3.21 BCD TO SEVEN SEGMENT DECODER

Visual display is one of the important part of electronic circuit. Often it is necessary to display the data in text form or the digits are displayed. Various types of display devices are commercially available. Light emitting diode or LED is one of the most widely used display device and it is economic, low power consuming easily compatible in electronic circuit. They are available in various sizes, shapes and colours. In seven segment display consisting of 7 LED's *a, b, c, d, e, f* and *g* of certain shape and placed at certain orientation as in figure 3.63. For its shape and as each of the LEDs can be controlled individually, this display is called seven segment display. It is used to display BCD numbers *i.e.* from 0 to 9 by LED's.

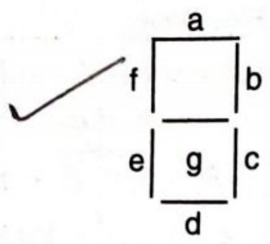


Fig. 3.67 (a). Orientation of seven LED's in a seven segment display.

Decimal digits 0 to 9 can be displayed by glowing some particular LEDs segments. For example digit '0' may be represented by glowing the segments *a, b, c, d, e* and *f* as in figure 3.65.

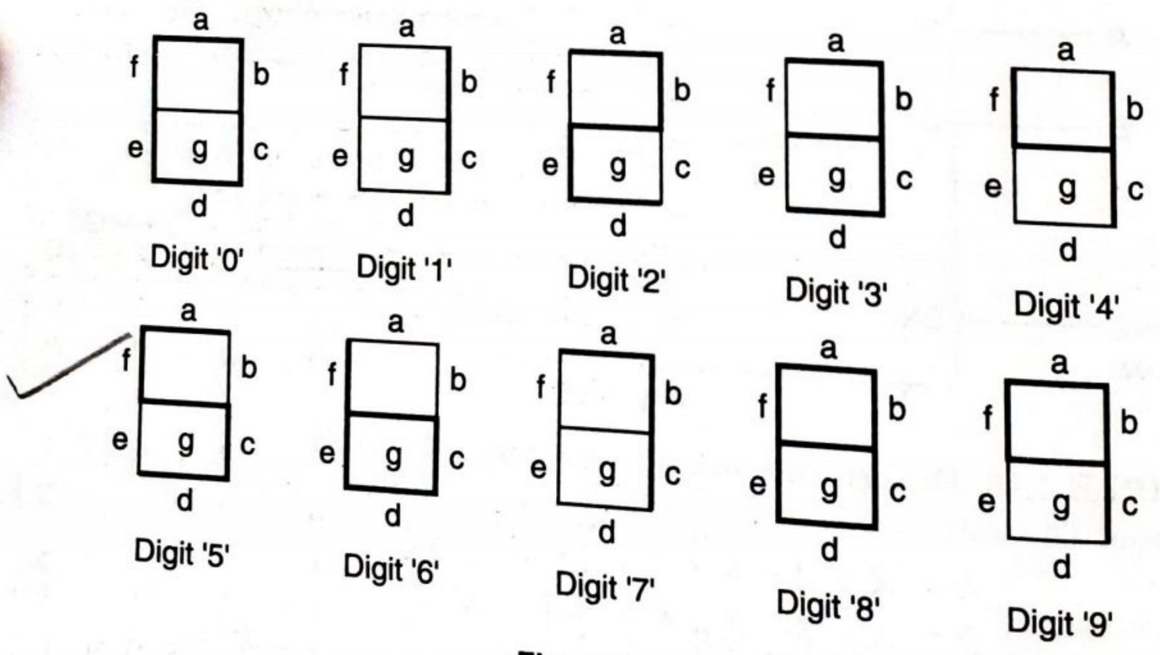
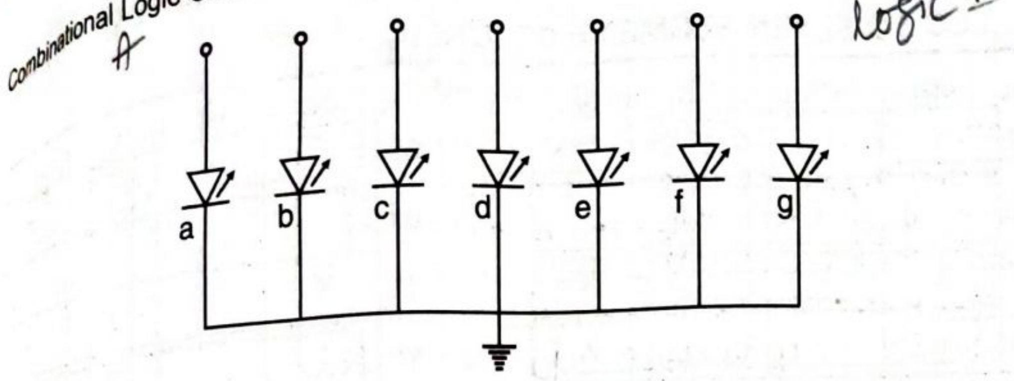


Fig. 3.67 (b).

