Digital Design & Computer Organization – BCS302(IPCC)
Module-2

Prepared By,

Dr. Vinod Kumar P

Associate Professor

Department of CSE-DS

ATME College of Engineering, Mysuru

COMBINATIONAL CIRCUIT DESIGN & SIMULATION USING GATES

Digital Logic Families

The following are the most popular digital logic families

TTL – Transistor Transistor Logic

ECL – Emitter Coupled Logic

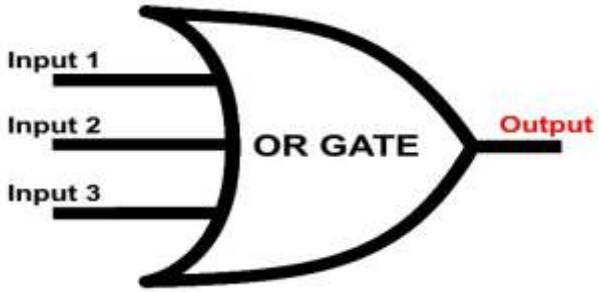
MOS – Metal-Oxide Semiconductor

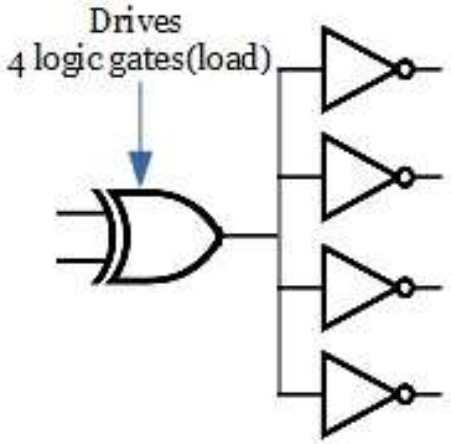
CMOS – Complementary Metal-Oxide Semiconductor

Characteristics of Logic Families

1. Fan-in
2. Fan-out
3. Propagation delay
4. Noise margin
5. Power dissipation

COMBINATIONAL CIRCUIT DESIGN & SIMULATION USING GATES





Fan-in: refers to the number of inputs in a digital logic gate family.

Fan-out: specifies the no. of standard loads that the output of typical gate can drive without impairing its normal operation

COMBINATIONAL CIRCUIT DESIGN & SIMULATION USING GATES

Propagation delay: is the average transition time for a signal to propagate from input to output.

Noise margin: is the maximum external noise voltage added to an input signal that does not cause an undesirable change in the output.

Power dissipation: it is the power consumed by the gate that must be available from power supply.

REVIEW OF COMBINATIONAL CIRCUIT DESIGN

Steps

- Set up a truth table which specifies the output(s) as a function of the input variables.
- Derive simplified algebraic expressions for the output functions using Karnaugh Maps, or Quine- McCluskey method, or any other similar procedure.
- The resulting algebraic expressions are then manipulated into the proper form, depending on the type of gates to be used in realizing the circuit.
- Minimum two-level AND-OR, or NAND-NAND circuits can be realized using the minimum sum-of-products. Minimum two-level OR-AND, or NOR-NOR circuits can be realized using the minimum product-of-sums.

DESIGN OF CIRCUITS WITH LIMITED GATE FAN-IN

- In practical logic design problems, the maximum number of inputs on each gate is limited.
- Depending on the type of gates used, this limit may be two, three, four, eight etc..
- If a two-level realization of a circuit requires more gate inputs than allowed, factoring the logic expression to obtain a multi-level realization is necessary.

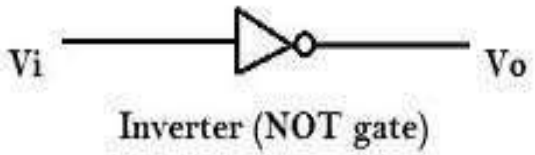
Example 1: *Realize the following functions using only two-input NAND gates and inverters.*

$f1 = \Sigma m\ (0,2,3,4,5)$
$f2 = \Sigma m\ (0,2,3,4,7)$
$f3 = \Sigma m(1,2,6,7)$

GATE DELAYS AND TIMING DIAGRAMS

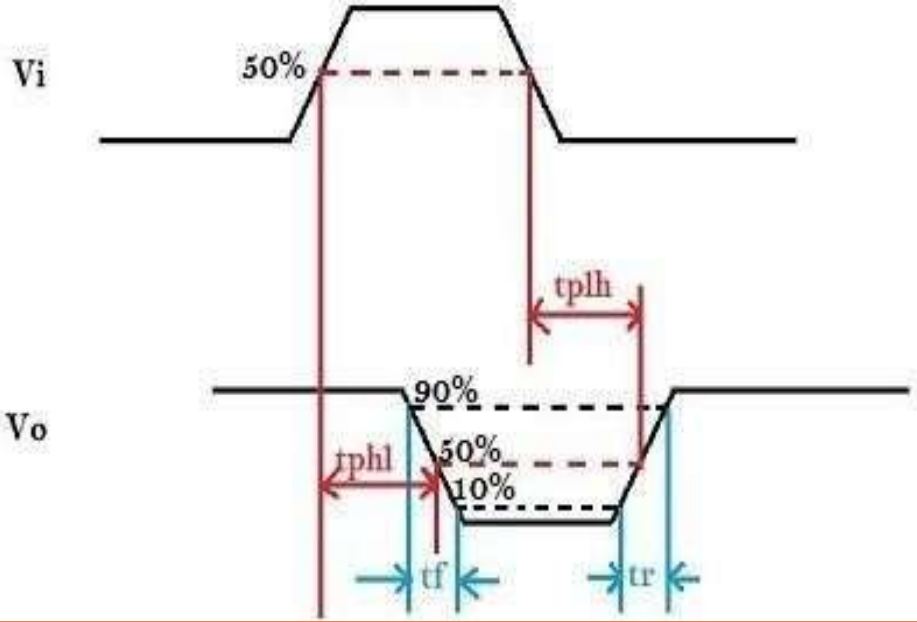
- When the input to a logic gate is changed, the output will not change instantaneously.
- The gates take a finite time to react to a change in input.
- So that the change in the gate output is delayed with respect to the input change.
- If the change in output is delayed by time, ε, with respect to the input, we say that, the gate has a propagation delay of ε.

Figure shows possible input and output waveforms for an inverter

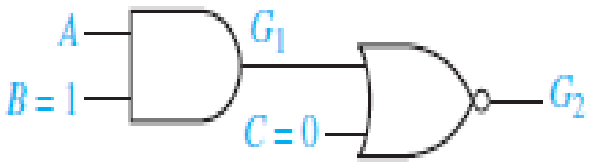


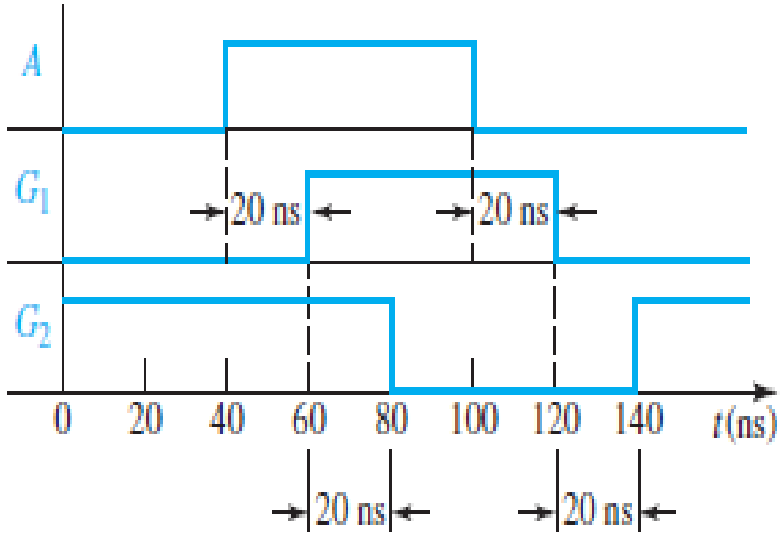
Inverter (NOT gate)

tr = Rise transition time
tf = Fall transition time
tphl = Propagation delay high-low
tplh = Propagation delay low-high



Timing diagram for a circuit with two gates





- Each gate has a propagation delay of 20 ns
- Gate inputs B and C are held at constant values 1 and 0.
- Input A is changed to 1 at *t = 40 ns* and then changed back to 0 at *t = 100 ns*.
- The output of gate G1 changes 20 ns after A changes, and the output of gate G2 changes 20 ns after G1 changes.

HAZARDS IN COMBINATIONAL LOGIC

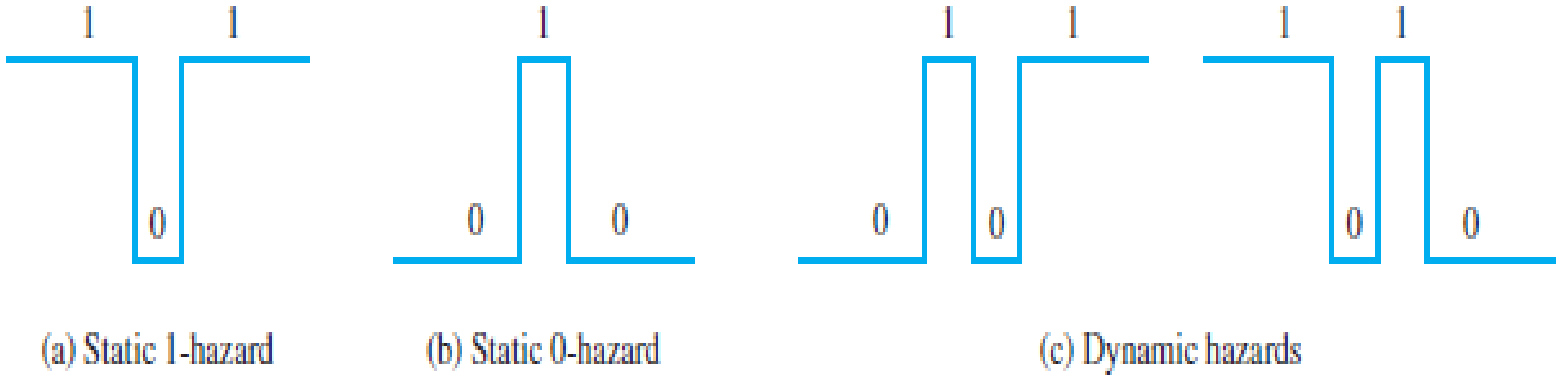
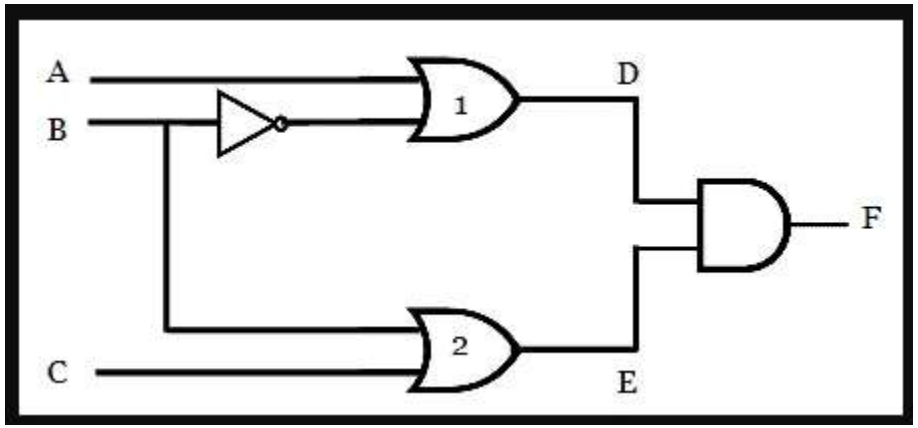


