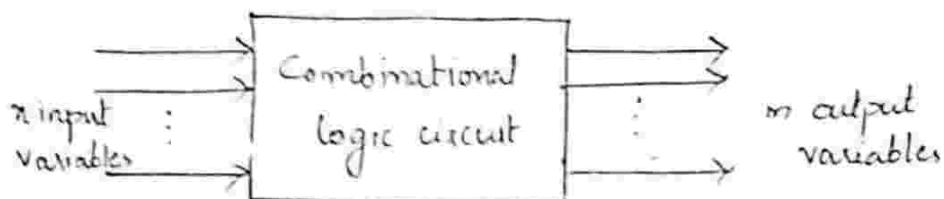


The output of a combinational circuit depends on its present inputs only.



The block diagram of a combinational circuit

Adders

The most basic arithmetic operation is the addition of 2 binary digits. This simple addition consists of 4 possible operations, namely

$$0+0=0, \quad 0+1=1, \quad 1+0=1, \quad 1+1=10$$

The 1st 3 operations produce a sum whose length is one digit

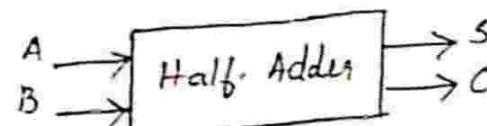
The 4th operation consists of 2 digits. The higher significant bit of this result is called a carry.

The Half-Adder

A half adder is a combinational circuit with 2 binary inputs

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

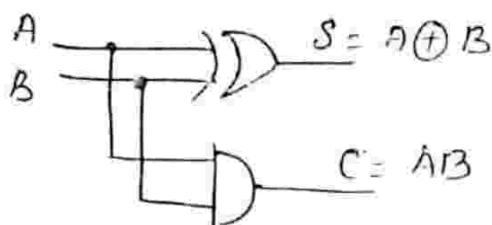
(a) Truth table.



(b) Block diagram

$$\text{The Sum } S = A\bar{B} + \bar{A}B \\ = A \oplus B$$

$$\text{Carry } C = AB$$

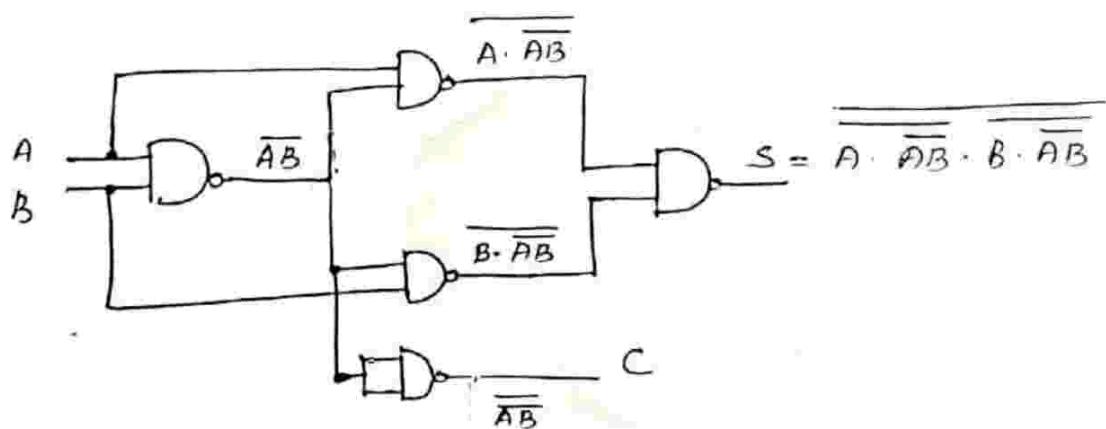


(a) logic diagram of half adder

NAND logic

$$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B} \\ = A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B}) \\ = A \cdot \overline{AB} + B \cdot \overline{AB} \\ = \overline{A \cdot AB} + \overline{B \cdot AB}$$

$$C = AB = \overline{\overline{AB}}$$



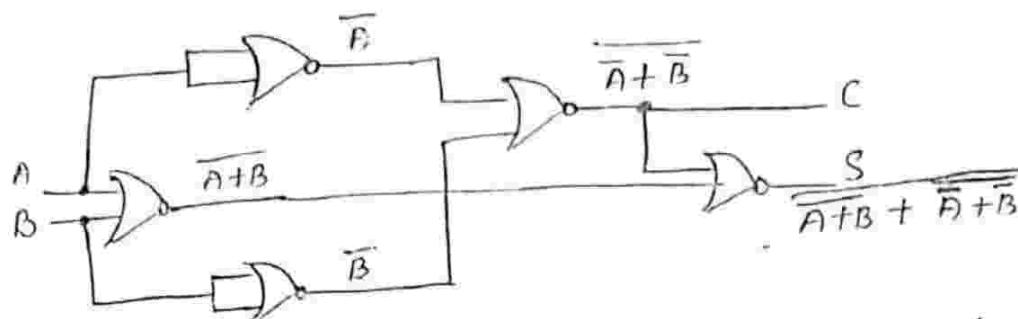
NOR logic

$$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B} \\ = A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B}) \\ = (A + B)(\bar{A} + \bar{B})$$

$$= \overline{A+B} + \overline{A} \cdot \overline{B}$$

2/VI

$$C = AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}}$$

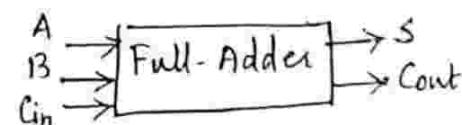


Full-Adder

A full adder is a combinational circuit that adds two bits & a carry and outputs a sum bit and a carry bit

Inputs

A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



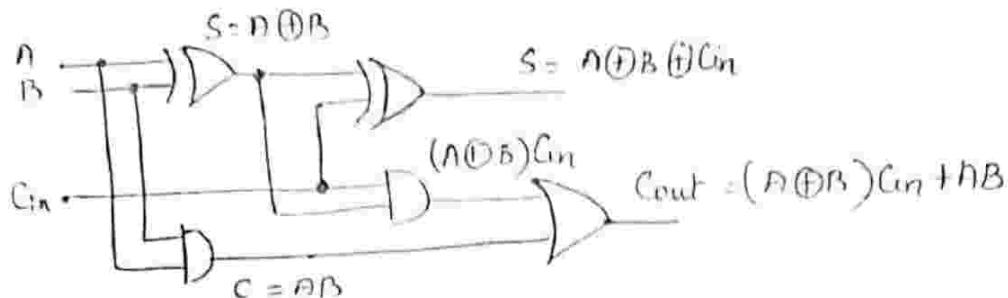
(b) Block diagram.

(a) Truth table

$$\begin{aligned} S &= \overline{A} \overline{B} \text{Cin} + \overline{A} B \overline{\text{Cin}} + A \overline{B} \overline{\text{Cin}} + A B \text{Cin} \\ &= (\overline{A} \overline{B} + \overline{A} B) \overline{\text{Cin}} + (A \overline{B} + A B) \text{Cin} \end{aligned}$$

$$= (A \oplus B) \bar{C}_{in} + (\bar{A} \oplus B) C_{in} = A \oplus B \oplus C_{in}$$

and $C_{out} = \bar{A}B C_{in} + A\bar{B} C_{in} + AB \bar{C}_{in} + ABC_{in}$
 $= AB + (A \oplus B) C_{in}$

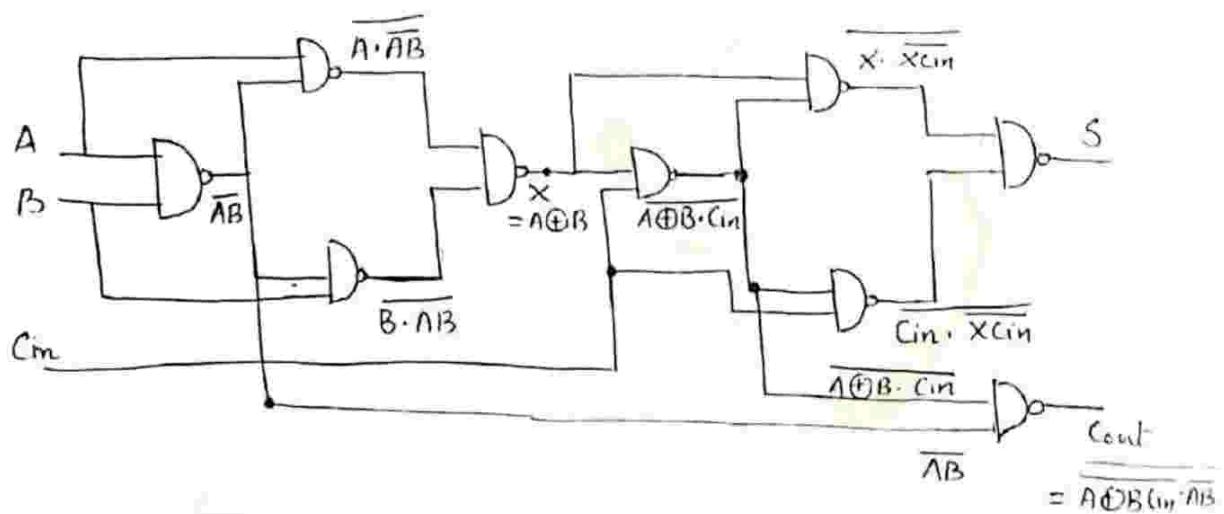


NAND Logic

$$A \oplus B = \overline{\overline{A} \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}} = x \text{ Then}$$

$$S = A \oplus B \oplus C_{in} = \overline{x \cdot \overline{x} C_{in}} \cdot \overline{C_{in} \cdot \overline{x} C_{in}} \\ = x \oplus C_{in}$$

$$C_{out} = C_{in} (A \oplus B) + AB \\ = \overline{C_{in} \cdot \overline{A \oplus B} \cdot \overline{AB}}$$



NOR logic

$$A \oplus B = X = \overline{A+B} + \overline{\bar{A}+\bar{B}}$$

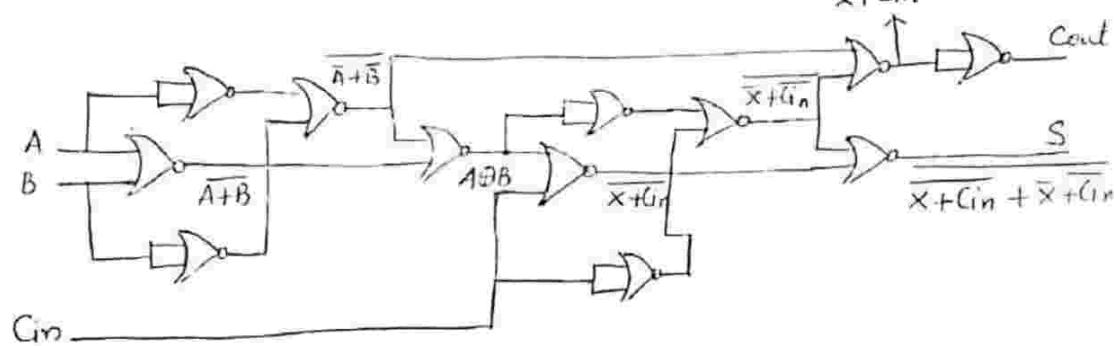
$$S = A \oplus B \oplus \text{Cin} = X \oplus \text{Cin}$$

$$= \overline{X+Cin} + \overline{\bar{X}+\bar{Cin}}$$

$$\text{Cout} = AB + \text{Cin} (A \oplus B)$$

$$= \overline{\bar{A}+\bar{B}} + \overline{\text{Cin} + \overline{A \oplus B}}$$

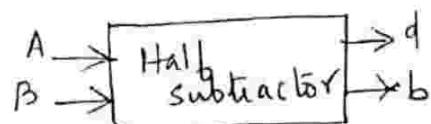
$$= \overline{\overline{X+Cin} + \overline{\bar{A}+\bar{B}}}$$

The Half-Subtractor

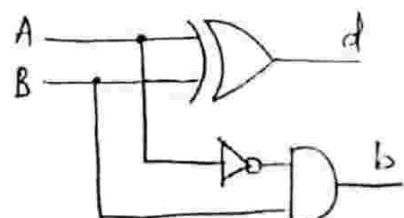
A half-subtractor is a combinational circuit that subtracts one bit from the other & produces the difference

Inputs		Outputs	
A	B	d	b
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

(a) Truth table



(b) Block diagram



$$\begin{aligned} \text{The difference } d &= A\bar{B} + \bar{A}B \\ &= A \oplus B \end{aligned}$$

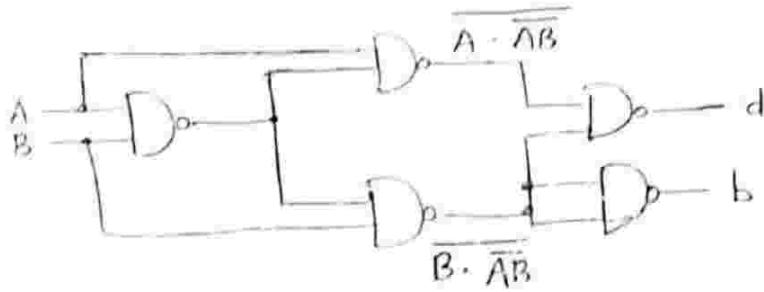
$$\text{Borrow } b = \bar{A}B$$

(c) Logic diagrams of a half-subtractor

NAND logic

$$d = A\bar{B} + \bar{A}B = \overline{\overline{A} \cdot \overline{AB}} + \overline{B \cdot \overline{AB}}$$

$$\begin{aligned} b &= \bar{A}B = \bar{B} + B\bar{B} = B(\bar{A} + \bar{B}) \\ &= B(\overline{AB}) \\ &= \overline{B(\overline{AB})} \end{aligned}$$