Two types of seven segment display modular are available – Common cathode type and common Anode type. From the equivalent circuit, it is clear that to glow a particular LED of common cathode type, logic '1' is to be applied at the anode of that LED as all the cathodes are grounded. Alternatively logic '0' is to be applied to glow certain LED of common anode type, as all the anodes are connected to high voltage V_{CC}.



K Fig. 3.68 (a). Common Cathode LED

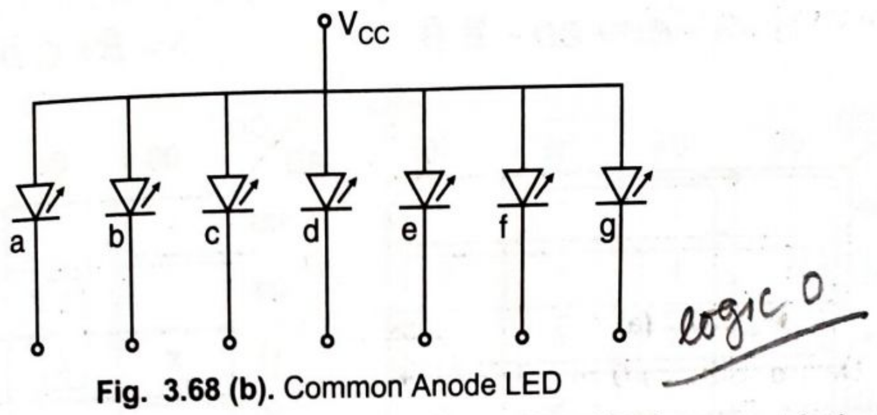


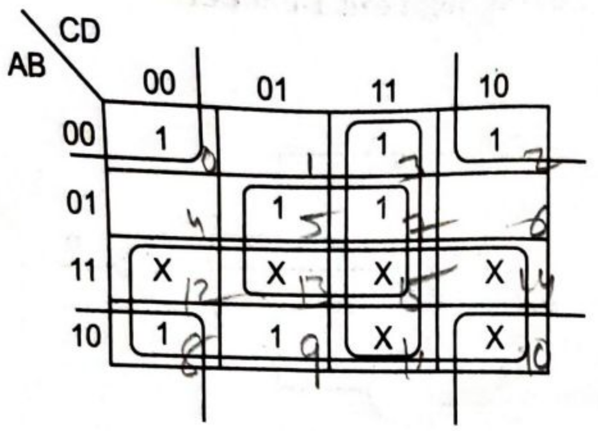
Fig. 3.68 (b). Common Anode LED

Every decimal digit of 0 to 9 is represented by the BCD data, consisting of four input variables A, B, C and D. The truth table can be made for each of the LED segment. Truth table for common cathode is given as follows. The Boolean expression for output variables a to g are obtained with the help of the Karnaugh Maps as shown below :

Decimal No.	*Input Variables*				*Output Variables Display as seven segment*						
	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	0	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	0	0	1	1

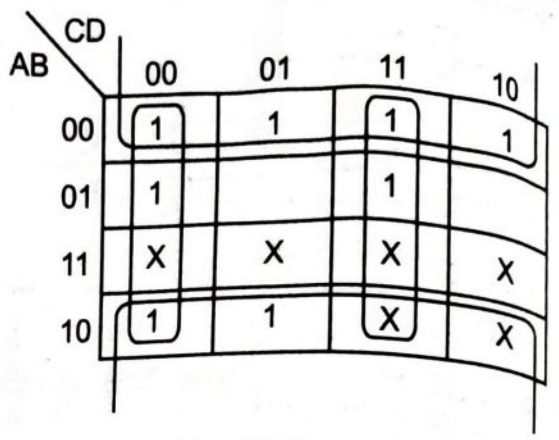
The Boolean expressions of the outputs of common anode type display are the complemented form of the respective output of common cathode type.

