DECODER

- A binary decoder is a combinational logic circuit that converts binary information from the **n** coded inputs to a maximum of 2^nunique outputs.
- We have following types of decoders 2x4,3x8,4x16….

2x4 decoder

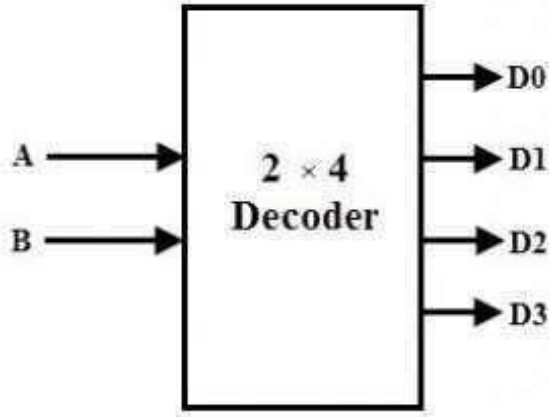


Fig 1: Block diagram

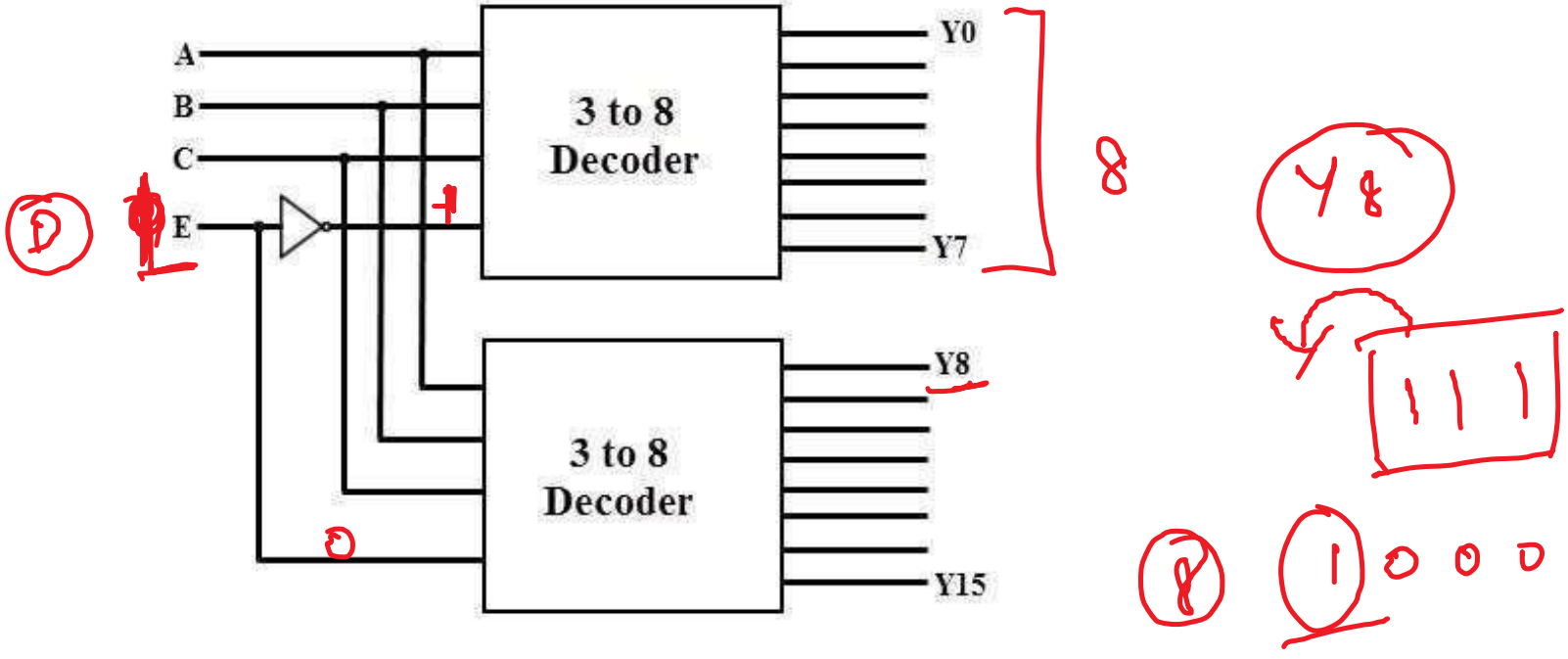
Inputs		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

Fig 2:Truth table

DECODERS

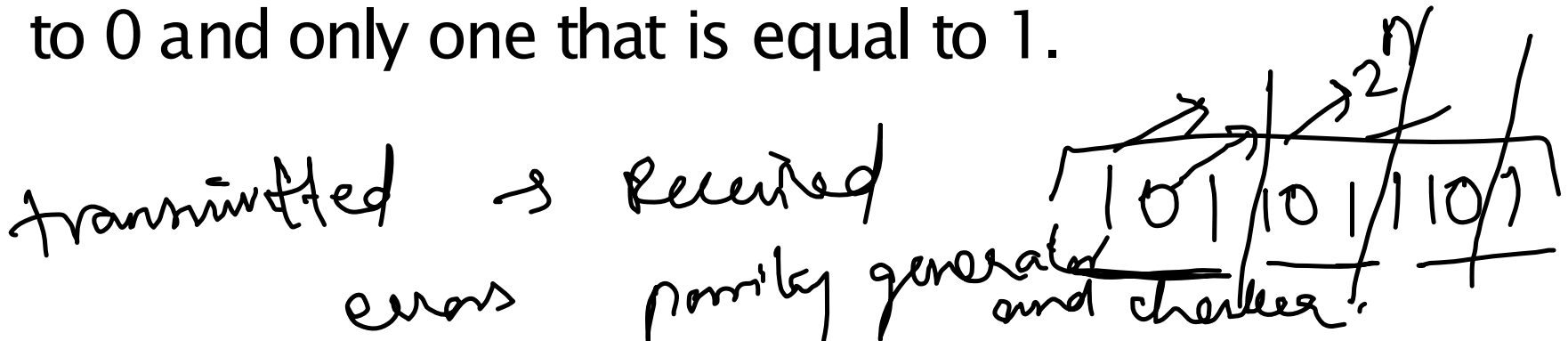
Higher order decoder implementation using lower order.

Ex:4x16 decoder using 3x8 decoders



Decoders → Demux

- The decoder is called n–to–m–line decoder, where $m \leq 2^n$.
- The decoder is also used in conjunction with other code converters such as a BCD–to–seven_segment decoder.
- 3–to–8 line decoder: For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1.



Implementation and truth table

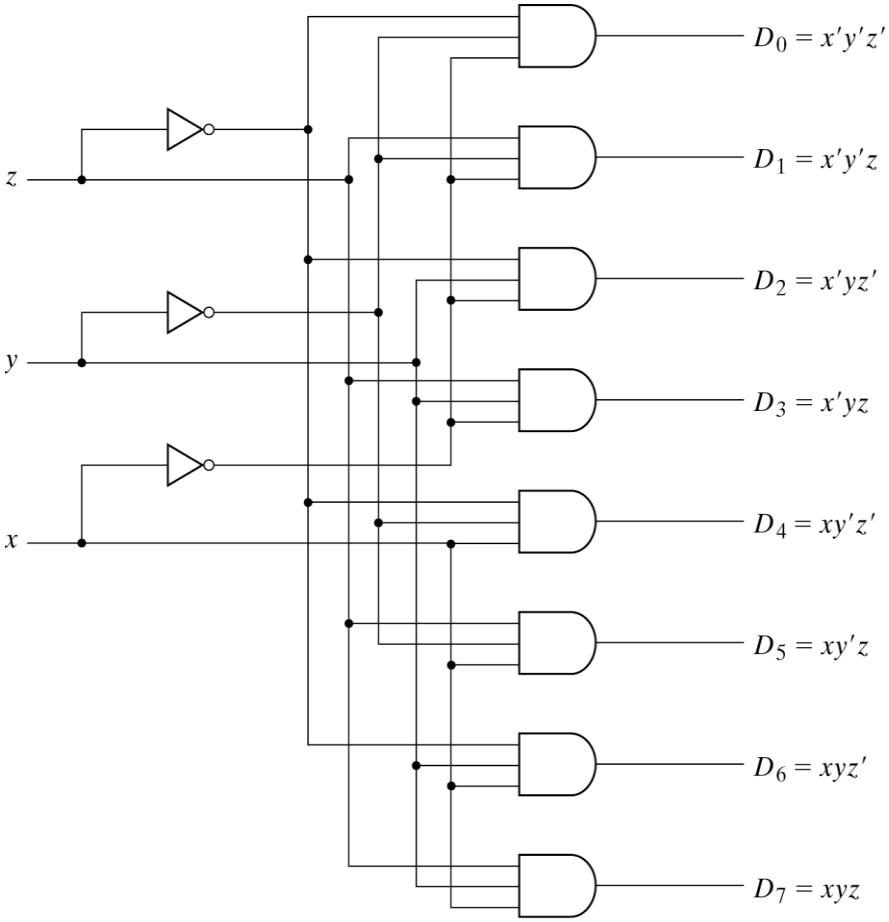


Fig. 4-18 3-to-8-Line Decoder

Table 4-6
Truth Table of a 3-to-8-Line Decoder

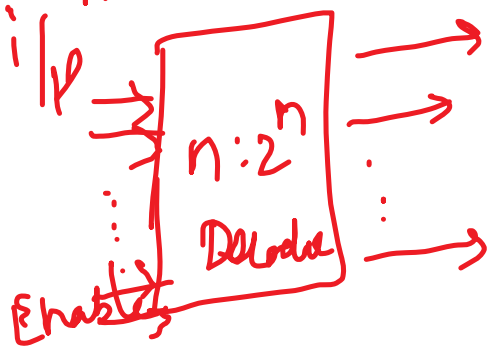
Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Binary decoder

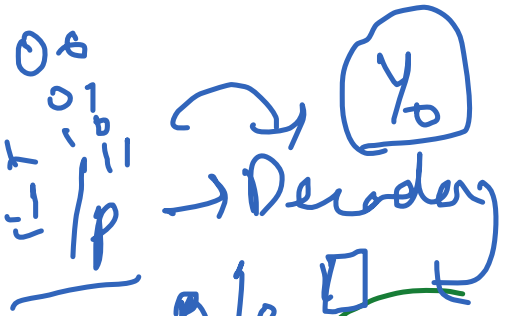
$n \rightarrow 2^n$

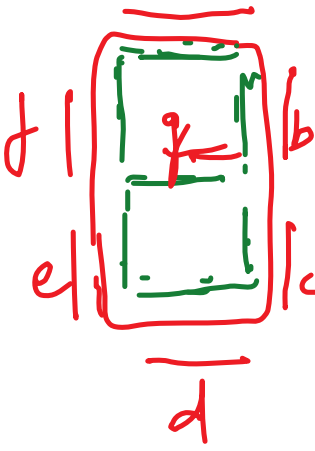
n-data

i/p → Transmitter → i/p.



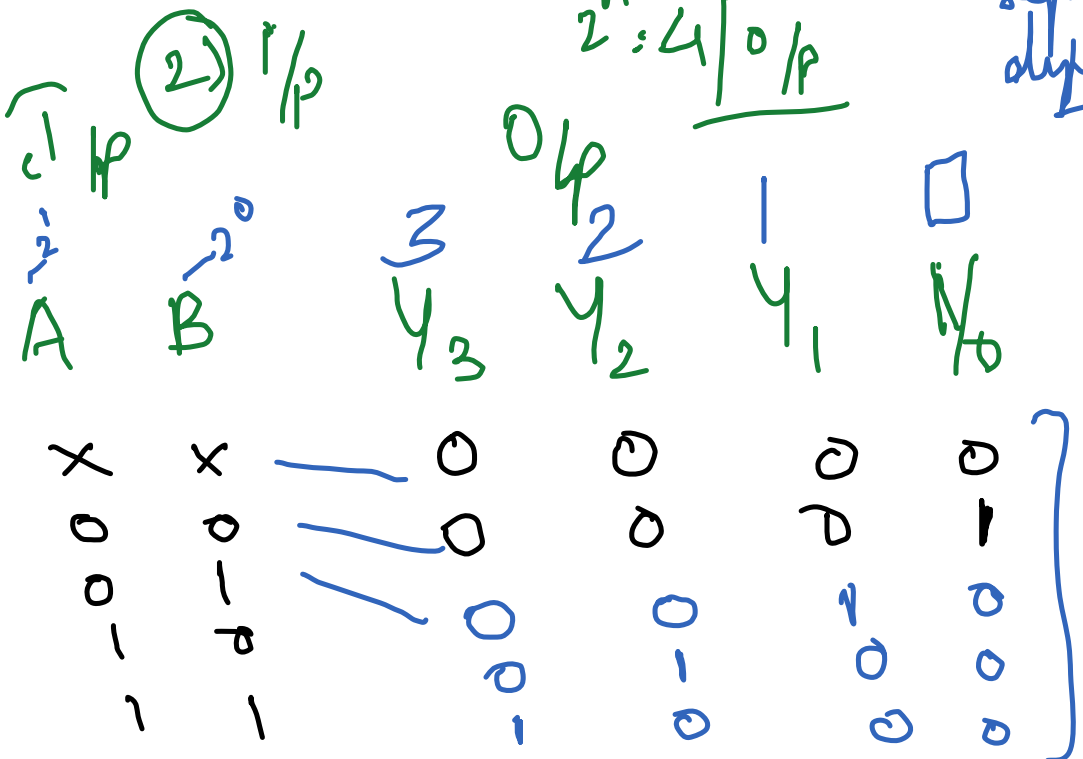
possible 2^n o/p.

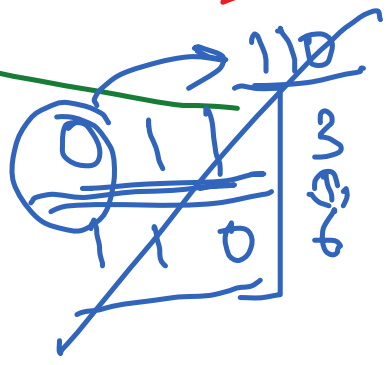




BCD to (0-9) Seven segment display

$2^n : 4$ o/p

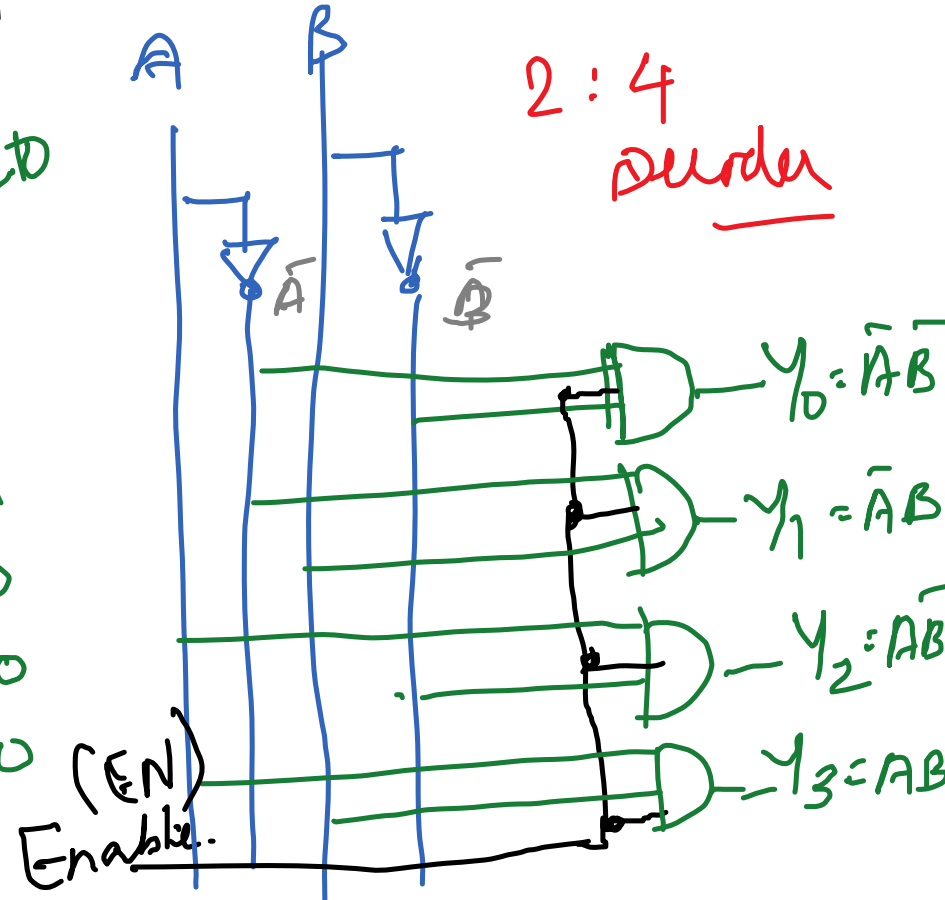




$n \rightarrow 2^n$ $\quad 2 \rightarrow 2^4 = 2$ to 4 decoder.