Logic diagram of a half-subtractor using only 2-input NAND gates

NOR logic

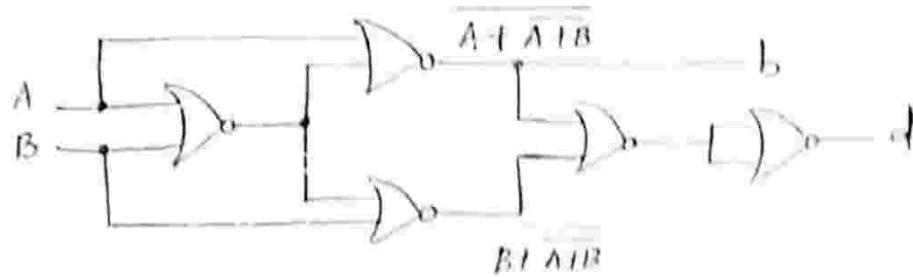
$$\begin{aligned} d &= A \oplus B = A\bar{B} + \bar{A}B \\ &= A\bar{B} + B\bar{B} + \bar{A}B + A\bar{A} \\ &= \overline{B(A+B)} + \overline{A(A+B)} \\ &= \overline{\overline{B(A+B)}} + \overline{\overline{A(A+B)}} \end{aligned}$$

$$\begin{aligned} b &= \bar{A}B = \overline{A(A+B)} \quad [\overline{A}\bar{B} + \overline{A}A = \overline{A}(A+B)] \\ &= \overline{\overline{A(A+B)}} \\ &= \overline{\overline{A}\overline{(A+B)}} \end{aligned}$$

$\overline{\overline{B(A+B)}}$ Apply deMorgan's theorem

$$= \overline{\overline{\overline{B}} + \overline{A+B}}$$

$$= \overline{B + (\overline{A+B})}$$



Logic diagram of a half-subtractor using 2 input NOR gates

The Full-Subtractor

A full-subtractor is a combinational circuit with three inputs (A, B, b_i) & two outputs d (difference) and b (borrow). The 1's & 0's of the o/p variables are determined from the subtraction of $A - B - b_i$.

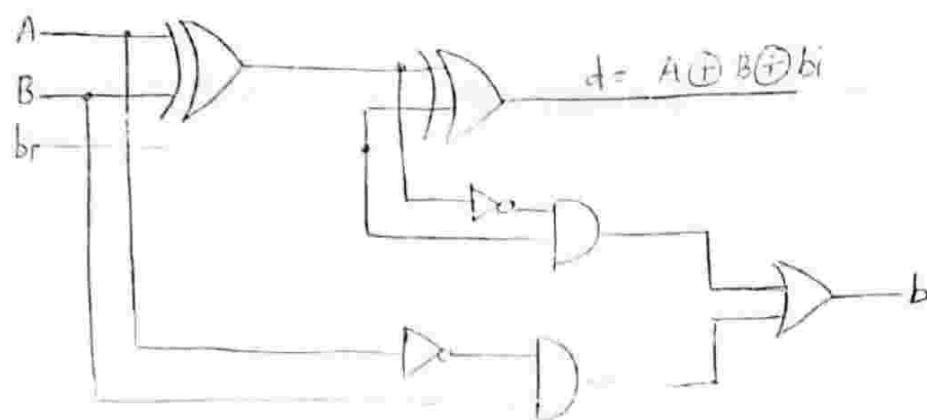
Inputs	Difference			Borrow b	$\overbrace{A \quad B \quad b_i}^{\text{Full subtractor}}$	d
	A	B	b_i			
0 0 0				0		
0 0 1				1		
0 1 0				1		
0 1 1				0		
1 0 0				1		
1 0 1				0		
1 1 0				0		
1 1 1				1		

(b) Block diagram

(a) Truth
table

$$\begin{aligned}
 d &= \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + AB\bar{b}_i + ABb_i \\
 &= b_i (AB + \bar{A}\bar{B}) + \bar{b}_i (A\bar{B} + \bar{A}B) \\
 &= b_i (\overline{A \oplus B}) + \bar{b}_i (A \oplus B) \\
 &= A \oplus B \oplus b_i
 \end{aligned}$$

$$\begin{aligned}
 b &= \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + AB\bar{b}_i + ABb_i \\
 &= \bar{A}B(b_i + \bar{b}_i) + b_i (AB + \bar{A}\bar{B}) \\
 &= \bar{A}B + b_i (\overline{A \oplus B})
 \end{aligned}$$

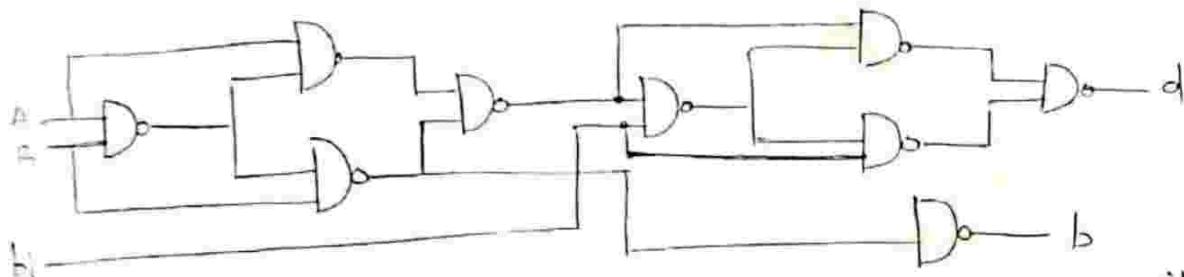


NAND Logic

$$d = A \oplus B \oplus b_i$$

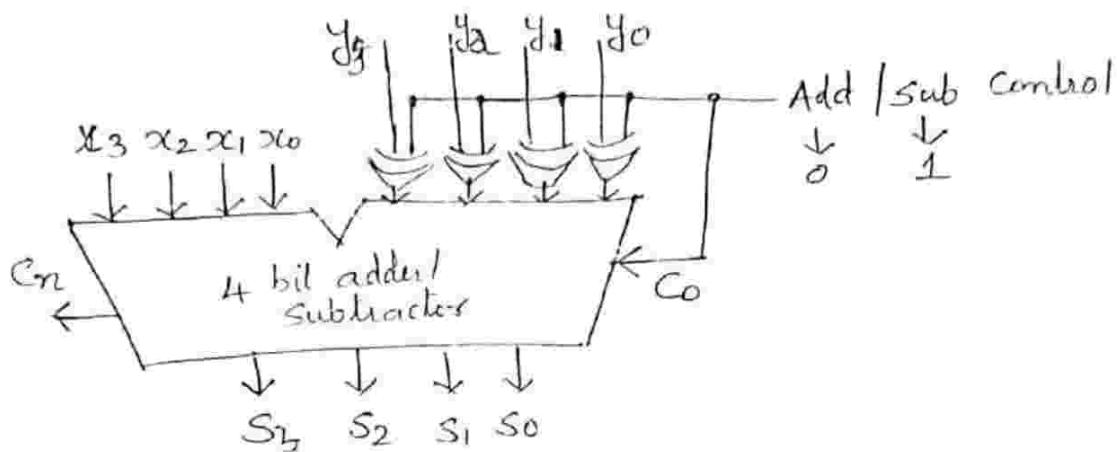
$$= \bar{A} \cdot \bar{B} \cdot b_i$$

$$\begin{aligned} b &= \overline{\bar{A}B + b_i(\bar{A} \oplus \bar{B})} = \overline{\bar{A}B} \cdot \overline{b_i(\bar{A} \oplus \bar{B})} \\ &= \overline{\bar{A}B} \cdot \overline{b_i(\bar{A} \oplus \bar{B})} = \overline{B(\bar{A} + \bar{B})} \cdot \overline{b_i(\bar{b}_i + (\bar{A} \oplus \bar{B}))} \\ &= \overline{B \cdot \bar{A}\bar{B}} \cdot \overline{b_i \cdot (\bar{A} \oplus \bar{B})} \end{aligned}$$



Logic diagram of a full-subtractor using only 2 input
NAND gates

4 bit Adder/ Subtractor



→ The above logic circuit can be used to perform either addition or subtraction based on the value applied to the Add/Sub input control line.

→ The add/sub ctrl line is zero for addition & applying 'Y' vector unchanged to one of the adder's inputs along with a carry-in signal, $C_0 = 0$

$$\begin{array}{l} \text{Example } \quad x_3 \quad x_2 \quad x_1 \quad x_0 \\ (1010 + 0011) \Rightarrow \begin{matrix} 1 & 0 & 1 & 0 \\ y_3 & y_2 & y_1 & y_0 \\ 0 & 0 & 1 & 1 \end{matrix} \end{array}$$

$$\begin{array}{r}
 \begin{array}{c} \text{xOR} \\ \hline \end{array} \\
 \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \quad (\text{Add Ctrl} = 0) \\
 \hline
 \begin{array}{cccc} 0 & 0 & 1 & 1 + \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 1 \end{array} \quad (C_0 = 0) \\
 \hline
 \begin{array}{c} \text{(Y vector)} \\ \hline \end{array} \\
 \begin{array}{cccc} 1 & 0 & 1 & 0 \end{array} \quad (\text{x vector}) \\
 \hline
 \begin{array}{cccc} 1 & 1 & 0 & 1 \end{array} \quad (\text{s vector}) \\
 \hline
 \begin{array}{cccc} s_3 & s_2 & s_1 & s_0 \end{array}
 \end{array}$$

→ when the Add/Sub = 1

→ Y vector is 1's complemented by XOR gate

→ Co is set to 1 to complete the 2's complementation of Y

Example : $1010 - 0011 = \underline{0111}$

$1010 + 2's \text{ complement of } [0011]$

$1010 + 1101 = \underline{\underline{0111}}$

$x_3 \quad x_2 \quad x_1 \quad x_0$

1 0 1 0

$y_3 \quad y_2 \quad y_1 \quad y_0$

0 0 1 1

$$\begin{array}{r} \text{XOR} \\ \hline 1 & 1 & 1 & 1 \end{array} \quad (\text{Add/Sub ch1} = 1)$$

$$\begin{array}{r} + \\ \hline 1 & 1 & 0 & 0 \end{array} \quad (C_0 = 1)$$

$$\begin{array}{r} + \\ \hline 1 & 1 & 0 & 1 \end{array} \quad (Y \text{ vector})$$

$$\begin{array}{r} + \\ \hline 1 & 0 & 1 & 0 \end{array} \quad (X \text{ vector})$$

$\boxed{1}$

$$\begin{array}{r} \hline 0 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

Multiplexers (Data Selectors)

35 copy

Multiplexing means sharing

2 types of multiplexing - time multiplexing & Frequency multiplexing.

Time multiplexing - one & only one device is using the line, at any given time

Frequency multiplexing - several devices share a common line by transmitting at different frequencies.

A multiplexer is a logic circuit that accepts several data inputs and allows only one of them at a time to get through to the output

The routing of the desired data input to the output is controlled by SELECT inputs.

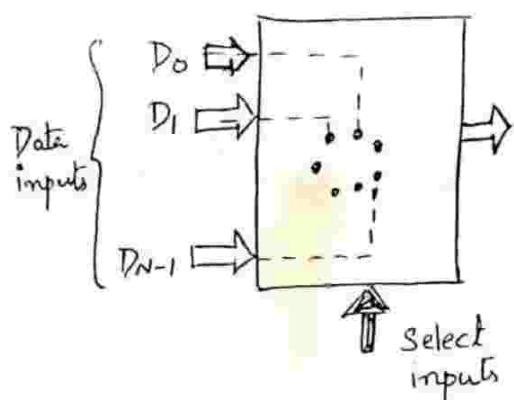


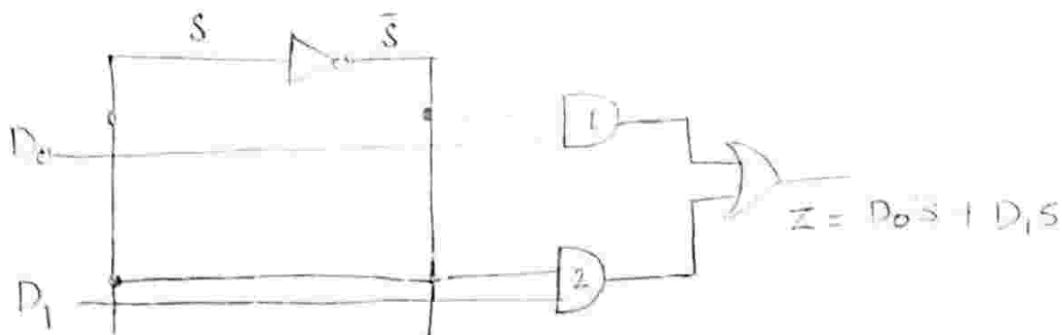
Fig : Functional diagram of a digital multiplexer

Normally there are 2^n input lines & n select lines whose bit combinations determine which input is selected.

The digital code applied to the select inputs determines which data inputs will be switched to the output.

Multiplexer selects 1 out of N input data sources
 & transmits the selected data to a single output channel.