The K-map output for a to g is shown as follows :



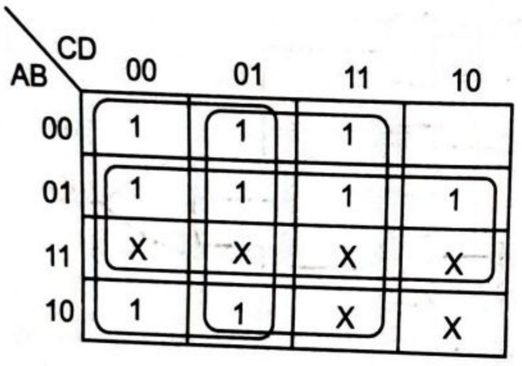
AB \ CD	00	01	11	10
00	1		1	1
01		1	1	
11	X	X	X	X
10	1	1	X	X

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



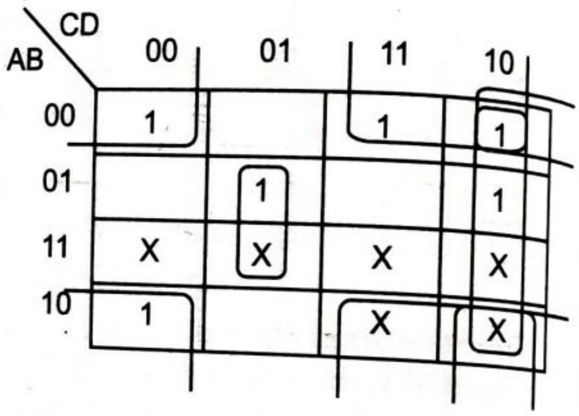
AB \ CD	00	01	11	10
00	1	1	1	1
01	1		1	
11	X	X	X	X
10	1	1	X	X

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



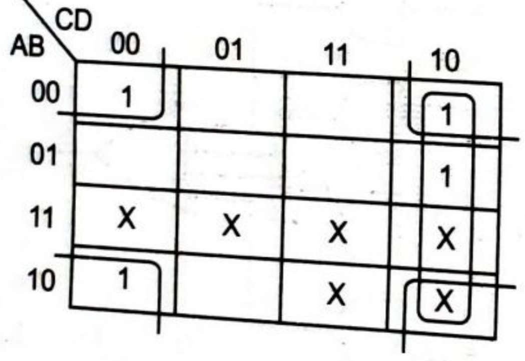
AB \ CD	00	01	11	10
00	1	1	1	
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

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



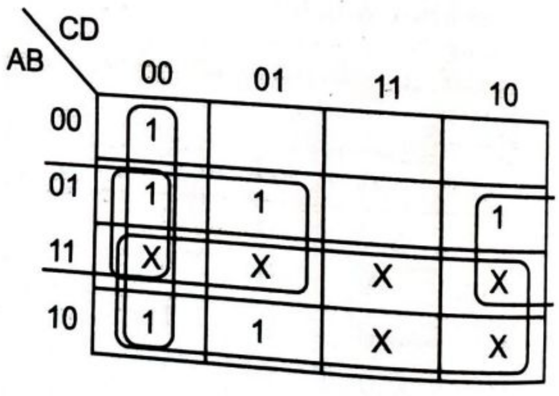
AB \ CD	00	01	11	10
00	1		1	1
01		1		1
11	X	X	X	X
10	1		X	X

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



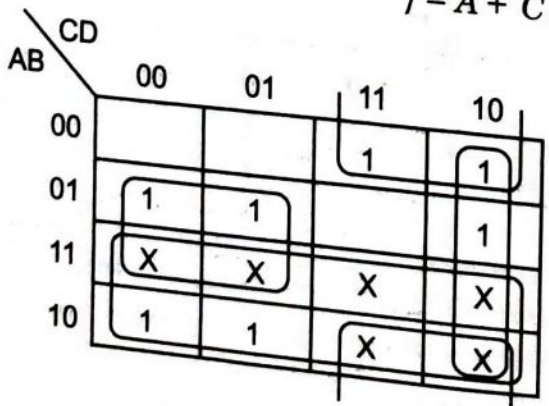
AB \ CD	00	01	11	10
00	1			1
01				1
11	X	X	X	X
10	1		X	X