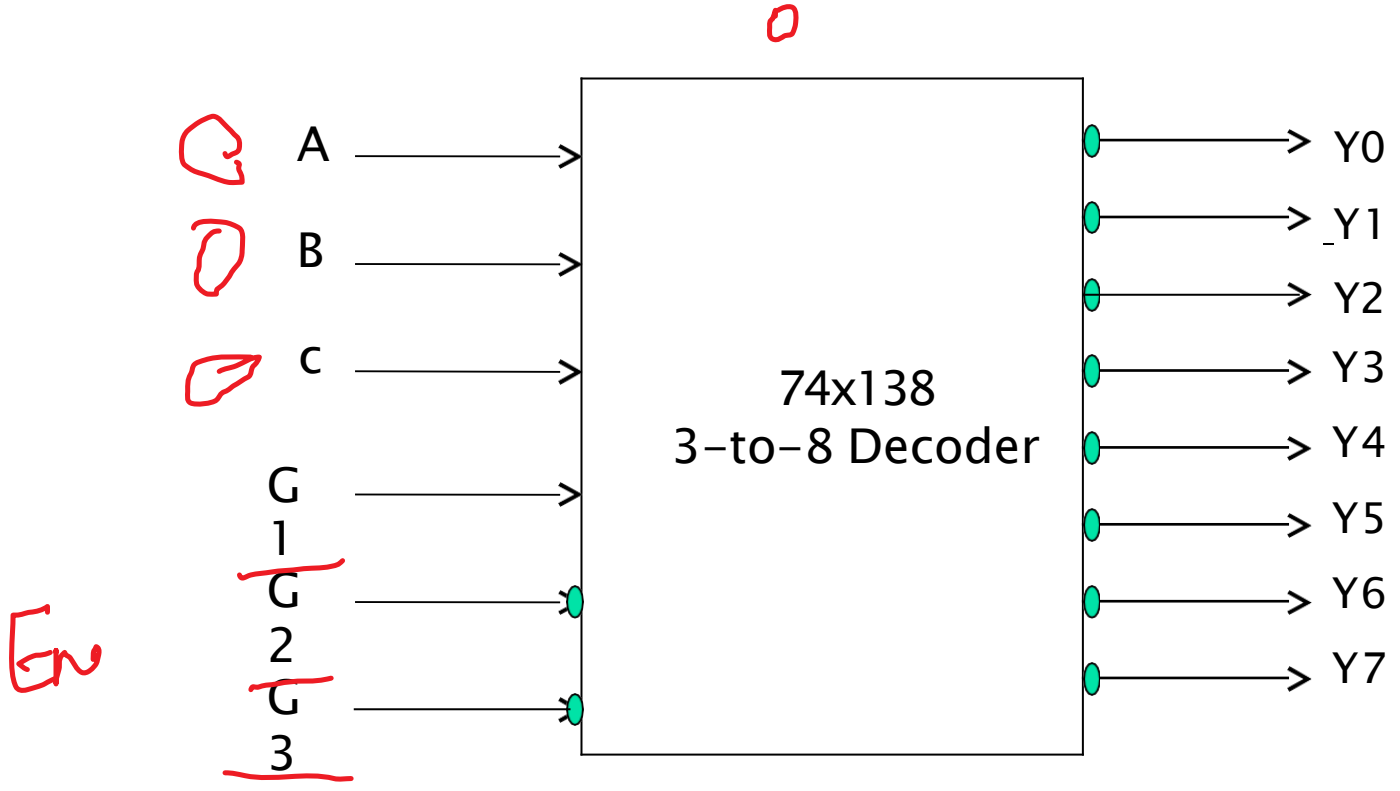
→ A decoder which has a n-bit binary i/p code and a one activated o/p out of 2^n o/p code is called binary decoder.

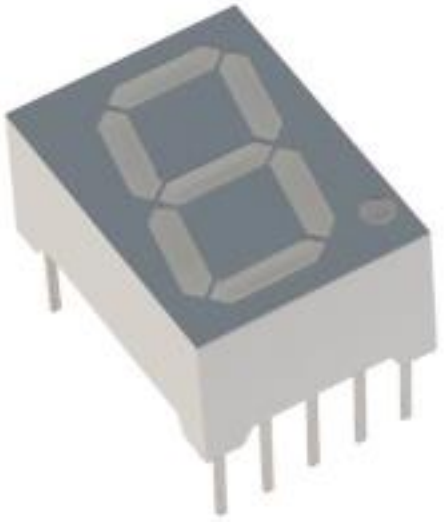
En	A	B	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

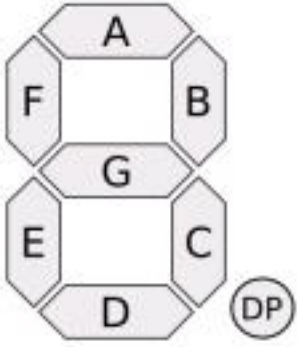


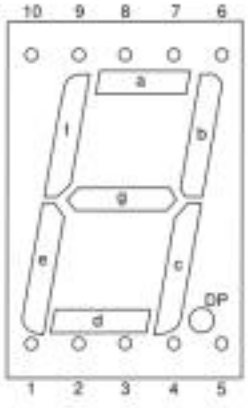
2 to 4 Decoder IC 74x139

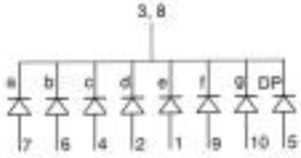
3 to 8 Decoder IC 74x138











BCD

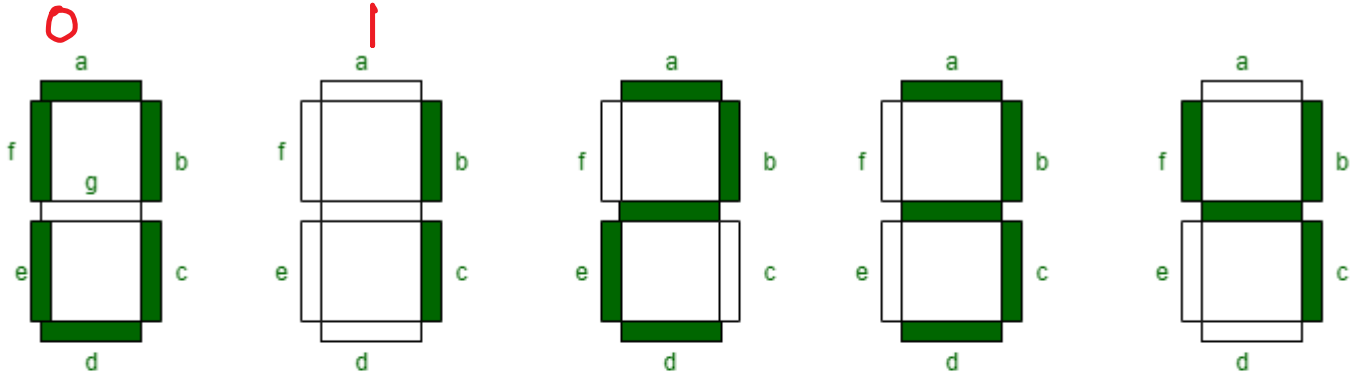
0 0 0 0

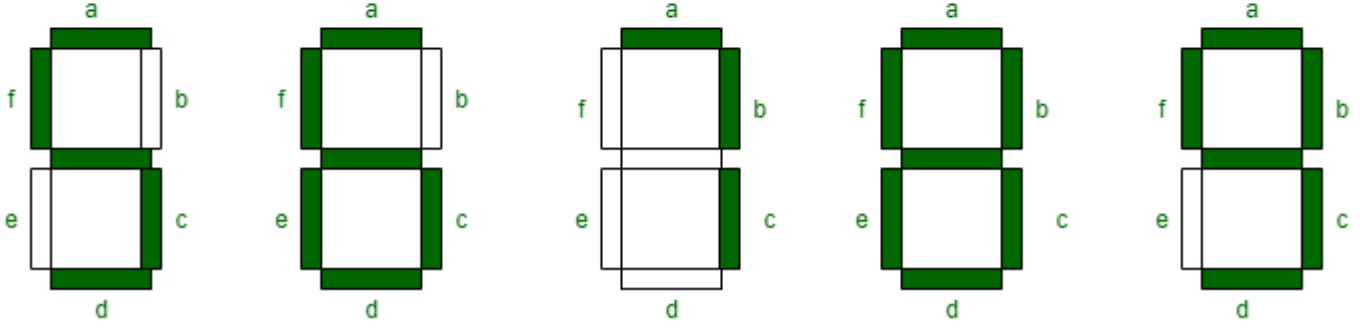
0 0 0 1

0 0 1 0

0 0 1 1

0 1 0 0

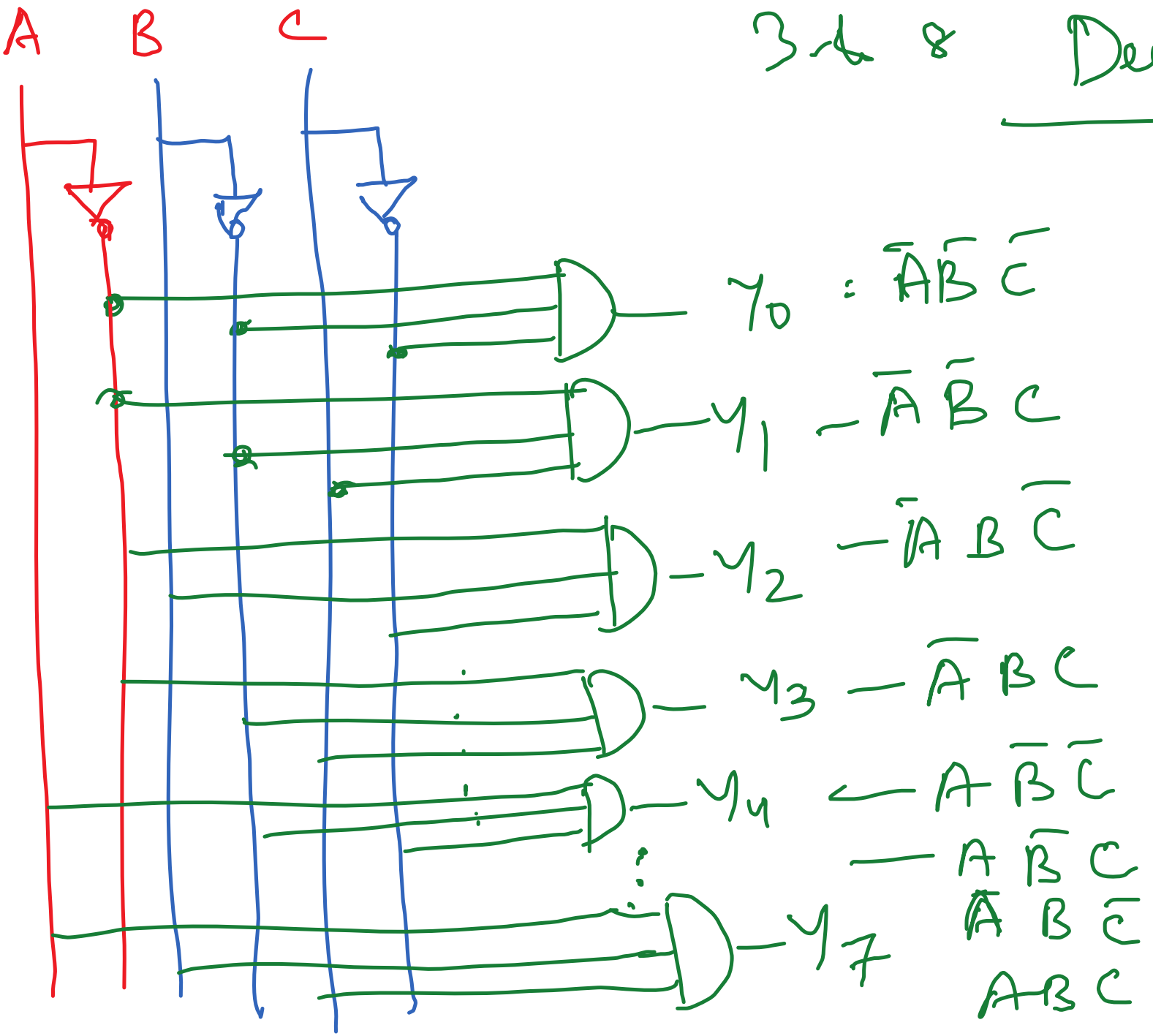




3 to 8 priority decoder.

En	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

3 to 8 Decoder



4.2.2 The 74X138 3-to-8 Decoder

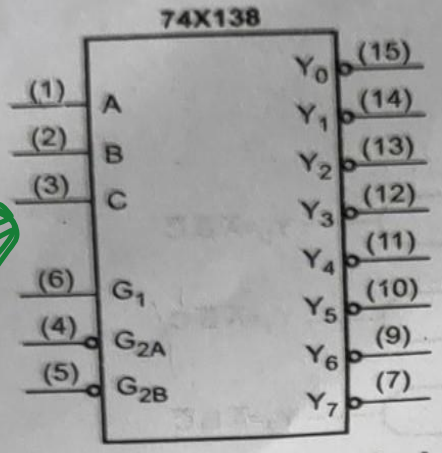


Fig. 4.9 Logic symbol

The 74X138 is a commercially ava 3-to-8 decoder. It accepts three binary i (A, B, C) and when enabled, provides individual active low outputs (Y_0 - Y_7). device has three enable inputs : two low ($\overline{G}_{2A}$, $\overline{G}_{2B}$) and one active high Fig. 4.9 and Table 4.4 show logic symbo function table respectively.

Inputs						Outputs							
G_{2B}	G_{2A}	G_1	C	B	A	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

Table 4.4 Function table

2 to 4 Decoder using IC 74x139

74x138 3 to 8 line decoder

74x139 2 to 4 Decoder

4.2.3 The 74X139 Dual 2-to-4 Decoder

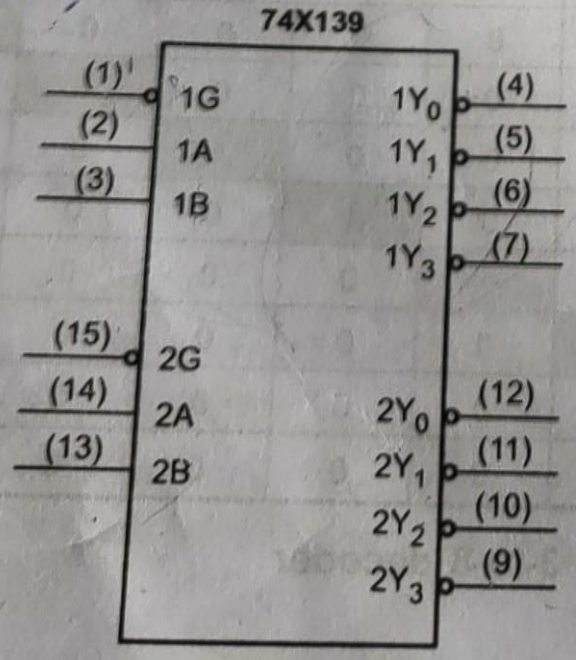


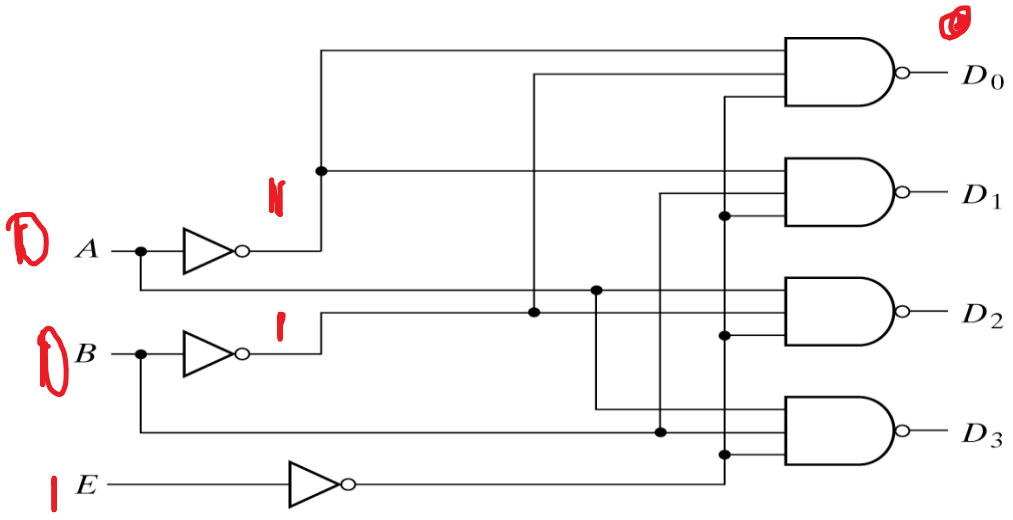
Fig. 4.10 Logic symbol for IC 74X139

The 74X139 consists independent and identical decoders. The enable inputs and of IC 74X139 are active low. shows the logic symbol and shows the function table for IC The Table 4.5 shows the truth one half of a 74X139 dual 2-to- The truth table for other half first half.

