Review of Combinational Circuit Design

First Step in the design of a combinational Switching circuit is usually to set up a Truth table which specifies the outputs as a function of input variables. For n i/p variables, truth table will have 2^n rows. If a given combination of values for the i/p variables can never occur at circuit inputs, the corresponding o/p values are don't cares.

Next Step is to derive Simplified algebraic expressions for the o/p functions using K-maps, QM method or similar procedure.

In some cases, if the number of variables is large & the number of terms is small, it may be desirable to go directly from the problem statement to algebraic equations, without writing truth table. The resulting equation can be simplified algebraically.

The number of levels in a gate circuit is equal to the maximum number of gates through which a signal must pass when going b/w the i/p & o/p terminals.

Minimum Two level AND-OR, NAND-NAND OR-NAND and NOR-OR ckt's can be realized using minimum sum of products.

lly minimum Two level OR-AND, NAND-NOR, AND-NOR and NAND-AND circuits can be realized using minimum POS. .

If the AND-OR circuits has a AND gate O/P connected to The Same Type of gate, Then extra inverTers must be added in The Conversion process.

Design of circuits with limited gate Fan-in

In Practical logic design Problems, the maximum number of inputs on each gate (or the fan in) is limited. It is neccessary to realize factoring the logic expression to obtain a multi level realization, if a two level realization of a circuit requires more gate inputs than allowed.

Realize the function, using only two input NAND gates & inverters. if we

$f_1 = b'c' + ab' + a'b$ $f_2 = b'c' + bc + a'b$ $f_3 = a'b'c + ab + bc'$

A \ BC	00 ($\bar{b}\bar{c}$)	01 ($\bar{b}c$)	11 (bc)	10 ($b\bar{c}$)
$\bar{A}$ 0	1	0	1	1
A 1	1	1	0	0

$f_1 = \Sigma_m(0,2,3,4,5)$

A \ BC	00	01	11	10
0	1	0	1	1
1	1	0	1	0

$f_2 = \Sigma_m(0,2,3,4,7)$

A \ BC	00	01	11	10
0	0	1	0	1
1	0	0	1	1

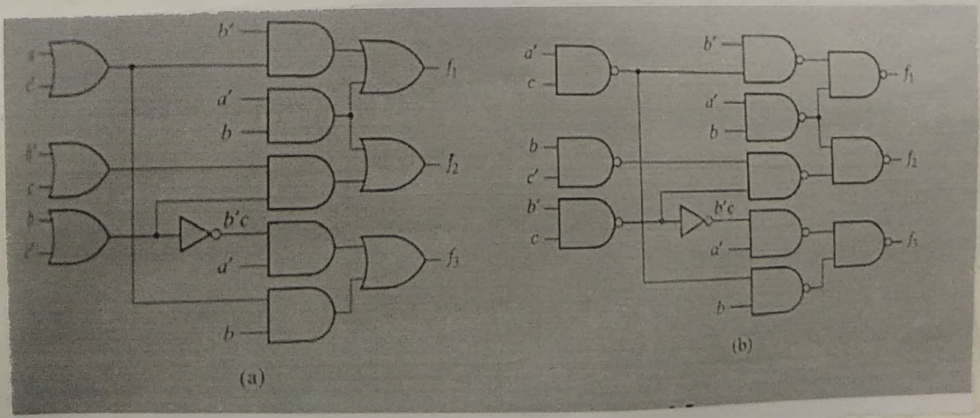
$f_3 = \Sigma_m(1,2,6,7)$

All function requires 3 input OR-gate, we will factor to reduce the number of gate i/p's

$f_1 = b'(a+c') + a'b$

$f_2 = b(a'+c) + b'c'$ or $f_2 = (b'+c)(b+c') + a'b$

$f_3 = a'b'c + b(a+c')$



Gate Delay & Timing Diagrams.

When The input to a logic gate is changed, The output will not change instantaneously. The switching elements within The gate take a finite time to react to change in input, So That the Change in the gate output is delayed with respect to the input change.

If the change in output is delayed by time ε, with respect to the i/p, Then gate has a propogation delay of ε.

propogation delay from 0 to 1 & 1 to 0 will be different. propogational delays for Itegrated circuit gate may be few nano Seconds (10^{-9} Seconds) which can be neglected in some case.