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



AB \ CD	00	01	11	10
00	1			
01	1	1		1
11	X	X	X	X
10	1	1	X	X

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



AB \ CD	00	01	11	10
00			1	1
01	1	1		1
11	X	X	X	X
10	1	1	X	X

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

Designing using Gates for BCD to Seven Segment Decoder

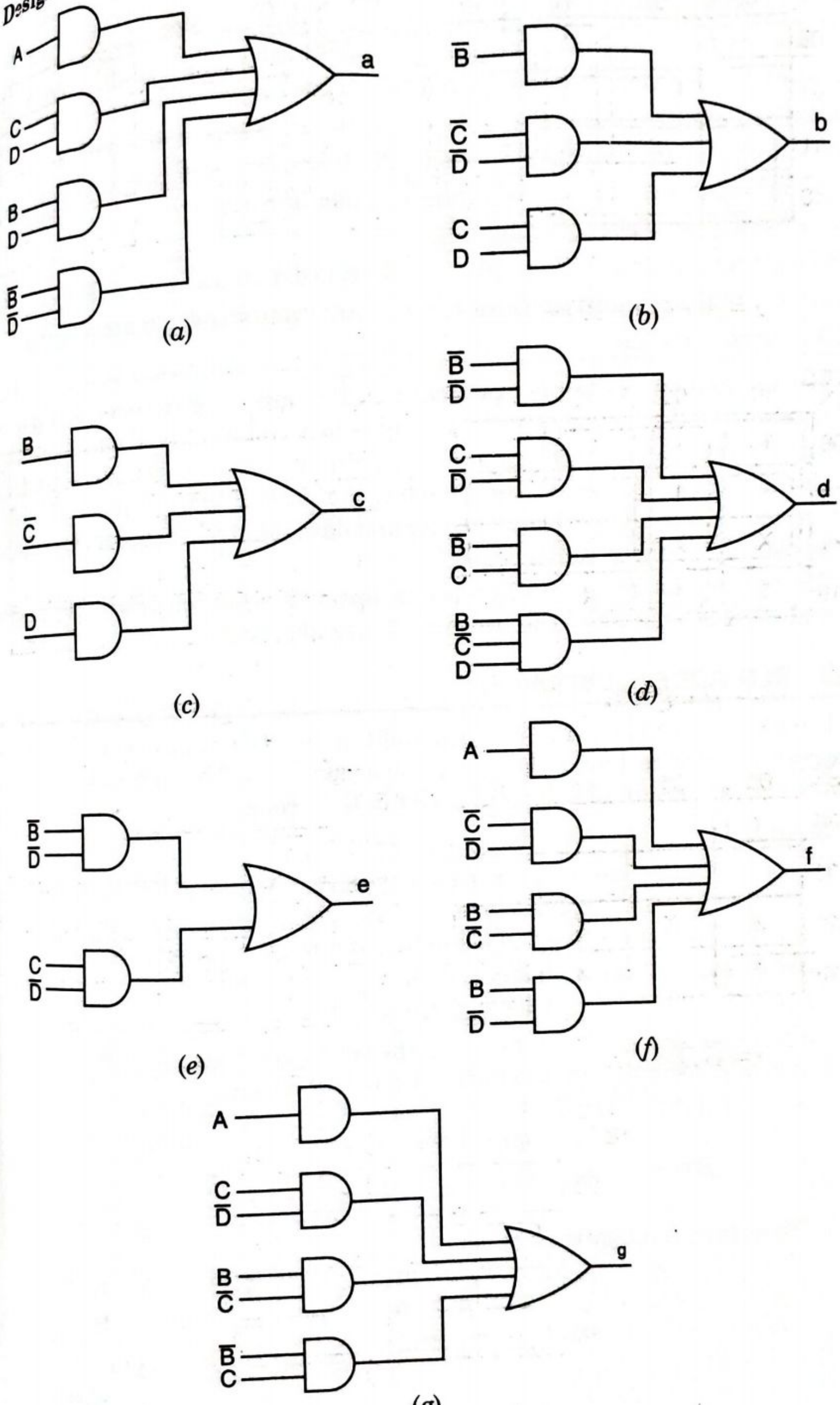


Fig. 3.69.