INPUT			OUTPUT			
G bar	B	A	Y3 bar	Y2bar	Y1bar	Y0bar
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Decoder with enable input

- Some decoders are constructed with NAND gates, it becomes more economical to generate the decoder minterms in their complemented form.
- As indicated by the truth table , only one output can be equal to 0 at any given time, all other outputs are equal to 1.



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

2.4 Cascading Binary Decoders

Binary decoder circuits can be connected together to form a larger decoder
4.11 shows the 4 × 16 decoder using two 3 × 8 decoders.

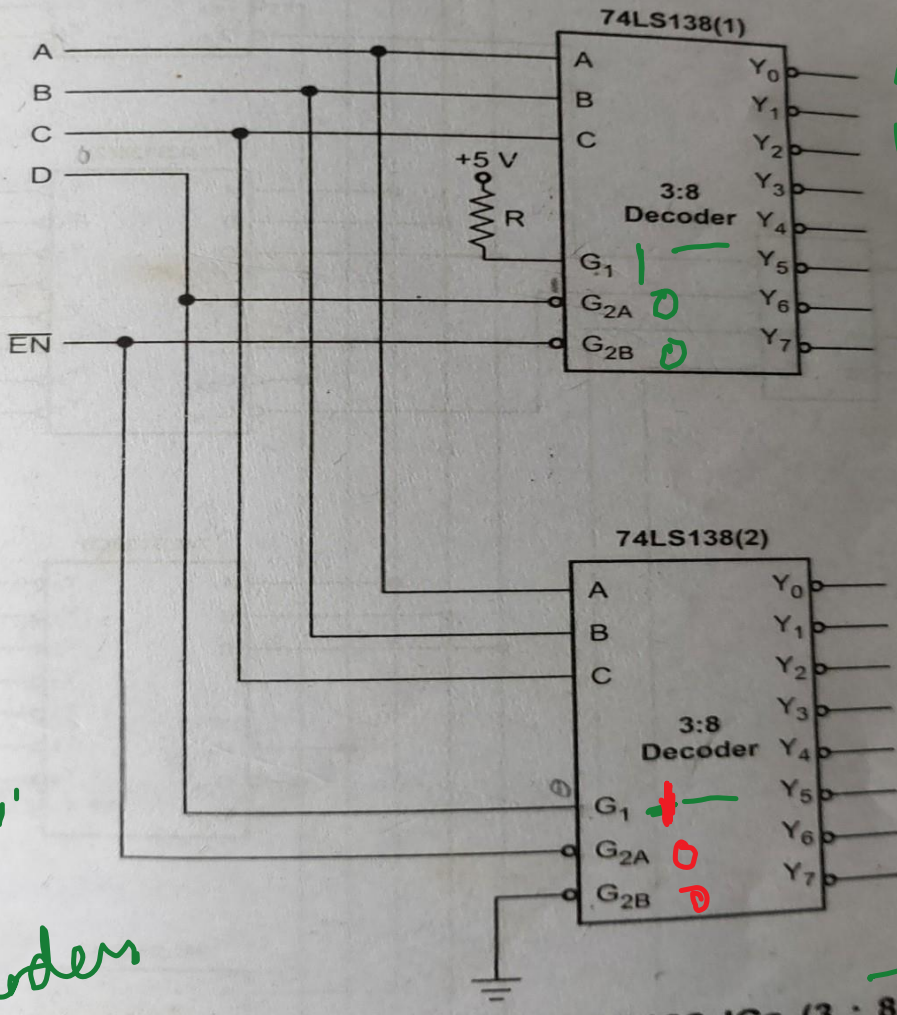


Fig. 4.11 4:16 decoder using two 74LS138 ICs (3 : 8 decoder)

Here, one input line (D) is used to enable/disable the decoders. When D
der is enabled and the other is disabled. Thus the bottom decoder outpu
the top eight outputs generate minterms 0 0 0 0 to 0 1 1 1. When D=

Example 4.4 : *Design 5-to-32 decoder using one 2-to-4 and four 3-to-8 decoder ICs.*

Solution : The Fig. 4.12 shows the construction of 5-to-32 decoder using four 74LS138s half 74LS139. The half section of 74LS139 IC is used as a 2-to-4 decoder to decode the higher order inputs, D and E. The four outputs of this decoder are used to enable one

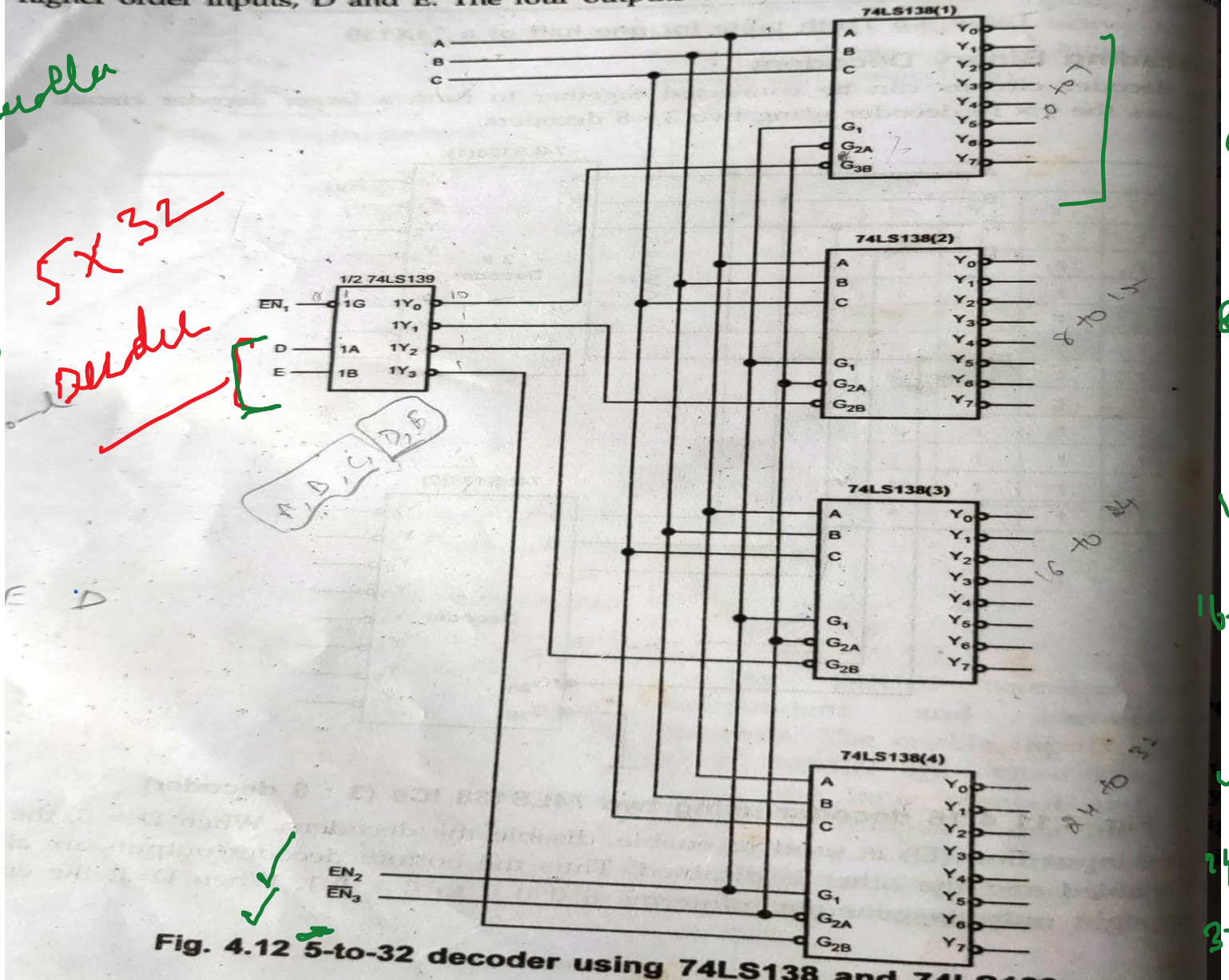
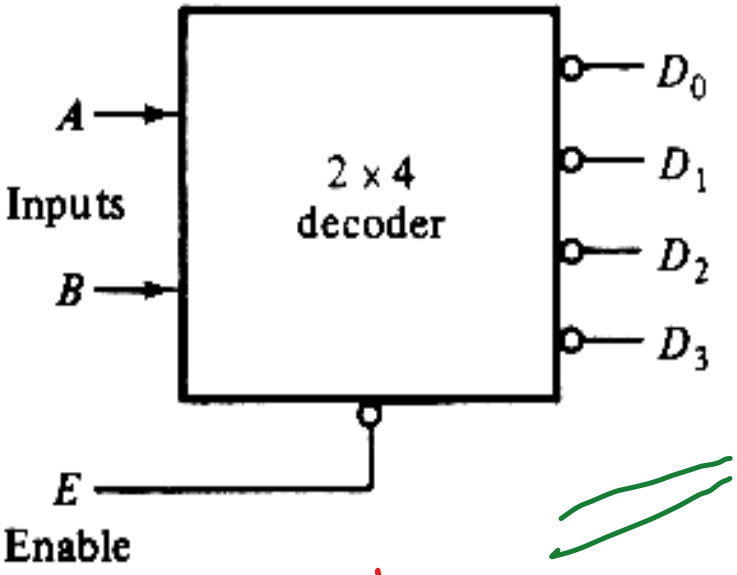


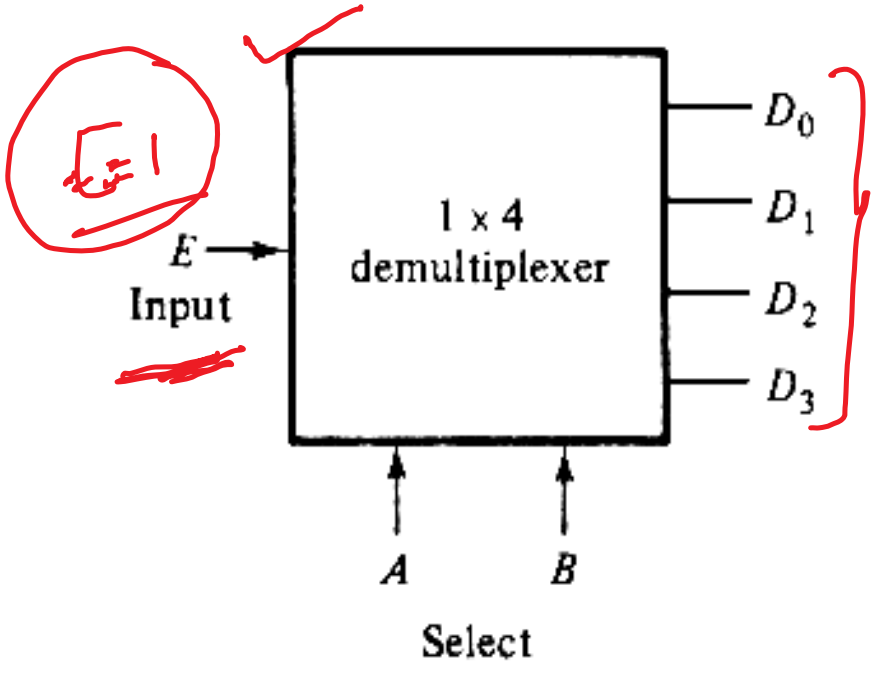
Fig. 4.12 5-to-32 decoder using 74LS138 and 74LS139

Demultiplexer

- A decoder with an enable input is referred to as a decoder/demultiplexer.
- The truth table of demultiplexer is the same with decoder.



(a) Decoder with enable



(b) Demultiplexer

3-to-8 decoder with enable implement the 4-to-16 decoder

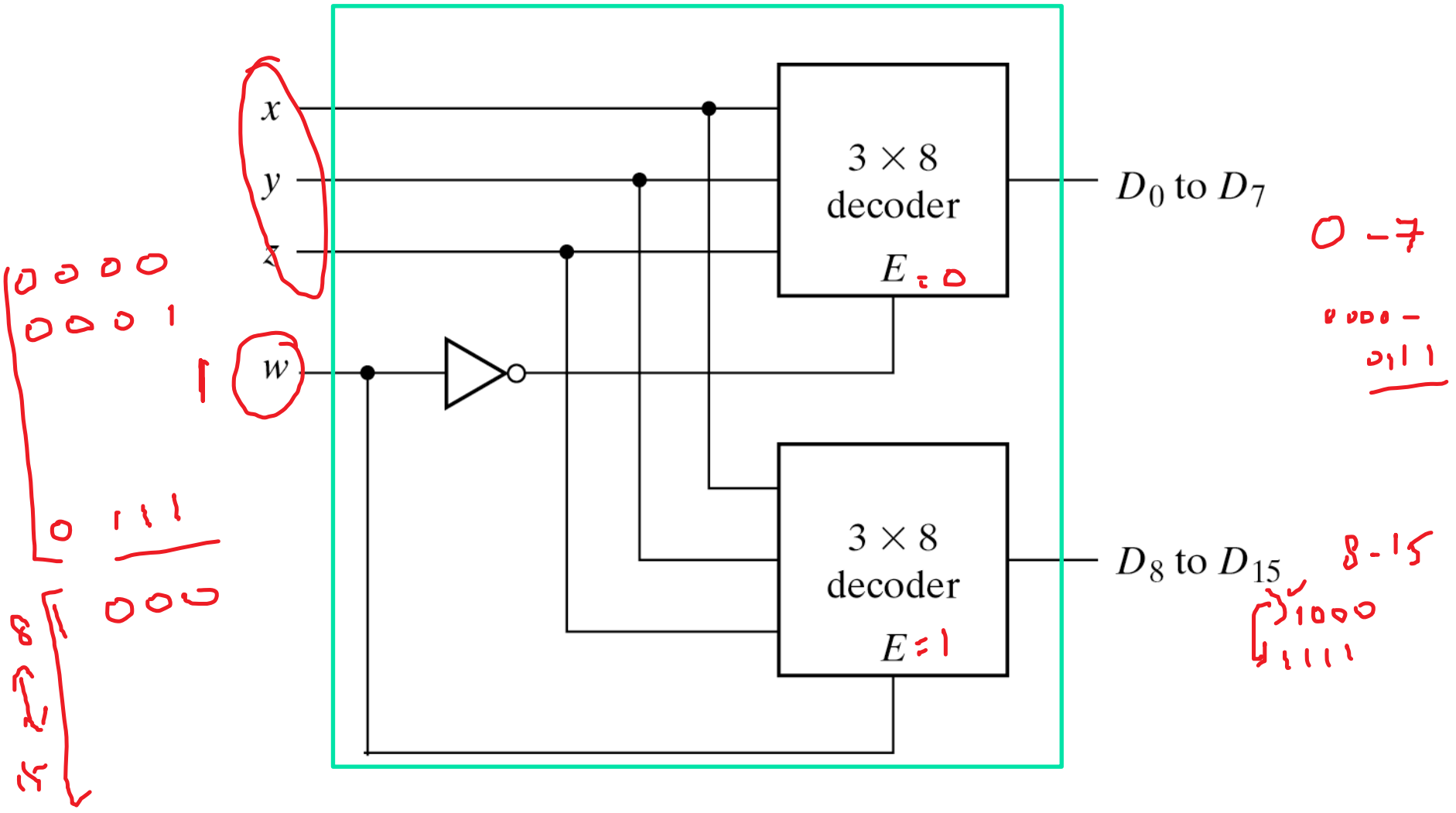
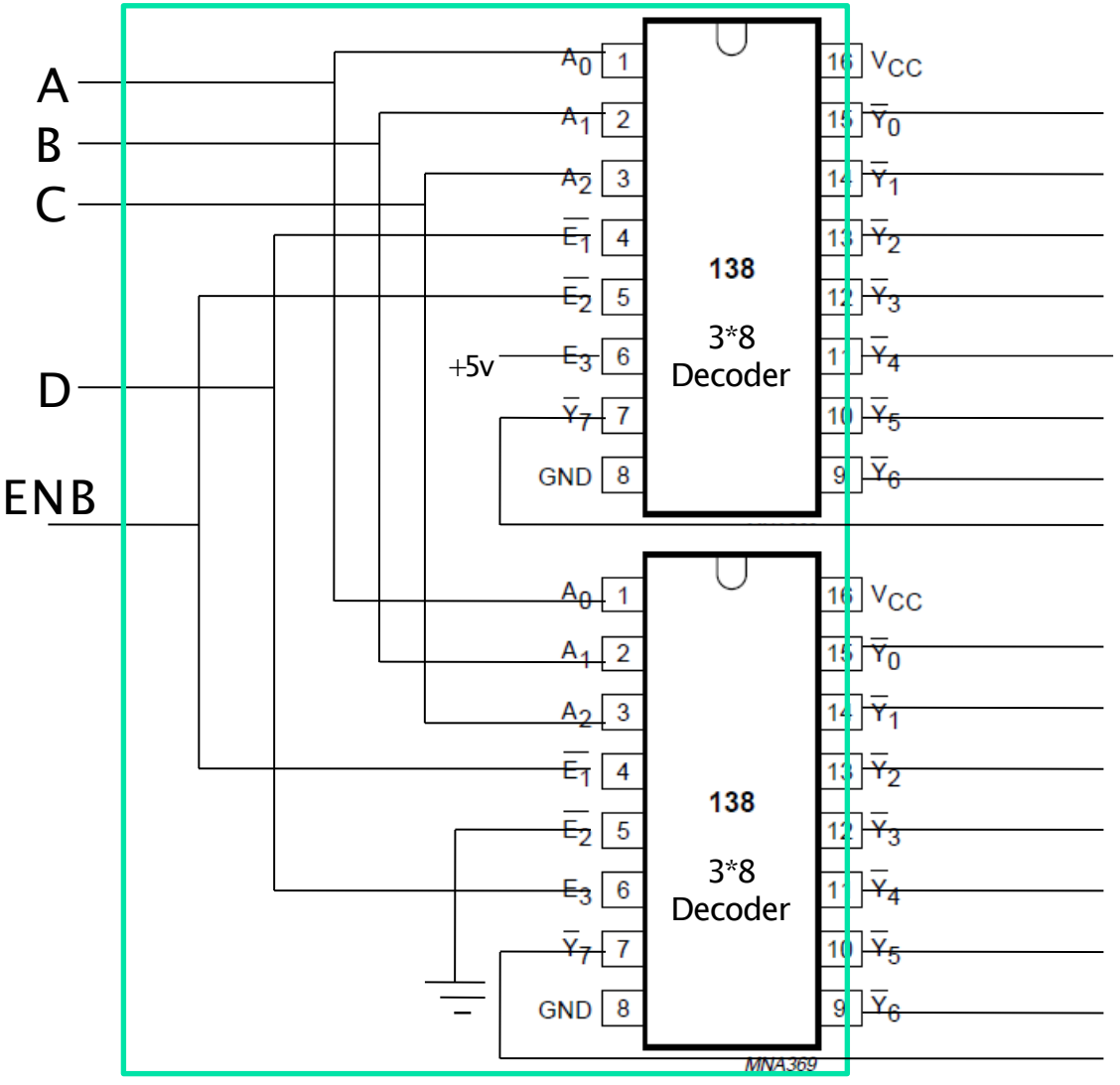


