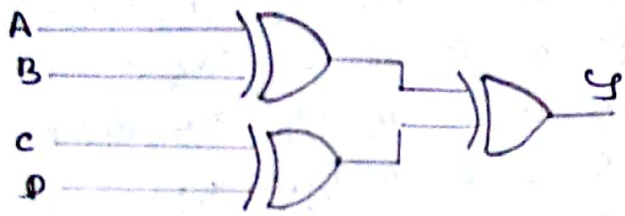


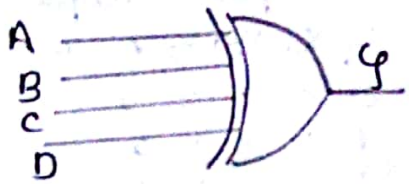
Logic Symbol

Pin diagram:



Four input XOR gate:





Write the truth table for 4 i/p XOR gate

Any Number of Inputs:

Using 2 input XOR gates we can produce XOR gates with any number of inputs.

3 input XOR gate

Parity Generators and Checkers:

Even parity means an n-bit input has an even number of 1s.
Eg: 110011 has even parity bcoz it contains 4 1s.

Odd parity means an n bit input has an odd number of 1s.
Eg: 110001 has odd parity bcoz it contains 3 1s.

Parity Checker:

XOR gates are ideal for checking the parity of a binary number bcoz they produce an o/p 1 when the i/p has an odd number of 1s. Therefore for even parity o/p will be 0 and odd parity o/p will be 1.

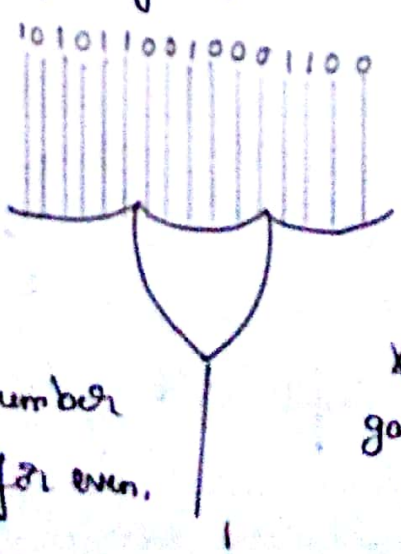


Fig. shows 16 input XOR gate. A 16 bit number drives the i/p. Output is 1 for odd parity 0 for even.

XOR gate with 16 i/ps.

Parity Generator.

A binary number may represent a instruction to addition/ subtraction etc operation to computer or binary number may represent a number, letter etc.. Sometimes we add a bit / binary number to produce a new binary number with even/odd parity.

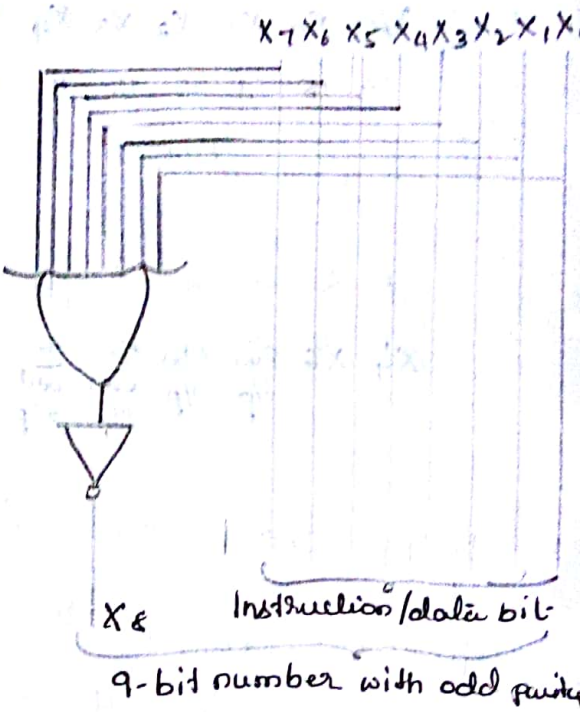


Fig shows the 8 bit binary number $X_7X_6X_5X_4\ X_3X_2X_1X_0$

if number is 0100 0001
number has even parity, which means XOR gate produces o/p 0
becoz of NOT gate $X_8 = 1$
final o/p will be 10100 0001
(number has odd parity now)

if number is 0110 0001
odd parity number, XOR produces o/p 1, NOT gate o/p will be 0
final o/p will be 0 0110 0001 (odd parity). Above circuit is an odd parity generator bcoz alway produces 9 bit o/p with odd parity.

Applications:

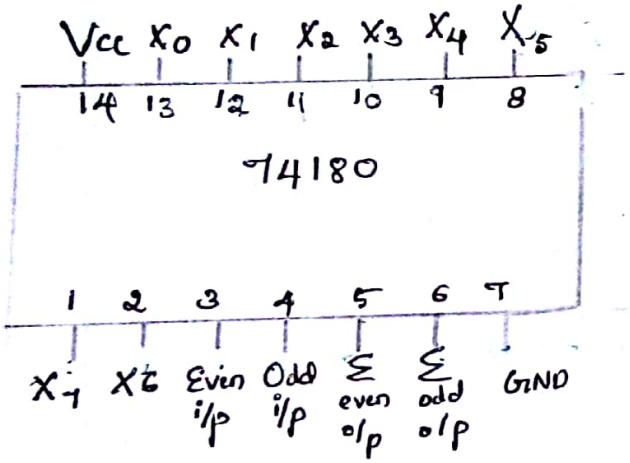
→ check for error at transmitting end receiving end of communication path.

→ they can check only 1 bit error using parity generator.

74180: TTL IC for parity generator (8 bit), 74180 can be used to detect even/odd parity. It can be set up to generate even or odd parity.

$X_0 \ldots X_7$ – input data bits, even & odd I/p control the operation of the chip. (21)

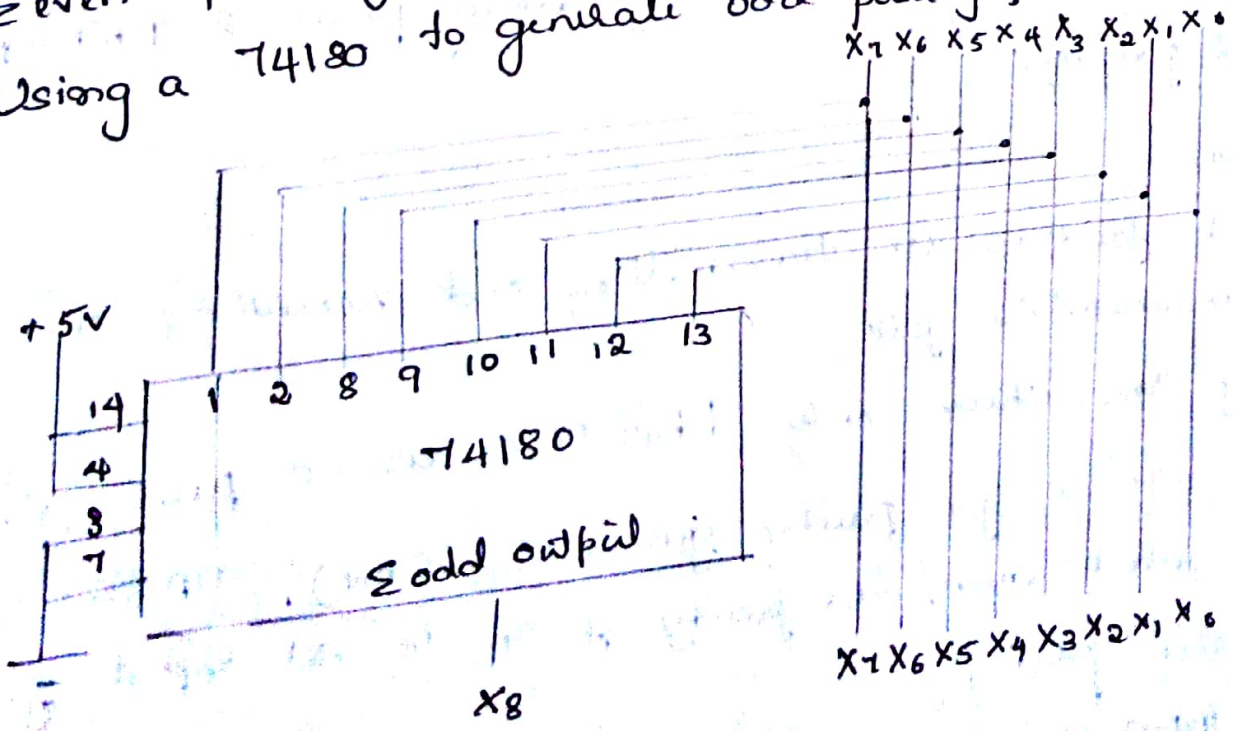
For example, suppose even i/p is high & odd i/p is low. When the i/p data has even parity the Σeven o/p is high & Σodd o/p is low. When the i/p data has odd parity, the Σeven o/p is low and the Σodd o/p is high.