Input and output waveforms for an Inverter

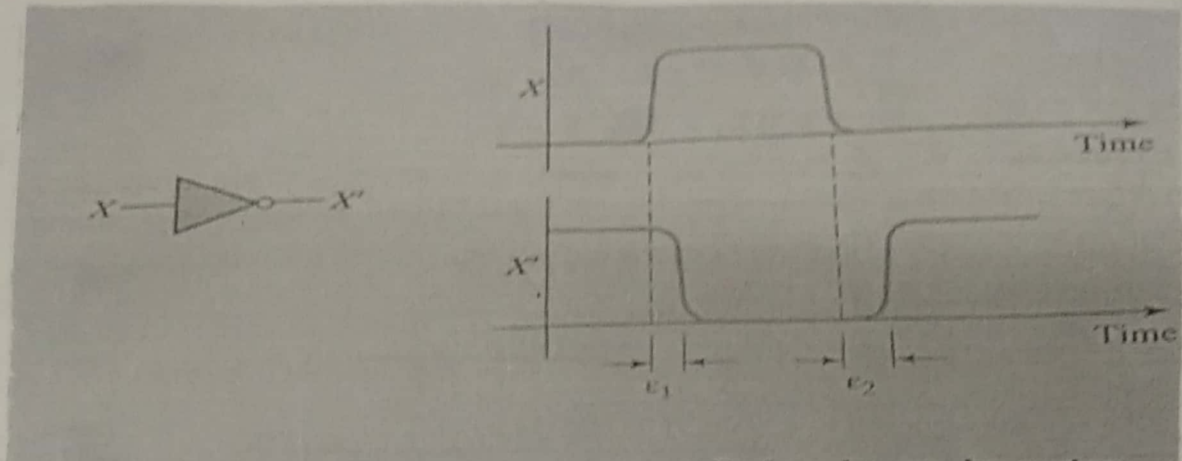
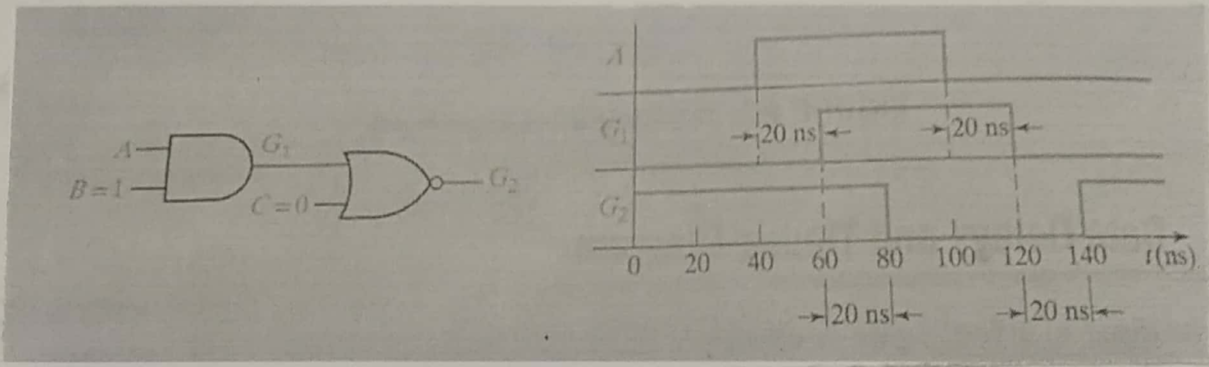