Fig. Hazards

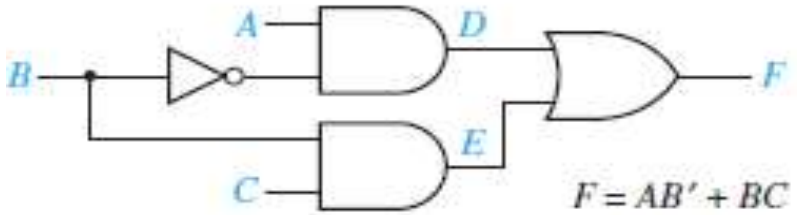
- If, in response to any single input change and for some combination of propagation delays, a circuit output may momentarily go to 0 when it should remain a constant 1, we say that the circuit has a ***static 1-hazard***.
- Similarly, if the output may momentarily go to 1 when it should remain a 0, we say that the circuit has a ***static 0-hazard***.
- If, when the output is supposed to change from 0 to 1 (or 1 to 0), the output may change three or more times, we say that the circuit has a ***dynamic hazard***.

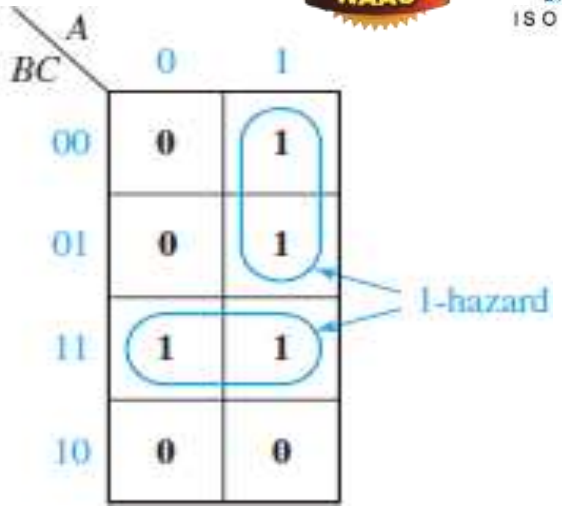
Static 0-hazard.



- If $A = B = C = 0$ and then output F =0.
- When B changes from 1 to 0 then output F should remain a constant 0. Now E will go to 1 before D goes to 0, resulting in a momentary 1 (a glitch) appearing at the output F .
- So F momentarily goes to 1. This is called Static 0 Hazard.

Static 1-hazard.





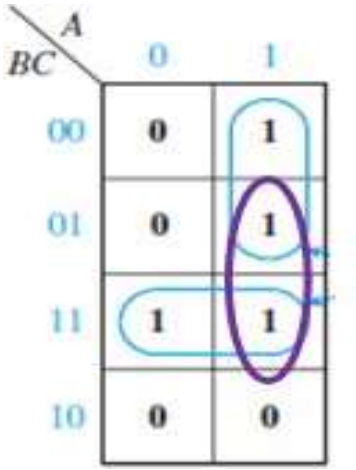
- If *A* = *C* = *1* and B=1 then output F =1.
- When *B* changes from 1 to 0 then output F should remain a constant 1. Now *E* will go to 0 before *D* goes to 1, resulting in a momentary 0 (a glitch) appearing at the output *F* .
- So *F* momentarily goes to 0. This is called Static 1 Hazard.

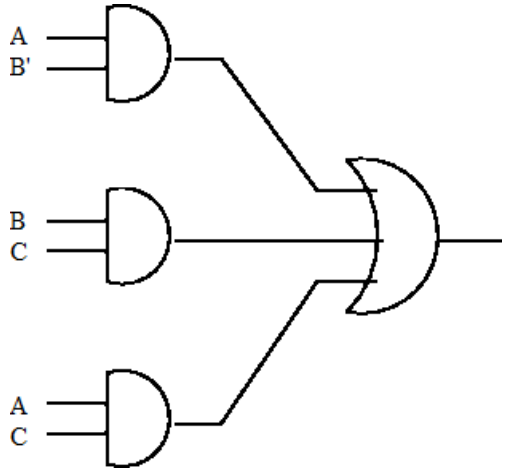
Detection of Static 1 Hazard

Hazards can be detected using a Karnaugh map.

- Write down the sum-of-products expression for the circuit.
- Plot each term on the map and loop it.
- If any two adjacent 1's are not covered by the same loop, a 1-hazard exists for the transition between the two 1's.

To Eliminate Static 1 Hazard





- If we add a loop to the map of above Figure and, then, add the corresponding gate to the circuit (as shown in the following Figure), this eliminates the hazard.
- The term *AC* remains *1* while *B* is changing, so no glitch can appear in the output.

SIMULATION AND TESTING OF LOGIC CIRCUITS

- In logic design process, Logic circuits may be tested either by building them or by simulating them on a computer.
- As logic circuits become more and more complex, it is very important to simulate a design before building it.

Simulation is done for following reasons

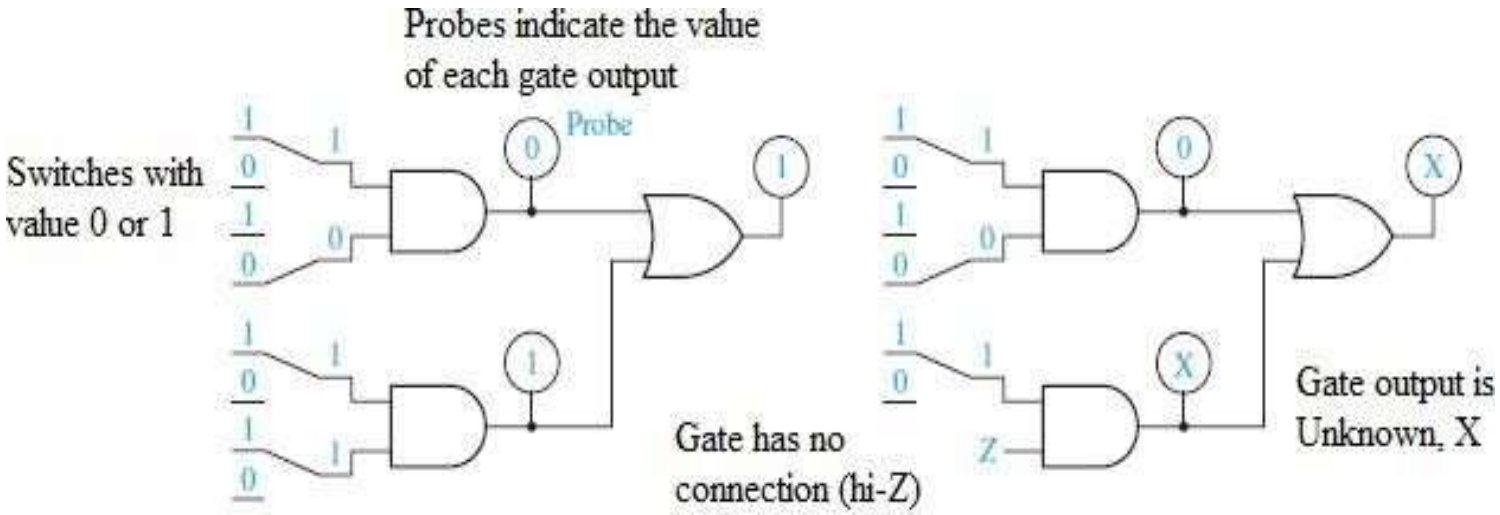
(1) Verification that the design is logically correct
(2) Verification that the timing of the logic signals is correct
(3) Simulation of faulty components in the circuit as an aid to finding tests for the circuits.

A simple simulator for combinational logic works as follows

- The circuit inputs are applied to the first set of gates in the circuit, and the outputs of those gates are calculated.
- The outputs of the gates which changed in the previous step are fed into the next level of gate inputs. If the input to any gate has changed, then the output of that gate is calculated.

Four-Valued Logic Simulator:

- The two logic values, ***0*** and ***1***, are not sufficient for simulating logic circuits.
- The value of a gate input or output may be ***unknown***, and we will represent this unknown value by ***X***.
- If no logic signal at an input, as in the case of an open circuit we use the logic value ***Z*** to represent an ***open circuit***, or ***high impedance*** (hi-Z) connection.



The following Figure shows a typical simulation screen on a personal computer.

•	0	1	X	Z
0	0	0	0	0
1	0	1	X	X
X	0	X	X	X
Z	0	X	X	X

+	0	1	X	Z
0	0	1	X	X
1	1	1	1	1
X	X	1	X	X
Z	X	1	X	X

The following Table shows AND and OR functions for four-valued logic simulation.