Basic 2-Input Multiplexing



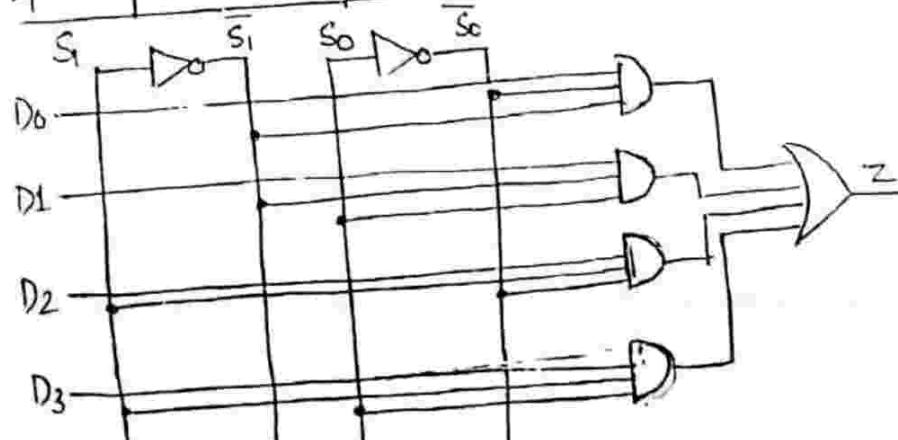
(a) 2- input multiplexer

<u>S</u>	<u>Output</u>	<u>C</u>	<u>D₀</u>	<u>the output</u>
0	D_0	0	D_0	$Z = D_0\bar{S} + D_1S$
1	D_1	1	D_1	$Z = D_1\bar{S} + D_1S$

(b) Function table

when $S=0$, AND gate 1 is enabled & AND gate 2 is disabled so, $Z=D_0$.
 When $S=1$, AND gate 1 is disabled & AND gate 2 is enabled so, $Z=D_1$.

4 input Multiplexer



(a) Logic diagram

Select Inputs	Outputs
$S_1 \quad S_0$	Z
0 0	D_0
0 1	D_1
1 0	D_2
1 1	D_3

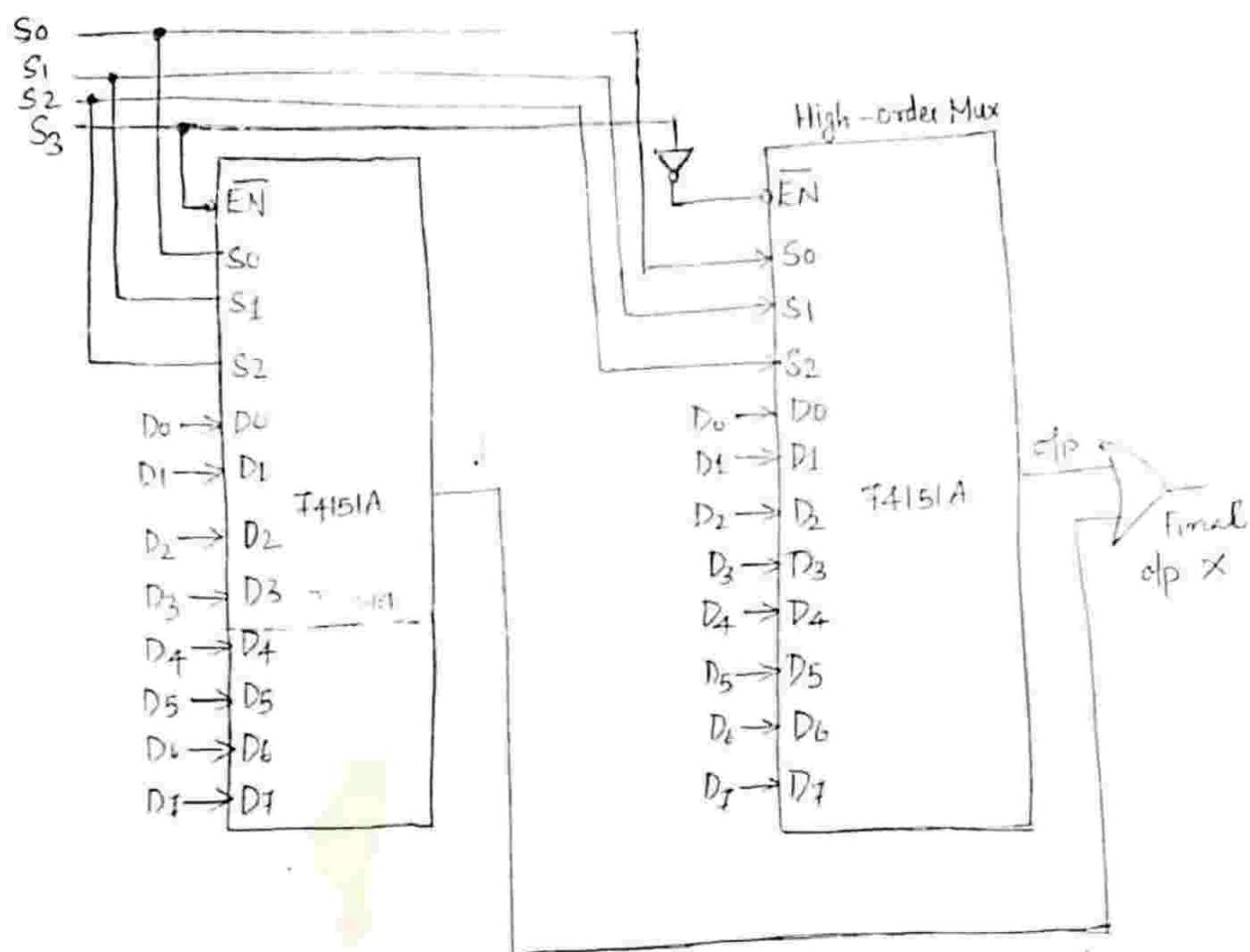
(b) Function table

16 Input Multiplexer From 2 8 Input Multiplexers

2 8-1/p multiplexers (T4151A) to get a 16 input multiplexer.

One OR gate and one inverter are also required.

The four select inputs S_3, S_2, S_1, S_0 will select one of the 16 inputs to pass through to X (output)



Logic diagram of cascading of two 8x1 muxes to get a 16-bit mux.

The S_3 input determines which multiplexer is enabled. When $S_3 = 0$, the left multiplexer is enabled and S_2, S_1, S_0 determine which of its date inputs will appear at its output & pass through OR gate to X .

When $S_3=1$, the eight multiplexor is enabled & $S_2, S_1 \in$ inputs select one of its data inputs for passage to output \Rightarrow

Applications of Multiplexers

Data selection

Data routing

Operation sequencing

Parallel to serial conversion

Waveform generation

Logic function generation

Q: Use a multiplexer to implement the logic function

$$F = A \oplus B \oplus C.$$

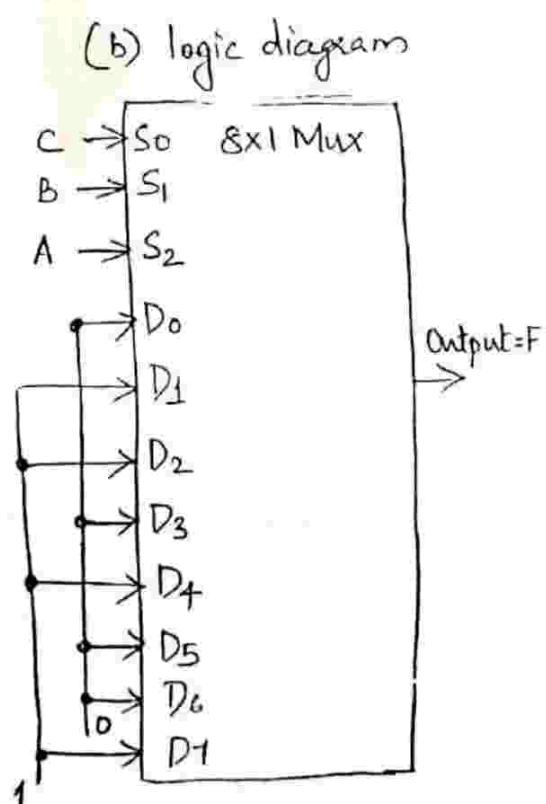
Since there are 3 input variables, we can use a multiplexer with 3 data select inputs (an 8 to 1 Mux).

Data select 3 inputs - S_2, S_1, S_0 for

Input variable - A, B, C

S_2	S_1	S_0	$F = A \oplus B \oplus C$
A	B	C	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a) Truth table

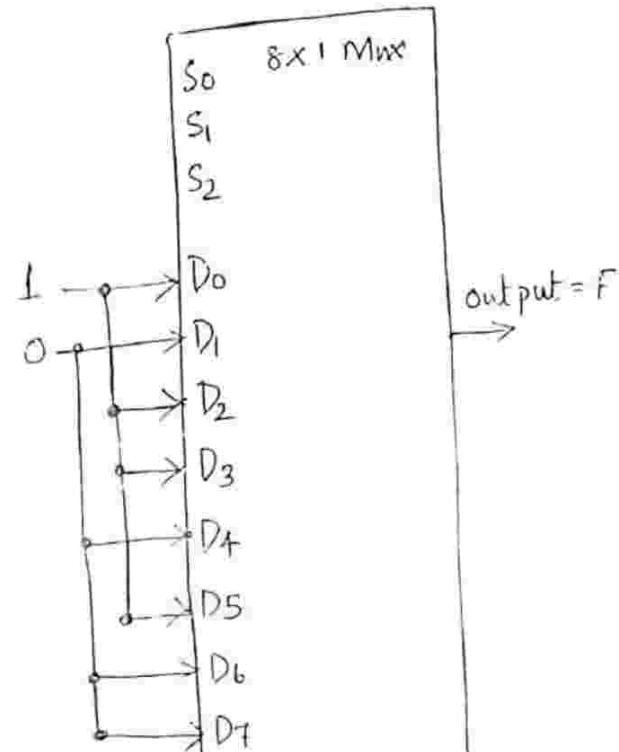


* Q2 Implement the following function using 3 to 1 Mux

$$F(x, y, z) = \Sigma m(0, 2, 3, 5)$$

S_2	S_1	S_0		F
x	y	z		
0	0	0	1	D_0
0	0	1	0	
0	1	0	1	D_2
0	1	1	1	D_3
1	0	0	0	
1	0	1	1	D_5
1	1	0	0	
1	1	1	0	

(a) Truth table



(b) Logic diagram

Logic 0 is connected to other data inputs D_1 , D_4 , D_6 & D_7 .

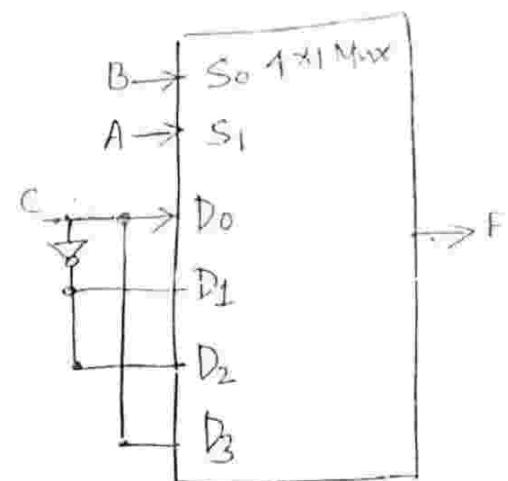
Logic 1 is connected to data inputs D_0 , D_2 , D_3 & D_5 .

Q3 Use a 4×1 mux to implement the logic function

$$F(A, B, C) = \Sigma m(1, 2, 4, 7)$$

S_1	S_0		
A	B	C	F
0	0	0	0
0	0	1	1 $F = C$
0	1	0	1 D_2
0	1	1	0 $F = \bar{C}$
1	0	0	1 D_4
1	0	1	0 $F = \bar{C}$
1	1	0	0 $F = C$
1	1	1	1 D_7

(a) Truth table



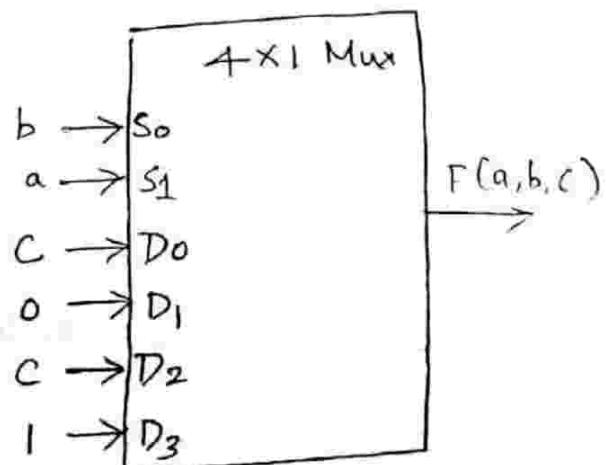
(b) Multiplexer implementation

Q3 Implement the function $F(a, b, c) = ab + \bar{b}c$ using 4:1 mux

$$\begin{aligned}
 F(a, b, c) &= ab + \bar{b}c \\
 &= ab(c + \bar{c}) + \bar{b}c(a + \bar{a}) \\
 &= abc + ab\bar{c} + a\bar{b}c + \bar{a}\bar{b}c \\
 &= \bar{a}\bar{b}c + a\bar{b}c + ab\bar{c} + abc \\
 &= \Sigma m(1, 5, 6, 7)
 \end{aligned}$$

S_1	S_0		
a	b	c	F
0	0	0	0
0	0	1	1 $F = C$
0	1	0	0 $F = 0$
0	1	1	0
1	0	0	0 $F = C$
1	0	1	1
1	1	0	1 $F = 1$
1	1	1	1

(a) Truth table



(b) Multiplexer implementation

q4 Implement the following Boolean function using an 8:1 multiplexer considering D as the input & A, B, C as the selection lines:

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

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

$$A\bar{B}(C + \bar{C})(D + \bar{D}) = (A\bar{B}C + A\bar{B}\bar{C})(D + \bar{D})$$