FIGURE 8-4: Propagation Delay in an Inverter

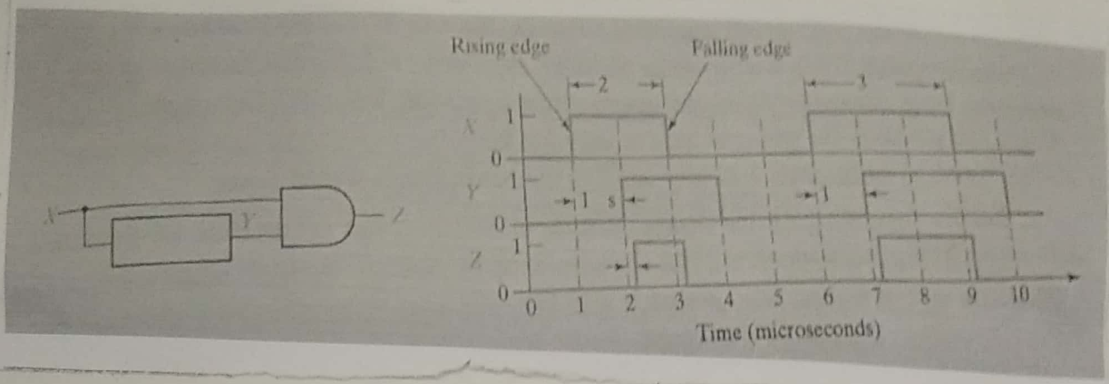
Timing diagram of a circuit with two gates



Assuming the propogation delay of 20 ns.

Input A is changed to 1 at t = 40ns and then changed back to 0 at t = 100 ns. The output of gate G_1 changes 20ns after A changes, & the output of gate G_2 changes 20ns after G_1 changes.

Timing diagram of Circuit with an added delay element.



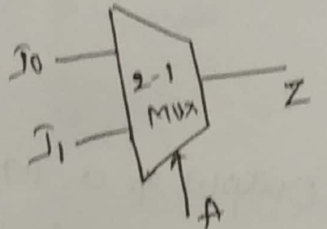
The input X Consists of 2 pulses, the first is 2 microsecond wide & Second is 3 microsecond wide. The delay element has an output Y which is the Same as The input except that it is delayed by 1 micro Second.

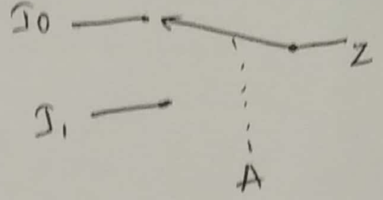
The output (Z) of the AND gate Should be 1 during the time interval in which both X & Y are 1. If we assume a Small Propogational delay in the AND gate (ε) then Z will be as Shown in the above figure.

MULTIPLEXERS

A multiplexers has a group of data inputs and a group of Control Inputs. The control inputs are used to select one of the data inputs and connect it to the o/p terminal.

2:1 MUX





when control input A is 0 → the switch is in upper position & Mux o/p is $Z = I_0$

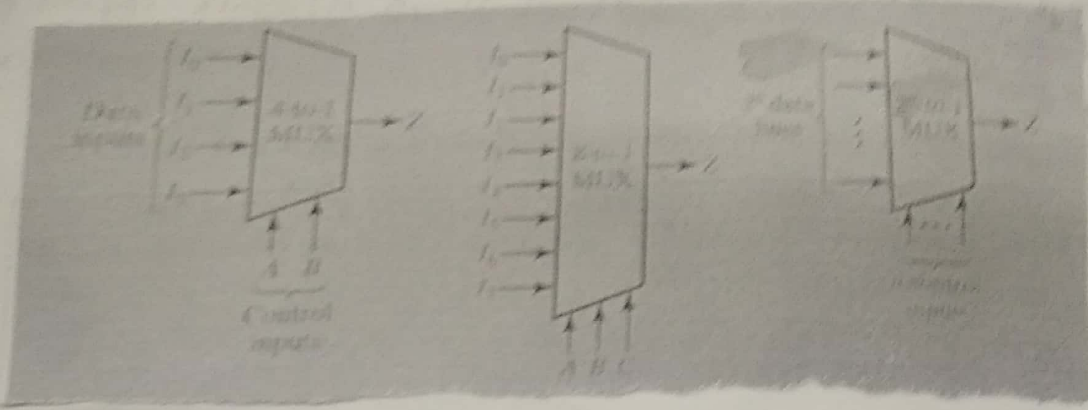
A is 1 → the switch is in lower position & Mux o/p is $Z = I_1$

$Z = A'I_0 + AI_1$ → logic equation of 2:1 MUX

4:1 MUX → $Z = A'B'I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$

8:1 MUX → $Z = \bar{A}\bar{B}I_0 + \bar{A}\bar{B}I_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + A\bar{B}\bar{C}I_4 + \bar{A}BCI_5 + AB\bar{C}I_6 + ABCI_7$

When ABC = 011, the o/p is I_3, & other o/p's are selected in similar manner.