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

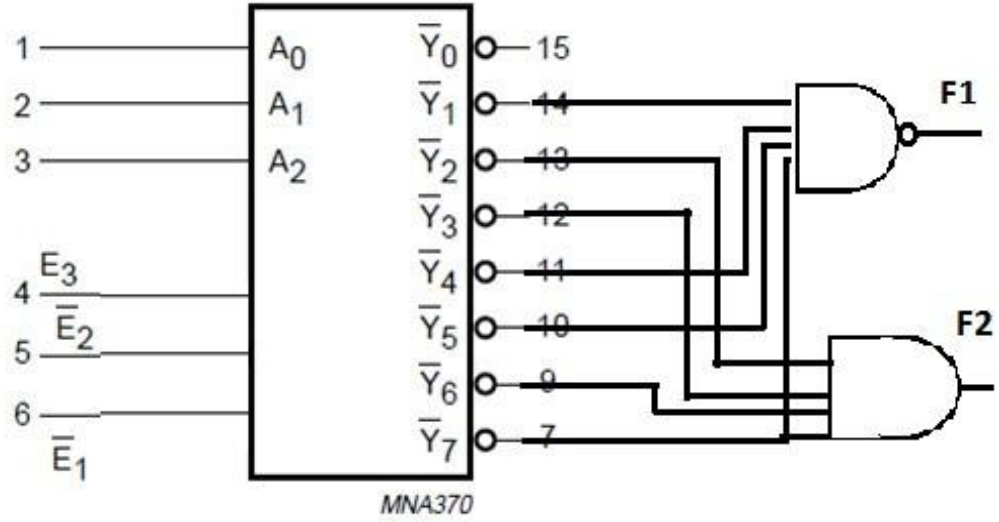
Cascading binary decoders

(4*16 Decoder using 3*8 decoder)



Example

- Implement the following multiple output function using 74LS138 and external gates. F1 (A,B,C)= Σm (1,4,5,7) & F2 (A,B,C)= ∏m (2,3,6,7)
- 74LS138 is an 3*8 decoder. The outputs of this ic have <u>active Low.</u> i.e in SOP form for F1 using NAND gate and POS function for F2 using AND gate.



Implementation of a Full Adder with a Decoder

- From table 4–4, we obtain the functions for the combinational circuit in sum of minterms:

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$
$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

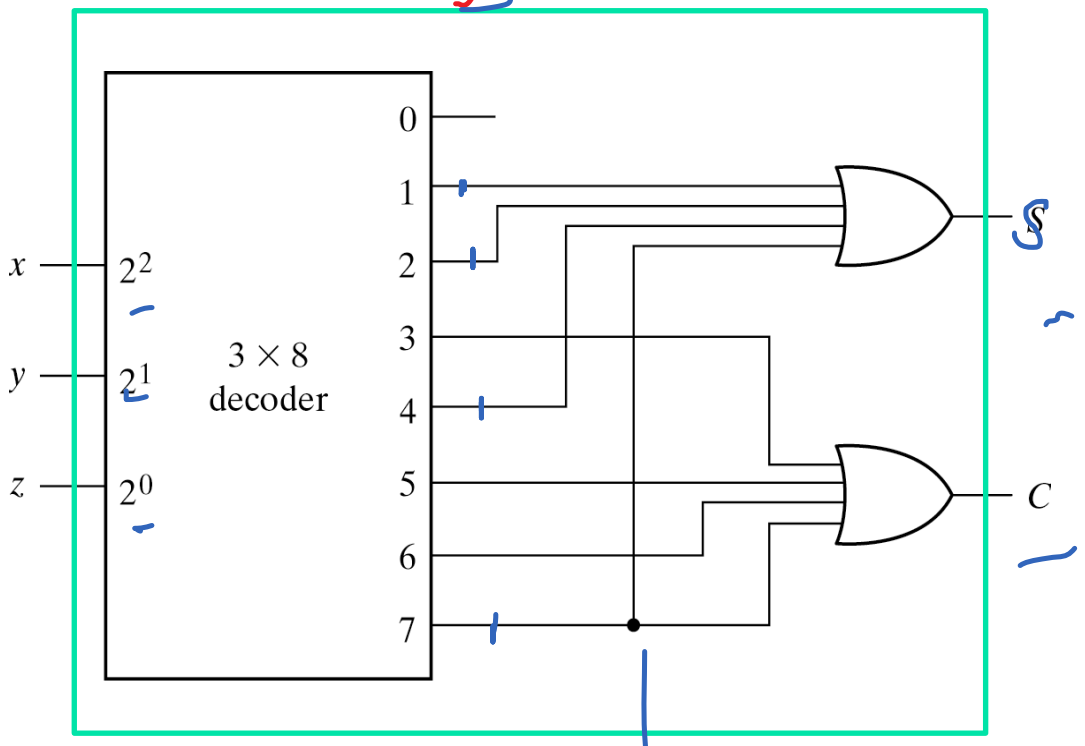


Fig. 4-21 Implementation of a Full Adder with a Decoder

$n \rightarrow 2^n$

Decoder

2 to 4 line D (IC 74 x 139)

3 to 8 line D (IC 74 x 138)

i/p

o/p

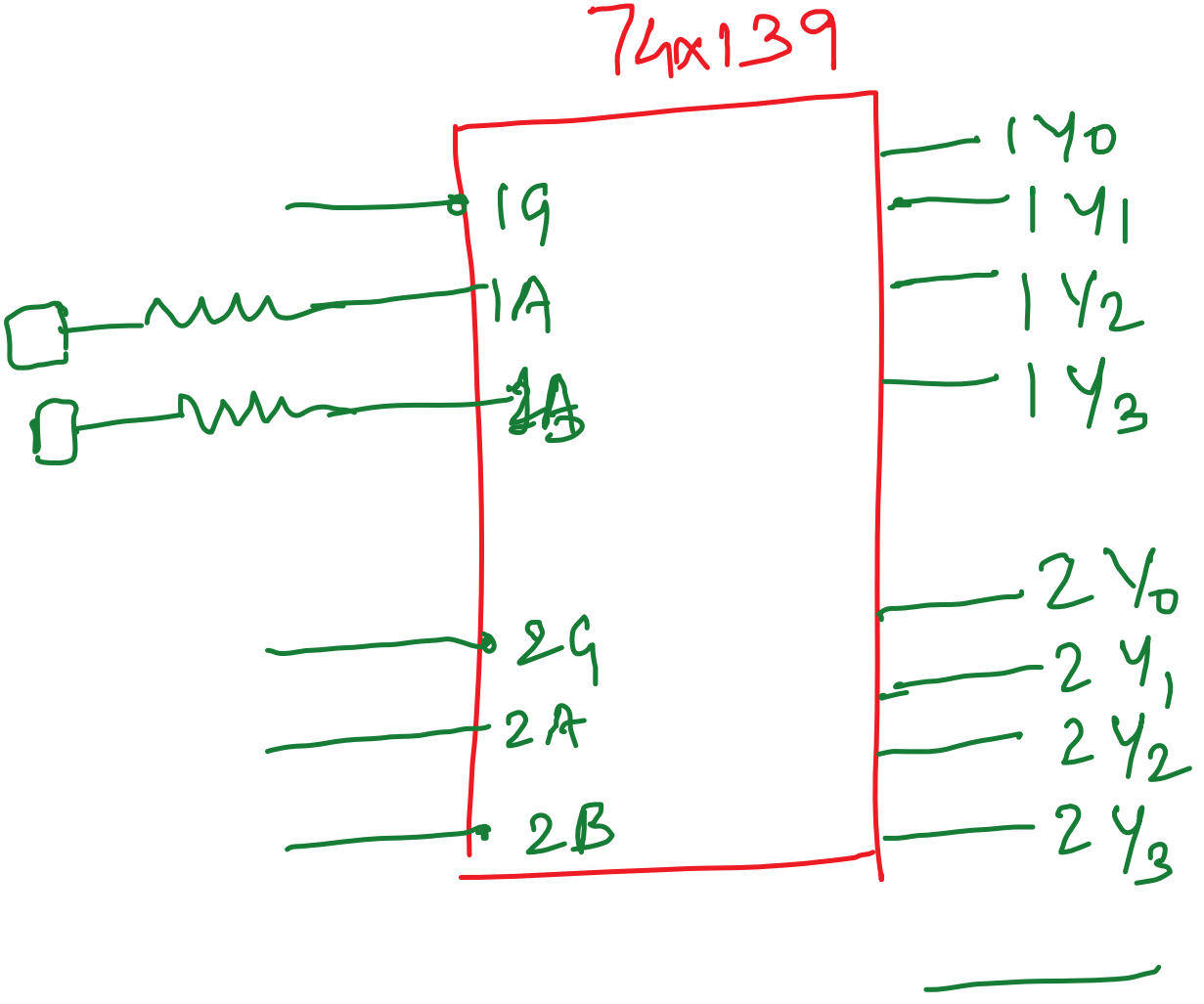
Demultiplexer.

3 bit → Decoder (4) → PU → Encoder (3 bit)

8 → 3 bit

i/p o/p

IC 74x139 (2 to 4 line Decoder)



2 bit i/p

A	B
00	01
01	10
⋮	⋮

G_1 B A Y_3 Y_2 Y_1 Y_0 $G \rightarrow 1$

$\overline{G} \rightarrow 0$

$-IG_0$

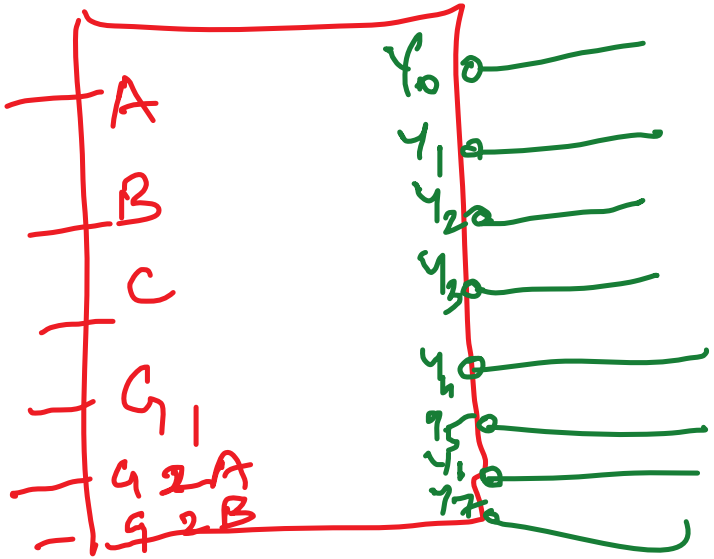
$-2G_0$

i/p

o/p

G	$\overline{G}$	B	A	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$
0	[redacted]	X	X	1	1	1	1
1	0	0	0	1	1	1	0
1	0	0	1	1	1	0	1
1	0	1	0	1	0	1	1
1	0	1	1	0	1	1	1

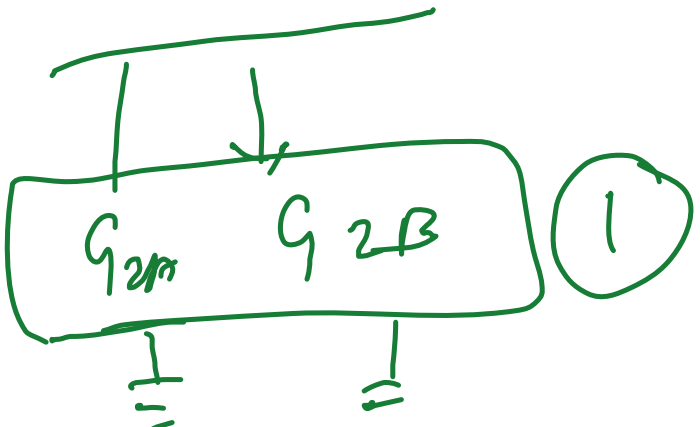
IC 74x138 3 to 8 line Decoder

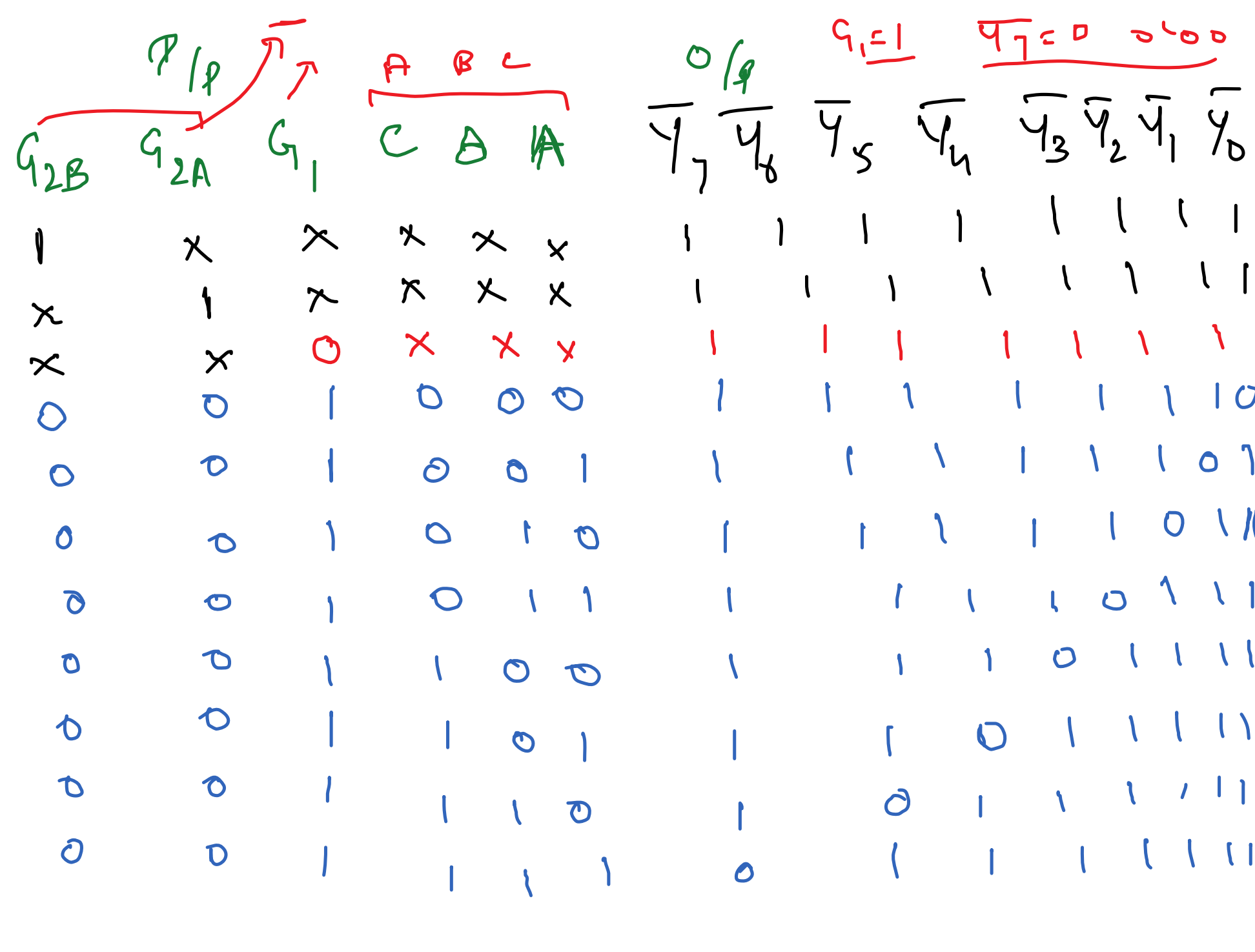


B → 0 (G_{2B})