3.22 COMPARISON BETWEEN DEMULTIPLEXER AND DECODER

Demultiplexer	*Decoder*
1. It has one input and many outputs. It simply selects an output line and acting as glorified switch.	1. A decoder takes n inputs and uses these inputs to determine which of the 2^n output line is high.
2. A demultiplexer is designed to set one output equal to input (whether it be high, low or changing signal).	2. A decoder is designed to simply keep one line high.
3. It is denoted as 1 : 4, 1 : 8, 1 : 16, where 1 indicate input line and 4, 8, 16 indicate outputs.	3. It is denoted as 2 : 4, 3 : 8 where 2, 3 are inputs and 4, 8 are outputs.

Let us take one example of decoder and demux 3 : 8 decoder has 3 data inputs and 1 : 8 demux has 3 control input (select lines), plus '1' data input, for a total of '4' inputs. In order to make decoder and demultiplexer same, we need to add additional feature to the decoder. For this 'Enable' input of decoder correspond to data bit of a demultiplexer and data input of a decoder correspond to the control bit of a demultiplexer.

In other words, the only difference is that interpretation of what is control bit and what is a data bit. The two circuits are identical.

3.23 BCD ADDER/SUBTRACTOR

BCD adder circuit is used to perform addition of BCD numbers. BCD code is used to represent decimal numbers *e.g.*, in a calculator. Therefore addition and subtraction are required to be performed in BCD code.

BCD Adder

The 4-bit binary adder can be used to perform addition of BCD numbers. In this if the four bit sum output is not a valid and digit or if a carry C_3 is generated, then decimal 6 (0110) is to be added to the sum to get the correct result.

Let us explain this with an example

Let
$$A = 8 \text{ and } B = 8$$

Since both are BCD numbers and the result is greater than 9. Therefore we had to add decimal 6 in it in order to get BCD output.

$$\begin{array}{lcllll} A = 8 & \Rightarrow & A_3 & A_2 & A_1 & A_0 \\ & & 1 & 0 & 0 & 0 \\ B = 8 & \Rightarrow & B_3 & B_2 & B_1 & B_0 \\ & & 1 & 0 & 0 & 0 \end{array}$$

The output of adder is given as

$$\begin{array}{c} A \\ +B \\ \hline \end{array} \Rightarrow \begin{array}{ccccc} & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ \hline C_3 & S_3 & S_2 & S_1 & S_0 \\ \uparrow & & & & \\ \text{Carry} & & & & \end{array} \Rightarrow \begin{array}{ccccc} & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ \uparrow & S_3 & S_2 & S_1 & S_0 \\ \text{Carry bit} & & & & \end{array}$$

The circuit diagram is represented by figure 3.70.

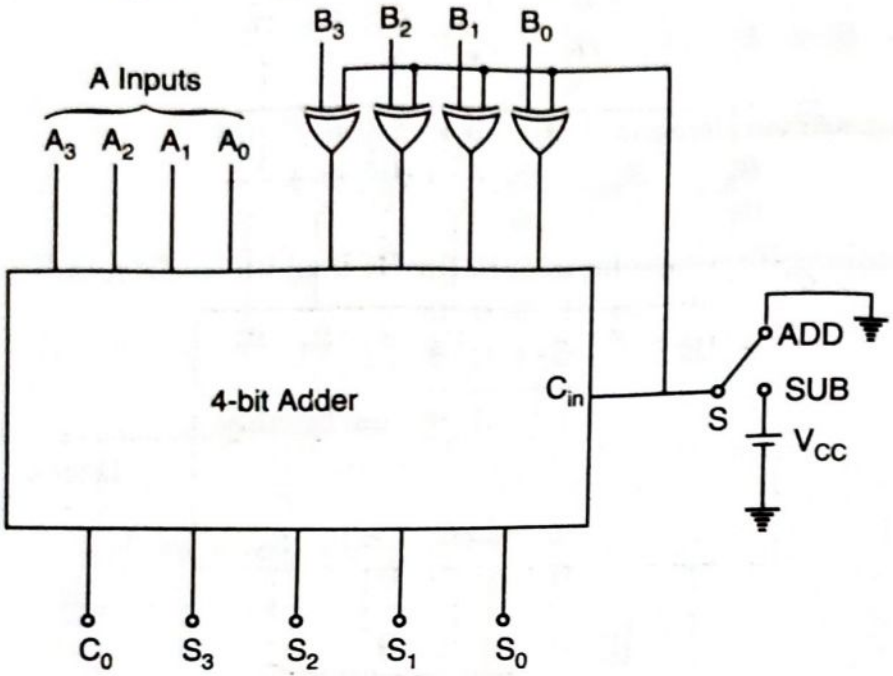


Fig. 3.72. 4 Bit Adder/subtractor circuit.