The general equation for the output of a MUX with n control inputs & 2^n data input is

$$Z = \sum_{K=0}^{2^n-1} m_K I_K$$

where m_K is a minterm of the n control variables & I_K is the corresponding data i/p.

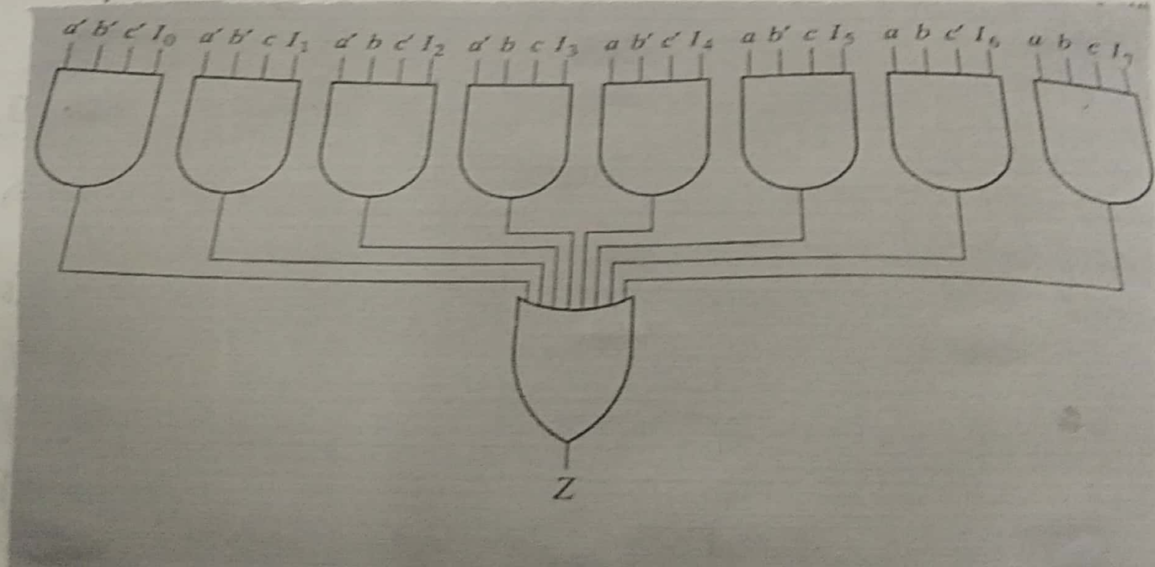
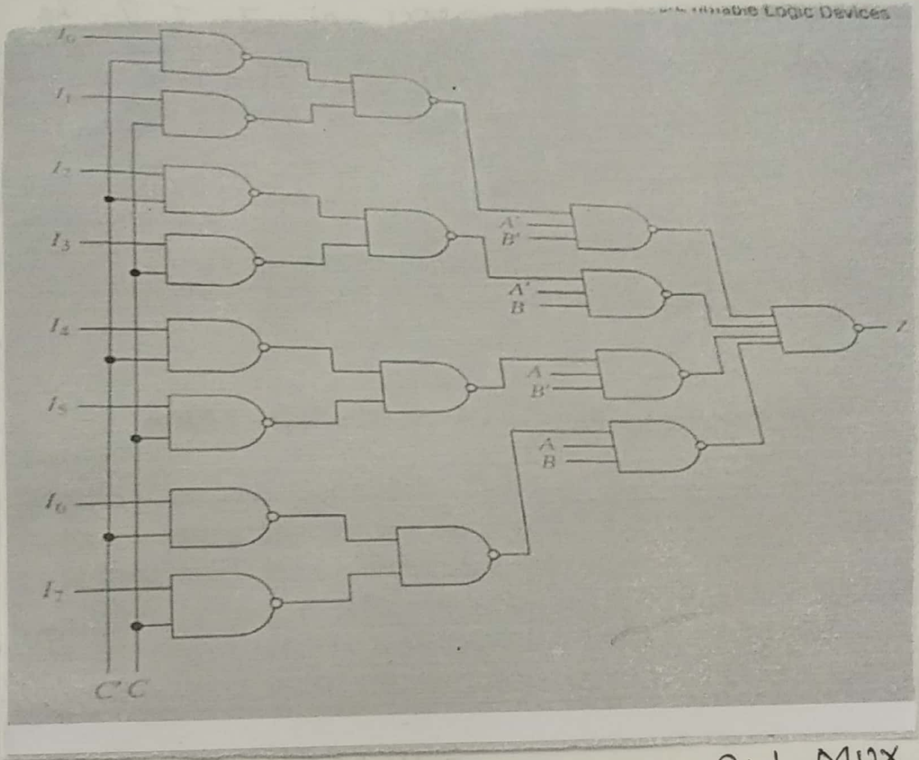