For an AND gate,

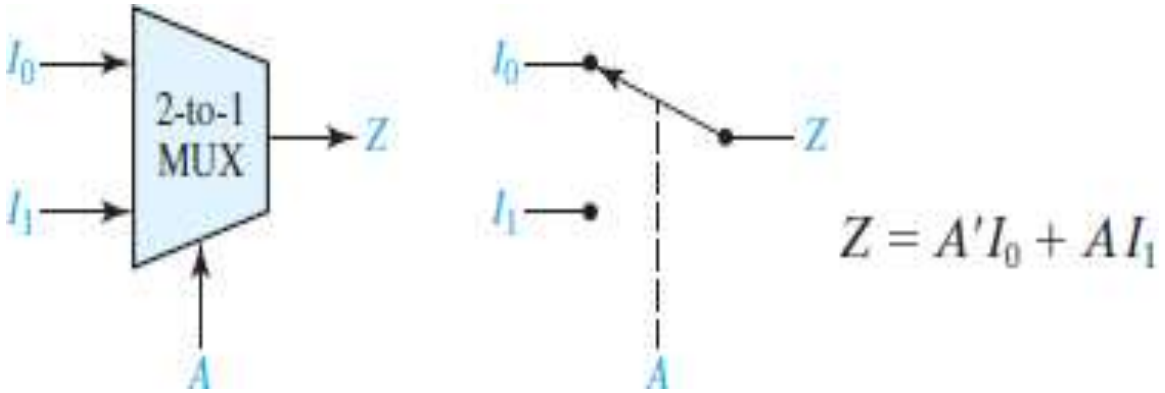
- If one of the inputs is *0*, the output is always *0* regardless of the other input
- If one input is *1* and the other input is *X* then the output is *X*.
- If one input is *1* and the other input is *Z* then the output is *X*.

For an OR gate,

- If one of the inputs is *1*, the output is **1** regardless of the other input
- If one input is 0 and the other input is *X* or *Z*, the output is unknown.

MULTIPLEXERS

- A *multiplexer* is a *data selector* which has a group of data inputs , a group of control inputs and single output.
- The control inputs are used to select one of the data inputs and connect it to the output terminal.

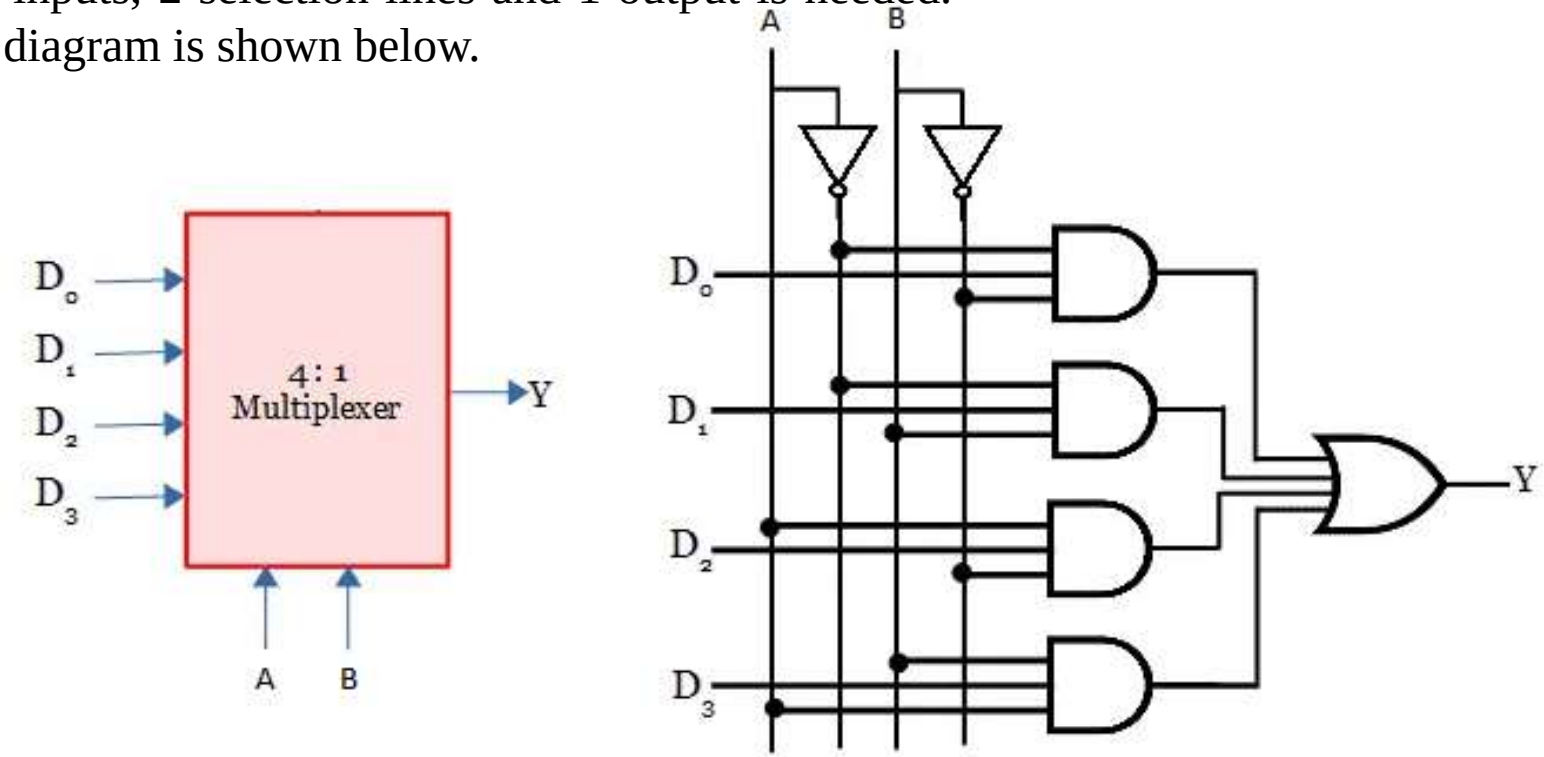


2-to-1 multiplexer

- When the control input A is 0, the switch is in the upper position and the MUX output is Z = I0.
- when A is 1, the switch is in the lower position and the MUX output is Z = I1.
- MUX acts like a switch that selects one of the data inputs (I0 or I1) and transmits it to the output .
- The logic equation for the 2-to-1 MUX is $Z = A'I_0 + AI_1$

4 to 1 Multiplexer

For 4 to 1 multiplexer, 4 data inputs, 2 selection lines and 1 output is needed.
The block diagram and circuit diagram is shown below.



Block diagram and logic circuit of 4 : 1 mux

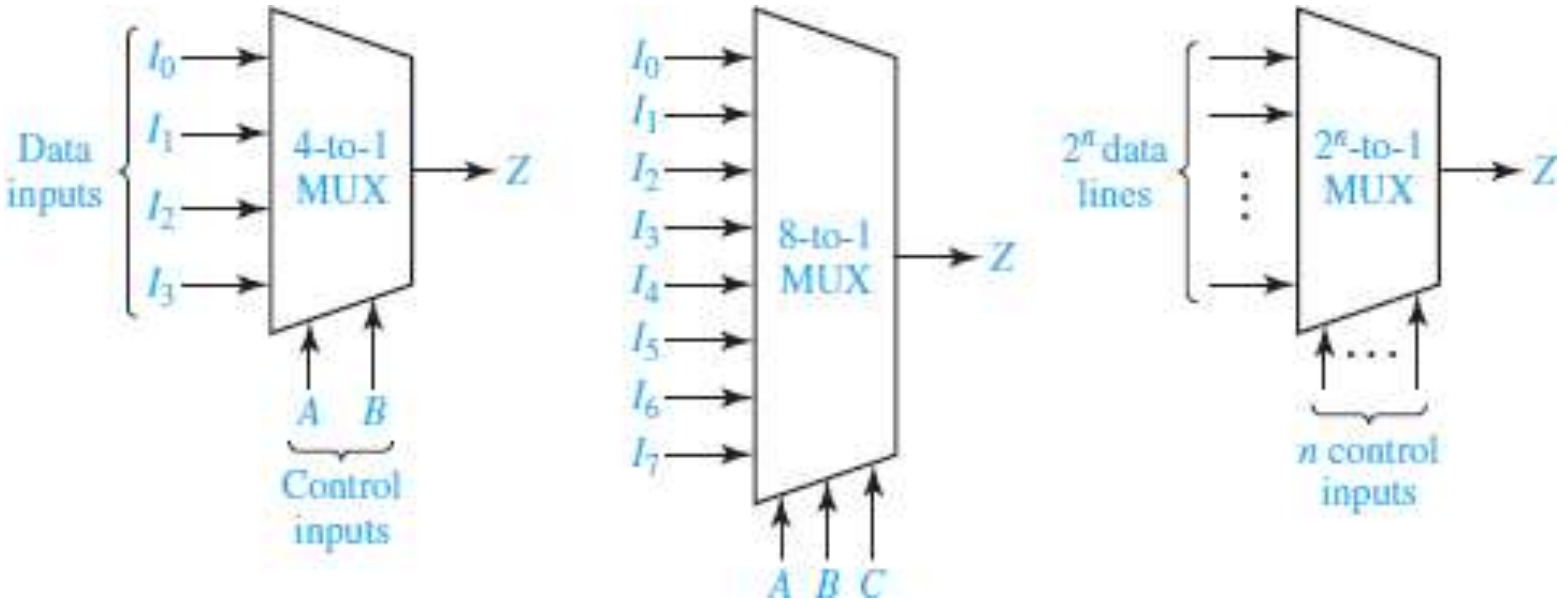
- For selection inputs, A= 0, B=0, first AND gate alone is enabled and the output produced is $Y = D_0 \overline{A}\ \overline{B}$
- For selection inputs, A= 0, B=1, second AND gate alone is enabled and the output produced is $Y = D_1 \overline{A}\ B$

- For selection inputs, A= 1, B=0, third AND gate alone is enabled and the output produced is $Y = D_2 \, A \, \overline{B}$
- For selection inputs, A= 1, B=1, forth AND gate alone is enabled and the output produced is $Y = D_3 \, A \, B$