	X_7 to X_0	Even	odd	Σeven	Σodd
H	Even	H	L	H	L
	Odd	H	L	L	H
L	Even	L	H	L	H
	Odd	L	H	H	L
	X	H	H	L	L
	X	L	L	H	H

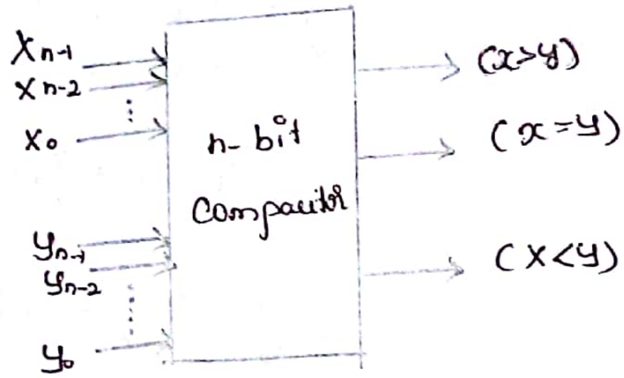
If we change the control i/ps (even odd) we can change the operation. Assume that the even i/p is low and the odd i/p is high. When the i/p data has even parity, the Σeven o/p is low and Σodd o/p is high. When the i/p data has odd parity, the Σeven o/p is high and the Σodd is low.

Using a 74180 to generate odd parity.



Magnitude Comparator:

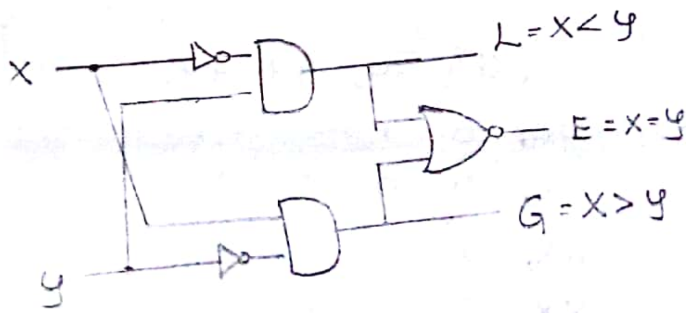
Magnitude Comparator compares magnitude of two n-bit binary numbers, say X and y and activates one of these 3 outputs X=y, X>y and X<y.



1 bit Magnitude Comparitor:

Truth Table:

Inputs		Outputs		
X	y	G X>y	E X=y	L X<y
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0



Solving Truth Table using Kmap

For G:

	$\bar{y}$	y
$\bar{X}$	0	0
X	1	0

$G = x\bar{y}$

For E

	$\bar{y}$	y
$\bar{X}$	1	0
X	0	1

$E = xy + \bar{x}\bar{y}$

$= \overline{(x\bar{y} + \bar{x}y)}$

$= \overline{(G+L)}$

For L

	$\bar{y}$	y
$\bar{X}$	0	1
X	0	0

$L = \bar{x}y$

2 - bit Comparator :

Truth Table

X		Y		G	E	L
X_1	X_0	Y_1	Y_0	$X>Y$	$X=Y$	$X<Y$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K map Simplification :

	$\bar{Y}_1\bar{Y}_0$	$\bar{Y}_1Y_0$	Y_1Y_0	$Y_1\bar{Y}_0$
$\bar{X}_1\bar{X}_0$	0	0	0	0
$\bar{X}_1X_0$	1	0	0	0
X_1X_0	1	1	0	1
$X_1\bar{X}_0$	1	1	0	0

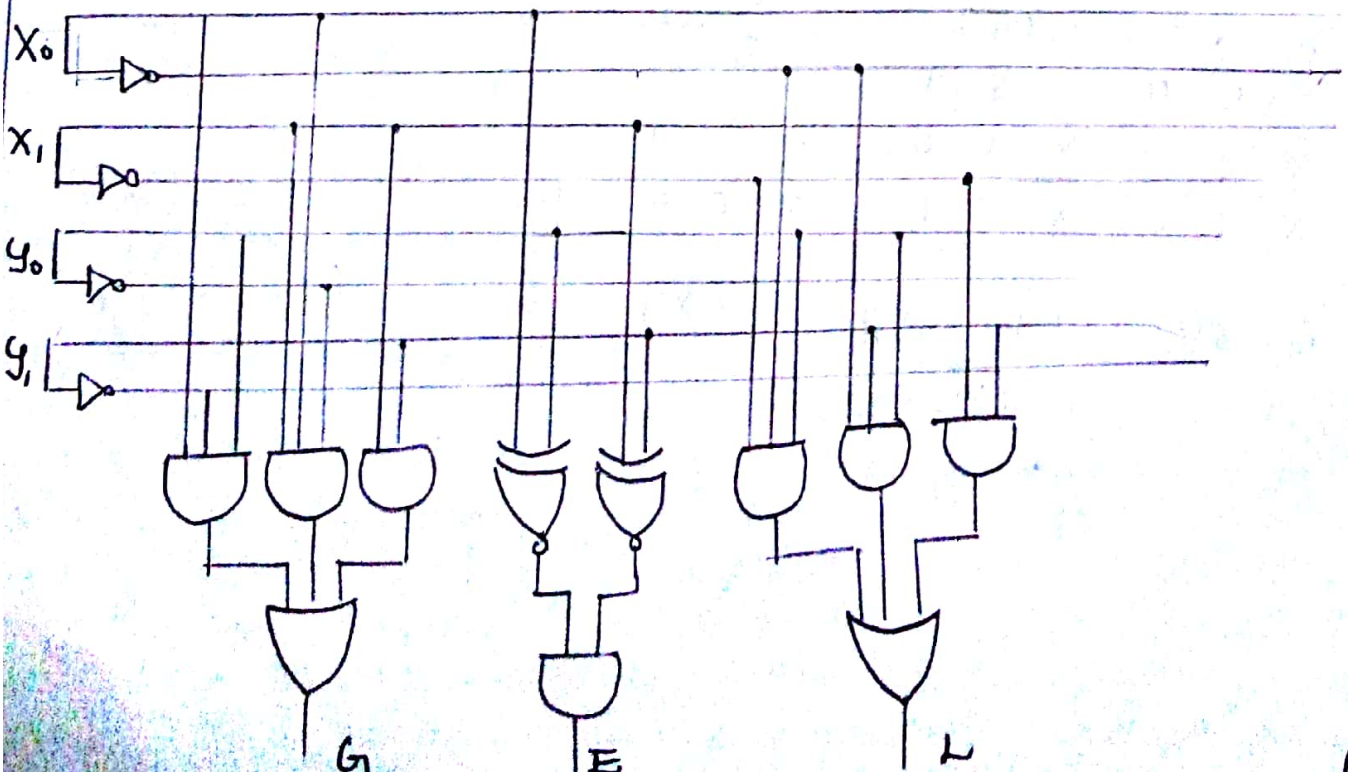
$$G = X_0\bar{Y}_1Y_0 + X_1X_0\bar{Y}_0 + X_1\bar{Y}_1$$

	$\bar{Y}_1\bar{Y}_0$	$\bar{Y}_1Y_0$	Y_1Y_0	$Y_1\bar{Y}_0$
$\bar{X}_1\bar{X}_0$	1	0	0	0
$\bar{X}_1X_0$	0	1	0	0
X_1X_0	0	0	1	0
$X_1\bar{X}_0$	0	0	0	1

$$E = \bar{X}_1\bar{X}_0\bar{Y}_1\bar{Y}_0 + \bar{X}_1X_0\bar{Y}_1Y_0 + X_1X_0Y_1Y_0 + X_1\bar{X}_0Y_1\bar{Y}_0$$
$$= \bar{X}_1\bar{Y}_1(\bar{X}_0\bar{Y}_0 + X_0Y_0) + X_1Y_1(X_0Y_0 + \bar{X}_0\bar{Y}_0)$$
$$= (X_0 \odot Y_0)(X_1 \odot Y_1)$$

	$\bar{Y}_1\bar{Y}_0$	$\bar{Y}_1Y_0$	Y_1Y_0	$Y_1\bar{Y}_0$
$\bar{X}_1\bar{X}_0$	0	1	1	1
$\bar{X}_1X_0$	0	0	1	1
X_1X_0	0	0	0	0
$X_1\bar{X}_0$	0	0	1	0

$$L = \bar{X}_1\bar{X}_0Y_0 + \bar{X}_0Y_1Y_0 + \bar{X}_1Y_1$$



Using truth table and getting logic equation through any simplification technique becomes very complex when magnitude of bit increases. So using 1 bit comparator logic equation we will find logic equation for 2 bit and 4 bit magnitude comparator.

bit-wise greater than terms (G): $G_1 = X_1\overline{Y_1}$ $\quad G_0 = X_0\overline{Y_0}$

bit-wise less than terms (L): $L_1 = \overline{X_1}Y_1$ $\quad L_0 = \overline{X_0}Y_0$

bit-wise equality terms (E): $E_1 = \overline{(G_1 + L_1)}$ $\quad E_0 = \overline{(G_0 + L_0)}$

From above equations 2 bit comparator outputs are

$$E = E_1 E_0 \qquad G = G_1 + E_1 G_0 \qquad L = L_1 + E_0 L_0$$

because $X = Y$ when both the bits are equal.