EXAMPLE 3.15. Design an 8-bit adder using two 4-bit adders.

Solution: An 8-bit adder can be formed by using two-4 bit adders. Cascading of two–4 bit are used in order to form 8-bit adder. A 4-bit adder has two 4-bit data inputs, a carry input, a carry output and 4-bit sum output. By cascading two such adders we can make an 8-bit adder.

In this case, the carry bit of first 4 bit adder is passed to the carry bit of 2nd 4-bit adder and outputs are S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7 and carry bit is C_7.

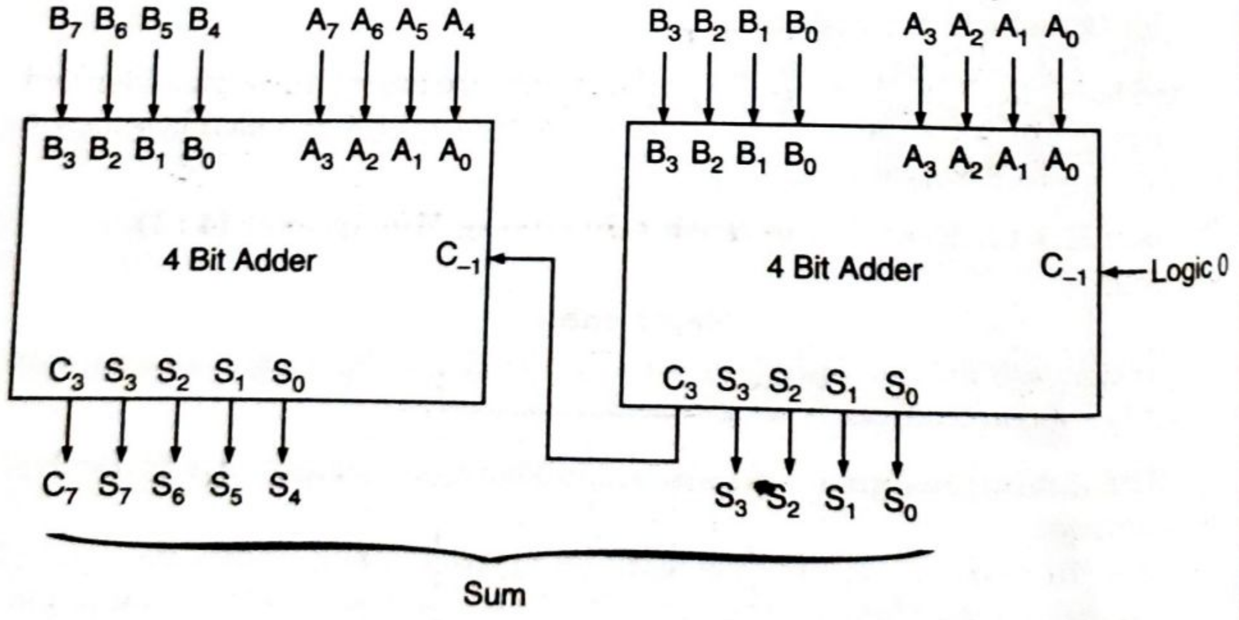


Fig. 3.73.

EXAMPLE 3.16. Design a nine's complement circuit.

Solution: Nine's complement of a BCD digit is given by nine minus that digit. This can be obtained by adding 1010 to the ones complement of the number. The circuit diagram is shown in figure 3.74