FIGURE 9-3: Logic Diagram for 8-to-1 MUX

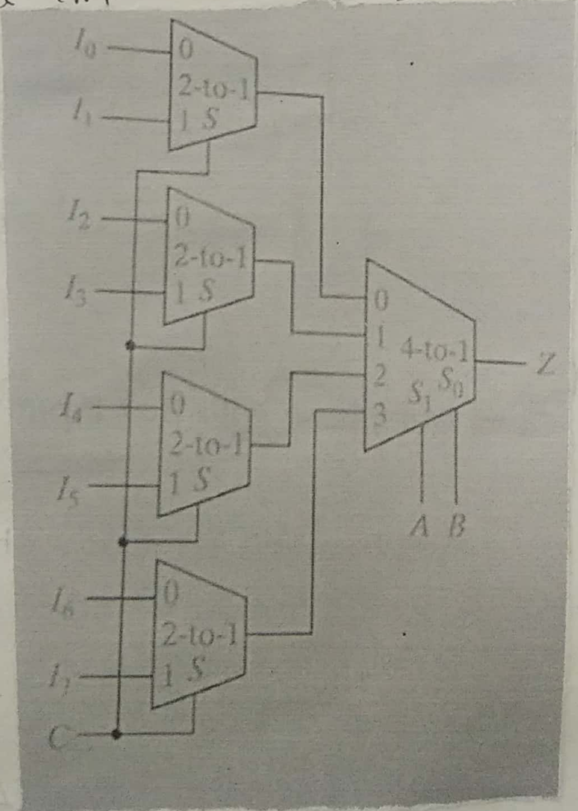
Logic Diagram for 8:1 MUX

$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 +$
$AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$

$Z = A'B'(C'I_0 + CI_1) + A'B(C'I_2 + CI_3) +$
$AB'(C'I_4 + CI_5) + AB(C'I_6 + CI_7)$



<u>Multilevel Implementation of an 8:1 MUX</u>



8:1 MUX using 2:1 & 4:1 MUX

Quadruple 2 to 1 MUX

→ It is used to Select one of two 4 bit data words

If $A = 0 \rightarrow$ value of x_0, x_1, x_2 & x_3 will appear at Z_0, Z_1, Z_2, Z_3

$A = 1 \rightarrow$ Values of y_0, y_1, y_2 & y_3 will apper at Z_0, Z_1, Z_2, Z_3

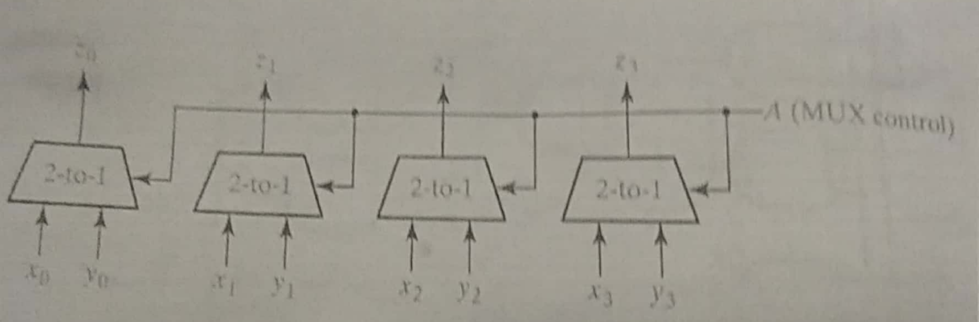


FIGURE 9-6: Quad Multiplexer Used to Select Data

Quad Multiplexer with Bus inputs & outputs

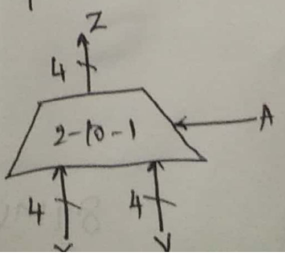
* Two bus inputs X & Y are used & bus o/p Z.