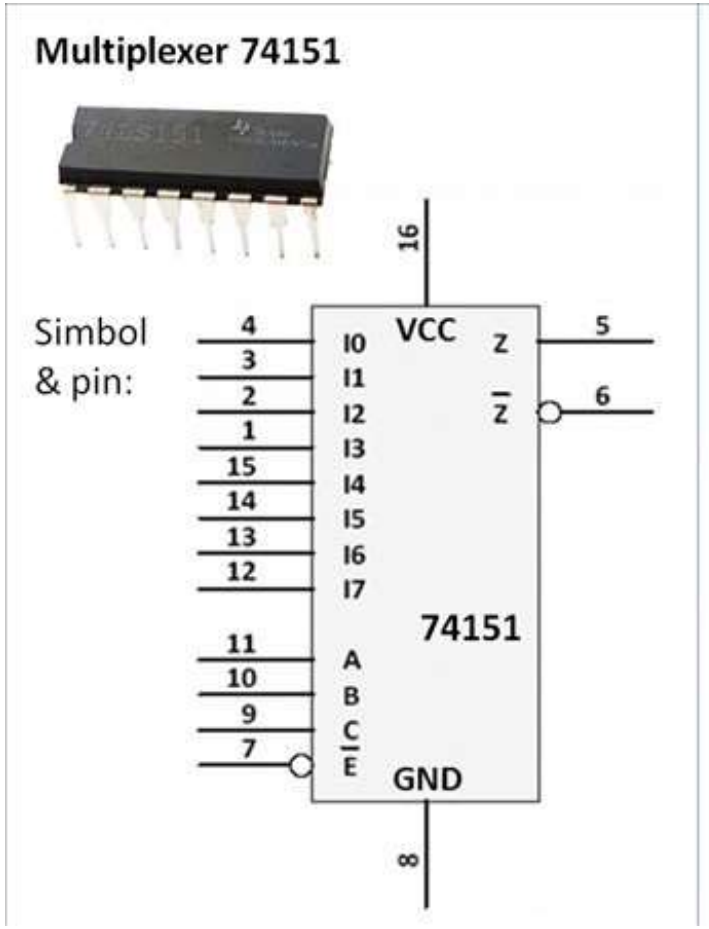
Truth table for 4 to 1 MUX is shown below

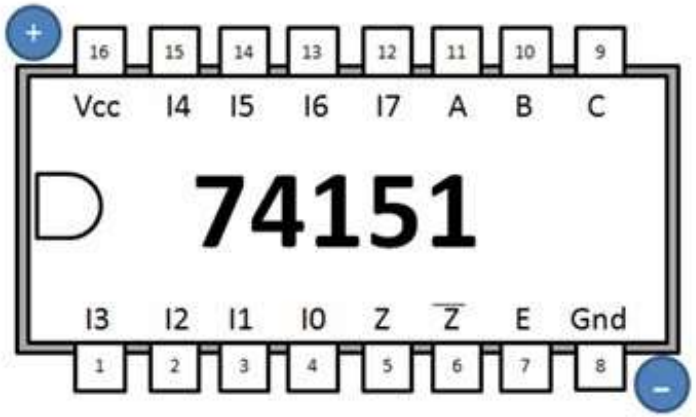
Select Data Inputs		Output
A	B	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Figure shows diagrams for a 4-to-1 multiplexer, 8-to-1 multiplexer, and 2^n-to-1 multiplexer.



8 to 1 Multiplexer IC 74151

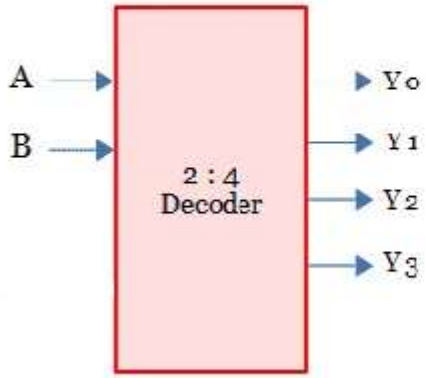




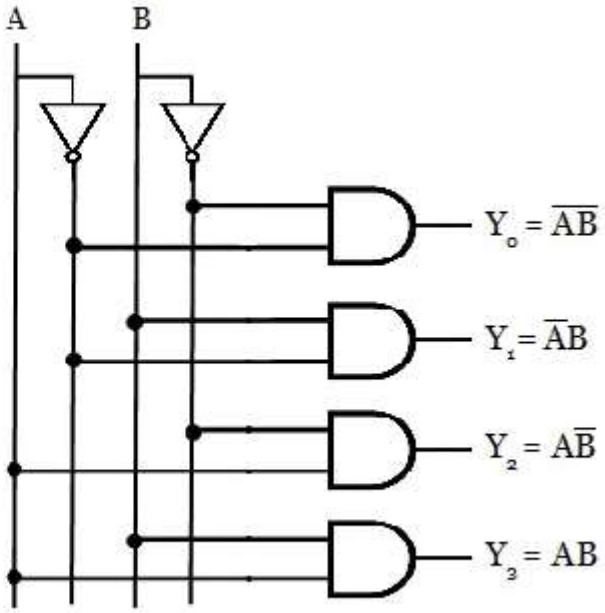
Decoder

A decoder is a multiple-input, multiple-output combinational logic circuit. It converts the n bit data inputs into the coded 2^n outputs.

2 to 4 binary decoder



Block diagram



Logic diagram

- The block diagram and Logic diagram is shown above. A and B are the two inputs and the output produced is one of the minterms.
- The circuit diagram has two inverters, which will provide the complement of two inputs A and B.
- Each AND gates generates one of the minters as an output.

Truth table

Inputs		Outputs			
A	B	Y3	Y2	Y1	Yo
x	x	0	0	0	0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Logic Expression

$$Y_o = \bar{A}\bar{B}$$

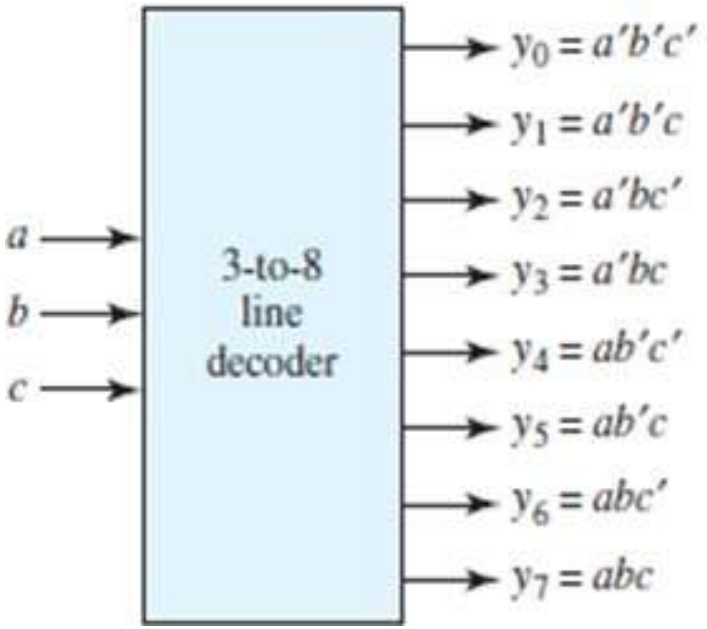
$$Y_1 = \bar{A}B$$

$$Y_2 = A\bar{B}$$

$$Y_3 = AB$$

From the above truth table, the operation can be understood.

- When both the inputs A and B are 0, Y_0 will be at active HIGH or logic 1 and the remaining output pins are active LOW or logic 0.
- When A = 0 and B = 1, Y_1 is at HIGH.
- When A = 1 and B = 0, Y_2 is at HIGH.
- When A = 1 and B = 1, Y_3 is at HIGH.