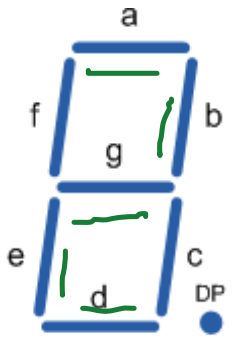
A = 0 (G_{2A})

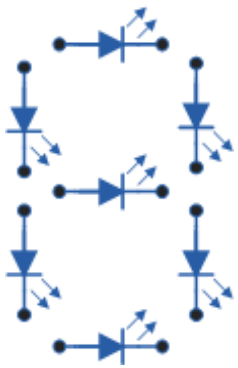
A B C
0 0 1



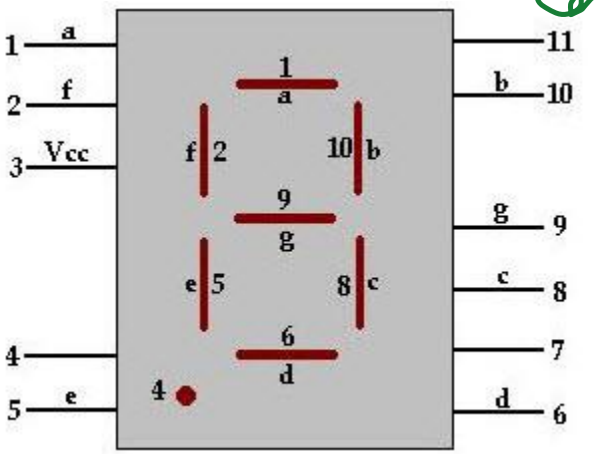

I/p
O/p
A B C
G_{2B}
G_{2A}
G_1
C B A
$\bar{Y}_7$ $\bar{Y}_6$ $\bar{Y}_5$ $\bar{Y}_4$ $\bar{Y}_3$ $\bar{Y}_2$ $\bar{Y}_1$ $\bar{Y}_0$
$G_1=1$
$\bar{Y}_7=0$ so on

BCD to 7 segment Display Decoder





Inputs					Outputs						
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

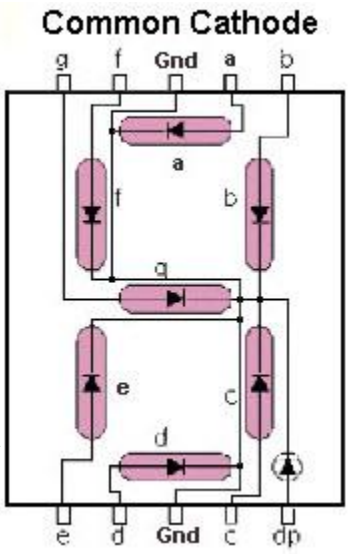


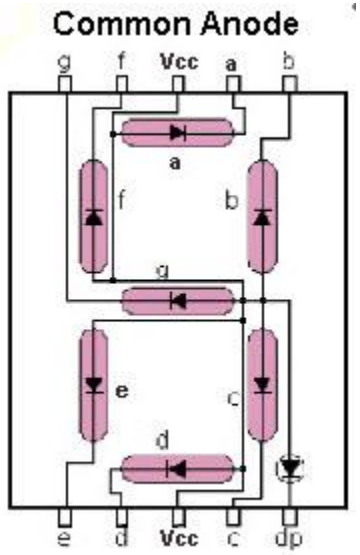
Common Cathode

1 7

1 _ 1

g, b, c, d, e, f





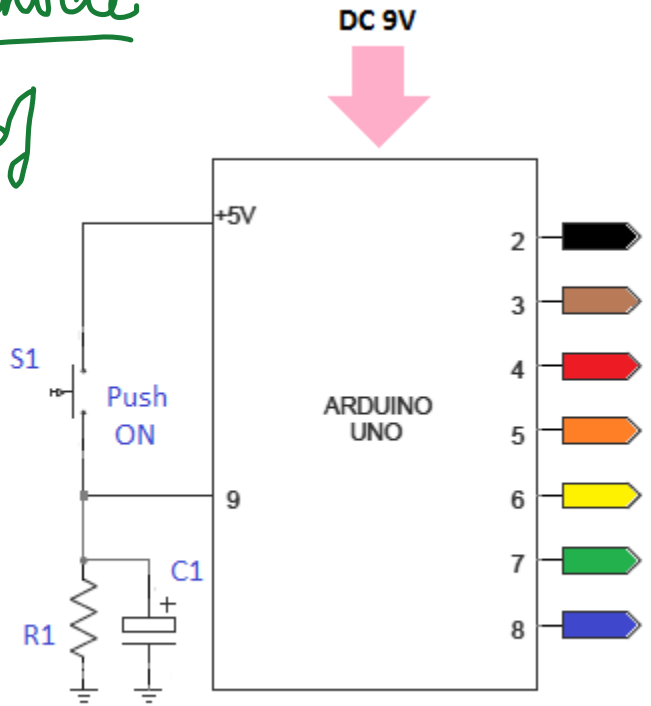
Common anode =

= 1 g = 0

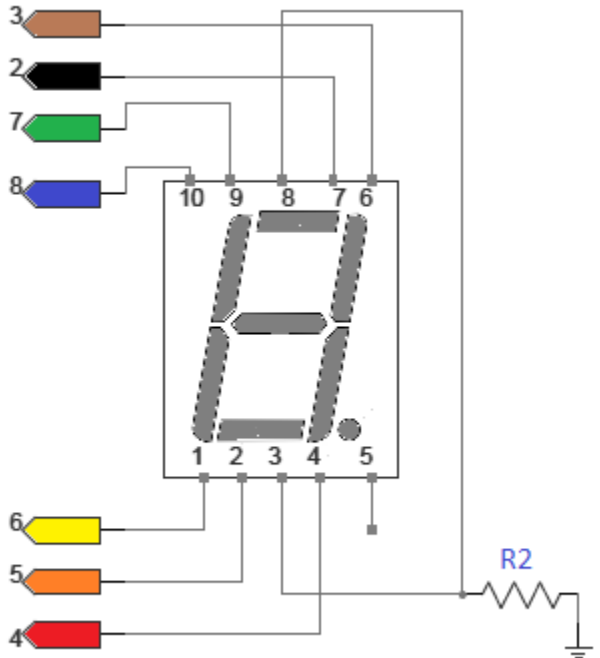
Reverse of common cathode

g = 1

g, b, c, d, e, f = 0



R1: 10K C1: 10uF/16V

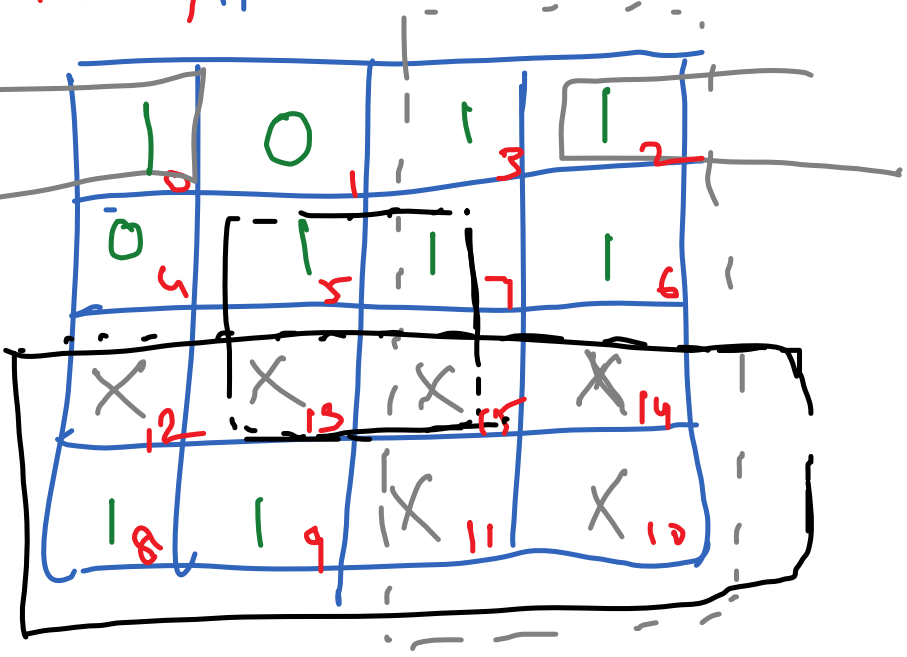


R2:330 R

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

(0-9) ← BCD to Common cathode 7-segment decoder.

K-map for a



10-15 = "X"

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

plot the same K-map for b, c, d, e, f, g

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

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

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

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

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

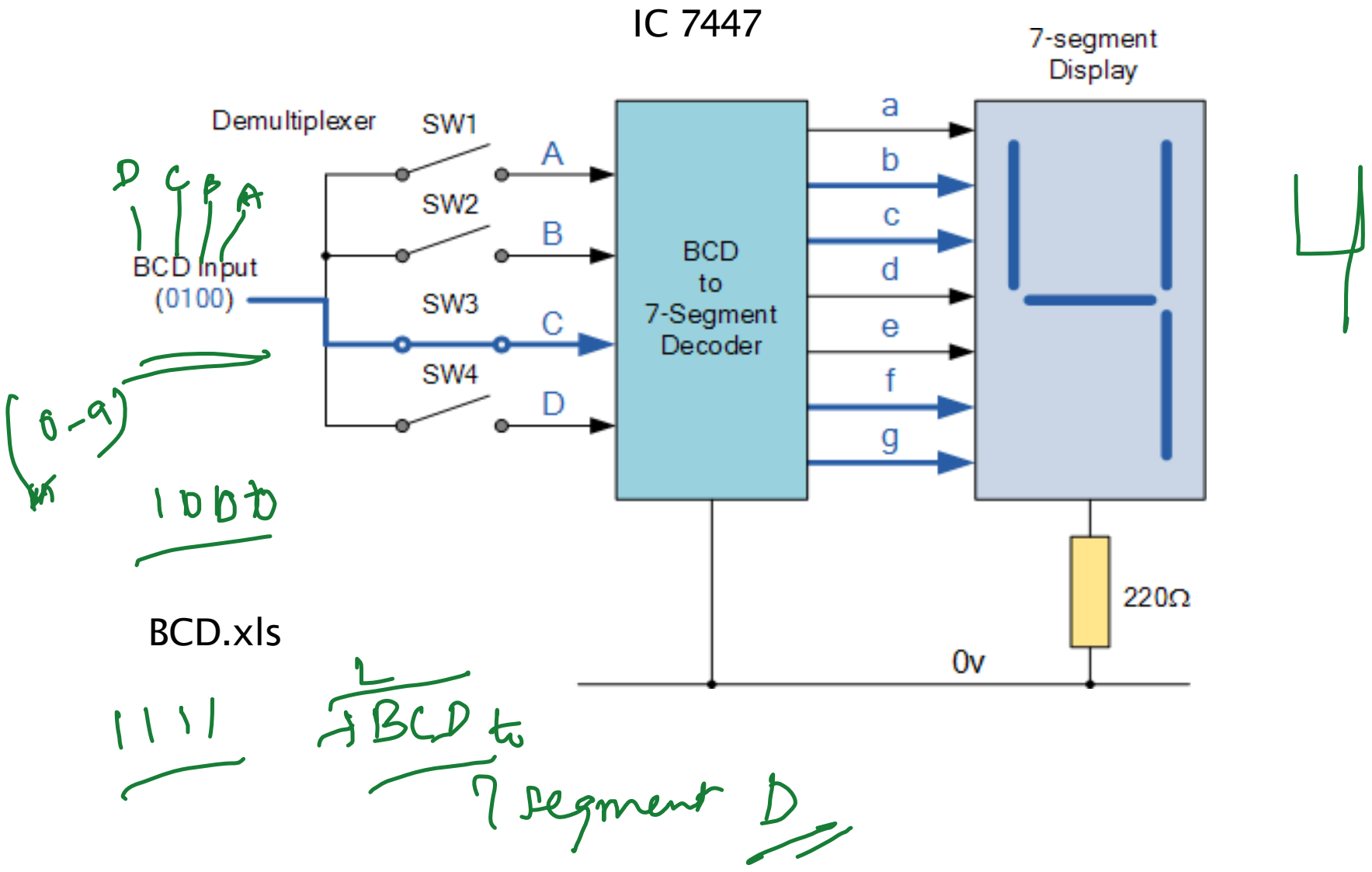
$g = A + B\bar{C} + \bar{B}C + C\bar{D}$.

Logic diagram for the values a to g

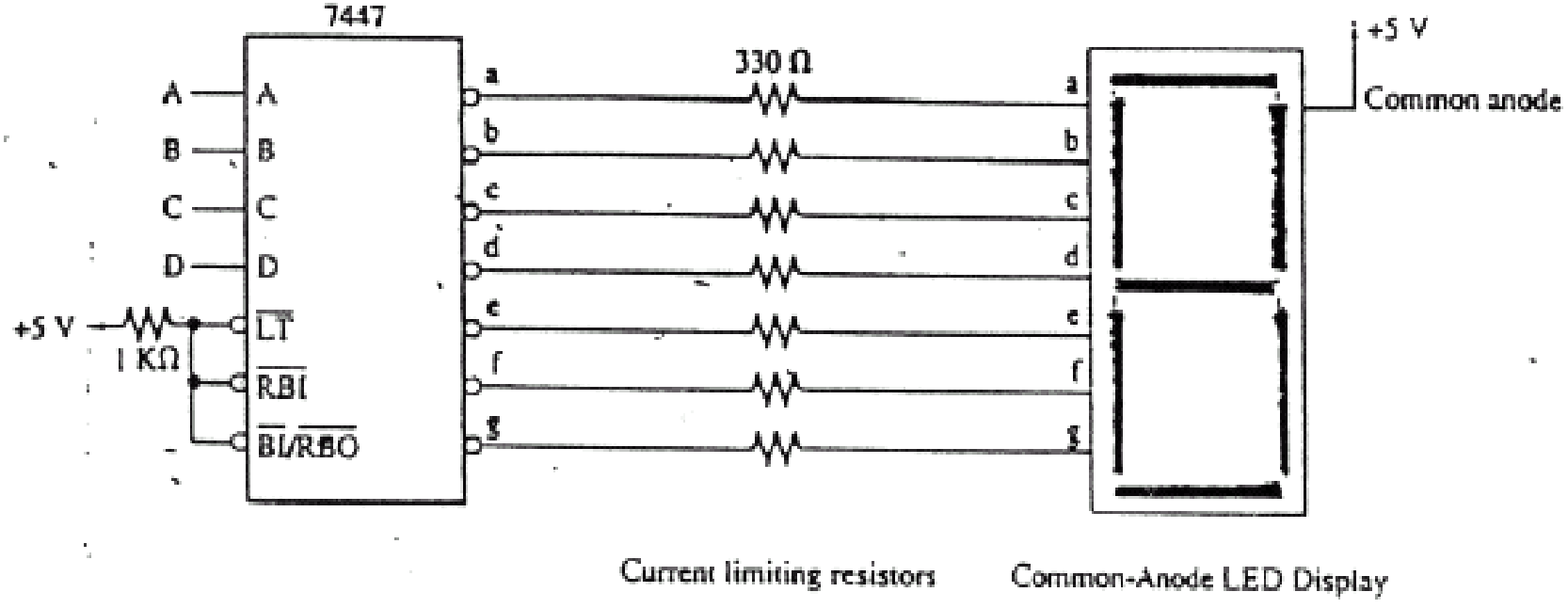
for common anode

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

Display Decoder



Common Anode LED Display



ENCODERS

- An Encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines.
- It will produce a binary code equivalent to the input, which is active High.