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

$$BD(A + \bar{A})(C + \bar{C}) = (ABD + \bar{A}BD)(C + \bar{C})$$

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

$$\bar{B}C\bar{D}(A + \bar{A}) = A\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D}$$

$$F(A, B, C, D) = ABCD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + ABC\bar{D}$$

$$+ \bar{A}BCD + AB\bar{C}D + \bar{A}B\bar{C}D + A\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D}$$

$$= 1011 + 1001 + 1010 + 1000 + 1111$$

$$+ 0111 + 1101 + 0101 + 1010 + 0010$$

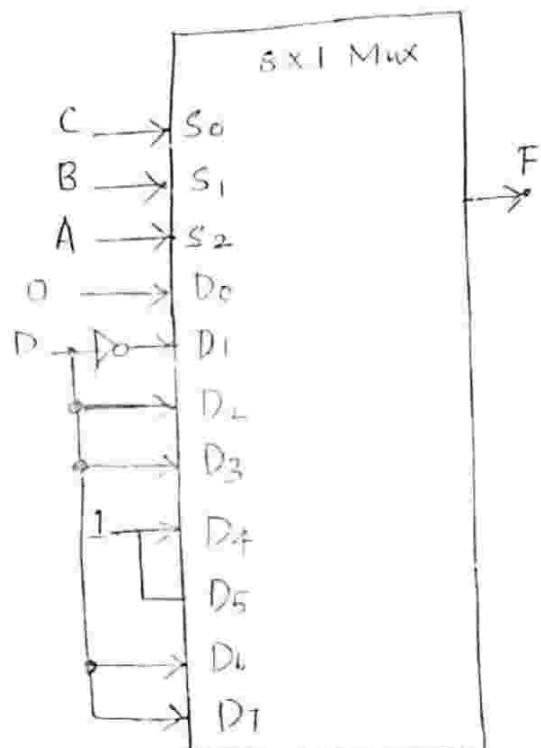
$$= \Sigma m(11, 9, 10, 8, 15, 7, 13, 5, 2)$$

$$= \Sigma m(2, 5, 7, 8, 9, 10, 11, 13, 15)$$

The truth table for the given four variable function F
is its implementation using an 8:1 mux with three select
inputs A, B & C

S_2	S_1	S_0	A	B	C	D	F
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	0	1	0	0	1	0	1
0	0	1	1	1	0	1	0
0	1	0	0	0	0	0	0
0	1	0	0	1	0	1	1
0	1	1	0	0	0	0	0
1	0	0	0	0	1	0	1
1	0	0	0	1	1	1	1
1	0	1	0	0	1	0	1
1	0	1	1	1	1	1	1
1	1	0	0	0	0	0	0
1	1	0	0	1	1	1	1
1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1

(a) Truth table



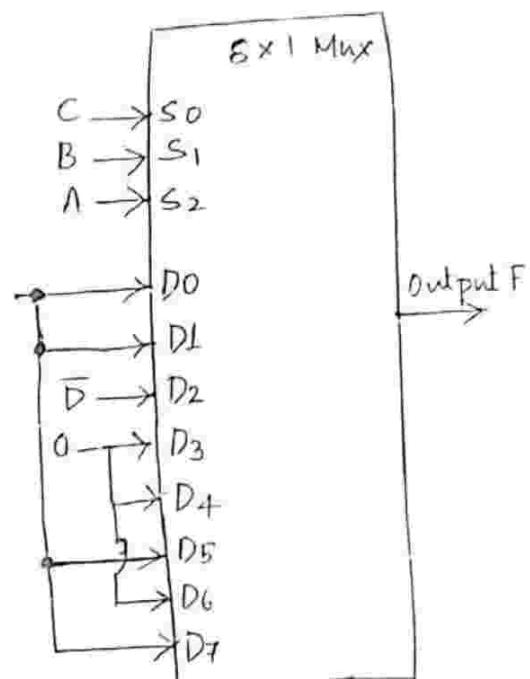
(b) Logic diagram

Q5 Use a multiplexer having 3 data select inputs to implement the logic for the function given below. Also realize the same using a 16:1 MUX.

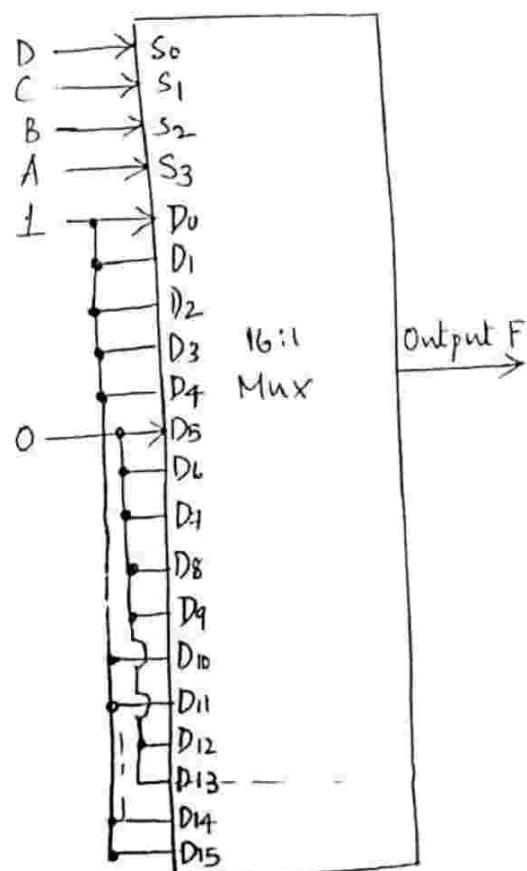
$$F = \Sigma m(0, 1, 2, 3, 4, 10, 11, 14, 15)$$

S_2	S_1	S_0	A	B	C	D	F
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	$F = 1$
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	$F = 1$
0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	$F = 0$
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	$F = 0$
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	$F = 1$
1	0	1	1	0	0	0	1
1	1	0	0	0	0	0	$F = 0$
1	1	0	1	0	0	0	$F = 0$
1	1	1	0	0	0	0	$F = 1$
1	1	1	1	0	0	0	1

(a) Truth table



(b) Logic diagram



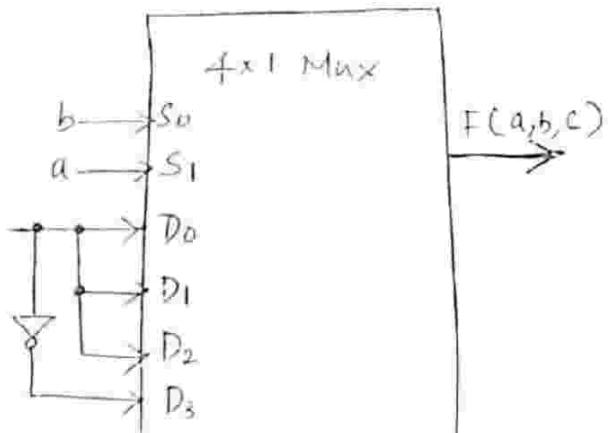
(c) Logic diagram

Q6 Implement the following function with a MUX.

$$F(a, b, c) = \sum m(1, 3, 5, 6)$$

S_1	S_0		
a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

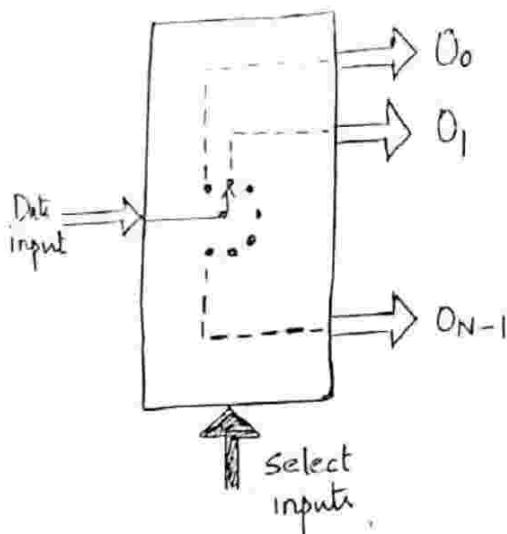
(a) Truth table



(b) Multiplexer Implementation

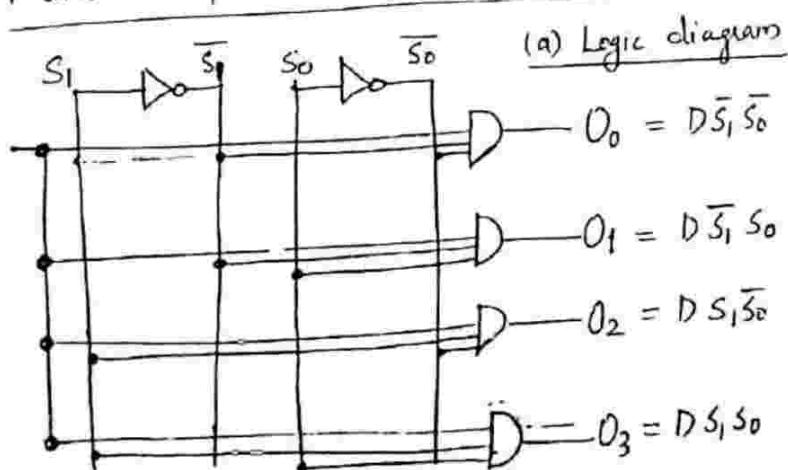
De Multiplexers (Data Distributors)

- A demultiplexer performs the reverse operation of multiplexer.
 - It takes a single input & distributes it over several outputs.
 - Demultiplexers are 'distributor', since it transmits the same data to different destinations.
- Multiplexer → N to 1 device
- Demultiplexer → 1 to N device
- The 'select input' code determines the output line to which the input data will be transmitted.



Functional diagram of a general demultiplexer

1 line to 4 line Demultiplexer

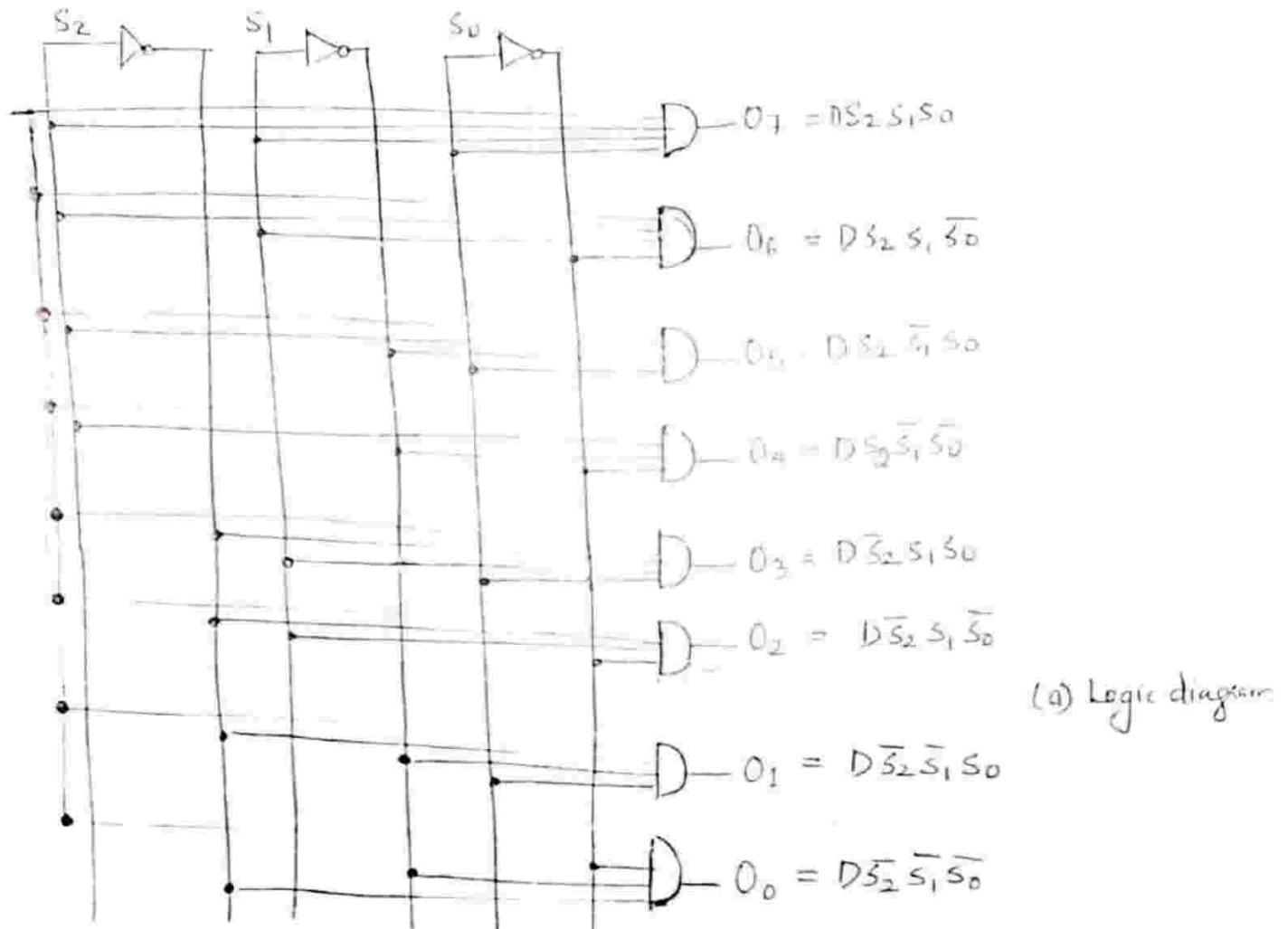


Select Code	Outputs
S ₁ S ₀	O ₃ O ₂ O ₁ O ₀
0 0	0 0 0 D
0 1	0 0 D 0
1 0	0 D 0 0
1 1	D 0 0 0

(b) Truth table

The input data line goes to all of the AND gates
 The 3 select lines S_2, S_1, S_0 enable only one gate at a time,
 and the date appearing on the input line will pass through
 the selected gate to the associated output line.