$X > Y$ if MSB of X is higher than MSB of Y

if MSB is equal $E_1 = 1$, then LSB of X & Y is checked.

Similarly $X < Y$ case.

For any 2 n bit numbers $X = X_{n-1} X_{n-2} \cdots X_0$

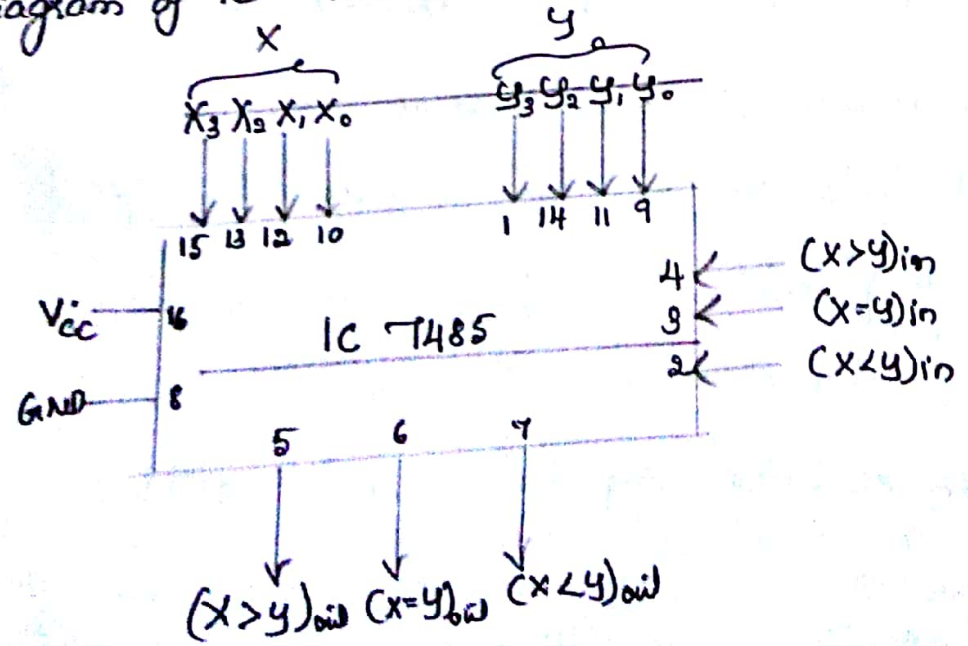
$Y = Y_{n-1} Y_{n-2} \cdots Y_0$

$$E(X=Y) = E_{n-1} \cdot E_{n-2} \cdots E_0$$

$$G(X>Y) = G_{n-1} + E_{n-1} G_{n-2} + \cdots + E_{n-1} E_{n-2} \cdots E_1 G_0$$

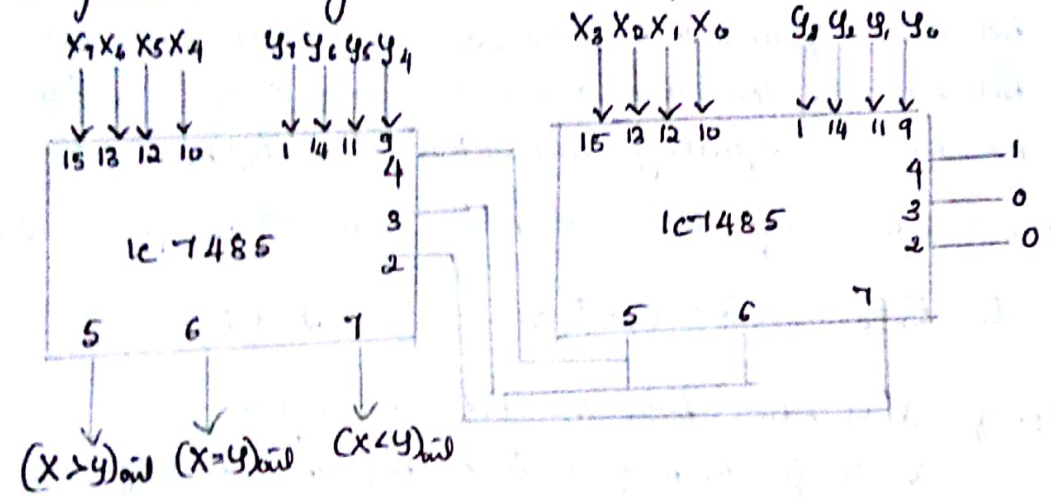
$$L(X<Y) = L_{n-1} + E_{n-1} L_{n-2} + \cdots + E_{n-1} E_{n-2} \cdots E_1 L_0$$

Block diagram of IC 7485 : 4 bit magnitude comparator.



These additional inputs $(X=Y)_{in}$ $(X>Y)_{in}$ $(X<Y)_{in}$ are used to connect more than one IC to compare numbers more than 4 bits. When IC 7485 is not used for cascade then $(X=Y)_{in} = 1$, $(X>Y)_{in} = 0$ and $(X<Y) = 0$.

8-bit comparator using 2 4-bit comparator.



Programmable Array Logic

PAL is a programmable array of logic gates on a single chip. Using PAL we can design solution to a SOP/POS/ multiplexer logic.

Programming a PAL:

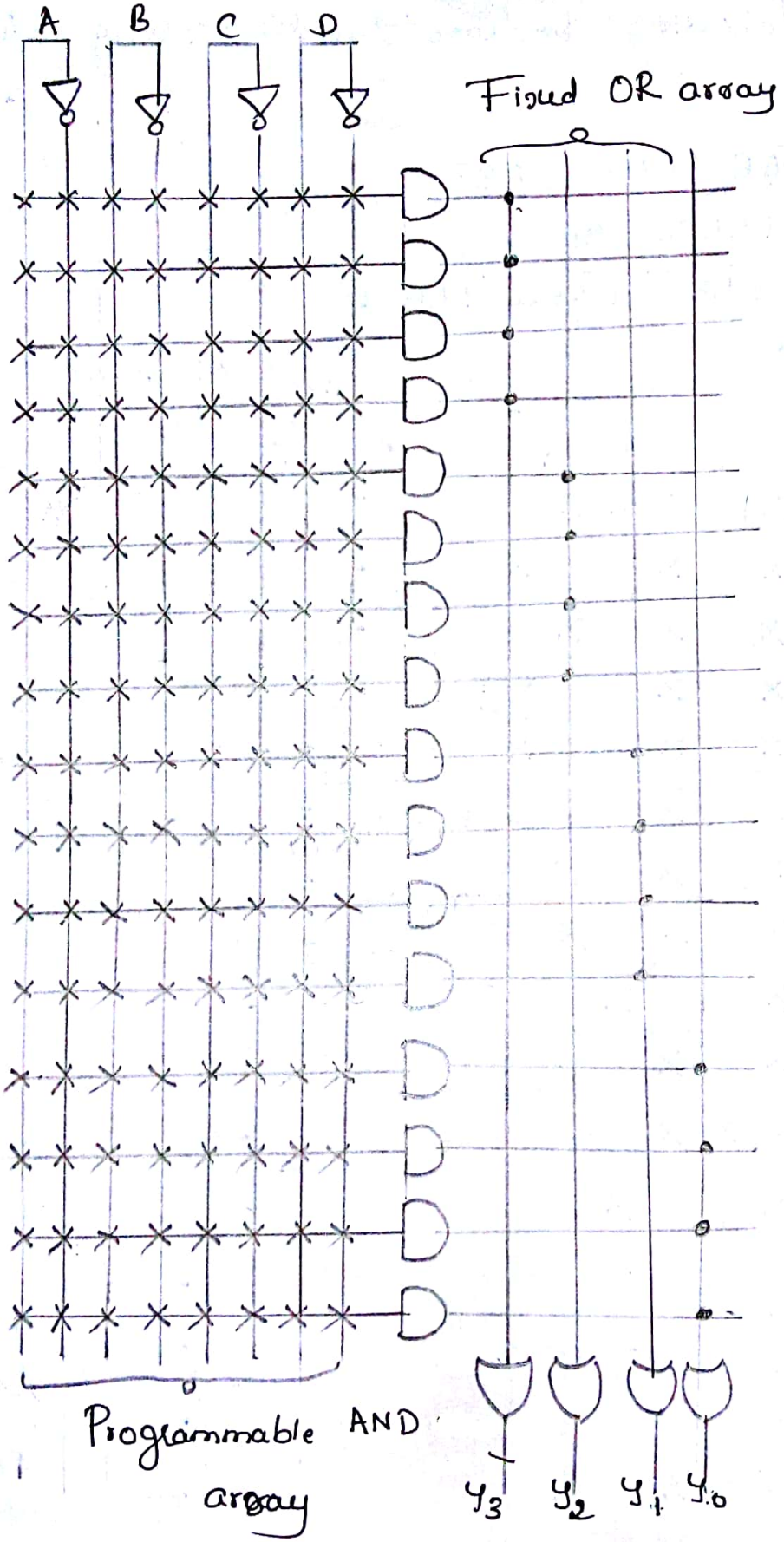
PAL has programmable AND array and a fixed OR array.

Consider a PAL with 4 inputs and 4 outputs.