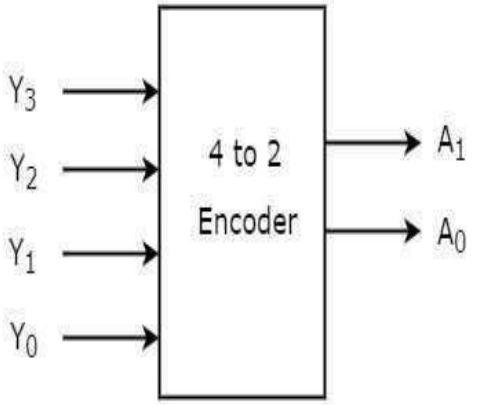


Fig 1:block diagram of 4x2 encoder

ENCODERS

Octal to binary encoder

Octal digits		Binary		
		A_2	A_1	A_0
D_0	0	0	0	0
D_1	1	0	0	1
D_2	2	0	1	0
D_3	3	0	1	1
D_4	4	1	0	0
D_5	5	1	0	1
D_6	6	1	1	0
D_7	7	1	1	1

Fig 2:Truth table

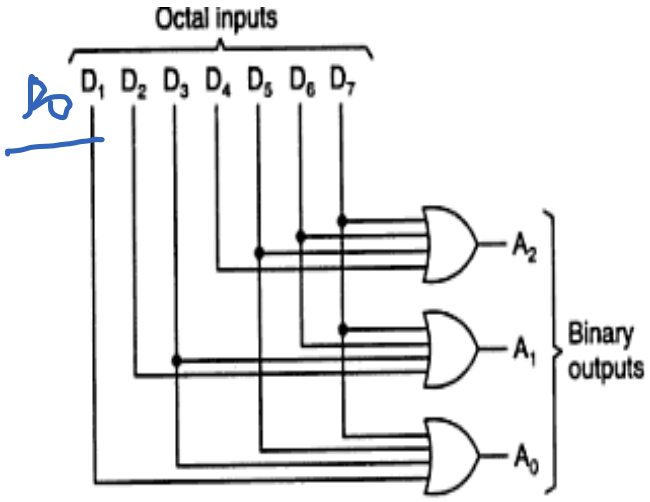


Fig 3: Logic diagram

ENCODER

Priority encoder

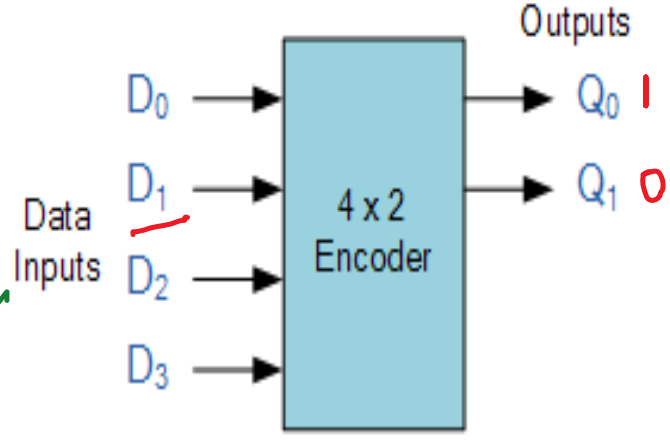
A 4 to 2 priority encoder has four inputs Y_3, Y_2, Y_1 & Y_0 and two outputs A_1 & A_0. Here, the input, Y_3 has the highest priority, whereas the input, Y_0 has the lowestpriority.

Inputs				Outputs		
Y_3	Y_2	Y_1	Y_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

Fig 4:Truth table

Encoders

4-to-2 Bit Binary Encoder



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

➢One of the main disadvantages of standard digital encoders is that they can generate the wrong output code when there is more than one input present at logic level "1". For example, if we make inputs D_1 and D_2 HIGH at logic "1" both at the same time, the resulting output is neither at "01" or at "10" but will be at "11" which is an output binary number that is different to the actual input present.

Decoders

3 x 8 D — 74 x 138

2 x 4 D — 74 x 139

↓

74 LS 138 @ 74 LS 139

Encoders →

Decimal to BCD Encoder

74 x x 147

9 x 4 (BCD o/p)

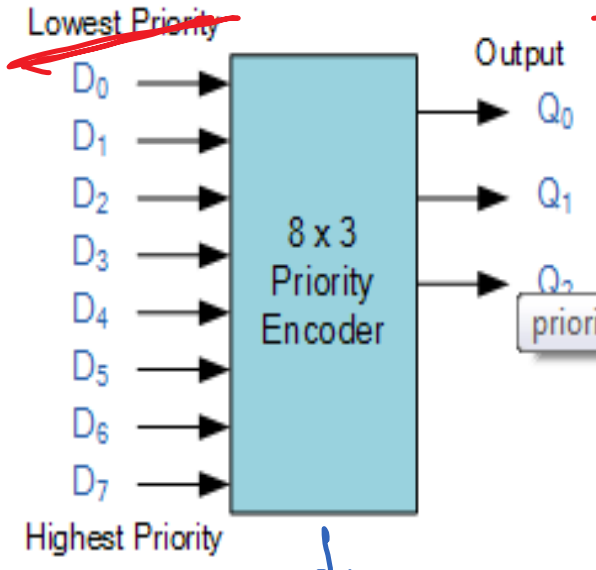
I/p.

74xx148

↓

priority encoder

8-to-3 Bit Priority Encoder



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

➢Priority encoders output the highest order input first for example, **if input lines "D2", "D3" and "D5" are applied simultaneously the output code would be for input "D5" ("101") as this has the highest order out of the 3 inputs.** Once input "D5" had been removed the next highest output code would be for input "D3" ("011"), and so on.

- An encoder is the inverse operation of a decoder.
- We can derive the Boolean functions by table 4–7

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

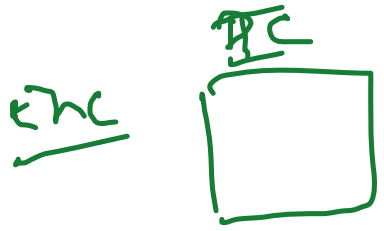
$$y = D_2 + D_3 + D_6 + D_7$$

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

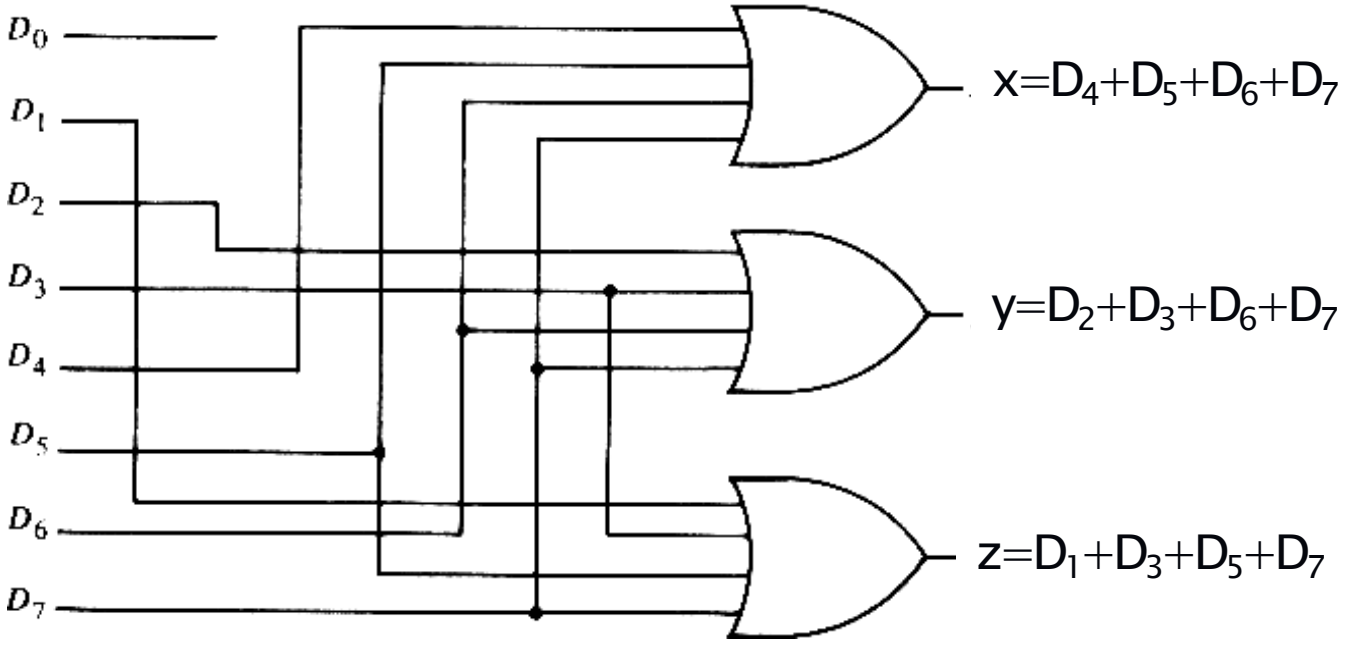
Table 4-7
Truth Table of Octal-to-Binary Encoder

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

Encoder



- An implementation



- limitations
  - illegal input: e.g. D3=D6=1
  - the output = 111 ('3 and '6)

Priority encoder

- If **two inputs are active simultaneously**, the output produces an undefined combination. We can establish an input priority to ensure that only one input is encoded.
- 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 output is the same as when D_0 is equal to 1.
- The discrepancy tables on Table 4–7 and Table 4–8 can resolve aforesaid condition by providing one more output to indicate that at least one input is equal to 1.

Priority encoder

4 x 2 encoder

V=0 · no valid inputs

V=1 · valid inputs

X's in output columns represent don't-care conditions

X's in the input columns are useful for representing a truth table in condensed form.

Instead of listing all 16 minterms of four variables.

Table 4-8
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

$V_{in} = 0$ off (0)

$V_{in} = 1$ (on)

Priority Encoder

- Resolve the ambiguity of illegal inputs
- Only one of the input is encoded

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

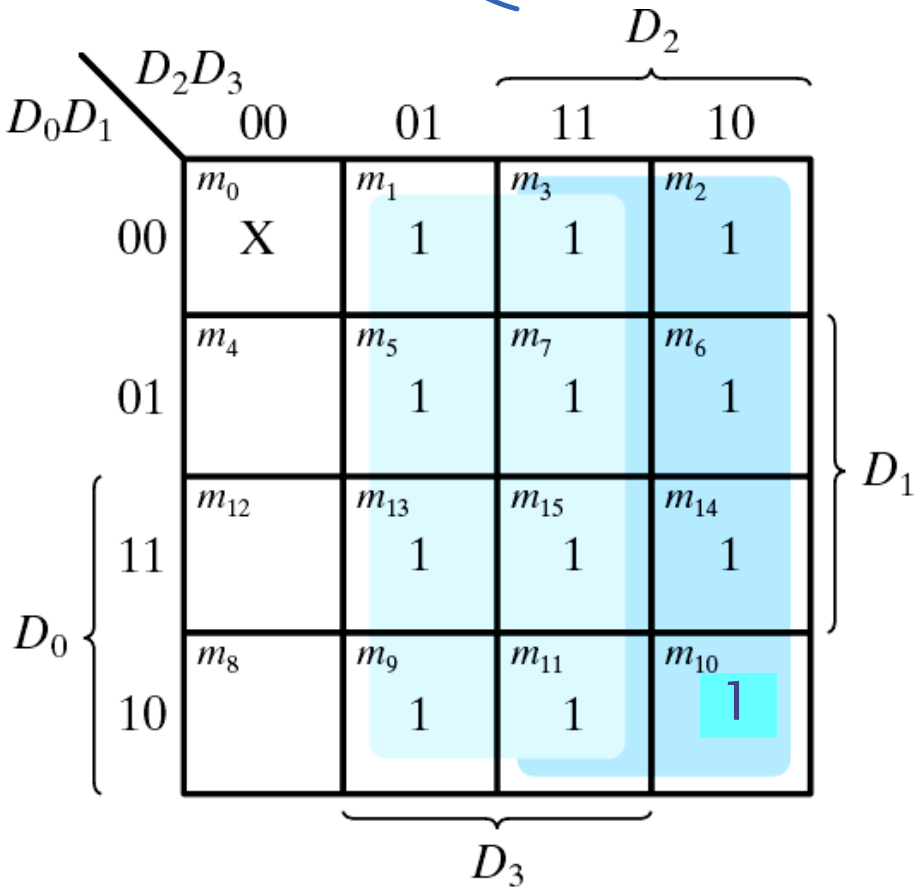
- D_3 has the highest priority
- D_0 has the lowest priority
- X: don't-care conditions = 1
- V: valid output indicator

Priority Encoder

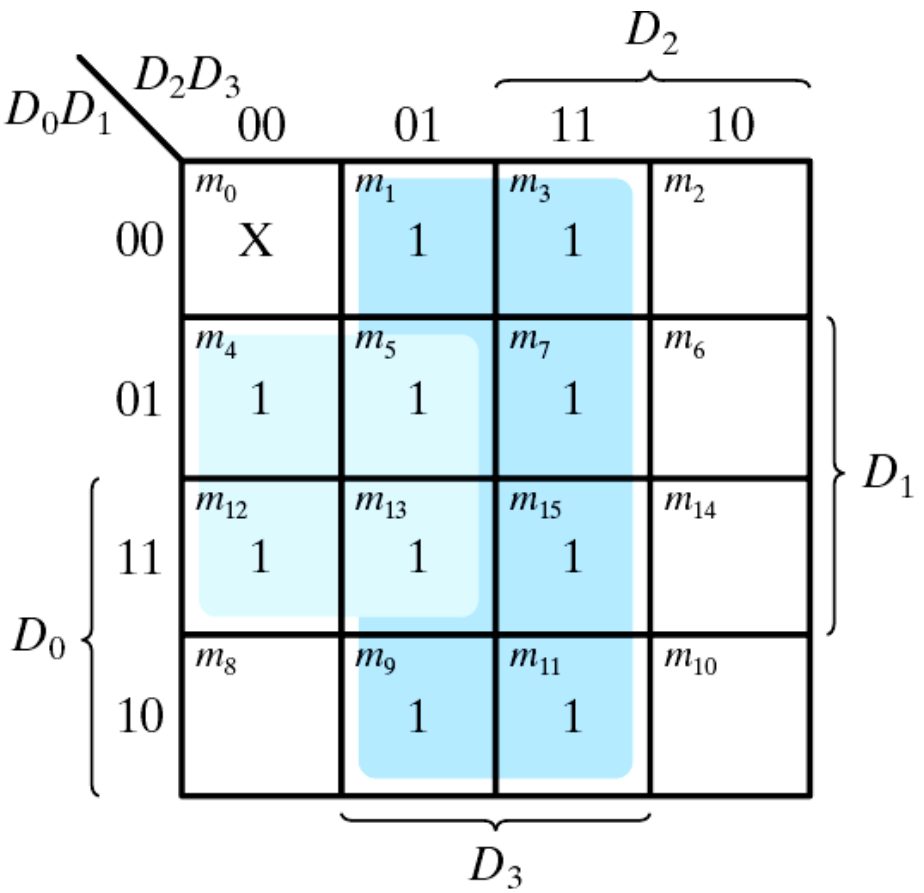
4 x 2 line P. E.