1 line to 8 line Demultiplexer



(b) Truth Table

Select Code			Outputs							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	D
0	0	1	0	0	0	0	0	0	D	0
0	1	0	0	0	0	0	0	D	0	0
0	1	1	0	0	0	0	D	0	0	0
1	0	0	0	0	0	D	0	0	0	0
1	0	1	0	0	D	0	0	0	0	0
1	1	0	0	D	0	0	0	0	0	0
1	1	1	D	0	0	0	0	0	0	0

Demultiplexer that distributes one g/p line to 8 o/p lines.
 The single data i/p line D is connected to all 8 AND gates, but only one of these gates will be enabled by the select i/p lines.

Q For the Boolean Function $F = (A+B)(\bar{A}+C)(B+C)$ Show how it can be implemented using a 1:8 de-multiplexer and one or more gates.

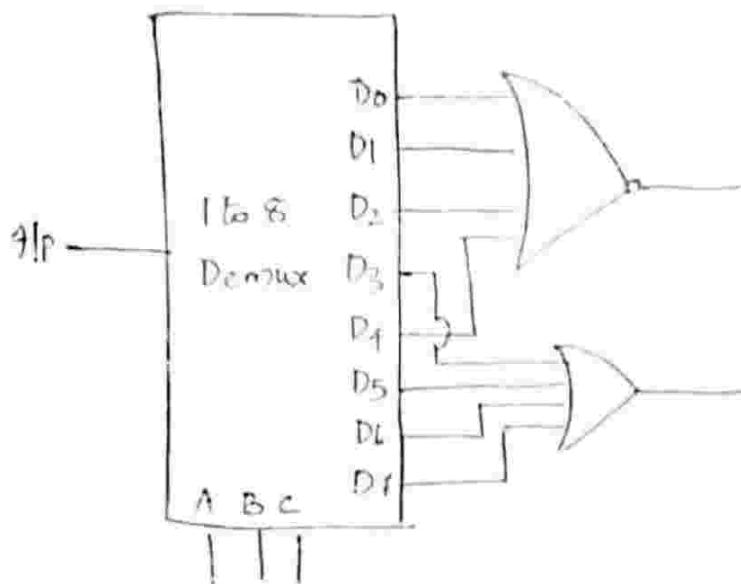
1st method

$$\begin{aligned}
 F &= (A+B)(\bar{A}+C)(B+C) \\
 &= (\bar{A}+B+\bar{C}\bar{C}) (\bar{A}+\bar{C}+\bar{B}\bar{B}) (B+C+A\bar{A}) \\
 &= (\bar{A}+B/\bar{C}) (\bar{A}+\bar{B}+\bar{C}) (\bar{A}+B/\bar{C}) (\bar{A}+\bar{B}+\bar{C}) (\bar{A}+B/\bar{C}) \\
 &\quad (\bar{A}+\bar{B}+\bar{C}) \\
 &= (\bar{A}+B+\bar{C}) (\bar{A}+\bar{B}+\bar{C}) (\bar{A}+\bar{B}+\bar{C}) (\bar{A}+\bar{B}+\bar{C}) \\
 &\quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \\
 &= \overline{\text{IM}}(0, 1, 2, 4) \\
 &= \overline{\text{S}}_m(3, 5, 6, 7)
 \end{aligned}$$

2nd method

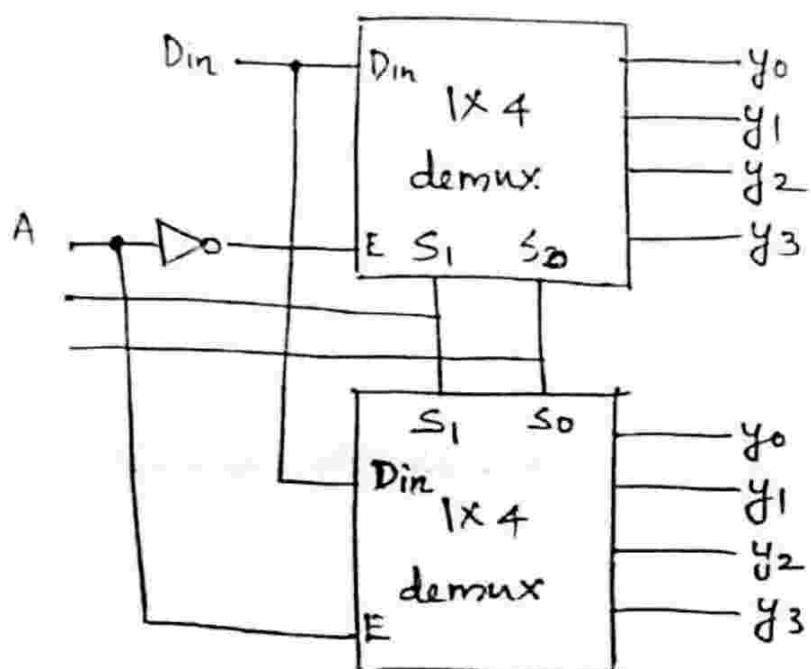
$$\begin{aligned}
 F &= (A+B)(\bar{A}+C)(B+C) \\
 &= (\bar{A}A + AB + \bar{A}C + BC)(B+C) \\
 &= (AB + AC + BC)(B+C) \\
 &= ABB + A\cancel{B}C + BBC + A\cancel{B}C + ACC + BCC \\
 &= AB + ABC + BC + AC + BC \\
 &= AB + ABC + AC + BC \\
 &= AB(C+\bar{C}) + ABC + AC(B+\bar{B}) + BC(A+\bar{A}) \\
 &= ABC + A\cancel{B}C + A\cancel{B}C + A\cancel{B}C + A\bar{B}C + A\bar{B}C + \bar{A}BC \\
 &= ABC + AB\bar{C} + A\bar{B}C + \bar{A}BC \\
 &\quad 1 \ 1 \ 1 \quad 1 \ 1 \ 0 \quad 1 \ 0 \ 1 \quad 0 \ 1 \ 1 \\
 &= \overline{\text{S}}_m(3, 5, 6, 7)
 \end{aligned}$$

$$= \lambda M (0, 1, 2, 4)$$



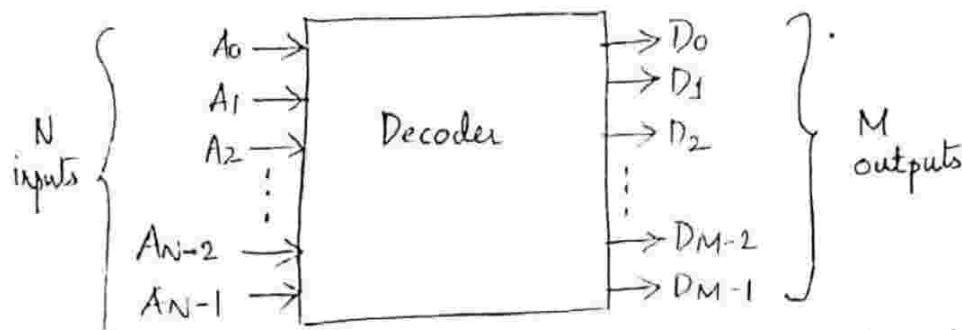
- Q) Design 1×8 demultiplexer using two 1×4 demultiplexers

The select inputs of 2 demux are connected together & the enable pins are connected directly to one demultiplexer & the other enable input is connected through NOT gate to the other demultiplexer



Decoder

A decoder is a logic circuit that converts an N -bit binary input code into M output lines such that only one output line is activated for each one of the possible combinations of inputs.



2^N input codes Only one output is high for each i/p code.

Fig: General Block diagram of a decoder

- The general decoder diagram with N inputs & M outputs.
- Since each of the N inputs can be a 0 or a 1, there are 2^N possible input combinations or codes.
- For each of these input combinations, only one of the M outputs will be active (HIGH), all the other outputs will remain inactive (LOW).
- Some decoders are designed to produce active LOW output, while all the other outputs remain HIGH.

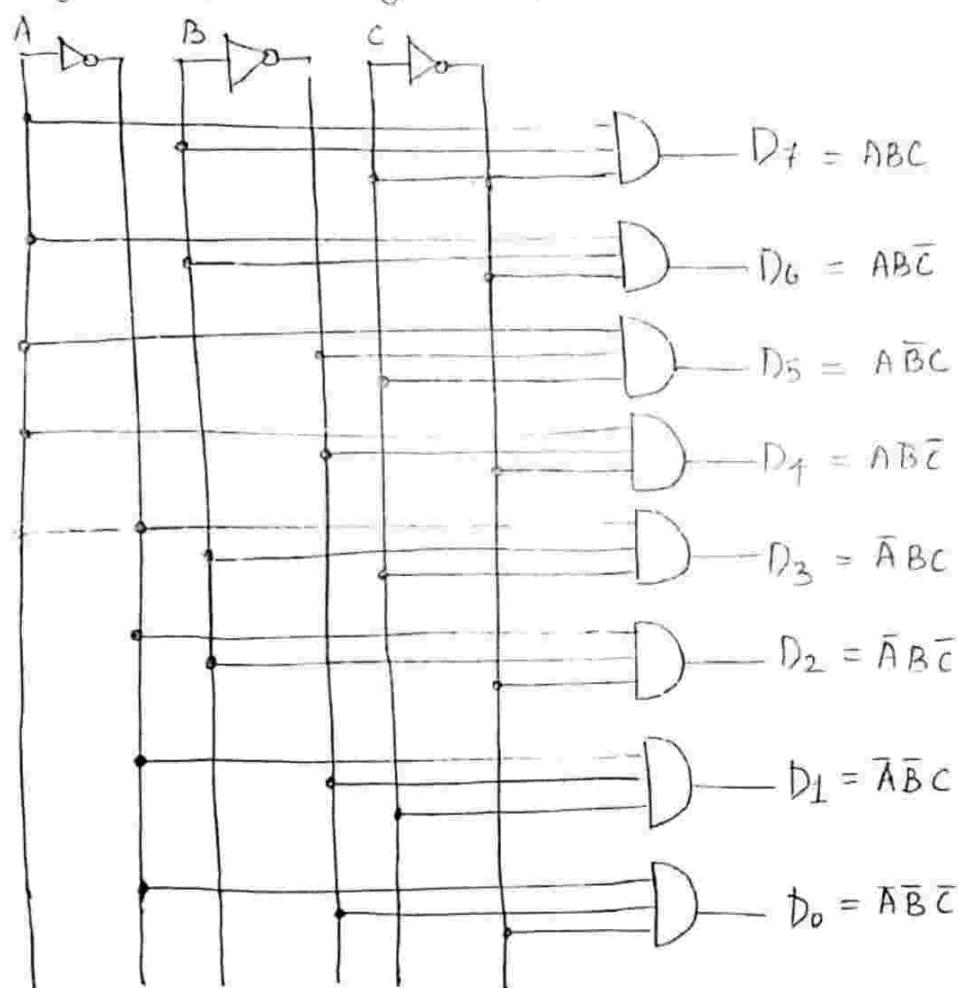
3 line to 8 line Decoder

Decoder with 3 i/p's & 8 o/p's.

It uses all AND gates & the o/p's are active - HIGH.
For active-Low outputs, NAND gates are used.

→ It is also called a binary to octal decoder because it takes a 3 bit binary input code & activates one of the eight (octal) outputs corresponding to that code.

It is also referred to as a 1-of-8 decoder because only one of the eight outputs is activated at one time.



(a) logic diagram

Inputs			Outputs							
A	B	C	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

(b) Truth table

Enable Inputs

Some decoders have one or more ENABLE inputs that are used to control the operation of the decoder.

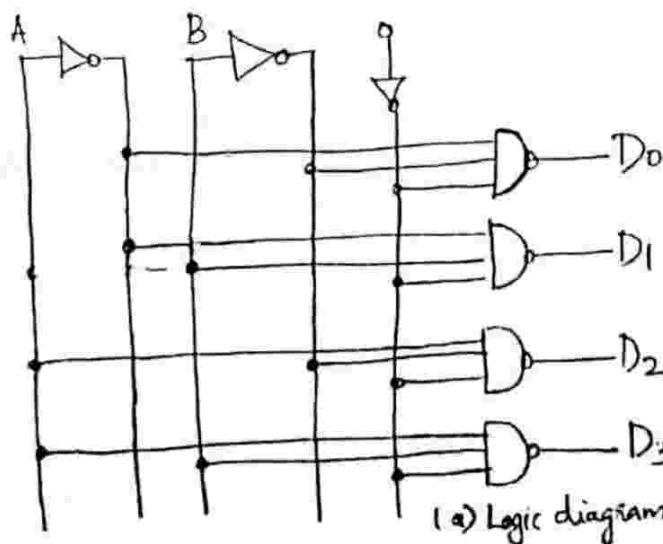
For BCD to Decimal Decoder

- The BCD to decimal decoder is also called a 4 line to 10 line or 4 to 10 decoder or 1 of 10 decoder.
- It has 4 i/p lines (for d_3, A_2, A_1, A_0) & 10 o/p lines (for $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9$).
- Only 1 o/p line is active at time.
- 6 of the 16 i/p combinations are invalid & for i/p combinations that are invalid for BCD none of the outputs will be activated.
- The i/p's & o/p's can be active high or active low.