✓ X bus represents the four signals x_0, x_1, x_2 & x_3
lly for y & Z

When $A = 0 \rightarrow$ Signals on bus X appear on bus Z

$A = 1 \longrightarrow$ Signals on bus Y appear on bus Z

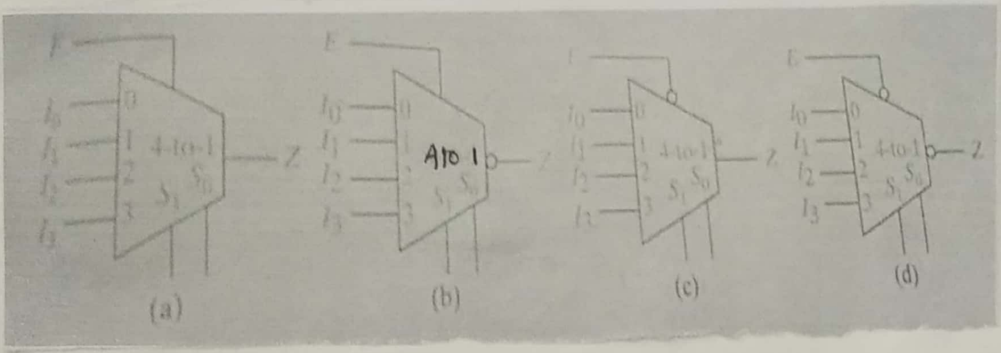
The diagonal Slash through a bus with number beside it Specifies the number of bits in the bus.



MULTIPLEXER with Enable pin

Four Combinations of Multiplexers with an enable are possible. The output can be active high or active low, where the enable Can be active high or active low.

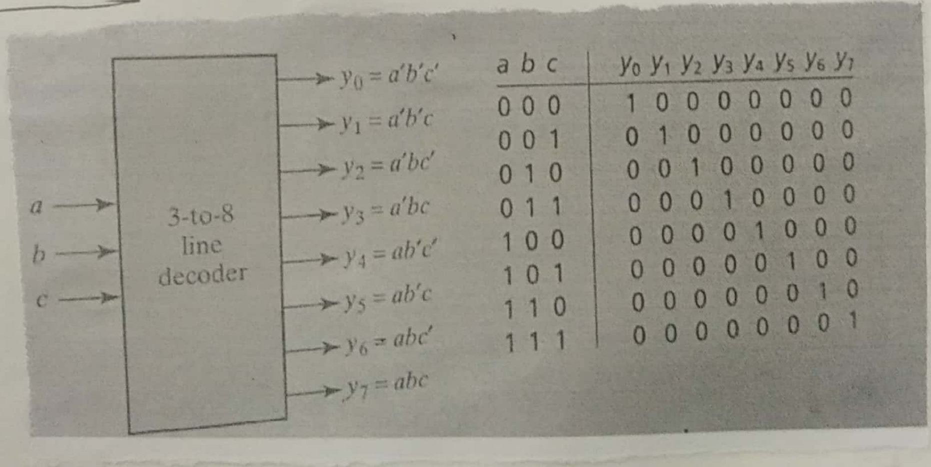
Active low is indicated by inserting bubble on the line to indicate the inclusion of an inversion.



Decoders and Encoders

Decoder generates all of the minterms of the three input variables. Exactly one of the output lines will be 1 for each combination of the values of the input variables.

3 to 8 decoder



a b c	y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_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

4 to 10 decoder

This decoder has inverted outputs. For each combination of the values of the inputs, exactly one of the output lines will be 0.