x

y



$x = D_2 + D_3$

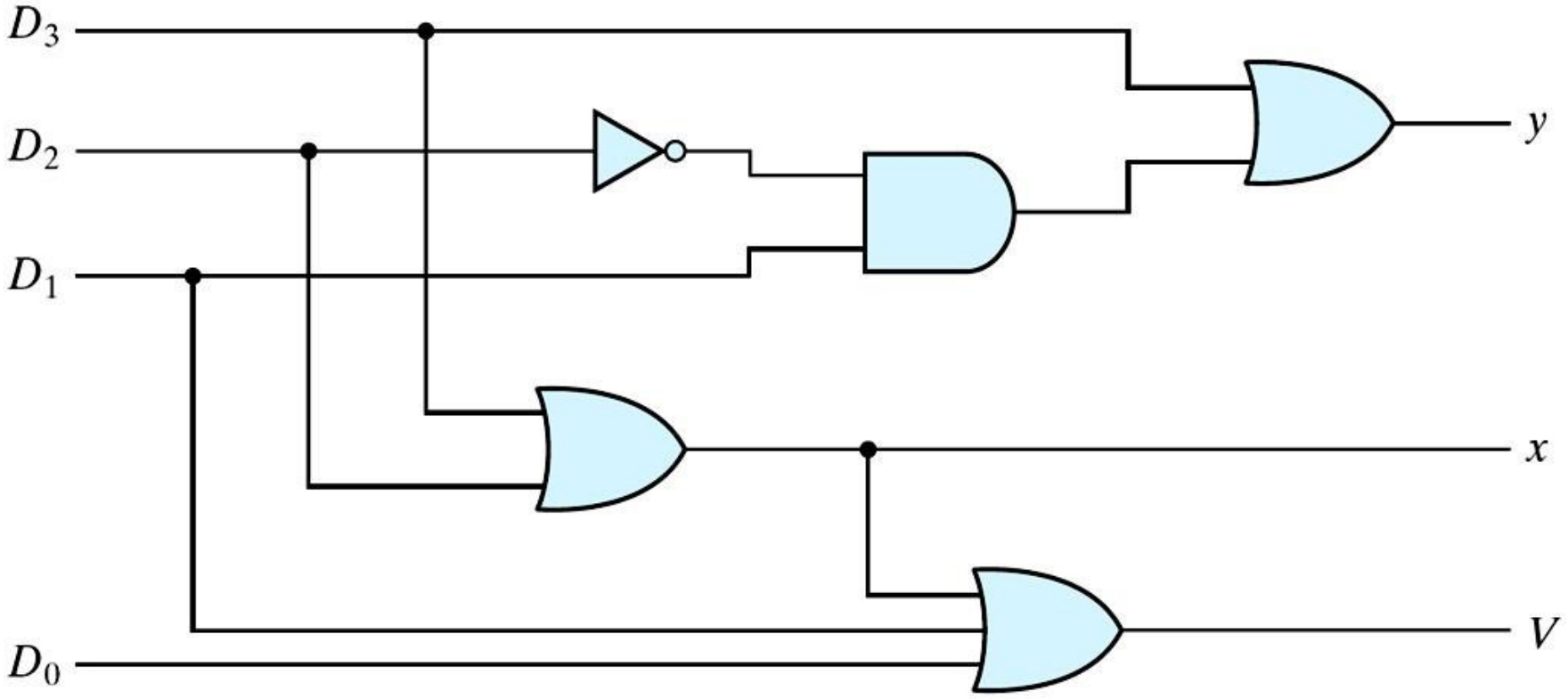


$y = D_3 + D_1D'_2$

V: 0

(00) (00)

Priority Encoder

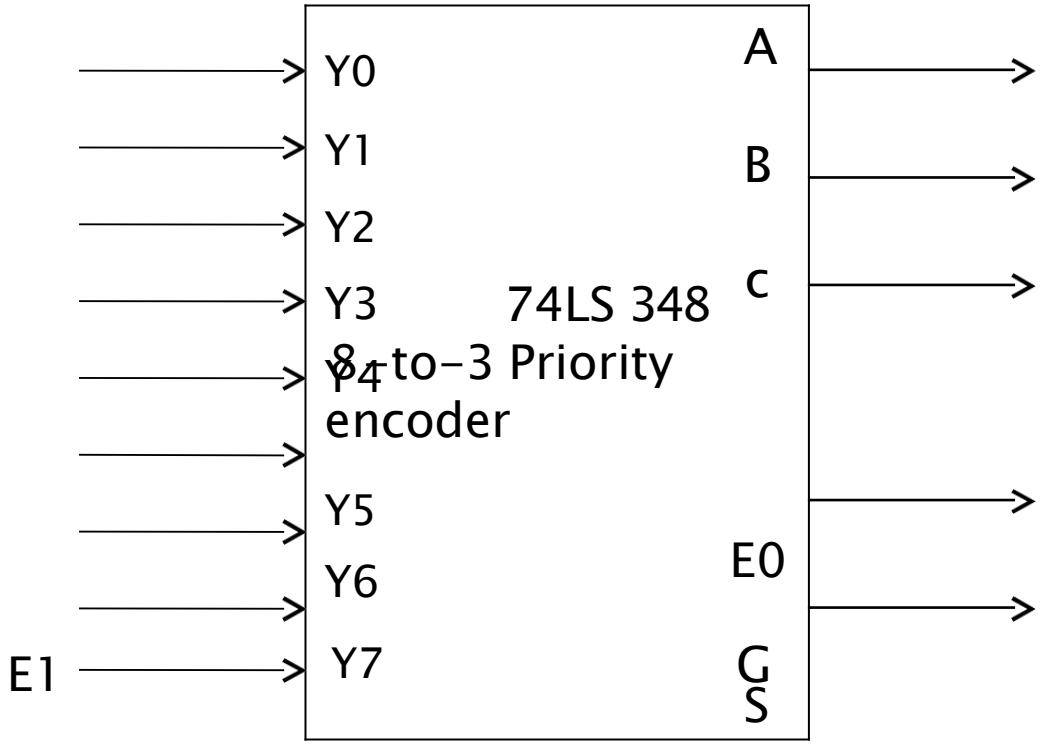


$$x = D_2 + D_3$$

$$y = D_3 + D_1 D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$

IC 74LS 348 8 TO 3 PRIORITY ENCODER



FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	Z	Z	Z	H	H
L	H	H	H	H	H	H	H	H	Z	Z	Z	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

H = high logic level, L = low logic level, X = irrelevant
Z = high-impedance state

MULTIPLEXERS

- Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.
- We have different types of multiplexers 2x1,4x1,8x1,16x1,32x1……

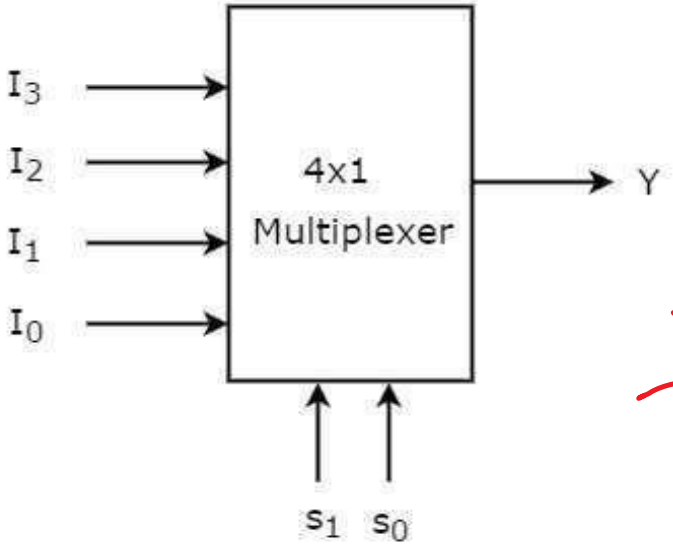


Fig 1: Block diagram

Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Fig 2: Truth table

MULTIPLEXERS

S_1	S_0	Y
0	0	Y_0
0	1	Y_1
1	0	Y_2
1	1	Y_3

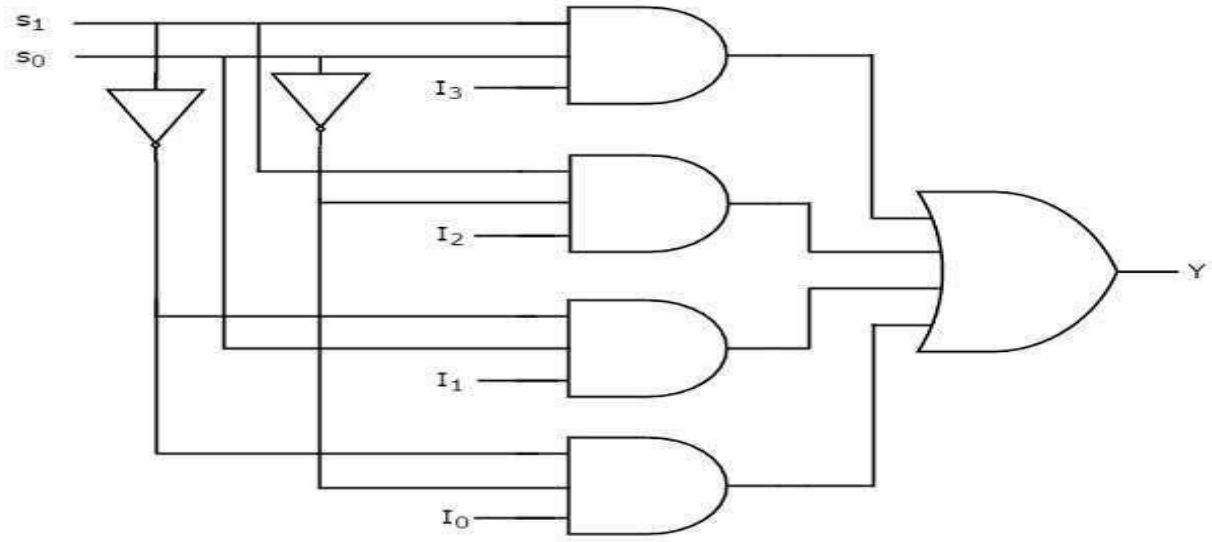


Fig 3: Logic diagram

- Now, let us implement the higher-order Multiplexer using lower-order Multiplexers.

MULTIPLEXERS

- Ex: 8x1 Multiplexer

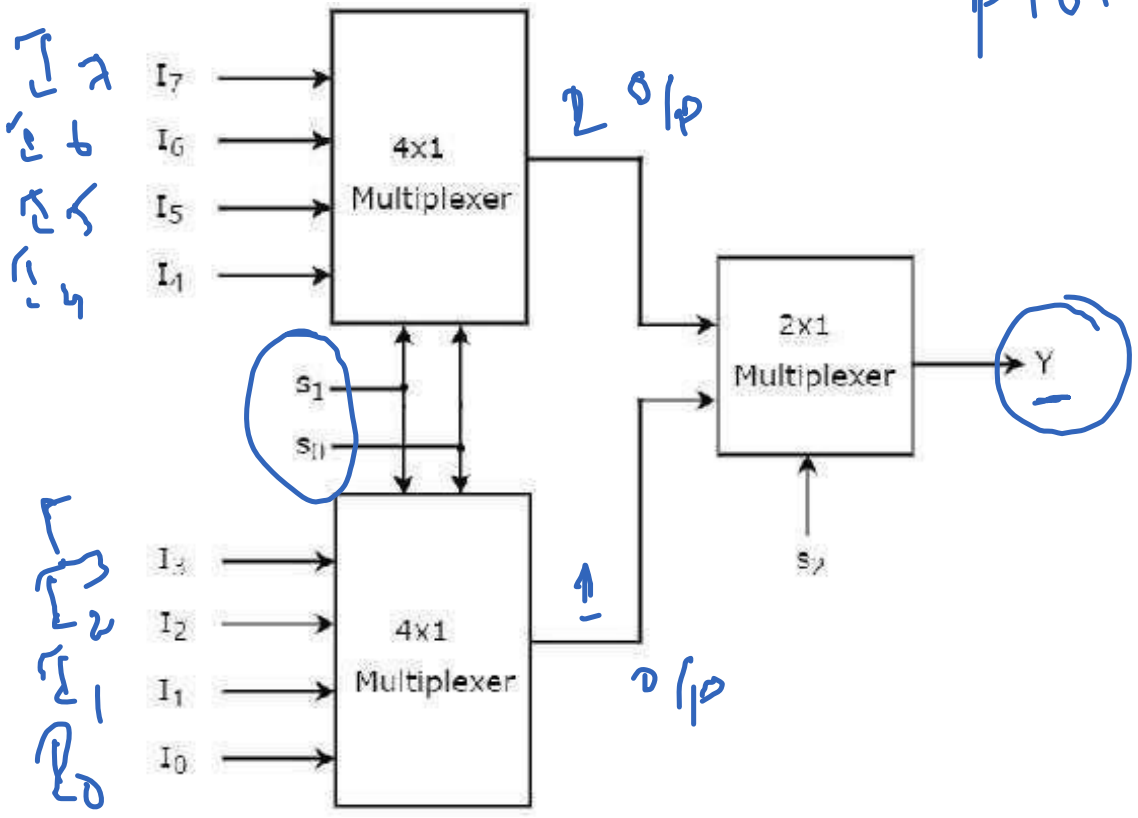
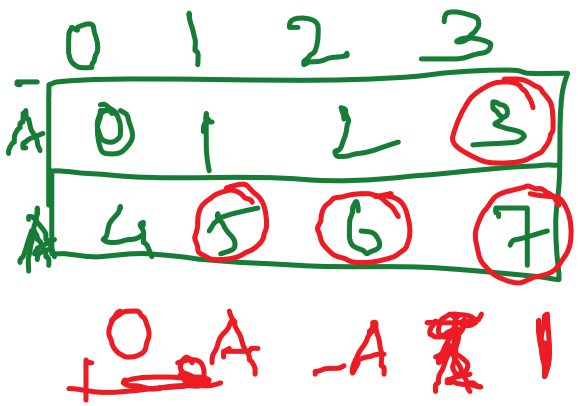
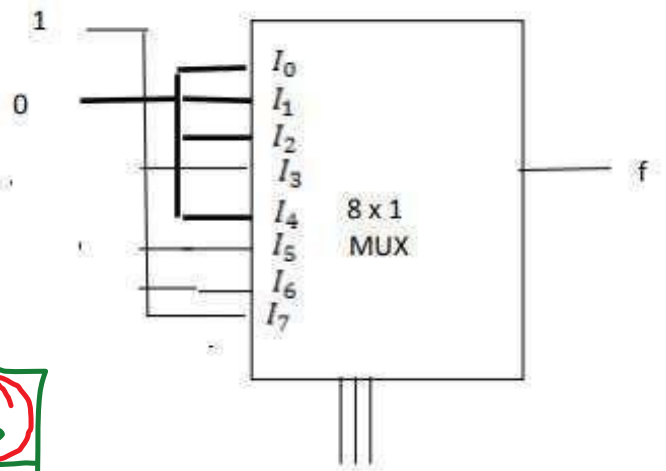


Fig 3: 8x1 Multiplexerdiagram

MULTIPLEXERS

- Implementation of Boolean function using multiplexer
- f(A1 , A2 , A3) =Σ(3,5,6,7) implementation using 8x1 mux



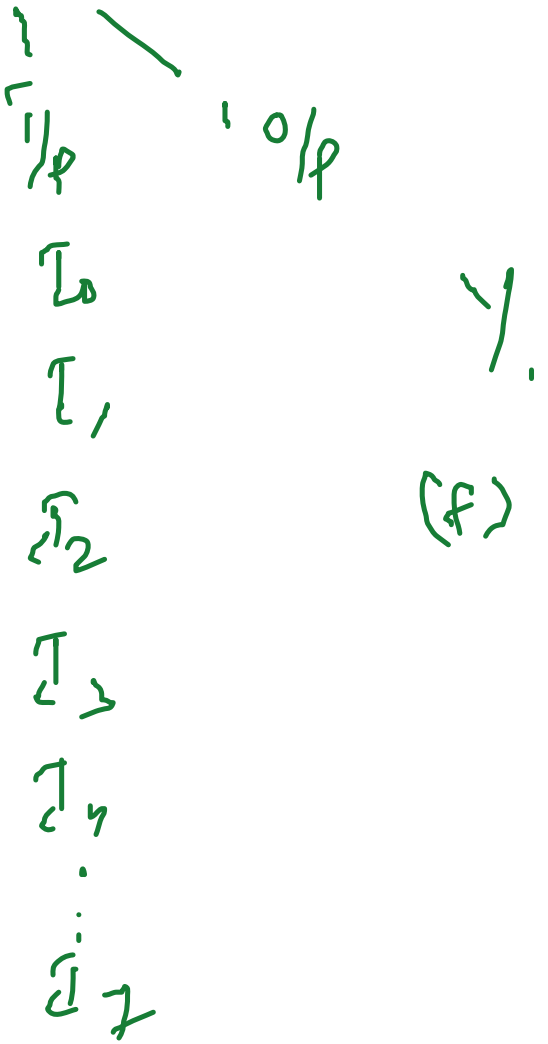


$A_1 A_2 A_3$

0 0 0

0 0 1

0 1 0

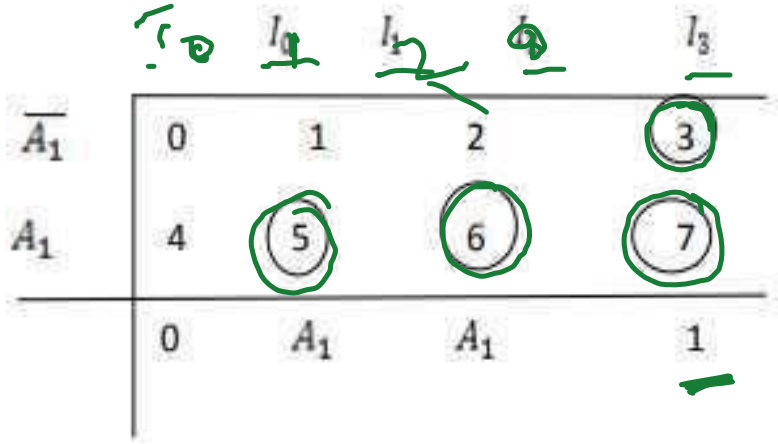


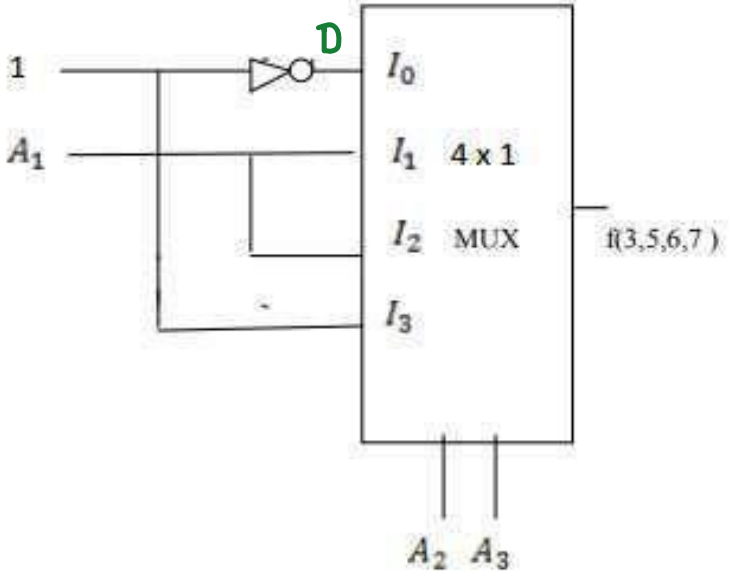
MULTIPLEXERS

f(A1 , A2 , A3) =Σ(3,5,6,7) implementation using 4x1 mux

Method:1

Minterms	A_1	A_2	A_3	f
0	0	0	0	0
1	0	0	1	0
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	1