Q2 2 line to 4 line Decoder With NAND Gates

Some decoders are constructed with NAND gates.

Since a NAND gate produces the AND operation with an inverted output, it becomes more economical to generate the decoder minterms in their complemented form.



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

A 2 to 4 line decoder with an enable input is constructed with NAND gates.

The circuit operates with complemented o/p's & complemented enable i/p.

The decoder is enabled when $E=0$.

The i/p's are active high & outputs active low.

Only 1 o/p can be equal to 0 at any given time, all other o/p's are equal to 1. (indicated by the truth table)

The circuit is disabled when $E=1$, regardless of the values of other 2 i/p's (A & B).

Q3- Combinational Logic Implementation

→ A decoder provides 2^n minterms of n input variables

→ Since any Boolean function can be expressed in sum of minterms, one can use a decoder to generate the minterms & an external OR gate to form the logic sum.

→ In this way any combinational circuit with n inputs & m outputs can be implemented with an n to 2^n decoder & m OR gates.

Full Adder Implementation

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in} = A \oplus B \oplus C_{in} = \Sigma m(1, 2, 4, 7)$$

$$\begin{aligned}C_{out} &= \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} = AB + (A \oplus B)C_{in} \\&= \Sigma m(3, 5, 6, 7)\end{aligned}$$

There are 3 inputs & a total of 8 minterms, we need a 3 to 8 line decoder.

The decoder generates the 8 minterms for A, B, Cin.

The OR gate for output S forms the logical sum of minterms 1, 2, 4 & 7. The OR gate for Cout forms the logical sum of the minterms 4, 5, 6 & 7.

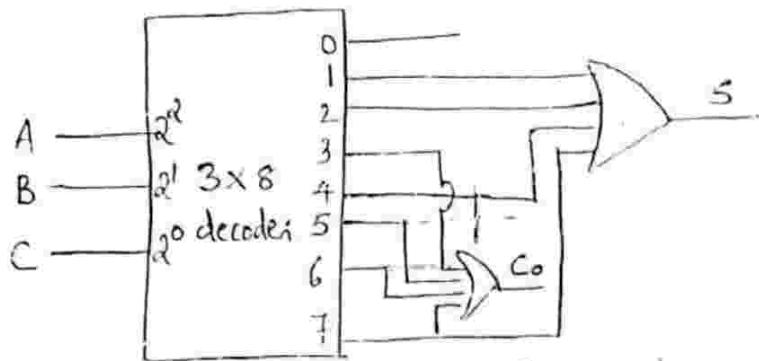


fig: Logic diagram of a full adder using a decoder

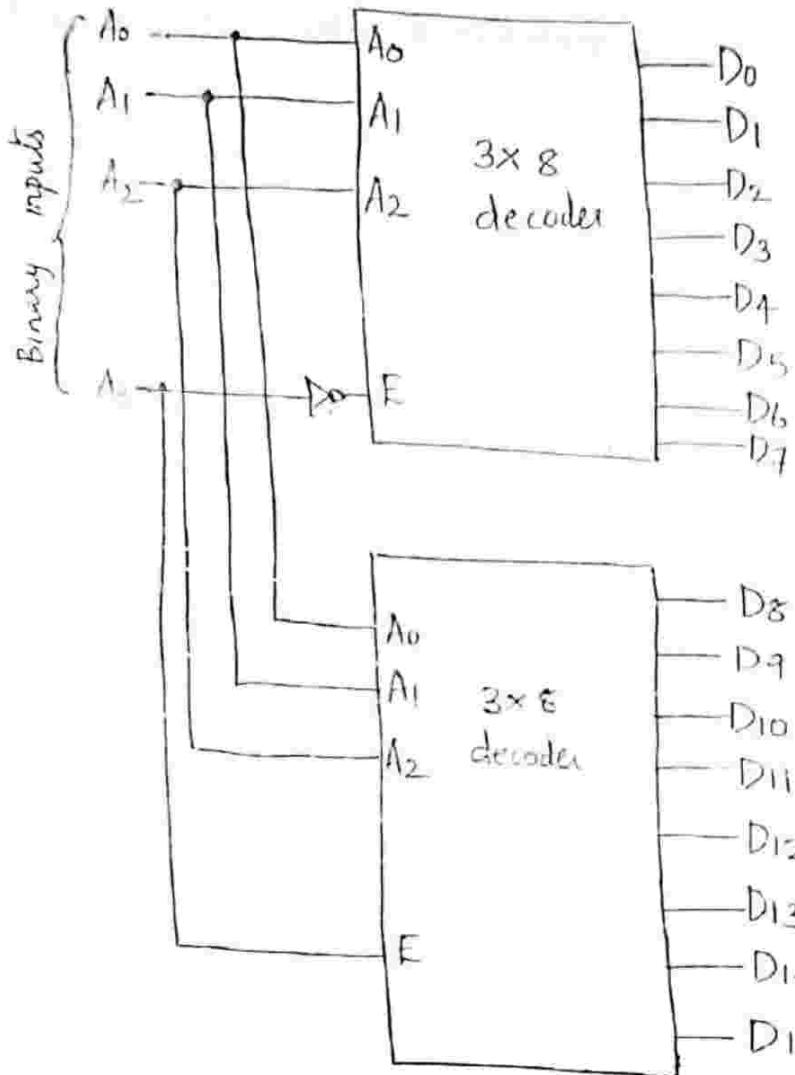
Q7 4 to 16 decoder from Two 3 to 8 decoders.

→ Two 3 to 8 decoders, to obtain a 4 to 16 decoder

→ The most significant input bit A_3 is connected through an inverter to \bar{E} on the upper decoder & directly to E on the lower decoder (for D8 through D15).

→ When A_3 is Low, the upper decoder is enabled & the lower decoder is disabled. The bottom decoder o/p's all 0's, and top 8 o/p's generate minterms

→ When A_3 is high, the lower decoder is enabled & the upper decoder is disabled. The bottom decoder o/p's generate minterms 1000 to 1111 while the o/p's of the top decoder are all 0's.



(a) Logic diagram

Binary inputs	Decimal output (active low)
0 0 0 0	D_0
0 0 0 1	D_1
0 0 1 0	D_2
0 0 1 1	D_3
0 1 0 0	D_4
0 1 0 1	D_5
0 1 1 0	D_6
0 1 1 1	D_7
1 0 0 0	D_8
1 0 0 1	D_9
1 0 1 0	D_{10}
1 0 1 1	D_{11}
1 1 0 0	D_{12}
1 1 0 1	D_{13}
1 1 1 0	D_{14}
1 1 1 1	D_{15}

(b) function table

Decoder Applications

Decoders are used whenever an output or a group of outputs is to be activated only on the occurrence of a specific combination of input levels.

These ifp levels are often provided by the ofp's of a counter or register.

BCD to Seven Segment Decoder

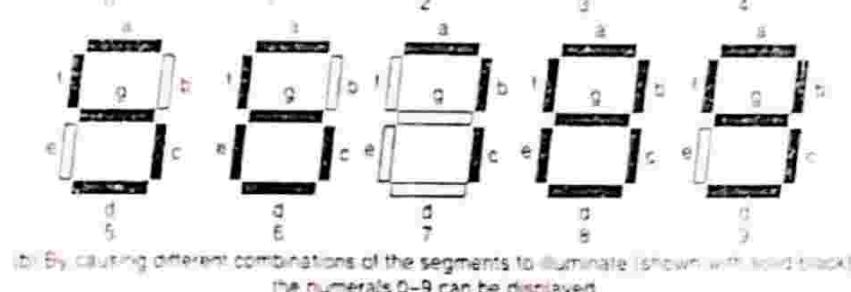
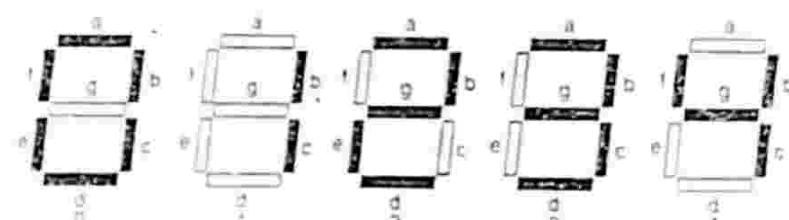
This type of decoder accepts the BCD code & provides outputs to energize seven segment display devices in order to produce a decimal read out.

Sometimes, the hex characters A through F may be produced. Each segment is made up of a material that emits light when current is passed through it.

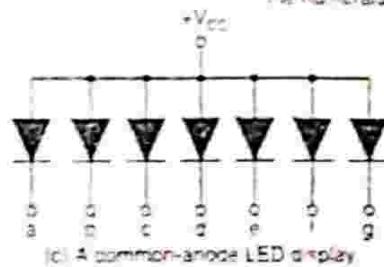
The most commonly used materials include LED's



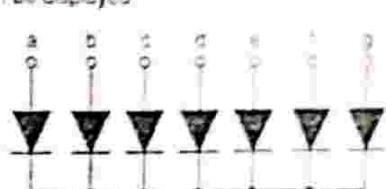
(a) Letters used to designate the segments



(b) By causing different combinations of the segments to illuminate (shown with bold block) the numerals 0-9 can be displayed.



(c) A common-anode LED display



(d) A common-cathode LED display

Figure 7.73 The seven segment display

Fig a shows a 7 segment display consisting of 7 light emitting segments. The segments are designated by letters a-g.

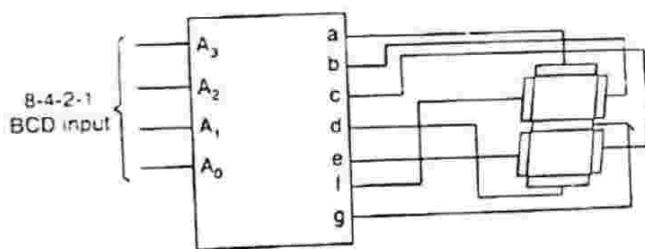
By illuminating various combinations of segments as shown in fig b, the nos 0-9 can be displayed.

Fig c & d show 2 types of LED display - the common anode & common cathode types

In the common-anode type, a low voltage applied to an LED cathode allows current to flow through the diode, which causes it to emit light.

In the common-cathode type, a high voltage applied to an LED anode causes the current to flow & produces the resulting light emission.

An 842-1 BCD to seven segment decoder is a logic circuit as shown in figure below



(a) Logic circuit

Since a 1 (HIGH) on any output line activates that line, we assume that the display is of the common-cathode type.

$$b = \bar{A}_3 \bar{A}_2 \bar{A}_1 \bar{A}_0 + \bar{A}_3 \bar{A}_2 \bar{A}_1 A_0 + \bar{A}_3 \bar{A}_2 A_1 \bar{A}_0 + \bar{A}_3 A_2 \bar{A}_1 \bar{A}_0 + \bar{A}_3 A_2 A_1 \bar{A}_0 \\ + A_3 \bar{A}_2 \bar{A}_1 \bar{A}_0 + A_3 \bar{A}_2 \bar{A}_1 A_0 = \Sigma m(0, 1, 2, 3, 4, 7, 8, 9)$$

$$d = \Sigma m(10, 11, 12, 13, 14, 15)$$

The function table for such a decoder is shown below

Decimal digit	$8 \cdot 4 \cdot 2 \cdot 1 \text{ BCD}$				Seven Segment Code						
	A_3	A_2	A_1	A_0	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	1	0	0	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	1	1	1

(b) Function table

$A_3 A_2$	$A_1 A_0$	00	01	11	10
00	1	1	1	1	
01	1		1		
11	x	x	x	x	
10	1	1	x	x	

(c) K map to derive simplified expression for driving segment (b)

$$b = \overline{A}_2 + \overline{A}_1 \overline{A}_0 + A_1 A_0$$

The K-map used to simplify the logic expression for driving segment b is shown above. Entries 10-15 are don't cares as usual.

Encoders

An encoder is a device whose inputs are decimal digits and/or alphabetic characters & whose outputs are the coded representation of those inputs.

An encoder is a device which converts familiar numbers or symbols into coded format.

In other words, an encoder may be said to be a combinational logic circuit that performs the reverse operation of the decoder. The opposite of the decoding process is called encoding. i.e. encoding is a process of converting familiar numbers or symbols into a coded format.

An encoder has a no: of input lines, only one of which is activated at a given time, and produces an N -bit output code depending on which input is activated.

The block diagram of an encoder with M inputs & N outputs. Here the inputs are active HIGH, which means they are normally LOW.