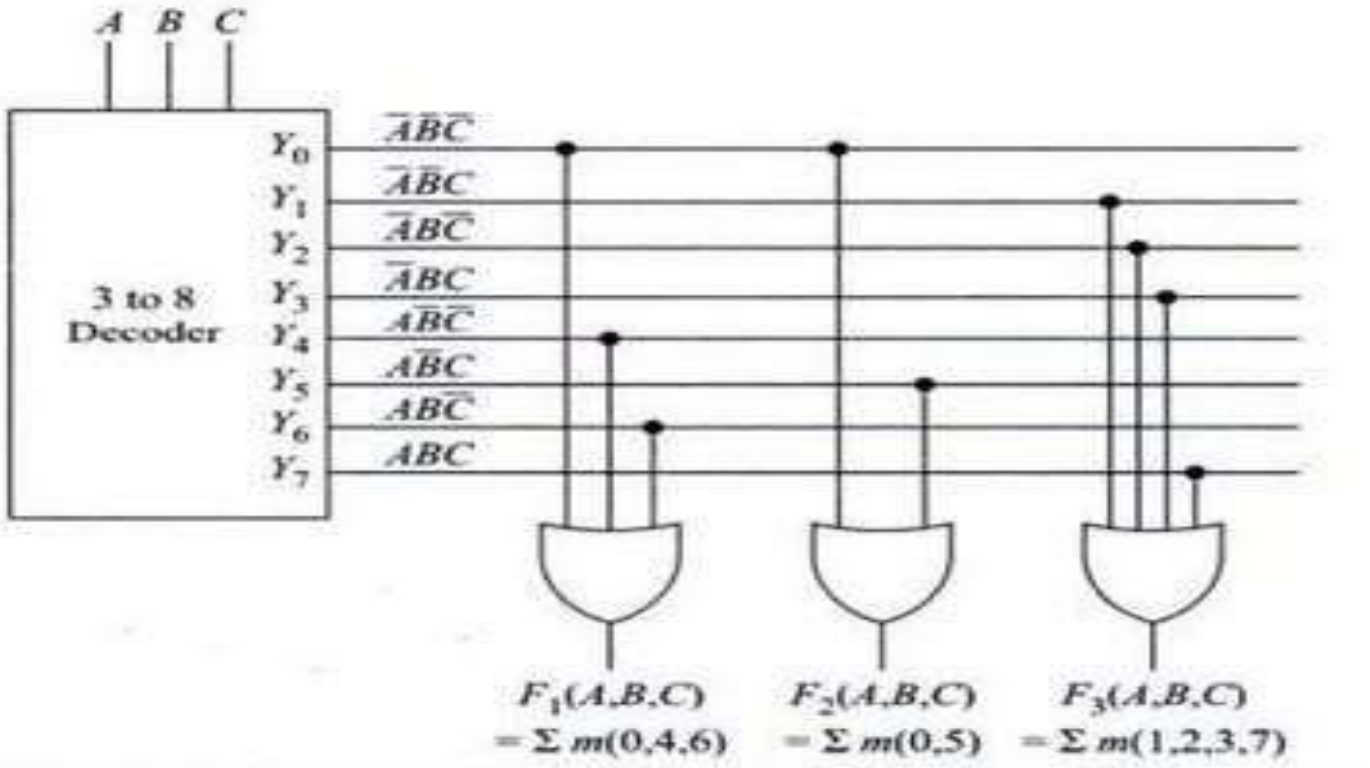


a	b	c	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

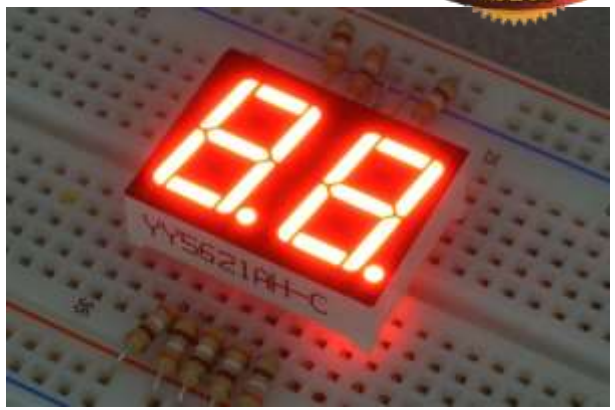
3-to-8-line decoder

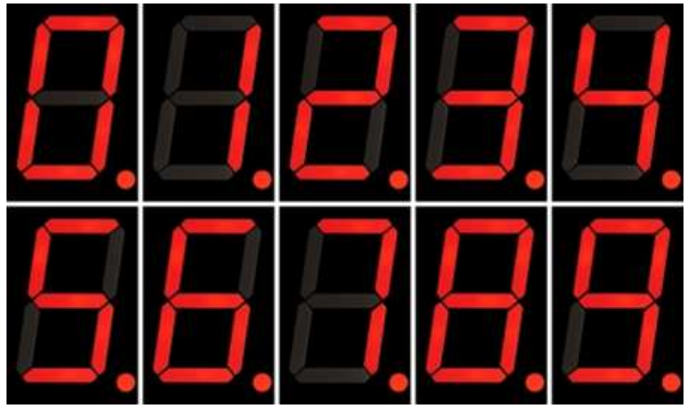
Problem: Show how using a 3-to8 decoder and multi-input OR gates following Boolean expression can be realized simultaneously.
F1 (A, B, C) = ∑m (0, 4, 6)
F2 (A, B, C) = ∑m (0, 5)
F3 (A, B, C) = ∑m (1, 2, 3, 7).

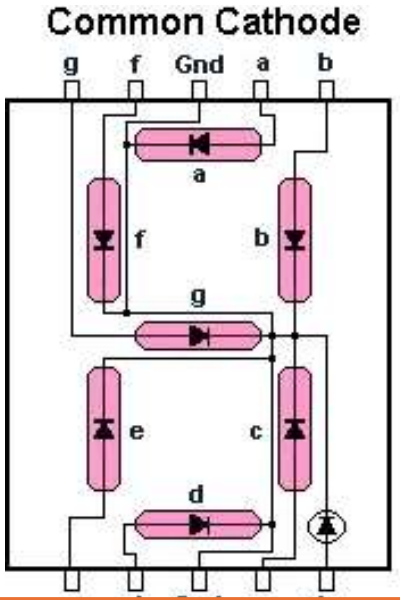
Solution:

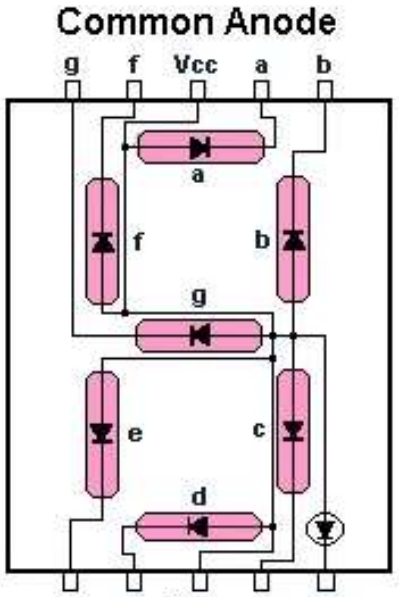


Seven-Segment Display

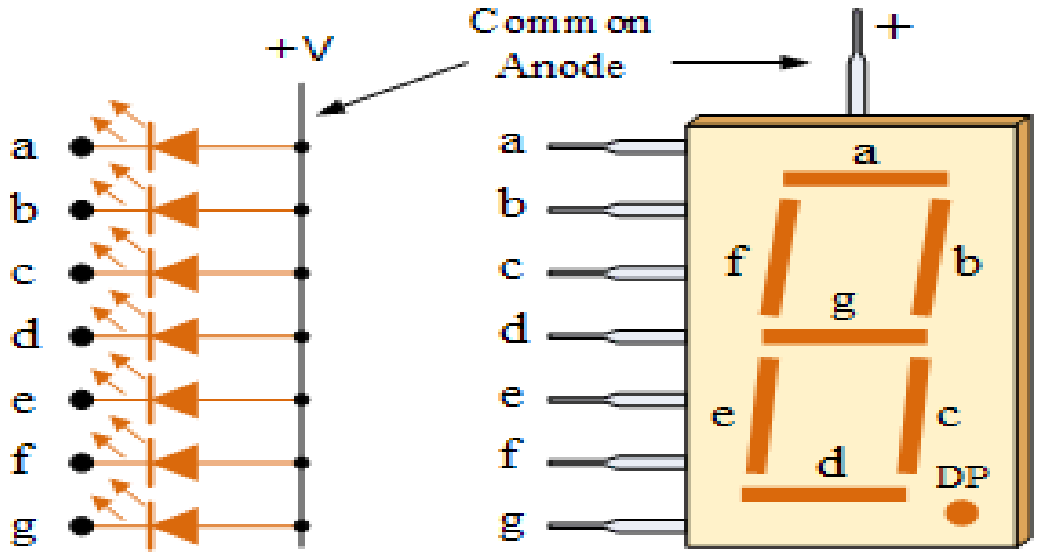




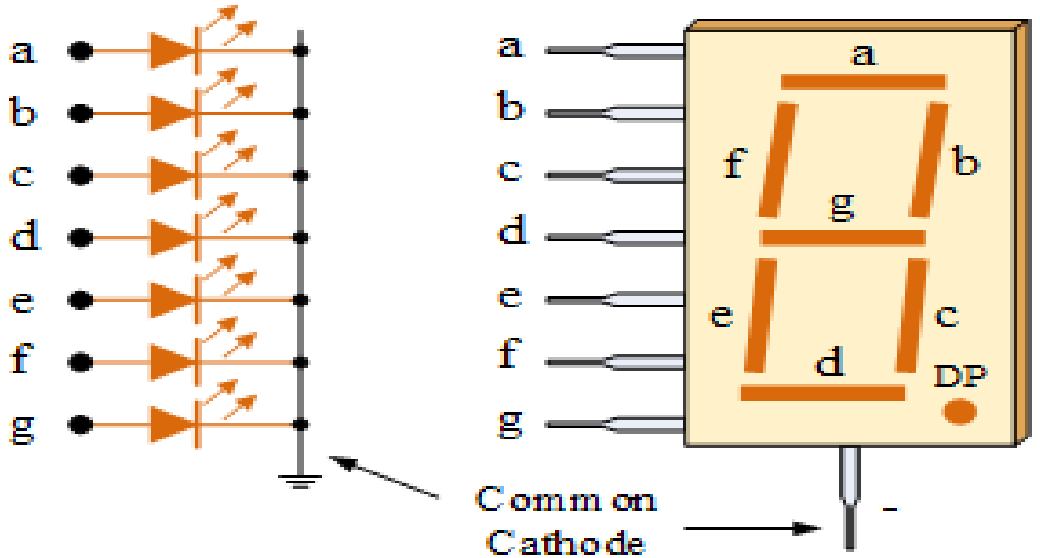




Common Anode Seven-Segment Display

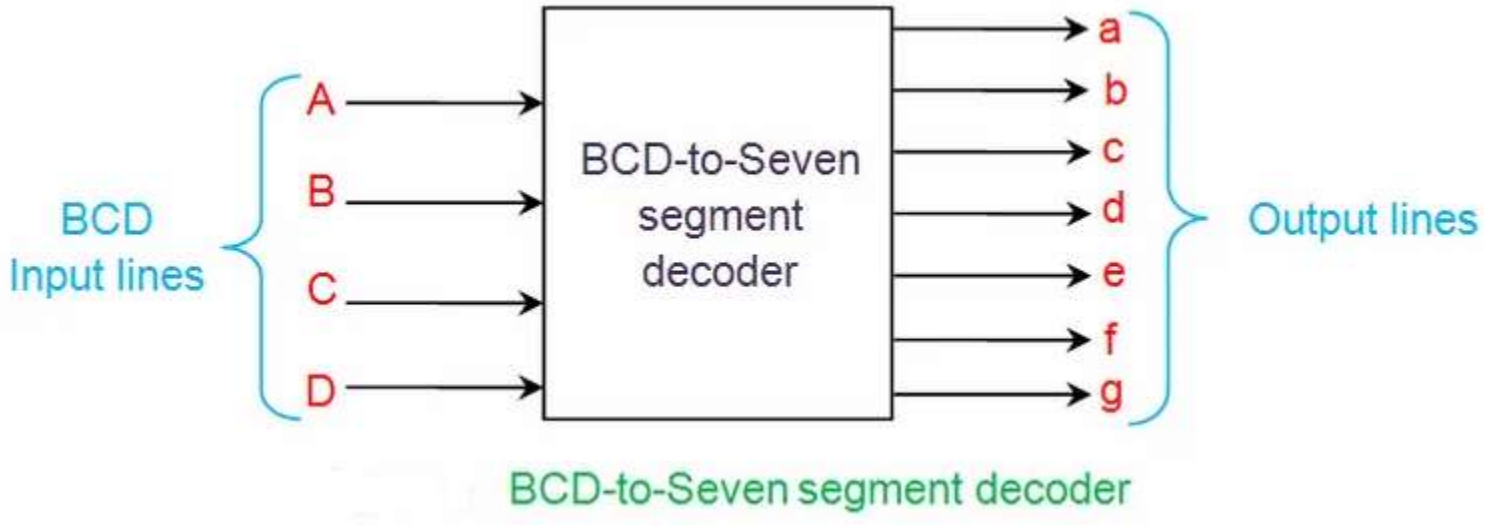


Common Cathode Seven-Segment Display



BCD to 7 Segment Decoder

- It converts BCD into 7 segment outputs

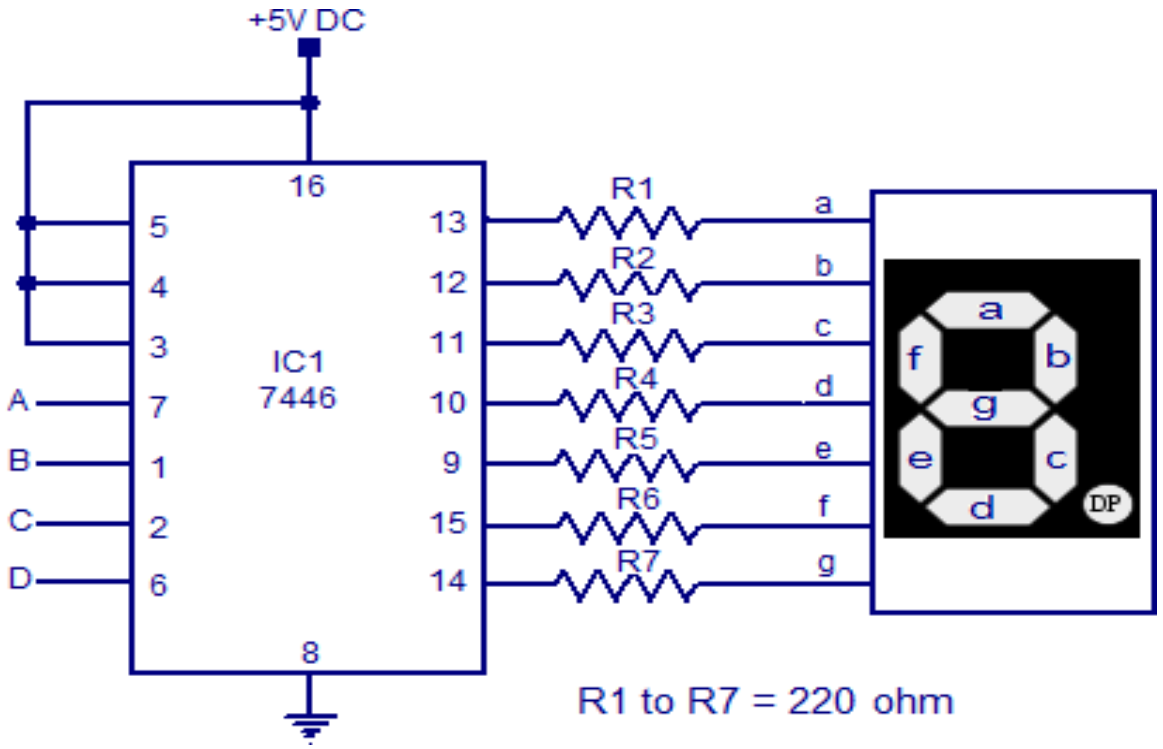


Truth Table

Truth table for common cathode type BCD to seven segment decoder

Decimal Digit	Input lines				Output lines							Display pattern
	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	

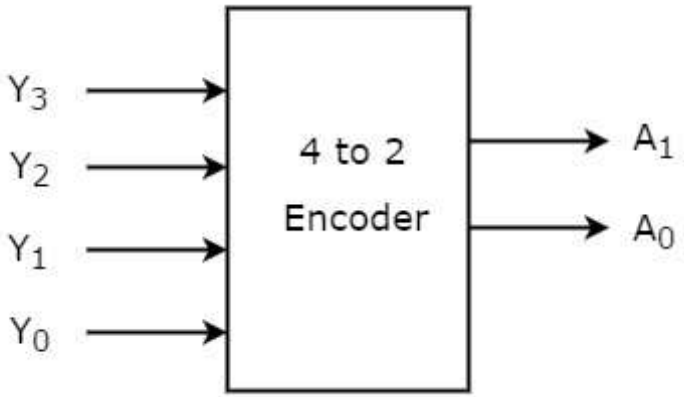
Seven-Segment Decoder IC 7446 with Seven Segment Display



ENCODERS

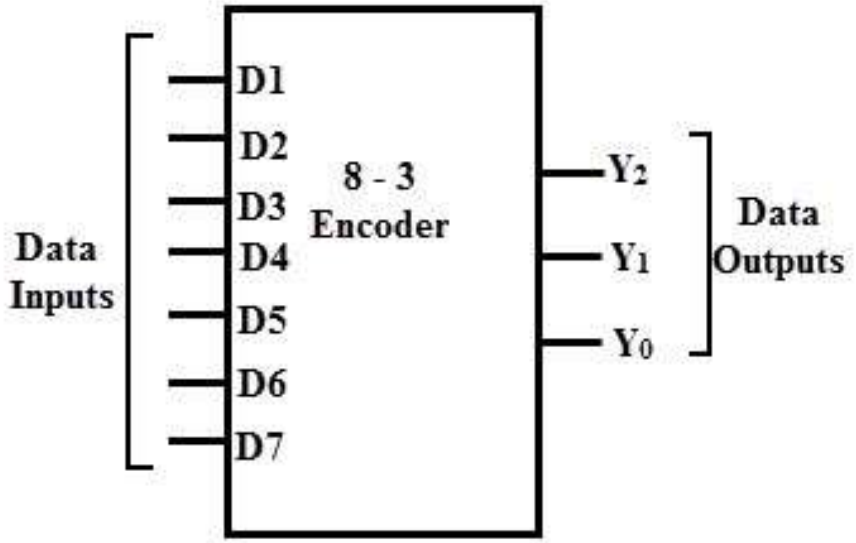
An ***encoder*** (*converts an active input signal to a coded output signal*) performs the inverse function of a decoder.

4 to 2 Encoder



INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

8 to 3 Encoder



Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
×	1	0	0	0	0	0	0	0	0	1
×	×	1	0	0	0	0	0	0	1	0
×	×	×	1	0	0	0	0	0	1	1
×	×	×	×	1	0	0	0	1	0	0
×	×	×	×	×	1	0	0	1	0	1
×	×	×	×	×	×	1	0	1	1	0
×	×	×	×	×	×	×	1	1	1	1

THREE STATE BUFFERS

- A gate output can only be connected to a limited number of other device inputs without degrading the performance of a digital system.
- A simple buffer may be used to increase the driving capability of a gate output.

The following Figure shows a three-state buffer and its logical equivalent

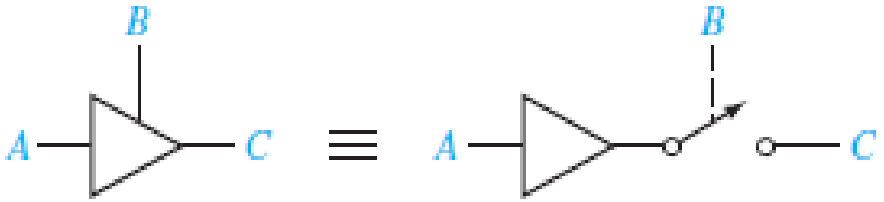
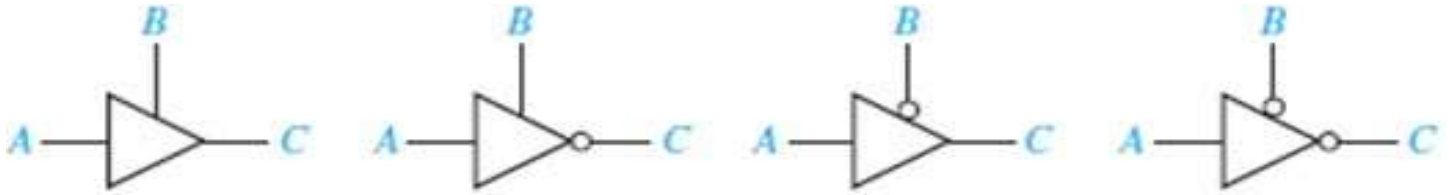


Fig: Three state buffer

- When the enable input B is 1, the output C equals A.
- when B is 0, the output C acts like an open circuit. This is called as high-impedance (Hi-Z) state.
- Three-state buffers are also called ***tri-state buffers***

Types of three-state buffers.



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

(b)

B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

(c)

B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

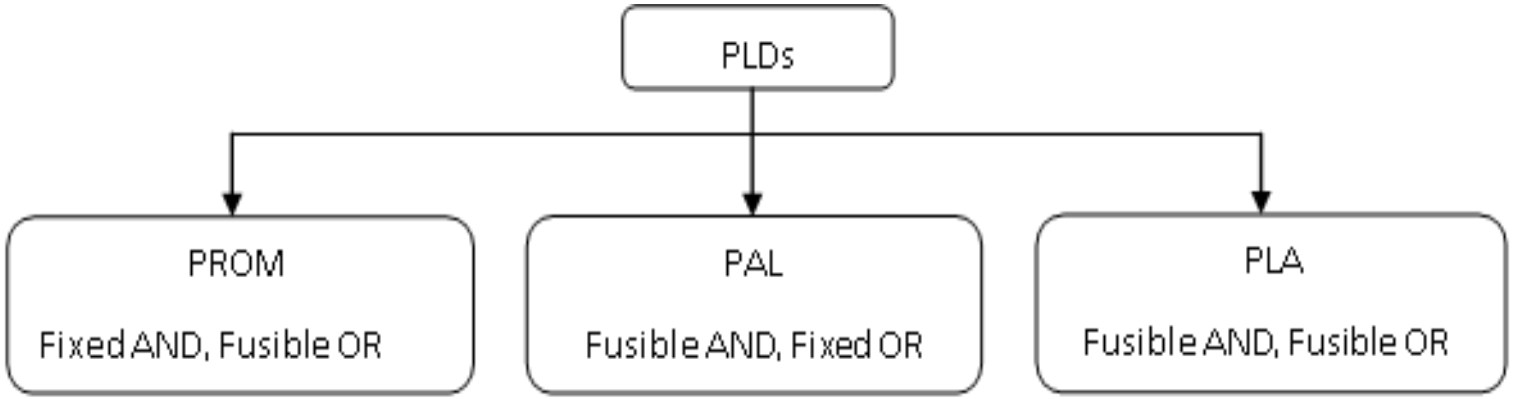
(d)

- In Figures (a) and (b) the buffer output is enabled when B = 1 and disabled when B = 0.
- In Figures (c) and (d) the buffer output is enabled when B = 0 and disabled when B = 1.