X's on input side are fusible links.

Solid bullets on the output side are fixed connection

Hence AND gates are programmable. Hence we can burn in the desired fundamental products, which are then ORed by the fixed output connections.



Structure of PAL

Commercially Available PALs.

10H8 : 10 input and 8 output AND-OR
16H2 : 16 input and 2 output AND-OR
14L4 : 14 input and 4 output AND-OR-INVERTER

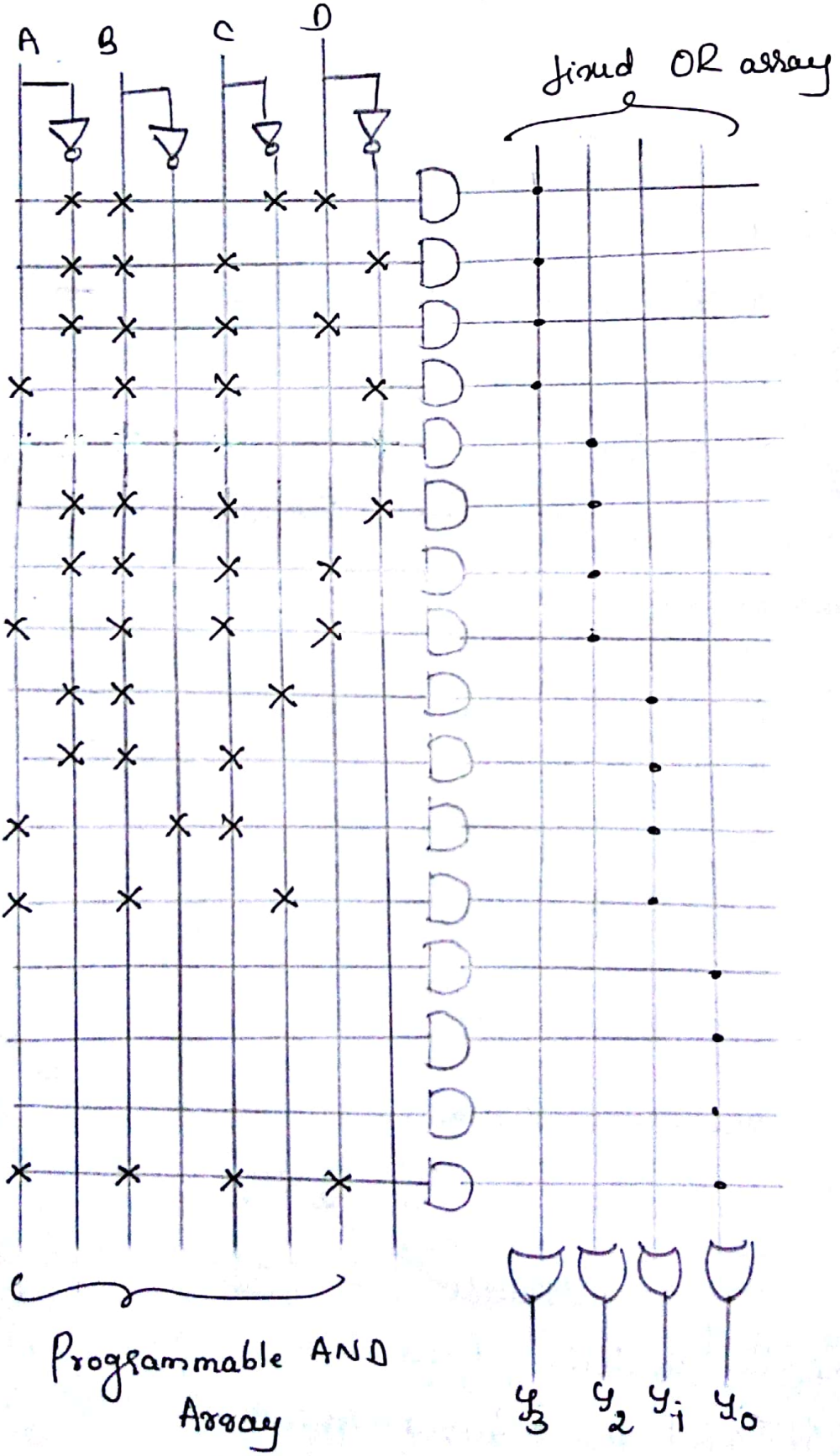
Implement following boolean functions using PAL :

$Y_0 = ABCD$

$Y_1 = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C}$

$Y_2 = \bar{A}BC\bar{D} + \bar{A}BCD + ABCD$

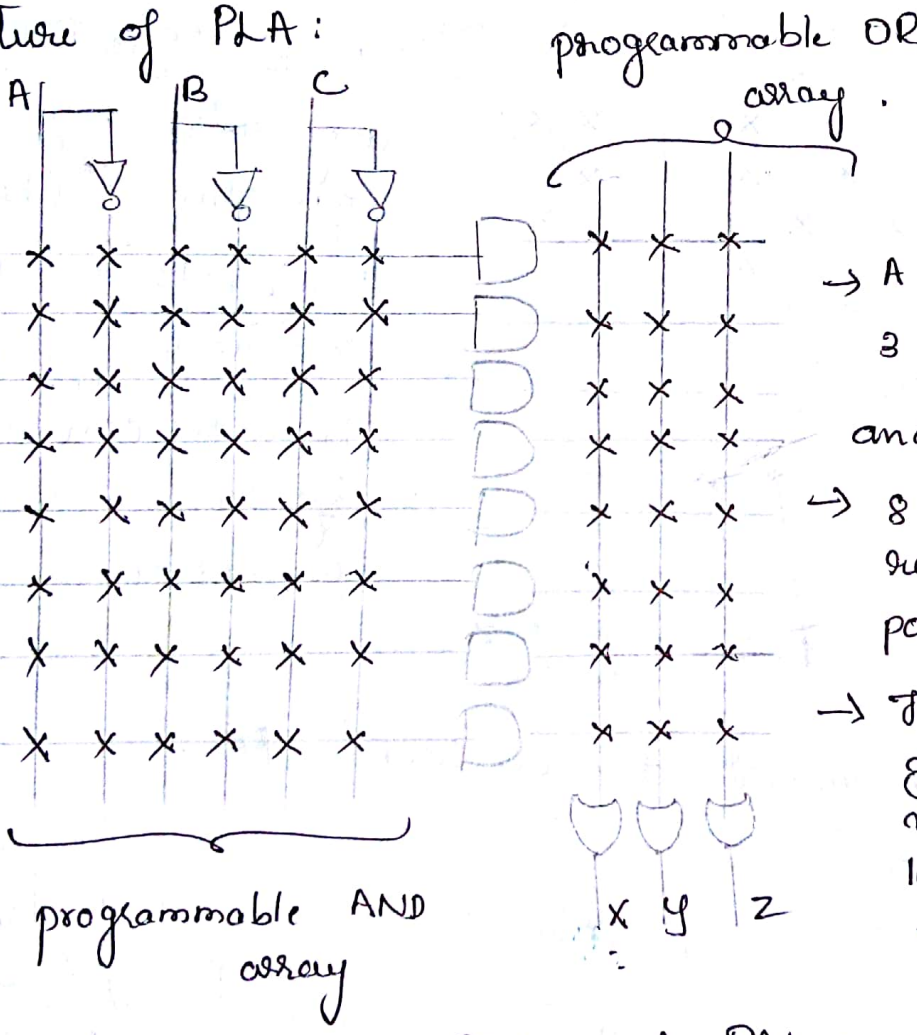
$Y_3 = \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + ABC\bar{D}$



Programmable Logic Arrays:

A programmable logic array is an another solution for SOP / POS / multiplexer logic. PLA consists of programmable AND array followed by programmable OR array. i.e both AND and OR array are fusible linked.

Structure of PLA:



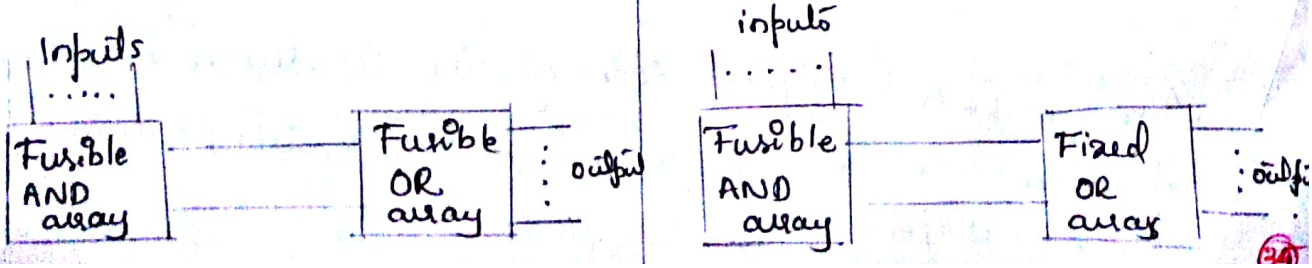
→ A PLA having 3 inputs (A, B, C) and 3 outputs (X, Y, Z)

→ 8 AND gates are required to decode 8 possible i/p states

→ There are 3 OR gates that can be used to generate logic functions at the o/p.