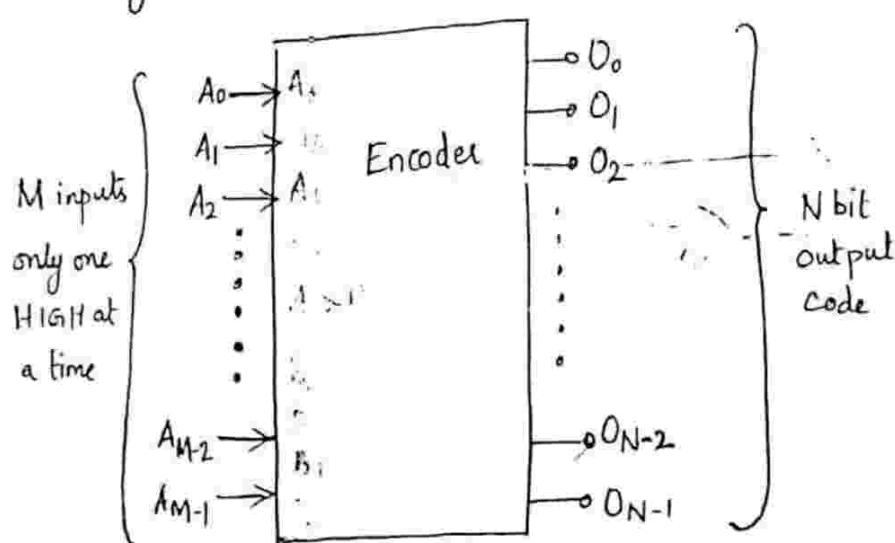


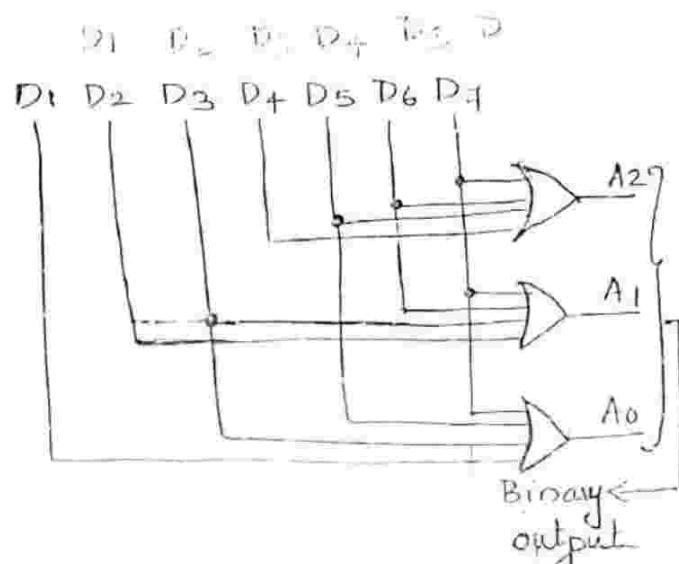
Fig: Block diagram of encoder

Octal to Binary Encoder

An octal to binary encoder (8 line to 3 line encoder) accepts 8 input lines & produces a 3 bit output code corresponding to the activated input.

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

(a) Truth table



(b) Logic diagram

fig: Octal to binary encoder

$$A_2 = D_4 + D_5 + D_6 + D_7$$

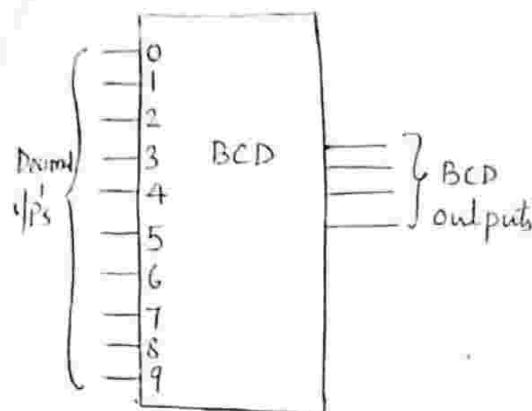
$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_0 = D_1 + D_3 + D_5 + D_7$$

D₀ is not present in any of the expressions. So D₀ is a don't care.

Decimal to BCD Encoder

This type of encoder has 10 inputs - one for each decimal digit & 4 outputs corresponding to the BCD code



(a) Logic symbol

$$A_3 = D_8 + D_9$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

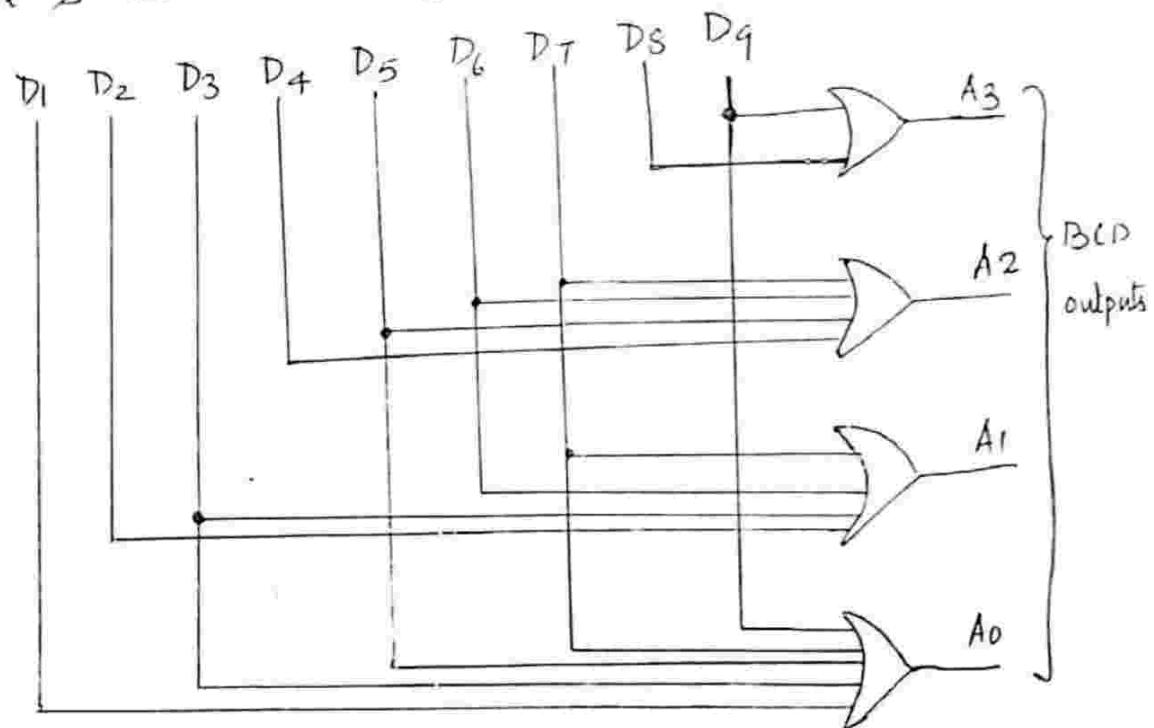
$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$$

The truthtable determine the relationships between each BCD bit & the decimal digits

Decimal inputs		Binary			
		A ₃	A ₂	A ₁	A ₀
D ₀	0	0	0	0	0
D ₁	1	0	0	0	1
D ₂	2	0	0	1	0
D ₃	3	0	0	1	1
D ₄	4	0	1	0	0
D ₅	5	0	1	0	1
D ₆	6	0	1	1	0
D ₇	7	0	1	1	1
D ₈	8	1	0	0	0
D ₉	9	1	0	0	1

(b) Truthtable



Priority Encoders

The encoders discussed so far will operate correctly, provided that one & only one decimal i/p is HIGH at any given time. In some practical systems, 2 or more decimal i/p's may become HIGH at the same time. For eg, a person operating a keyboard might press a second key before releasing the first. Let us say he presses key 3 before releasing key 4. In such case the output will be 7_{10} (0111) instead of being 4_{10} or 3_{10} .

(A) Priority encoder is a logic circuit that responds to just one i/p in accordance with some priority system, among all those that may be simultaneously HIGH.

(The most common priority system is based on the relative magnitudes of the i/p's; whichever decimal i/p is the largest, is the one that is encoded. Thus in the above example, a priority encoder would encode decimal 4 if both 3 & 4 are simultaneously HIGH.)

In some practical applications, priority encoders may have several i/p's that are routinely HIGH at the same time, & the principal function of the encoder in those cases is to select the i/p with the highest priority. This function is called arbitration.

4-Input Priority Encoder

The truth-table of a 4-input priority encoder is given below

The truthtable of a 4 input priority Encoder is given in fig(a)

In addition to the outputs A & B , the circuit has a 3rd output designated by V . This is a valid bit indicator that is set to 1 when one or more inputs are equal to 1. If all inputs are 0, there is no valid input & V is equal to 0. The other two outputs are not specified when V equals 0 & are specified as don't care conditions.

According to the truthtable, the higher the subscript number, the higher the priority of the input. Input D_3 has the highest priority. So regardless of the values of other inputs, when this input is 1, the output for AB is 11 (binary 3).

D_2 has the next priority level. The output is 10 if $D_2 = 1$ provided that $D_3 = 0$ regardless of the values of the other two lower priority inputs. The output for D_1 is generated only if higher priority inputs are 0. & so on down the priority levels. From the truthtable

$$\begin{aligned} A &= D_3 + \overline{D}_3 D_2 = D_3 + D_2 \\ B &= D_3 + \overline{D}_3 \overline{D}_2 D_1 = D_3 + \overline{D}_2 D_1 \\ V &= D_3 + D_2 + D_1 + D_0 \end{aligned}$$

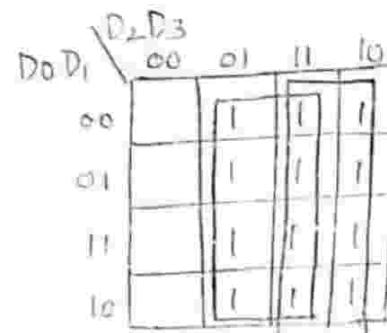
The condition for output V is an OR function of all ip variables. Although the table has only 5 five rows, when each x in a row is p+ replaced by 0 & then by 1, we obtain all 16 possible input combinations. The minterms for the two functions A & B are

$$A = \Sigma_m (1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15)$$

$$B = \Sigma_m (1, 3, 4, 5, 7, 9, 11, 12, 13, 15)$$

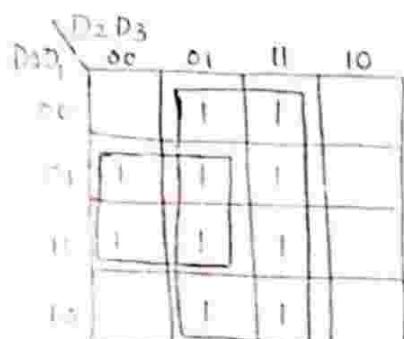
Inputs				Outputs		
D ₀	D ₁	D ₂	D ₃	A	B	V
0	0	0	0	x	x	0
D ₀ 1	0	0	0	0	0	1
D ₁ X	1	0	0	0	1	1
D ₂ X	X	1	0	1	0	1
D ₃ X	X	X	1	1	1	1

(a) Truth table



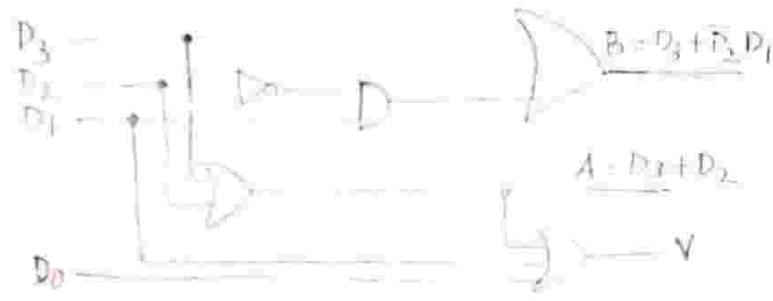
$$A = D_3 + D_2$$

(b) K map for A



$$B = D_3 + \overline{D}_2 D_1$$

(c) K-map for B



(d) Logic diagram

fig: 4-bit priority Encoder

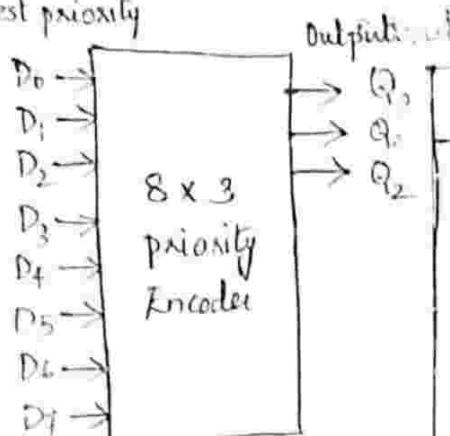
Decimal to BCD priority Encoder

This type of encoder performs the same basic function of encoding the decimal digits into 4 bit BCD outputs, as that performed by a normal decimal-to-BCD encoder. It however offers the additional facility of providing priority. That is, it produces a BCD op corresponding to the highest order decimal digit appearing on the ip's & ignores all others.

The purpose of the priority detection logic circuitry is to prevent a low-order digit input from disrupting the

The priority encoders output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

Lowest priority



Inputs	Outputs	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0 0 0 0 0 0 0 1	0 0 0	0	0	0	0	0	1
0 0 0 0 0 0 1 x	0 0 1	0	0	1	0	1	1
0 0 0 0 0 1 x x	0 1 0	0	1	0	0	1	0
0 0 0 0 1 x x x	0 1 1	0	1	1	0	0	1
0 0 0 1 x x x x	1 0 0	1	0	0	1	0	0
0 0 1 x x x x x	1 0 1	1	0	1	0	1	1
0 1 x x x x x x	1 1 0	1	1	0	1	1	0
1 x x x x x x x	1 1 1	1	1	1	1	1	1

Highest priority

when x equals don't care that is logic 0 or a logic 1.

$$Q_0 = \Sigma(1, 3, 5, 7)$$

$$Q_0 = \overline{D}_7 \overline{D}_6 \overline{D}_5 \overline{D}_4 \overline{D}_3 \overline{D}_2 D_1 + \overline{D}_7 \overline{D}_6 \overline{D}_5 \overline{D}_4 D_8 + \overline{D}_7 \overline{D}_6 D_5 + D_7$$

$$Q_1 = \Sigma(2, 3, 6, 7)$$

$$Q_1 = \overline{D}_7 \overline{D}_6 \overline{D}_5 \overline{D}_4 \overline{D}_3 D_2 + \overline{D}_7 \overline{D}_6 \overline{D}_5 \overline{D}_4 D_3 + \overline{D}_7 D_6 + D_7$$

$$Q_2 = \Sigma(4, 5, 6, 7)$$

$$Q_2 = \Sigma(\overline{D}_7 \overline{D}_6 \overline{D}_5 D_4 + \overline{D}_7 \overline{D}_6 D_5 + \overline{D}_7 D_6 + D_7)$$

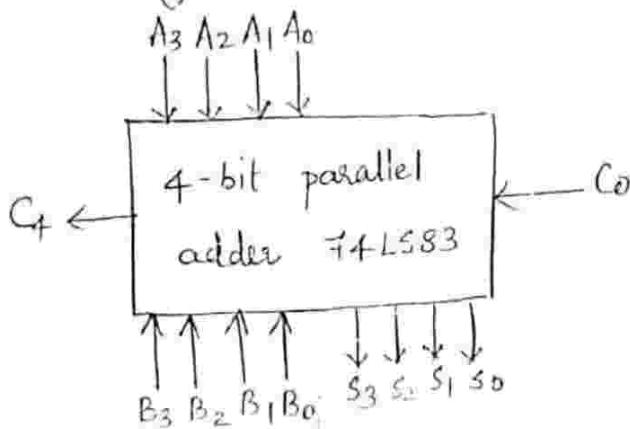
$$Q_0 = D_1 + D_3 + D_5 + D_7$$

$$Q_1 = D_2 + D_3 + D_6 + D_7$$

$$Q_2 = D_4 + D_5 + D_6 + D_7$$

4-bit Adder

The most common parallel adder is a 4-bit parallel adder IC that contains four interconnected full adders & a look-ahead carry circuit.