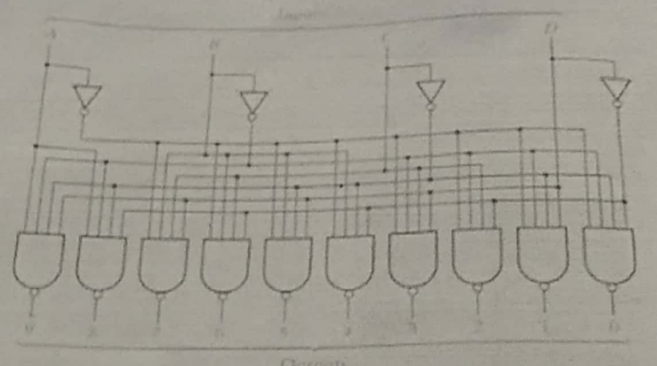
An n to 2^n line decoder generates all 2^n minterms (or Maxterms) of the n input variables.

The outputs are defined by equations

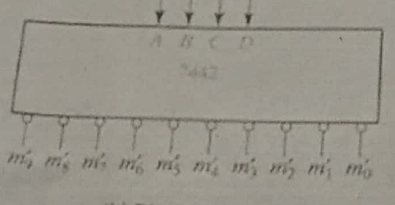
$y_i = m_i = M_i'$ $i = 0$ to $2^n - 1$ (non inverted O/P's)

OR

$y_i = m_i' = M_i$, $i = 0$ to $2^n - 1$ (inverted O/P's)



(a) Logic diagram

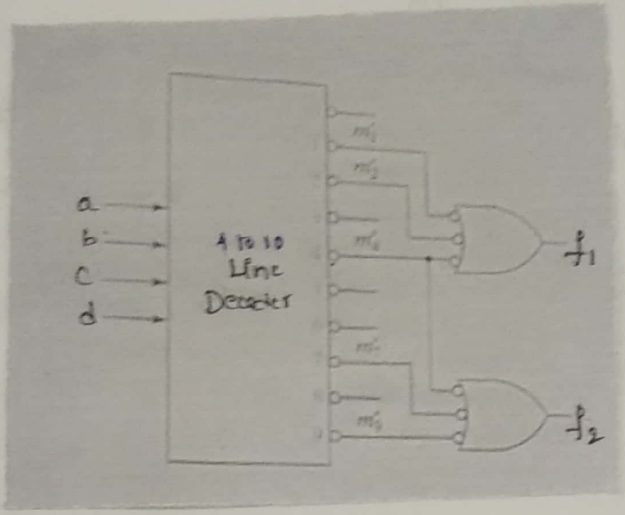


A B C D
7442
m_9' m_8' m_7' m_6' m_5' m_4' m_3' m_2' m_1' m_0'

(b) Block diagram

BCD Input				Decimal Output									
A	B	C	D	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Implement $f_1(abcd) = m_1 + m_2 + m_4$ and
$f_2(a,b,c,d) = m_4 + m_7 + m_9$



8 to 3 Priority Encoder

Encoder Performs the inverse function of a decoder.

8 to 3 priority encoder with inputs y_0 through y_7

If input y_i is 1 and the other inputs are 0,

Then the abc outputs represent a binary number

equal to i.

for example: if $y_3 = 1$, then abc = 011.

If more than one input can be 1 at the Same time, the output can be defined using a priority Scheme.

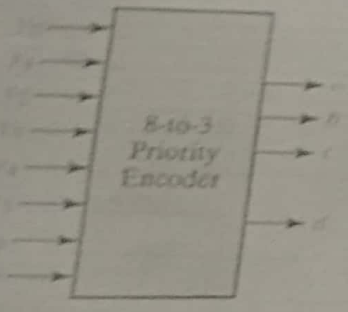
If more than one input is 1, the highest numbered input determines the O/P.

If i/p's y_1, y_4 & y_5 are 1, the output is

The X's in the table are don't-cares,

example: if y_5 is 1, we do not care what inputs y_0 through y_4 are.

output d is 1 if any input is 1, otherwise d=0. This signal is needed to distinguish the case of all 0 inputs from the case where only y_0 is 1.



y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1

FIGURE 9-20: An 8-to-3 Priority Encoder