Comparision between PLA and PAL:

PLA	PAL
Both AND and OR arrays are programmable	OR array is fixed and AND array is programmable
Costliest and complex than PAL	Cheaper and simpler.



Implement following functions using PLA

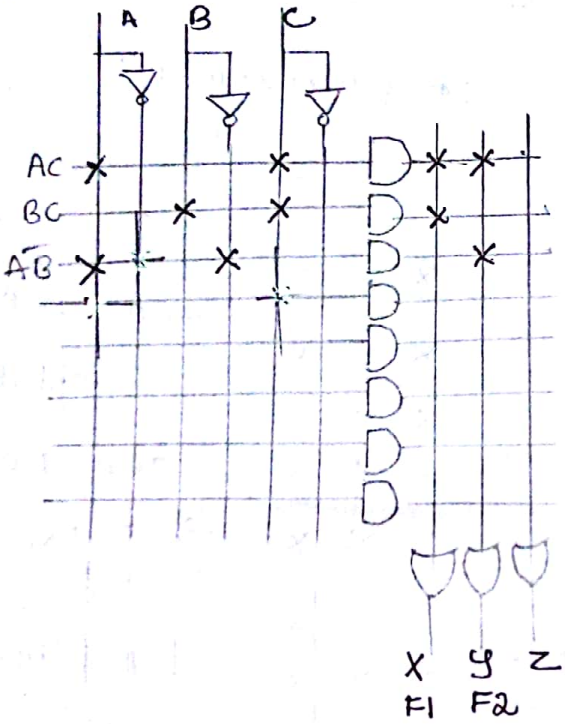
$F_1 = \Sigma m\,(3,5,7)$ $F_2 = \Sigma m\,(4,5,7)$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0_0	0_1	1_3	0_2
A	0_4	1_5	1_7	0_6

$F_1 = AC + BC$

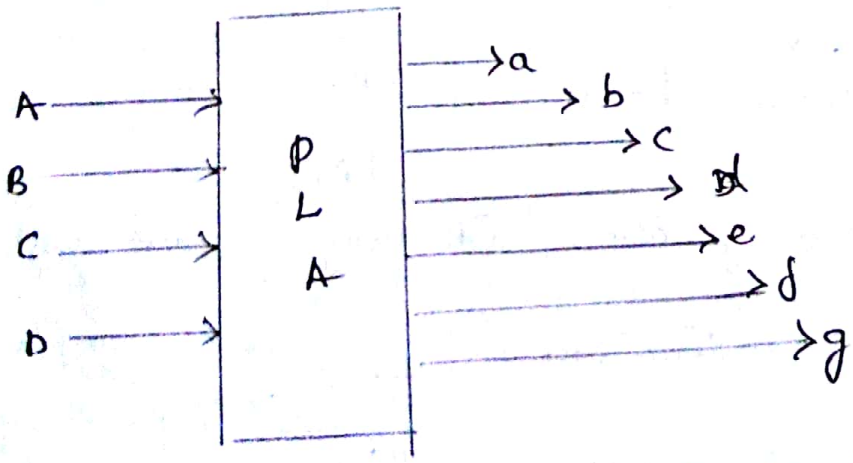
	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0_0	0_1	0_3	0_2
A	1_4	1_5	1_7	0_6

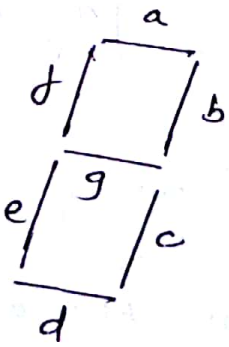
$F_2 = A\bar{B} + AC$



Note: PLAs, PALs, ROMs are called programmable logic device (PLDs)
Many PLDs are programmable only at the factory
PLDs that can be programmed by the user is called field programmable.

Implement 7 segments decoder using PLA.





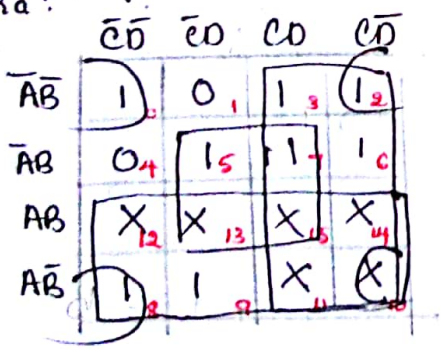
Use a PLA to suppress each of the 10 decimal digits represented in binary form and to correctly drive a 7 segment display.

PLA must have 4 inputs and 4 bits (ABCD) are required to represent the 10 decimal numbers. There must be 7 outputs (abcdefg), 1 output to drive each of the 7 segments of the indicator.

Digit	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

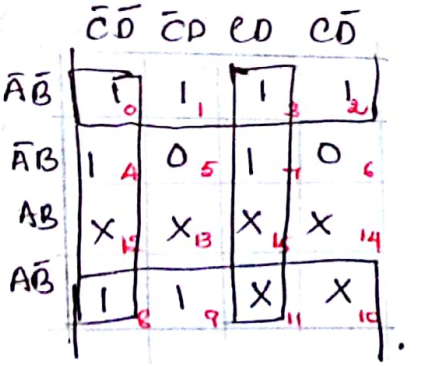
K Map Simplification:

For a:



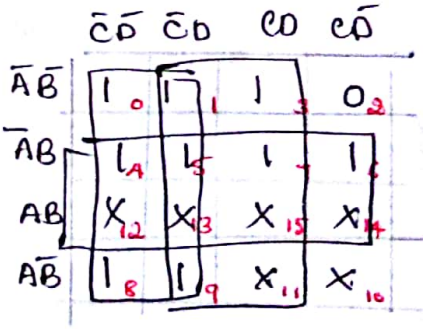
$a = A + C + BD + \bar{B}\bar{D}$

For b:



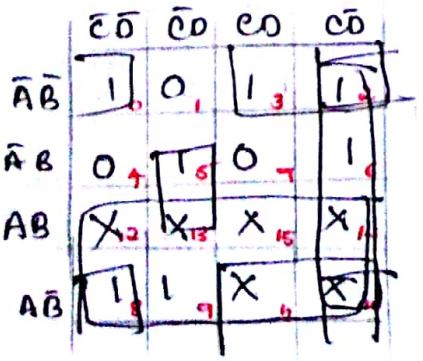
$b = \bar{B} + \bar{C}\bar{D} + CD$

For c



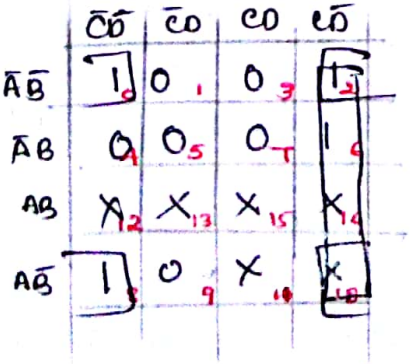
$c = \bar{C} + D + B$

For d:



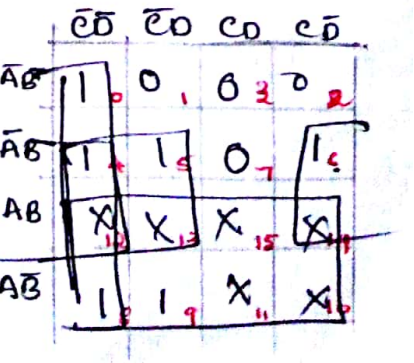
$d = A + C\bar{D} + \bar{B}C + \bar{B}\bar{D} + B\bar{C}D$

For e:



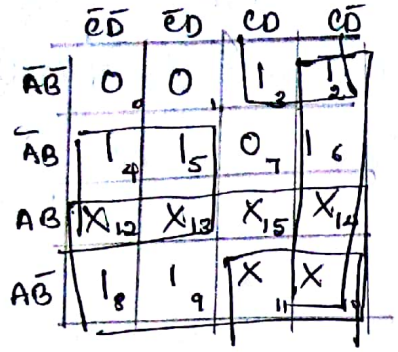
$e = C\bar{D} + \bar{B}\bar{D}$

For f:

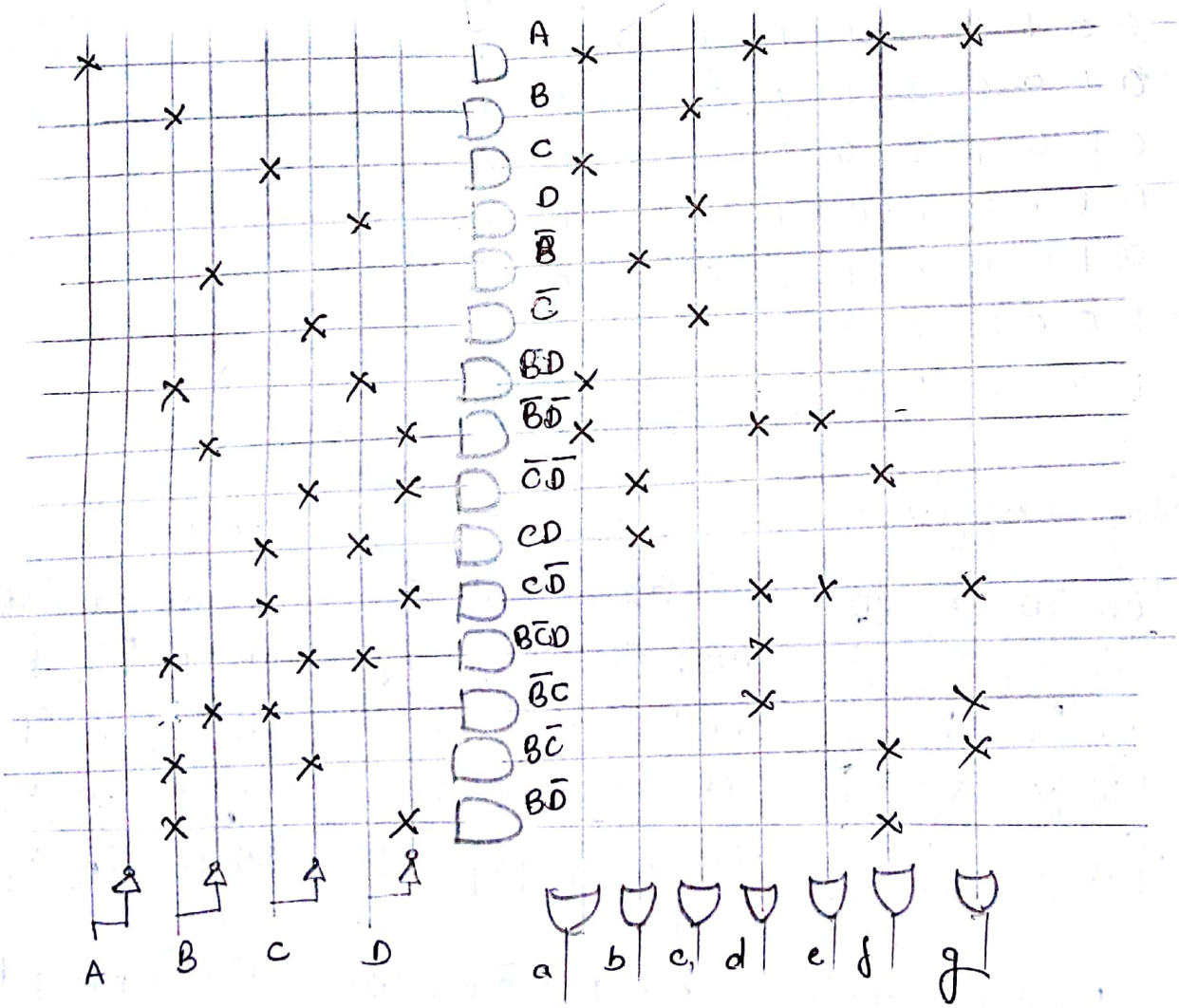


$f = A + \bar{C}\bar{D} + \bar{C}B + B\bar{D}$

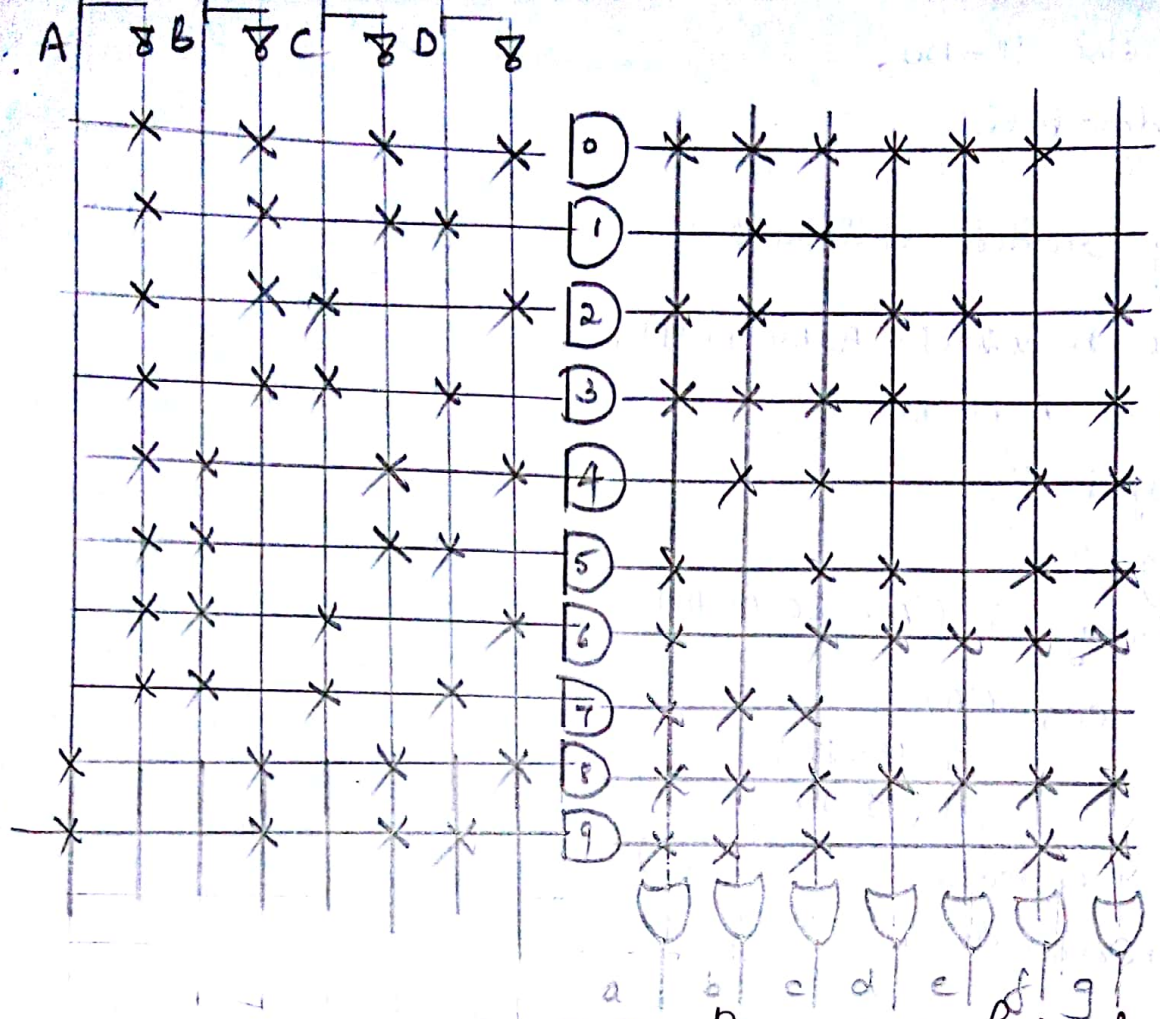
For f



$f = A + B\bar{C} + \bar{B}C + C\bar{D}$



We can implement directly without simplification.



HDL Implementation Of Data Processing Circuits:

2:1 MUX using Dataflow model

```
module mux2to1 (A, D0, D1, Y);
  input A, D0, D1;
  output Y;
  assign Y = (~A&D0) | (A&D1);
 // assign Y = A? D1: D0;
endmodule
```

2:1 MUX using Behavioural model

```
module mux2to1 (A, D0, D1, Y); // using else if
  input A, D0, D1;
  output Y;
  reg Y;
  always @ (A or D0 or D1)
     if (A==1) Y=D1;
```

```
    else y = D0;
endmodule

//using switch statement
module mux2to1 (A, D0, D1, y);
  input A, D0, D1;
  output y;
  reg y;
  always @ (A or D0 or D1)
    case (A)
      0: y = D0;
      1: y = D1;
    endcase
endmodule
```

Design a 4to1 MUX

i) using assign
ii) using case

```
i) module mux4to1 (A, B, D0, D1, D2, D3, y);
  input A, B, D0, D1, D2, D3;
  output y;
  assign y = A ? (B ? D3 : D2) : (B ? D1 : D0);
endmodule

ii) module mux4to1 (A, B, D0, D1, D2, D3, y);
  input A, B, D0, D1, D2, D3;
  output y;
  reg y;
  always @ (A or B or D0 or D1 or D2 or D3)
    case ({A, B})
      0: y = D0;
```

```
    1 : y = D1;
    2 : y = D2;
    3 : y = D3;
  endcase
endmodule
```

Write verilog code for a combinatorial logic circuit that compares 2 4 bit number A & B and generate 3bit output y :

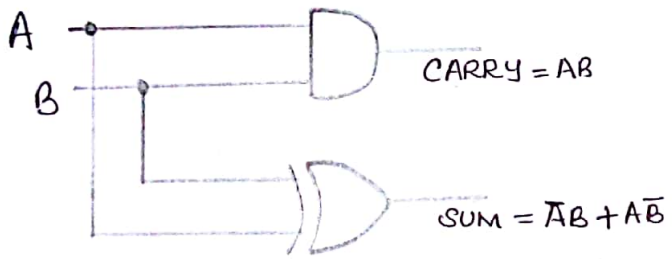
```
module comparator4b (A, B, y)
input [3:0] A, B;
output [2:0] y;
reg [2:0] y;
always @ (A or B)
    if (A<B) y = 3'b001
    else if (A>B) y = 3'b010
    else y = 3'b100
endmodule
```

Arithmetic Building Blocks:

The half adder, full adder, and the controlled inverter are used as arithmetic building blocks.