Logic symbol of 74LS83

The BCD addition process is

- 1) Add the 4-bit BCD code groups for each decimal digit position using ordinary binary addition
- 2) For those positions where the sum is 9 or less, the sum is in proper BCD form & no correction is needed
- 3) When the sum of 2 digits is greater than 9, a correction of 0110 should be added to that sum, to produce the proper BCD result. This will produce a carry to be added to the next decimal position.

A BCD adder circuit must be able to operate in accordance with the above steps. In other words, the circuit must be able to do the following:

- 1) Add two 4-bit BCD code groups, using straight binary addition

- Determine, if the sum of this addition is greater than 1001 (decimal 9); if it is add 0110 (decimal 6) to this sum generate a carry to the next decimal position.
- If the two BCD code groups $A_3 A_2 A_1 A_0$ & $B_3 B_2 B_1 B_0$ are applied to a 4 bit parallel adder, the adder will output $S_4 S_3 S_2 S_1 S_0$ where S_4 is actually c_4 , the carry out of the MSB bit. The logic output X that will go HIGH only when the sum is greater than 1001
- * will be High i) whenever $S_4 = 1$ (sum greater than 15)
 - ii) whenever $S_3 = 1$ and either S_2 or S_1 or both are 1 (sums 10 to 15)

This condition can be expressed as

$$X = S_4 + S_3(S_2 + S_1)$$

Whenever $X = 1$, it is necessary to add the correction factor 0110 to the sum bits & to generate a carry.

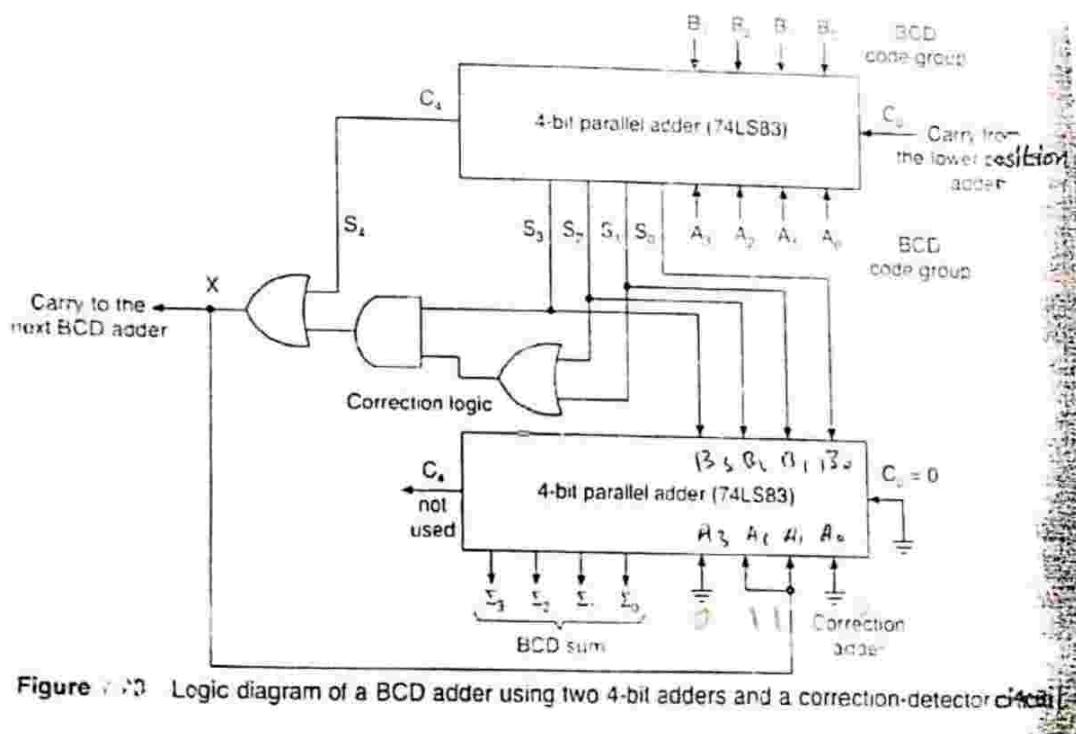


Figure 7.13 Logic diagram of a BCD adder using two 4-bit adders and a correction-detector circuit.

BCD Code groups are A_3, A_2, A_1, A_0 & B_3, B_2, B_1, B_0 to produce the sum $S_4, S_3 S_2, S_1, S_0$. The lower 4 bit adder will add the correction 0110 to the sum bits only when $X=1$, producing the final BCD sum output.

represented by $\Sigma_3 \Sigma_2 \Sigma_1 \Sigma_0$. The X is carry out when sum is greater than 1001 when $X=0$ there is no carry & no addition of 0110. In such cases $\Sigma_3 \Sigma_2 \Sigma_1 \Sigma_0 = S_3 S_2 S_1 S_0$.

Comparators

A comparator is a logic circuit used to compare the magnitudes of two binary numbers.

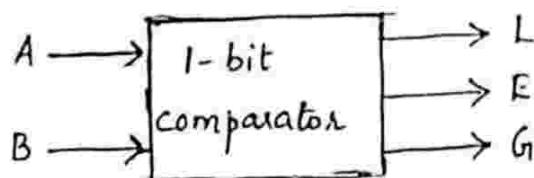
The X-NOR gate (coincidence gate) is a basic comparator, because its output is a 1 only if its two input bits are equal ie the output is 1 if & only if the input bits coincide.

Two binary numbers are equal, if & only if all their corresponding bits coincide. For eg 2 4-bit binary numbers, $A_3 A_2 A_1 A_0$ & $B_3 B_2 B_1 B_0$ are equal, if & only if $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ & $A_0 = B_0$. Thus equality holds when A_3 coincides with B_3 , A_2 coincides with B_2 , A_1 coincides with B_1 & A_0 coincides with B_0 . The implementation of this logic,

$$\text{EQUALITY} = (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) (A_0 \odot B_0)$$

is straight-forward.

The block diagram of a 1-bit comparator



1-Bit Magnitude Comparator

The logic for a 1-bit magnitude comparator: Let the 1-bit numbers be $A = A_0$ & $B = B_0$. If $A_0 = 1$ & $B_0 = 0$, then $A > B$.

$$\therefore A > B : G = A_0 \overline{B_0}$$

If $A_0 = 0$ & $B_0 = 1$, then $A < B$

$$\therefore A < B : L = \overline{A_0} B_0$$

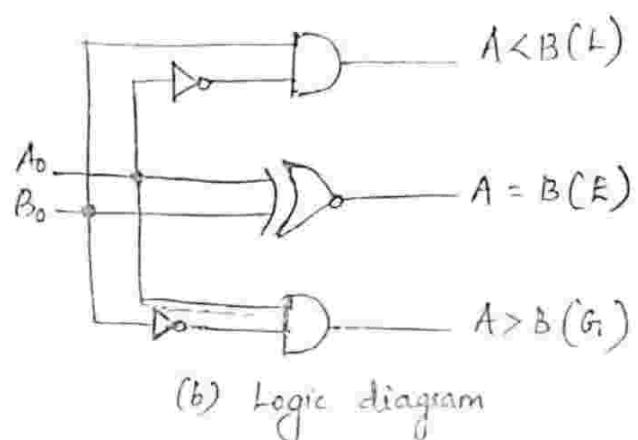
If $A_0 \& B_0$ coincide, i.e. $A_0 = B_0 = 0$ or if $A_0 = B_0 = 1$
then $A = B$

$$\therefore A = B : E = A_0 \oplus B_0$$

The truth table & the logic diagram for the 1-bit comparator
are shown below. The logic expressions for G , L & E can
also be obtained from the truth table

A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

(a) Truth table



(b) Logic diagram

2 Bit Magnitude Comparator

The logic for a 2-bit magnitude comparison: Let two
2 bit numbers be $A = A_1 A_0$ & $B = B_1 B_0$

1) If $A_1 = 1$ & $B_1 = 0$, then $A > B$ or

2) If A_1 & B_1 coincide & $A_0 = 1$ & $B_0 = 0$, then $A > B$. So
the logic expression for $A > B$ is

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

1) If $A_1 = 0$ & $B_1 = 1$ then $A < B$ or

2) If A_1 & B_1 coincide and $A_0 = 0$ & $B_0 = 1$, then
 $A < B$. So the expression for $A < B$ is

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

If $A_1 \& B_1$ coincide & if $A_0 \& B_0$ coincide then $A = B$.
So the expression for $A = B$ is

$$A = B : E = (A_1 \oplus B_1) (A_0 \oplus B_0)$$

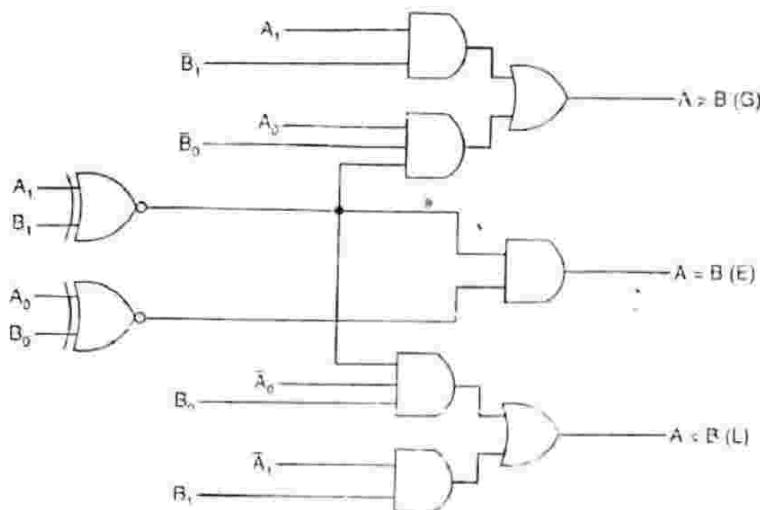
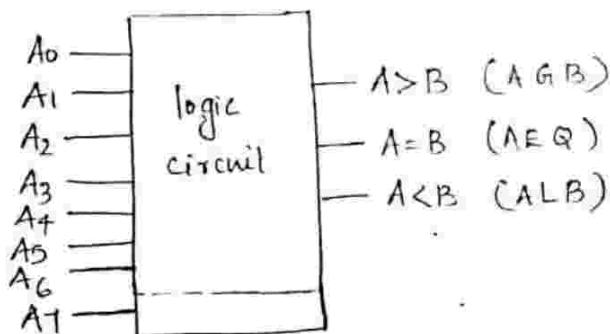


Figure 7.57 Logic diagram of a 2-bit magnitude comparator.

3 bit Magnitude Comparator



Let the 3 bit numbers be $A = A_2 A_1 A_0$ & $B = B_2 B_1 B_0$

Step 1

1) If $A_2 = 1$ & $B_2 = 0$ then $A > B$ or

2) If $A_2 = B_2$ & $A_1 = 1$ & $B_1 = 0$ then $A > B$ or

3) If $A_2 = B_2$ & $A_1 = B_1$ & $A_0 = 1$ & $B_0 = 0$, then $A > B$

$$A > B : G = A_2 \bar{B}_2 + (A_2 \oplus B_2) A_1 \bar{B}_1 + (A_2 \oplus B_2)(A_1 \oplus B_1) A_0 \bar{B}_0$$

Step 2

1) If $A_2 = 0$ & $B_2 = 1$ then $A < B$ or

2) If $A_2 = B_2$ & $A_1 = 0$ & $B_1 = 1$ then $A < B$ or

i) If $A_2 = B_2$ & $A_1 = B_1$ & $A_0 = 0$ & $B_0 = 1$, then $A < B$

$$A < B : L = \bar{A}_2 B_2 + (A_2 \oplus B_2) \bar{A}_1 B_1 + (A_2 \oplus B_2) (A_1 \oplus B_1) A_0 B_0$$

ii) A_2 & B_2 coincide & if A_1 & B_1 coincide & if A_0 & B_0 coincide then $A = B$

So the expression for $A = B$ is

$$A = B : E = (A_2 \oplus B_2) (A_1 \oplus B_1) (A_0 \oplus B_0)$$