PROGRAMMABLE LOGIC DEVICES (PLDs)

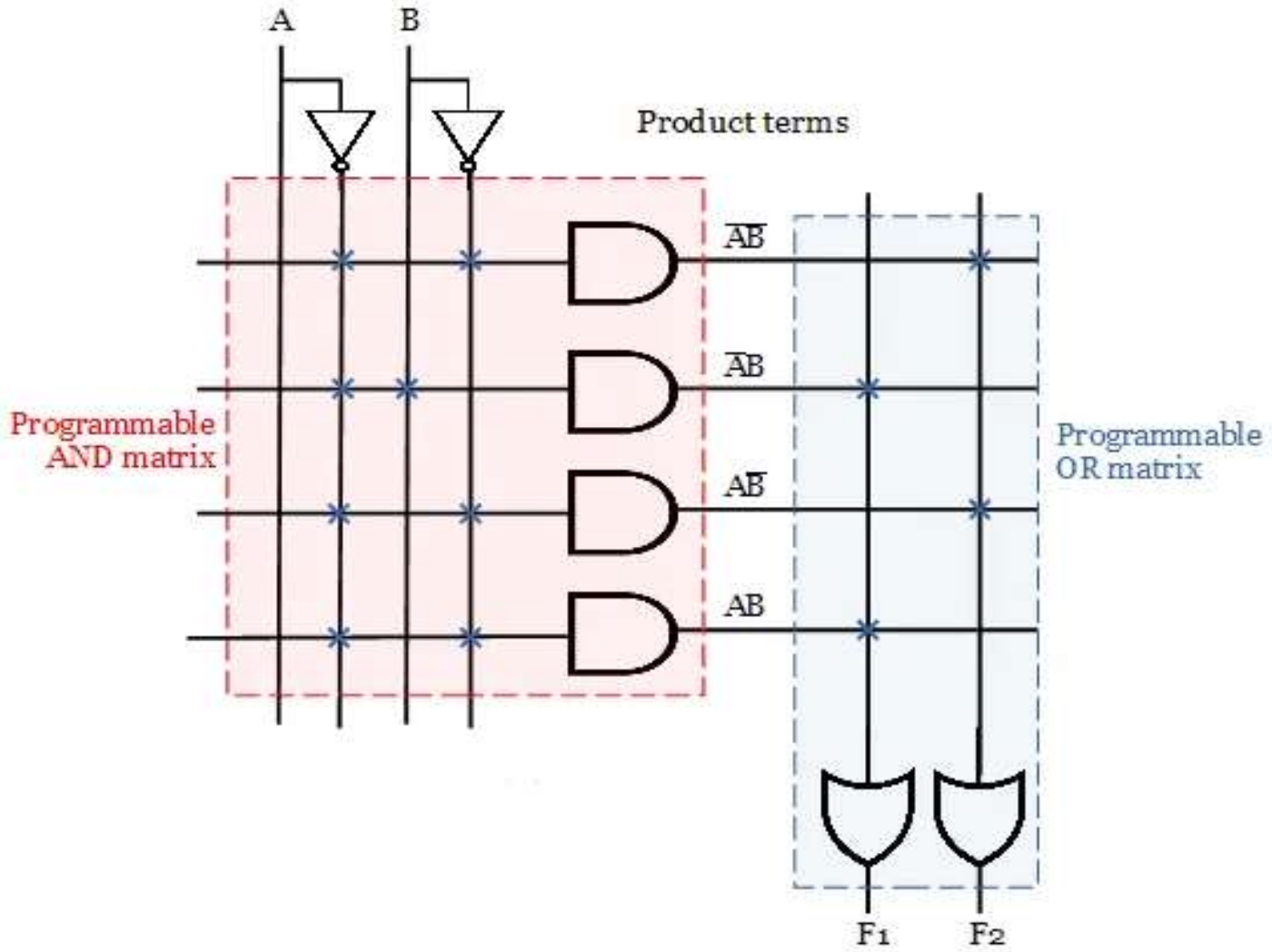
- A *programmable logic device* (*PLD*) is a digital integrated circuit capable of being programmed to provide different logic functions.

Classification



- Programmable Read Only Memory (PROM)
- Programmable Array Logic (PAL)
- Programmable Logic Array (PLA)

An example of PLA circuit of a combinational circuit is shown below. As you can observe from the circuit diagram, AND array consists of fuses, to program according to the user requirements.

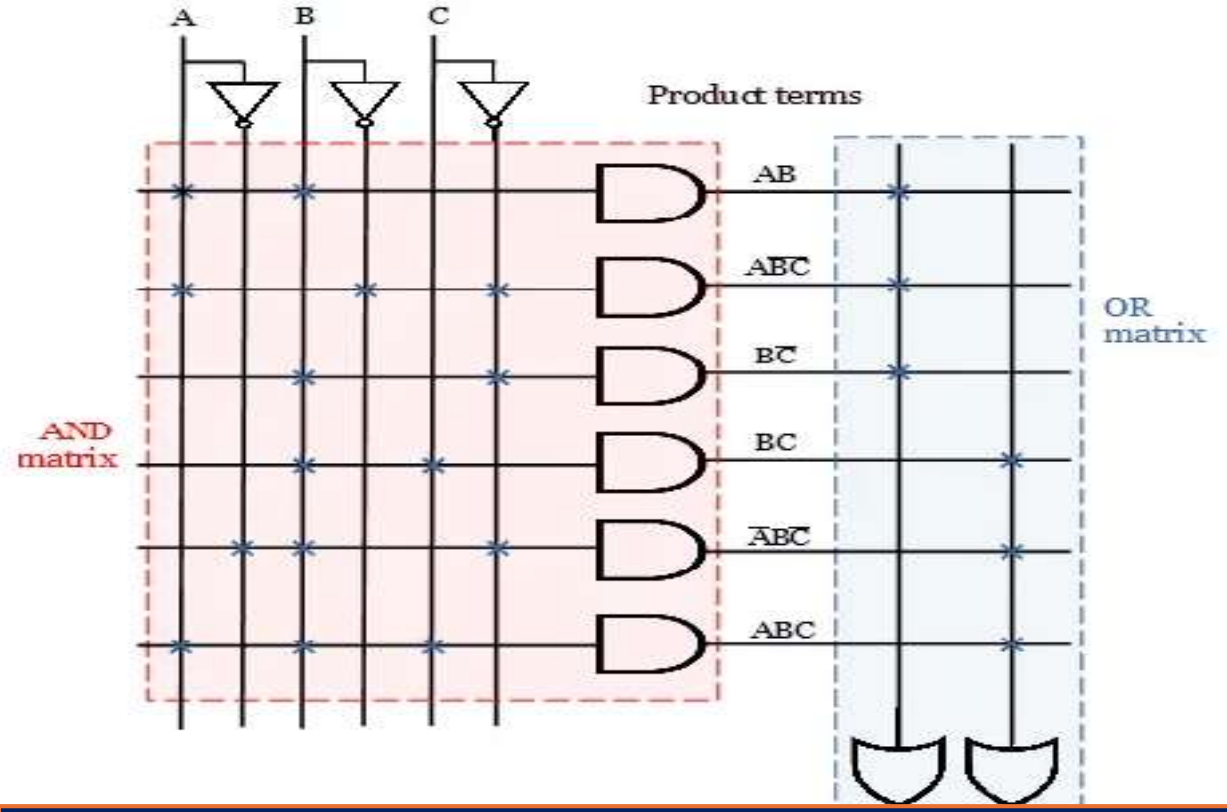


Example 1

Realize the Boolean expression
W = AB + AB'C' + BC' and
X = BC + A'BC' + ABC using Programmable Logic Array.

Solution:

- There are three inputs(A, B, C) and two outputs(W, X). The complement of three inputs are obtained through NOT gates.
- The given expression has six product terms.
- Two OR gate arrays are used at the output to realize the the two functions.



Example 2

Realize a boolean functions
F1(A, B, C) = ∑ m(1, 3, 6, 7) and
F2(A, B, C) = ∑ m(0, 2, 4, 5) using PLA.

Solution

To obtain the expression, the given function is implemented using Karnaugh map.

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

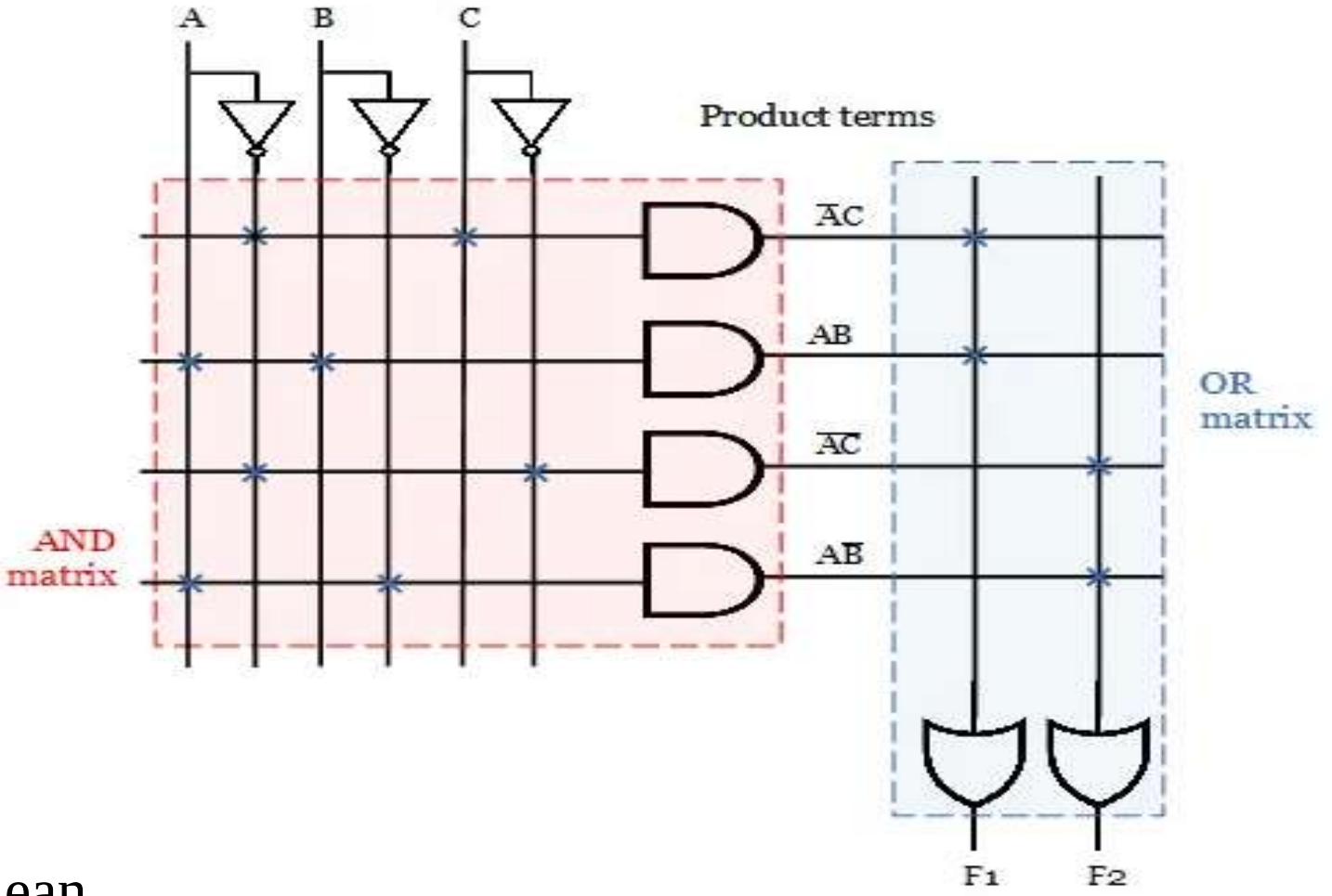
$F_1 = \overline{A}C + A\overline{B}$

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

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

- Thus for the two obtained expressions, the PLA circuit is realized. There are three inputs(A, B, C) and two outputs(F1, F2). In the obtained expressions, there are four product terms and so four AND gate array is used. Two OR gates are used to generate the two boolean fuctions.