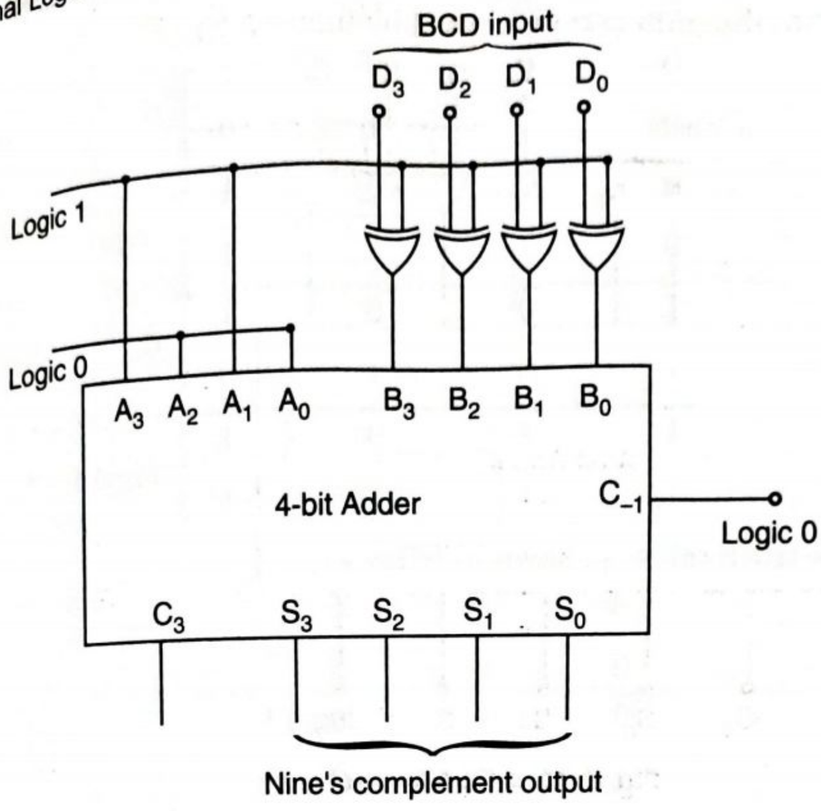


Fig. 3.74.

Let $D_3\ D_2\ D_1\ D_0$ and the 9's complement of 0001 (*i.e.* '1') is (9 – 1) = 8.
$\ \ \ \ \ \ \ 0\ \ 0\ \ 0\ \ 1$

When D_3, D_3, D_2, D_0 is 0 0 0 1 and 2nd input of
EX-OR gate is 1 1 1 1 and the output of EX-OR is
(0 EXOR 1) = "1 1 1 0" and the
$B_3\,B_2\,B_1\,B_0$ = "1 1 1 0".
The $A_3\,A_2\,A_1\,A_0$ = 1 0 1 0
and both the $B_3\,B_2\,B_1\,B_0$ and $A_3\,A_2\,A_1\,A_0$ bits are passed through adder and the output is given as "1000" and hence the output is "1000" which is binary 8 and the required answer.

EXAMPLE 3.17. Realize the truth table using Multiplexer (4 : 1).
Solution:

Truth table

A	*B*	*C*	*Y*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

The given truth table can be realized by using 4 : 1 Multiplexer. The minimization of the truth table is shown as follows :

A	B	C	Y	
0	0	0	1	$\bar{C}$
0	0	1	0	
0	1	0	1	logic 1
0	1	1	1	
1	0	0	1	$\bar{C}$
1	0	1	0	
1	1	0	1	logic 1
1	1	1	1	

The new truth table is shown in follows :

A	B	Y
0	0	$\bar{C}$
0	1	logic 1
1	0	$\bar{C}$
1	1	logic 1

The designing is as follows :

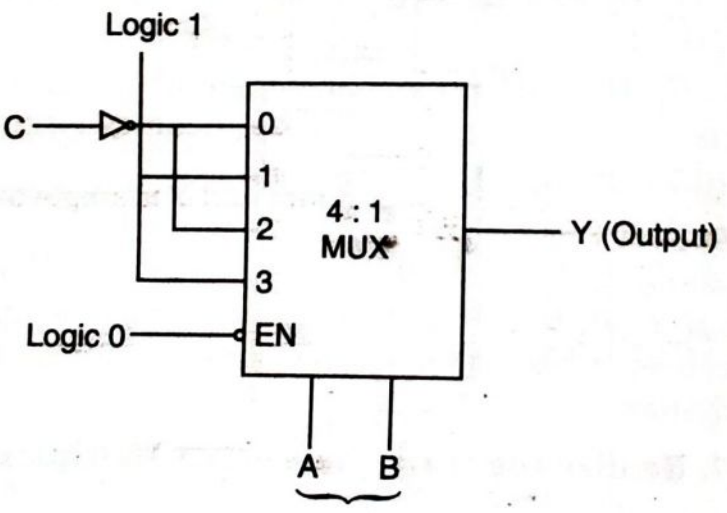


Fig. 3.75. Select lines

EXAMPLE 3.18. How to get Logic Gates using Multiplexers

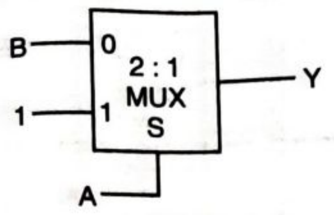
1. OR Gate using 2 : 1 MUX

Truth table of OR Gate :