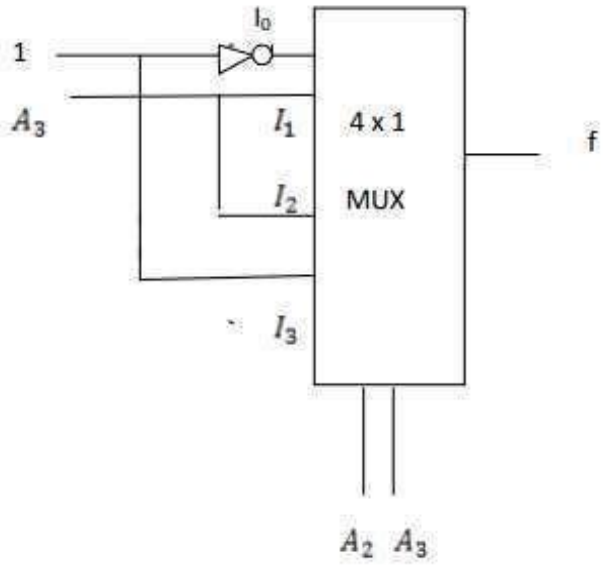




MULTIPLEXERS

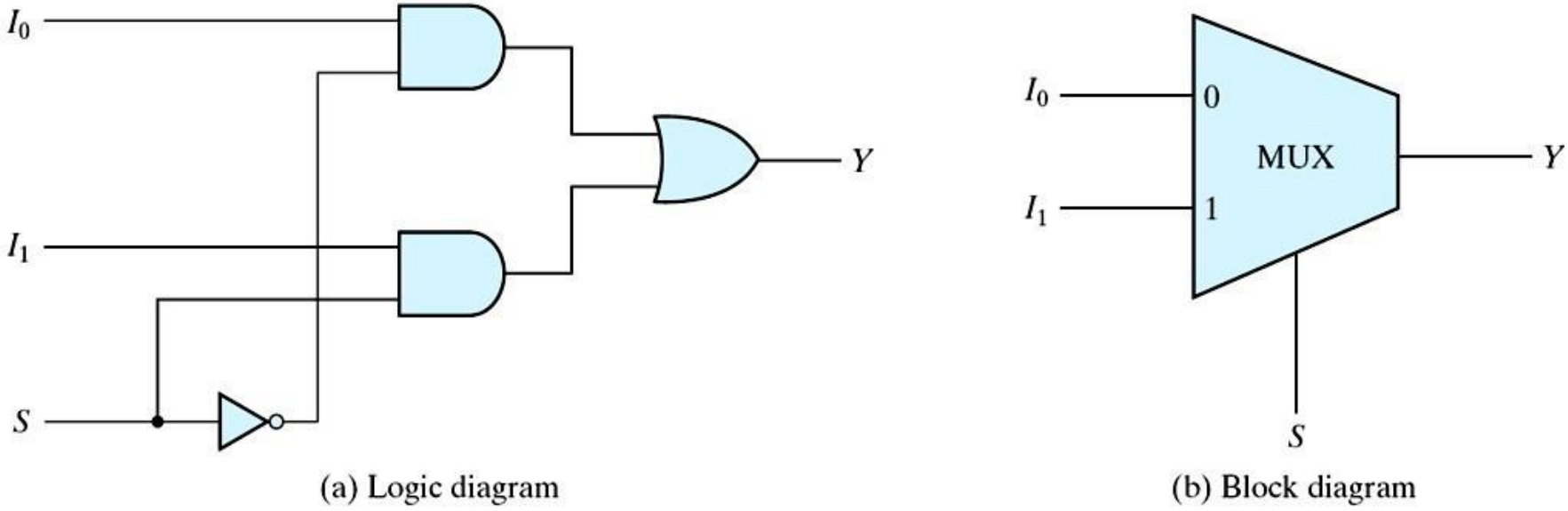
Method:2

Minterm	A_1	A_2	A_3	f	
0	0	0	0	0	
1	0	0	1	0	f= 0 I_0
2	0	1	0	0	
3	0	1	1	1	f= A_3 I_1
4	1	0	0	0	
5	1	0	1	1	f= A_3 I_2
6	1	1	0	1	
7	1	1	1	1	f= 1 I_3



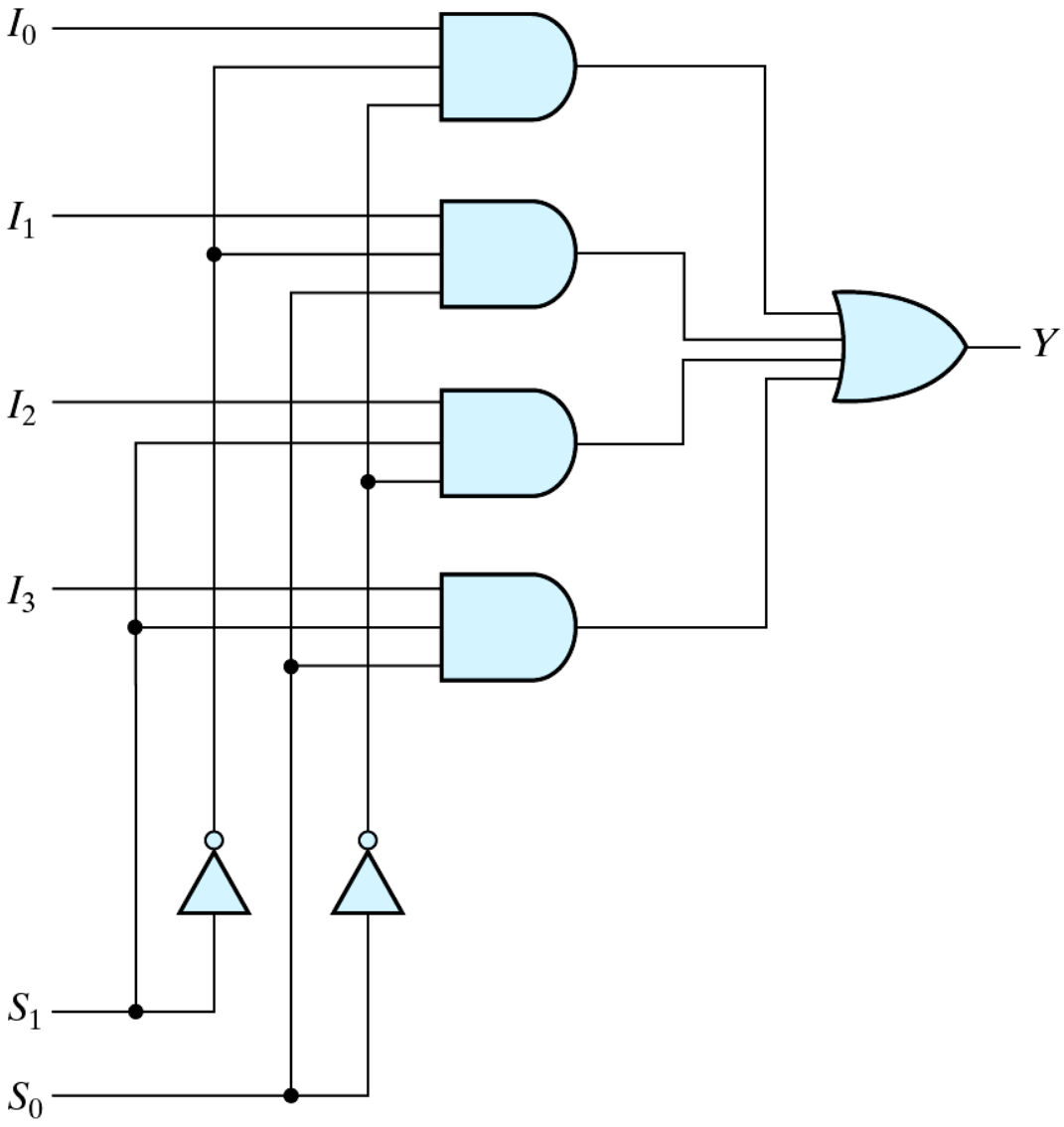
Multiplexer

- Select binary information from one of many input lines and direct it to a single output line
- 2 input lines, n selection lines and one output line
- E.g.: 2-to-1-line multiplexer



Two-to-one-line multiplexer

4-to-1-Line Multiplexer

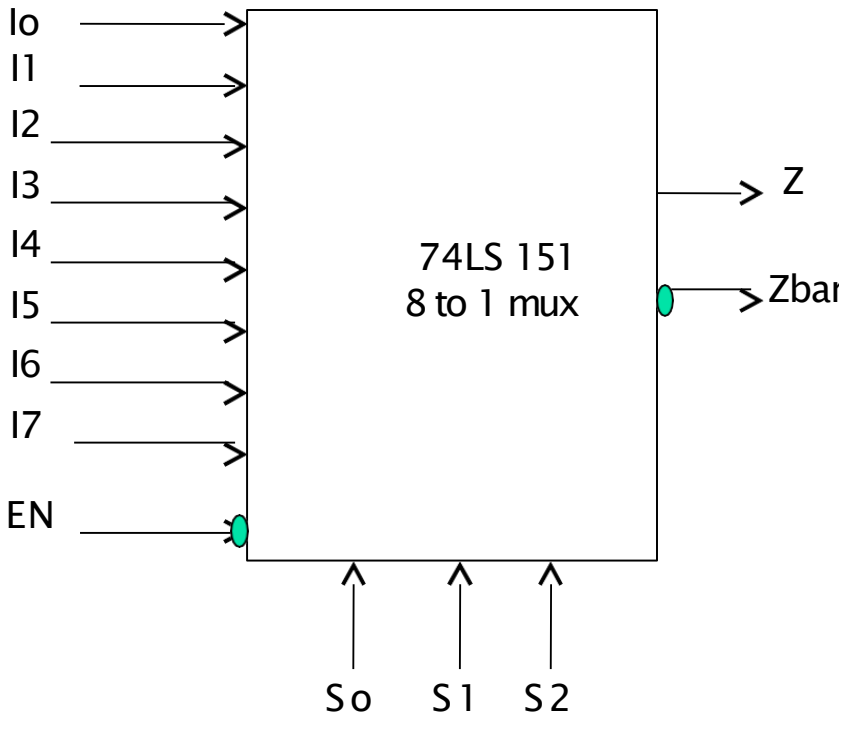


(a) Logic diagram

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

(b) Function table

8 to 1 Multiplexer using IC 74LS 151



TRUTH TABLE

$\overline{E}$	S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	$\overline{Z}$	Z
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	L	X	X	X	X	H	X	X	X	L	H
L	H	L	H	X	X	X	X	X	L	X	X	H	L
L	H	L	H	X	X	X	X	X	H	X	X	L	H
L	H	H	L	X	X	X	X	X	X	L	X	H	L
L	H	H	L	X	X	X	X	X	X	H	X	L	H
L	H	H	H	X	X	X	X	X	X	X	L	H	L
L	H	H	H	X	X	X	X	X	X	X	H	L	H

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

- For **Dual 4 to 1 Multiplexer** we are using IC 74LS153. it has two set of input 4 Lines and with two set of Enables with two set of output lines.

- For **16 to 1 Multiplexer** we are using IC 74LS150. It has 16 input lines with 4 Select lines along with one Enable Active low and one output

Boolean Function Implementation Using MUX

- MUX: a decoder + an OR gate
- 2^n-to-1 MUX can implement any Boolean function of *n* input variable.
- Procedure:
  - assign an ordering sequence of the input variable
  - the rightmost variable (D) will be used for the input lines
  - assign the remaining n–1 variables to the selection lines w.r.t. their corresponding sequence
  - construct the truth table
    - consider a pair of consecutive minterms starting from m_0
    - determine the input lines

DEMULTIPLEXER

- A demultiplexer is a device that takes a single input line and routes it to one of several digital output lines.
- A demultiplexer of 2^n outputs has n select lines, which are used to select which output line to send the input.
- We have 1x2,1x4,8x1…. Demultiplexers.

1x2
1x4, 1x8 ....

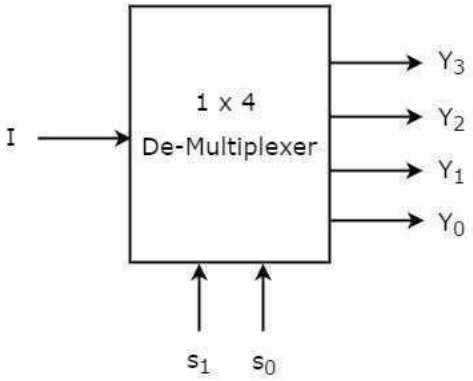


Fig:1 Block diagram

Selection Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

Fig :2 Truth table

DEMULTIPLEXER

Boolean functions for each output as

$$Y_3 = s_1 s_0 I$$

$$Y_2 = s_1 s_0' I$$

$$Y_1 = s_1' s_0 I$$

$$Y_0 = s_1' s_0' I$$

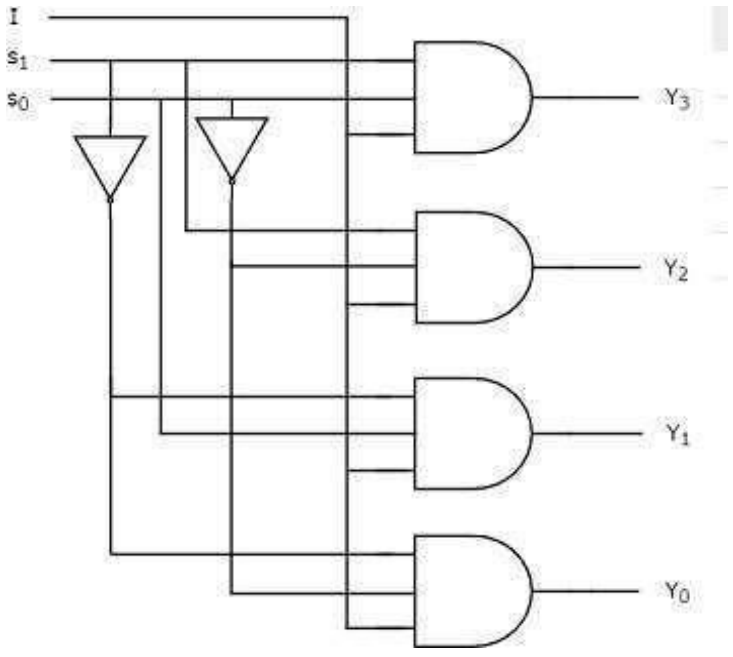


Fig:3 Logic diagram

BOOLEAN FUNCTION IMPLEMENTATION USING MULTIPLEXERS

- $F(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$
- This function can be implemented using multiplexer of various sizes
- 16 x 1 – A, B, C, D are selection lines
- 8 x 1
  - A as input line and B, C, D as selection lines
  - B as input line and A, C, D as selection lines
  - C as input line and A, B, D as selection lines
  - D as input line and A, B, C as selection lines
- 4 x 1
  - A, B as input lines and C, D as selection lines and vice-versa
  - A, C as input lines and B, D as selection lines and vice-versa
  - A, D as input lines and B, C as selection lines and vice-versa

Implementation using Multiplexer

$F(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$ using 16 x 1 Multiplexer