The realization of the given Boolean function is drawn.



Programmable Logic Arrays (PLA)

- In PLA the product terms of the input variables is realized by an AND array and the Ol array to form the output functions.
- Hence, a PLA implements a sum-of-products expression.
- PLA has “n” input lines and “m” output lines.

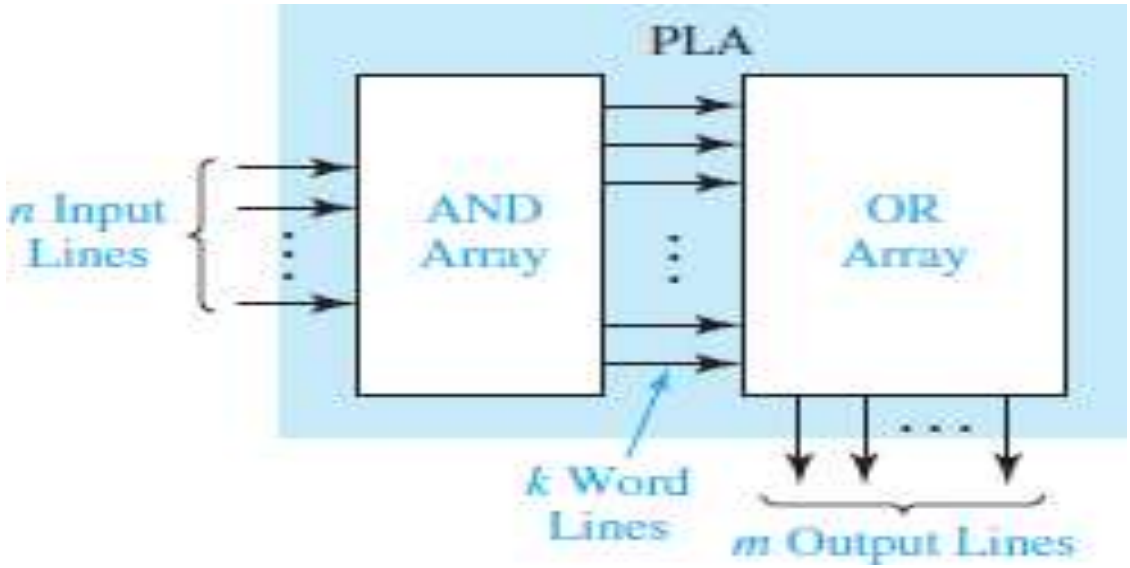


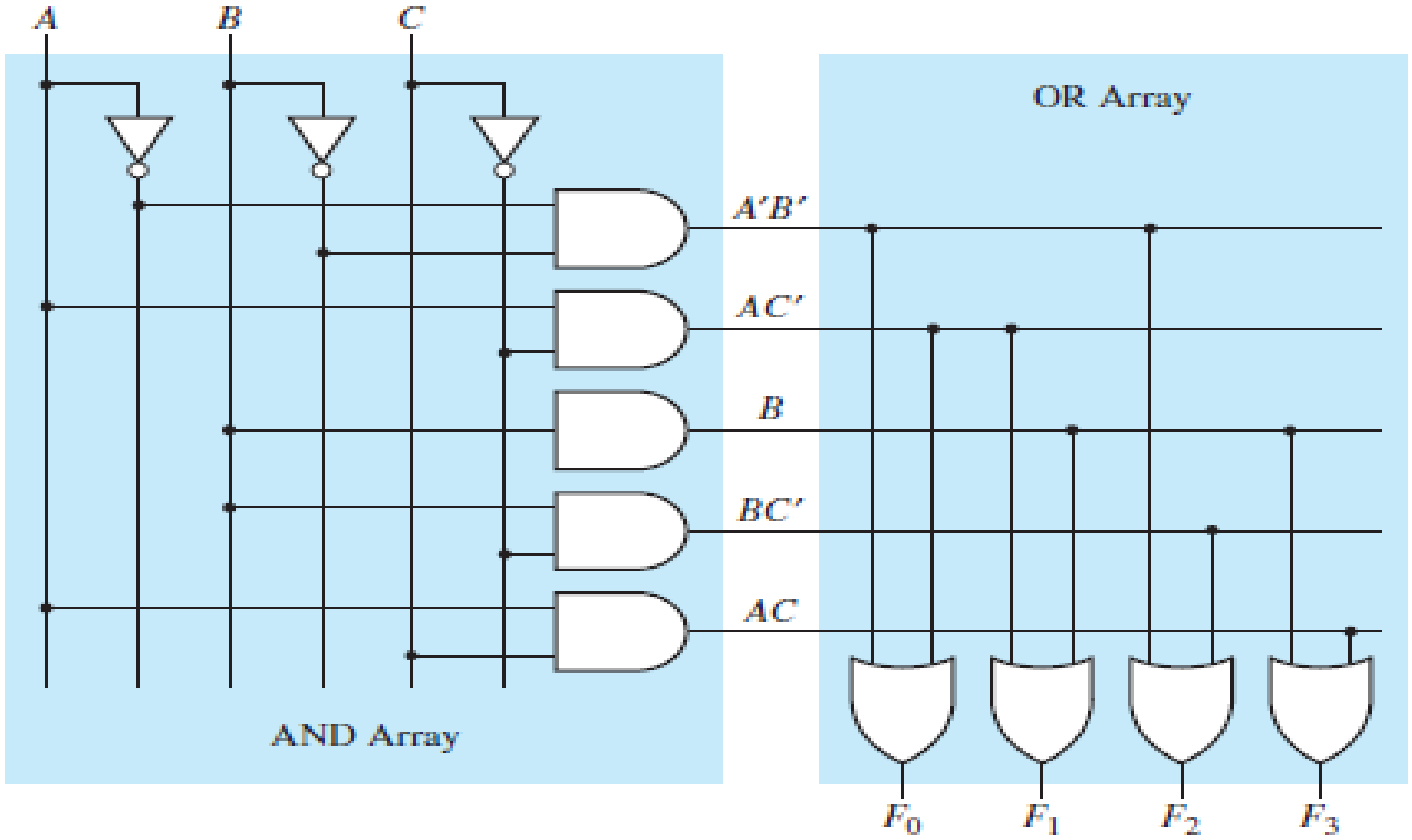
Fig: PLA Structure

Example: *Realize the following functions using PLA*

$$F0 = \sum m\,(0, 1, 4, 6) = A'B' + AC' \quad F1 = \sum m\,(2, 3, 4, 6, 7) = B + AC'$$

$$F2 = \sum m\,(0, 1, 2, 6) = A'B' + BC' \quad F3 = \sum m\,(2, 3, 5, 6, 7) = AC + B$$

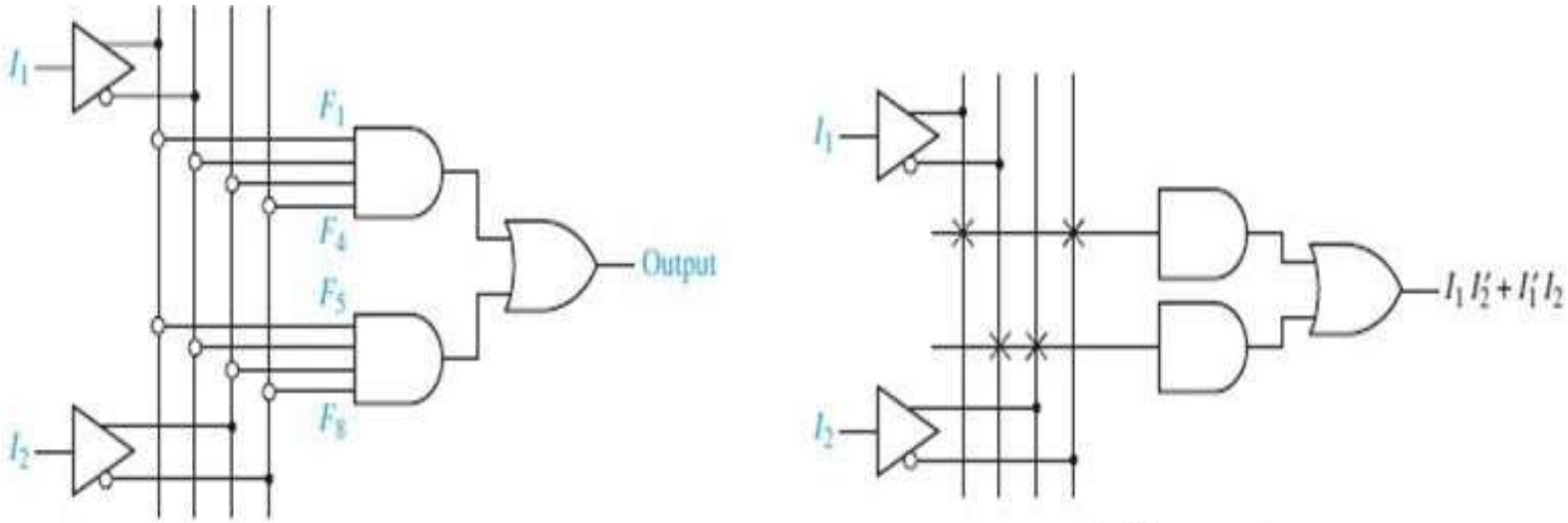
Solution



Programmable Array Logic (PAL)

A *PAL* is a programmable logic array (PLA) in which the AND array is programmable and the OR array is fixed.

Consider the PAL segment of the following Figure (a), used to realize the function $I1I'2 + I'1I2$. The X's in the following Figure (b) indicate that $I1$ *and* $I'2$ lines are connected to the first AND gate, and The $I'1$*and* $I2$ lines are connected to the other gate.



(a) Unprogrammed (b) Programmed

Thank You