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

OR Gate can be constructed by using 2 : 1 MUX by four ways :

(1)

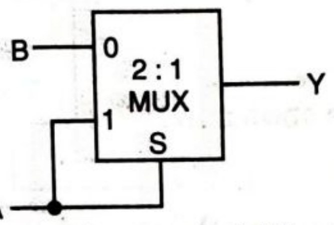


B & 1 are two inputs of 2 : 1 MUX and Y is taken is output. A is taken as select line of 2 : 1 MUX,

If $A = 0$ Then $Y = B$

else if $A = 1$ Then $Y = 1$

(2)

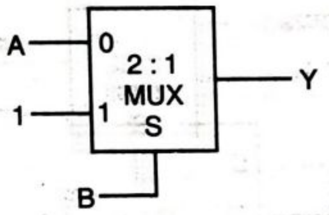


In this 2 : 1 MUX, B & A are two inputs and Y is single output. A is taken as single select line.

If $A = 0$ $Y = B$

else if $A = 1$ Then $Y = A$

(3)

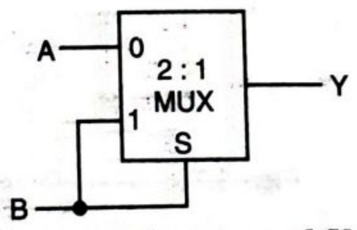


In this 2 : 1 MUX, A and 1 are two inputs and Y is single output. B is taken as single select line.

If $B = 0$ Then $Y = A$

else if $B = 1$ Then $Y = 1$

(4)



In this 2 : 1 MUX, A and B are two inputs and Y is single ouput. B is taken as single select line.

If $B = 0$ Then $Y = A$

else if $B = 1$ Then $Y = B$

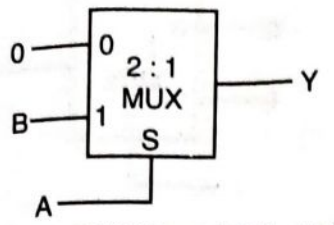
(2) AND Gate using 2 : 1 MUX

Truth table of AND gate

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

AND gate can be constructed by using 2 : 1 MUX by four ways :

(1)

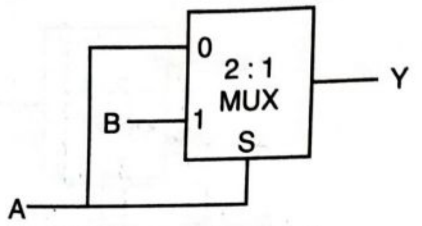


0 and B are two inputs of 2 : 1 MUX and Y is taken of output. A is taken as select line of 2 : 1 MUX.

If $A = 0$ Then $Y = 0$

else if $A = 1$ Then $Y = B$

(2)

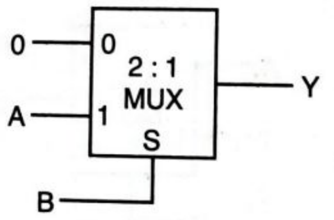


A & B are two inputs of 2 : 1 MUX and Y is taken as output. A is taken as select line of 2 : 1 MUX.

If $A = 0$ Then $Y = A$

else if $A = 1$ Then $Y = B$

(3)

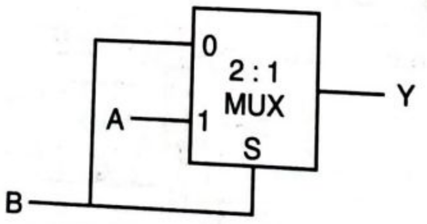


0 & A are two inputs of 2 : 1 MUX and Y is taken as output. B is taken as selects line of 2 : 1 MUX.

If $B = 0$ Then $Y = 0$

else if $B = 1$ Then $Y = A$.

(4)



B & A are two inputs of 2 : 1 MUX and Y is taken as output. B is taken as select line of 2 : 1 MUX.

If $B = 0$ Then $Y = B$

else if $B = 1$ Then $Y = A$

(3) NAND gate using 2 : 1 MUX

Truth table of NAND gate

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