Half-Adder:

The circuit used to add two bits with the possibility of a carry is called a half adder.



Circuit diagram

A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth Table

The output of the AND gate is called the carry and the output of the exclusive OR gate is called Sum.

Full-Adder:

A logic circuit that can add 3 bits at a time is called full adder. The third bit is the carry from a lower column.

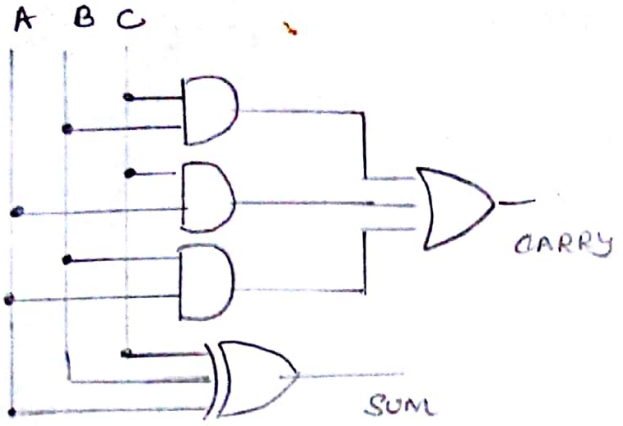
A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	0	0	1	0
1	1	1	1	1

C \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	0	1	0
C	0	1	1	1

Carry = $AB + BC + AC$

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	1	0	1
C	1	0	1	0

Sum = $A \oplus B \oplus C$



$CARRY = AB + BC + AC$

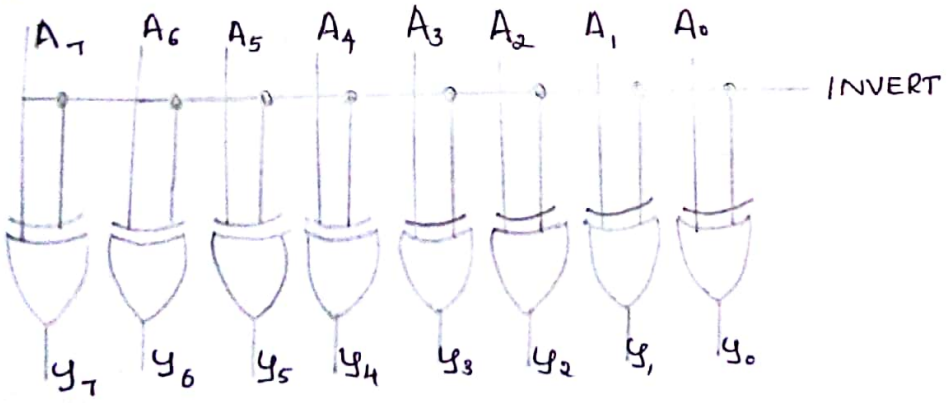
$SUM = A \oplus B \oplus C$

A general representation of full adder which adds i^{th} bit A_i and B_i of two numbers A and B takes carry from $(i-1)^{th}$ bit could be

$C_i = A_iB_i + B_iC_{i-1} + A_iC_{i-1}$ or $\quad S_i = A_i \oplus B_i \oplus C_{i-1}$

$C_i = A_iB_i + (A_i + B_i)C_{i-1}$

Controlled Inverter :



Above figure shows a controlled inverter. When INVERT is low, it transmits the 8 bit input to the output, when INVERT is high it transmits the 1's complement.

For example, if the input number is $A_7 \cdots A_0 = 01101110$

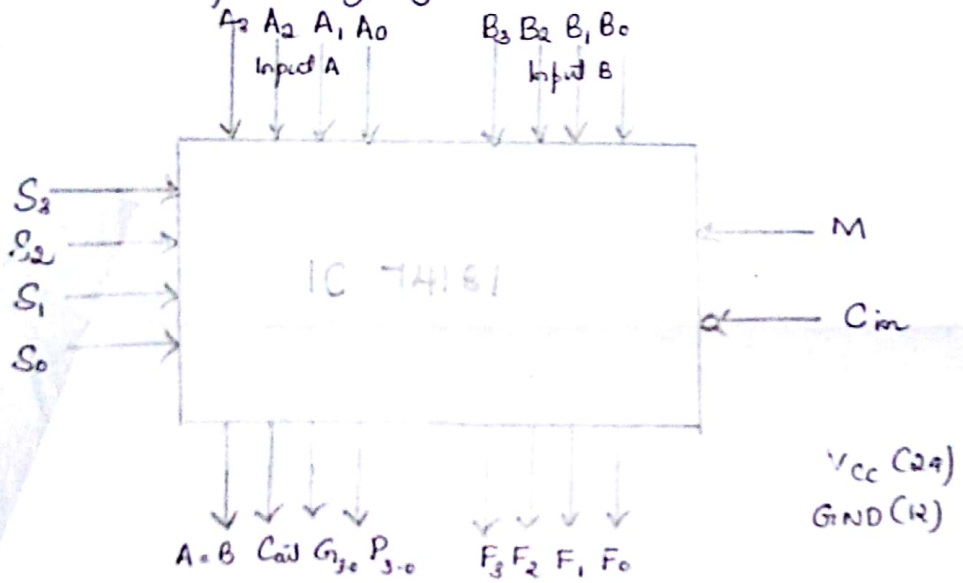
a low INVERT produces $Y_7 \cdots Y_0 = 01101110$

a high INVERT results in $Y_7 \cdots Y_0 = 10010001$

The controlled inverter is important because it is a step in the right direction.

Arithmetic Logic Unit:

Arithmetic Logic Unit (ALU) is multifunctional device that can perform both arithmetic and logic function. ALU is an integral part of central processing unit or CPU of a computer. It does wide range of functions. From normal addition, subtraction to increment, decrement operations. As logic unit it performs AND, OR, NOR, Ex-OR and many other complex logic functions.



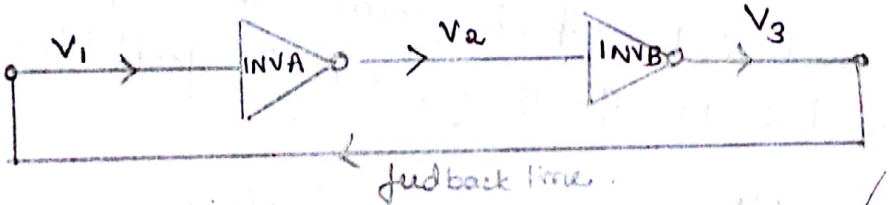
				M=1	M=0
S_3	S_2	S_1	S_0	Logic Function	Arithmetic Function $C_{in}=1$ (for $C_{in}=0$, add 1 to F)
0	0	0	0	$F = A'$	F = A
0	0	0	1	$F = (A+B)'$	F = A+B
0	0	1	0	$F = A'B$	F = A+B'
0	0	1	1	F = 0	F = minus 1
0	1	0	0	$F = (AB)'$	F = A plus (AB')
0	1	0	1	$F = B'$	F = (A+B) plus (AB')
0	1	1	0	$F = A \oplus B$	F = A minus B minus 1
0	1	1	1	$F = AB'$	F = AB' minus 1
1	0	0	0	$F = A'+B$	F = A plus (AB)
1	0	0	1	$F = (A \oplus B)'$	F = A plus B
1	0	1	0	F = B	F = (A+B') plus (AB)
1	0	1	1	F = AB	F = AB minus 1
1	1	0	0	F = 1	F = A plus A
1	1	0	1	F = A+B'	F = (A+B) plus A
1	1	1	0	F = A+B	F = (A+B') plus A
1	1	1	1	F = A	F = A minus 1

Flip-Flops:

A flip-flop is a bistable electronic circuit that has 2 stable state i.e its output is either logic 0 or 1. Flip-flop is memory device becoz its output will remain as set until something is done to change it. The flip-flop is also called a latch becoz it will hold/latch in either stable state.

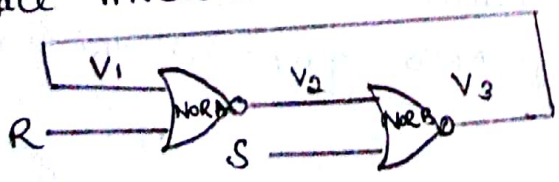
Flip-flops are used in construction of registers and counters in sequential circuit. flip-flops serve as key memory elements.

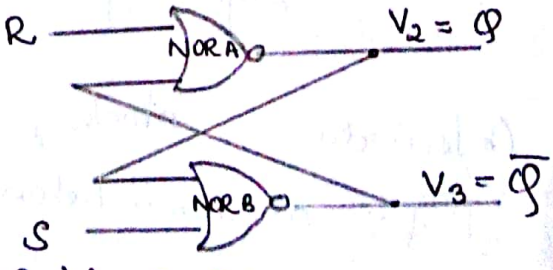
Flip-flop using inverter: when 2 inverters are connected in series, we can get a flip-flop.



If remove the feedback line and set $V_1 = +5V$. V_3 will be +5V representing logic 1. If V_1 is set to 0V then V_3 will also be 0V representing logic 0. We can use feed back line to hold V_1 value.

Replace Inverter with NOR Gate:





Circuit redrawn

The flip-flop has 2 output states Q and $\bar{Q}$ (i.e compliment of Q). 2 inputs are R and S so this flip-flop is called RS flip-flop.

If one of the inputs of NOR gate is 1 (H) then output will be 0 (L).

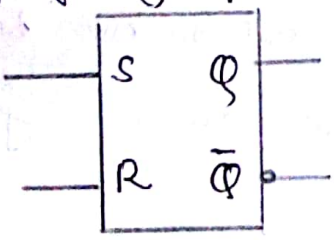
1) If R=0 and S=0, 0 at input of a NOR gate has no effect on its output, the flip flop remains in its present state Q is unchanged.

2) If R=0 and S=1, output of NOR B will be low, hence input of NOR A is low and output of NOR A is high (1). When R=0, S=1, it will SET the flip-flop (Q=1)

3) If R=1 and S=0, output of NOR A is low, both the inputs to NOR B is low, then output ($\bar{Q}$) will be high (1). which RESET the flip-flop (Q=0)

4) If R=1 and S=1, which forces the outputs of both NOR gates to low i.e $Q = \bar{Q} = 0$, so it is forbidden state.

1. RS flip-flop:

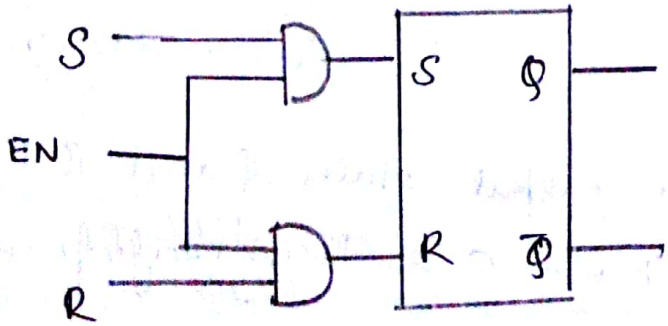


Logic symbol

R	S	Q	Action
0	0	Last state	No change
0	1	1	SET
1	0	0	RESET
1	1	?	Forbidden

Truth Table

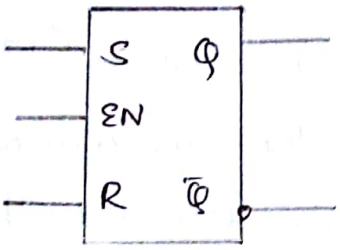
On introducing clock, adding 2 AND gates at the inputs R & S as shown below:



When EN is low, the AND gates outputs will be low, neither R & S value will effect the flip-flop output Q, the flip-flop is said to be disable.

When EN is high, information at the R and S inputs will be transmitted directly to the outputs. The flip-flop is said to be enabled.

Clocked flip-flop (RS-flip-flop)



EN	S	R	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	?
0	×	×	Q_n

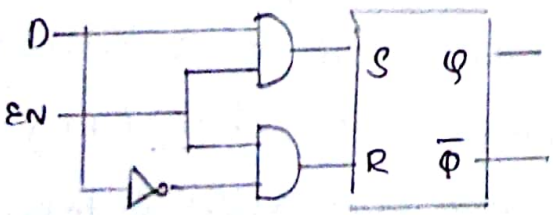
Clocked D Flip-flop:

Disadvantage of RS flip-flop:

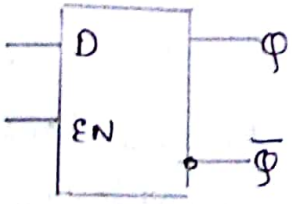
→ 2 input signals to drive a flip-flop

→ forbidden condition.

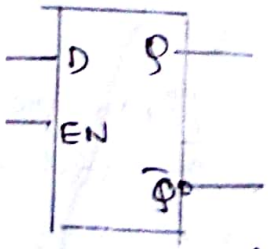
So D flip-flop needs only a single input.



A D flip-flop.



IEEE Symbol



Logic Symbol

When EN is low, both AND gates are disabled, Q remains unchanged (D value will not affect the output Q). When EN is high, both AND gates are enabled, Q is forced to equal the value of D

Truth Table for D:

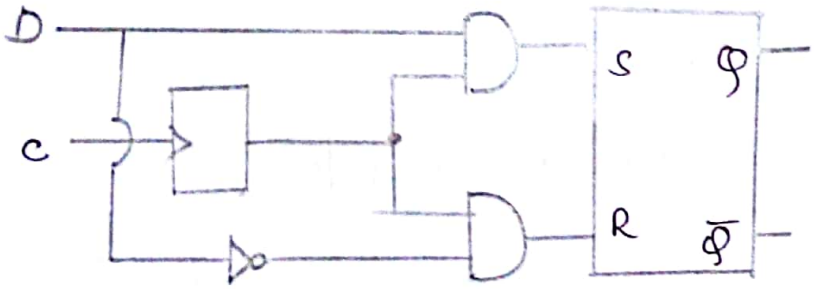
EN	D	Q_{n+1}
0	X	Q_n (last state)
1	0	0
1	1	1

There are many ways to Design a D flip-flop.

Edge-Triggered D flip-flops:

Edge triggering: A circuit responds only when the clock is in transition between its two voltage states.

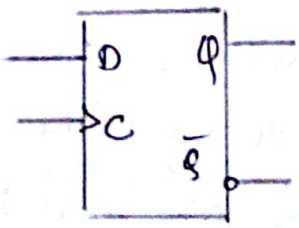
Fig below shows a positive pluse (PT) forming circuit at the input of a D flip-flop. The narrow positive pulse enables the AND gates for an instant. The effect is to activate the AND gates during the PT of C, which is equivalent to sampling the value of D for an instant.



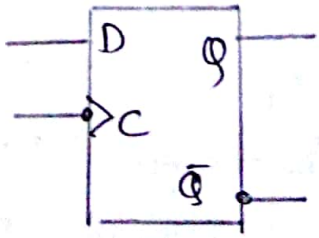
C	D	Q_{n+1}
0	X	Q_n (last state)
↑	0	0
↑	1	1

Positive edge triggered D flipflop.

When clock is low D is a dont-care and Q is latched in its last state. On the leading edge of the clock (PT) the data bit is loaded into the flip-flop and Q takes value of D.



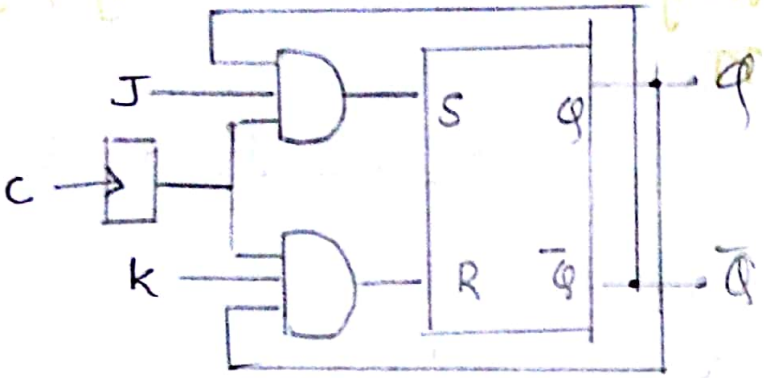
Positive edge triggered D



negative positive edge triggered D

Edge Triggered JK flip-flop:

In RS flip-flop if both R&S are 1 leads to illegal/forbidden state. So in JK flip-flop which accounts for this illegal input is more versatile circuit. For implementing counters we can use JK flip-flops.



JK flip-flip.

C	J	K	Q_{n+1}	Action
↑	0	0	Q_n (Last state)	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	$\bar{Q}_n$ (Toggle)	Toggle

Truth Table

Positive Edge Triggered JK flipflop.

Fig above shows the positive edge triggered JK flip flop with 2 additions.

i) Q output is connected back to the input of the lower AND gate
ii) $\bar{Q}$ output is connected back to the output of the upper AND gate

[Cross coupling from o/ps to i/p changes RS to JK flipflop]

1. When J & K = 0, both AND gate are disabled, clock pulse have no effect, Q retains its last value
2. When J = 0 & K = 1, upper gate is disabled, so there is no way to set the flip-flop, the only possibility is reset [the lower gate passes a Reset pulse as soon as reset pulse arrives] This forces Q = 0
3. When J = 1 & K = 0, lower gate is disabled, so its impossible to reset the flip flop. But we SET the flip flop. When Q = 0, $\bar{Q}$ = 1, upper gate passes a SET pulse on the

next PT (positive pulse), Q becomes 1.

4. When J and K are both 1, it's possible to set/reset the flip-flop. If Q is high(1) the lower gate passes a RESET pulse on next PT, when Q is low (0) the upper gate passes a SET pulse on the next PT. Therefore J=1 & K=1 mean the flip-flop will toggle on the next PT.

Propagation delay prevents the JK-flip-flop from having i.e toggling more than once during a PT)