

Combinational circuits - Analysis and Design
 procedures - Circuits for Arithmetic operations,
 Code conversions - Decoders and Encoders -
 Multiplexers and Demultiplexers - Introduction
 to HDL - HDL models of combinational
 Circuits.

Combinational Circuits

Digital circuits are classified into 2 categories:

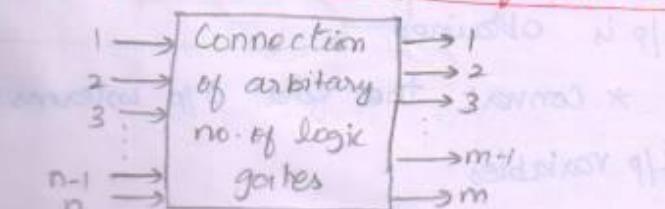
- * Combinational Circuits

- * Sequential Circuits

Combinational Circuits: A combinational circuit consists of an arbitrary number of logic gates which are connected together to form an output for a certain combination of input variables without any feedback.

The o/p at any instant of time depend on the i/p present at that instant of time only.

Block diagram of combinational circuit



Analysis and Design Procedure

The purpose of analysis of any device is to know the function of that device. Basically analysis of a combinational circuit is to obtain the O/P of that circuit whereas circuit design involves obtaining a digital logic circuit from the specified design objective (O/P).

Analysis procedure:- Analysis of a digital ckt results a Boolean fn, a truth table and sometimes an explanation of ckt also.

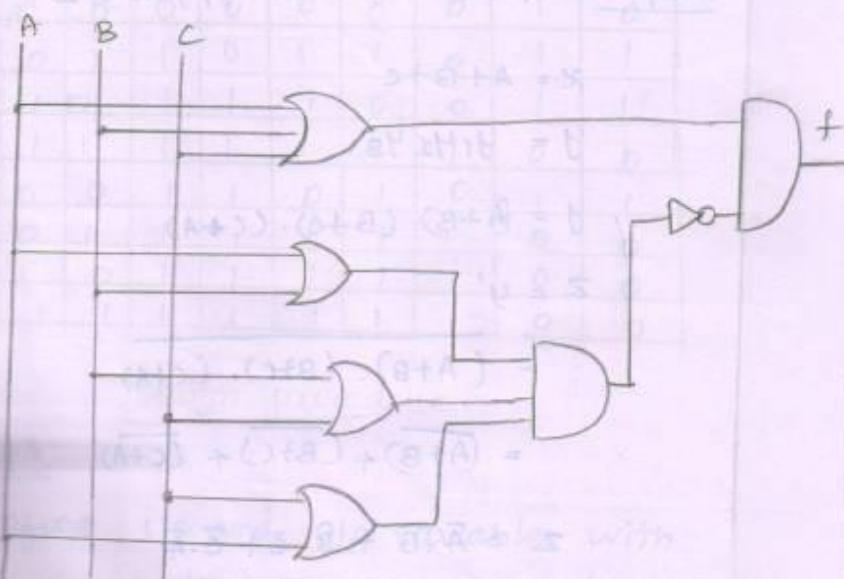
To obtain the Boolean fn of the gn ckt:-

- * Designate the O/P of selected gates with suitable alphanumeric symbols.
- * Find the O/P expression for each selected gate which is connected to i/p.
- * Find the O/P expression for the gate which is connected to previous gates.
- * Repeat this process until the final O/P is obtained
- * Convert the final O/P in terms of i/p variables

To obtain the truth table directly from the given digital circuit:-

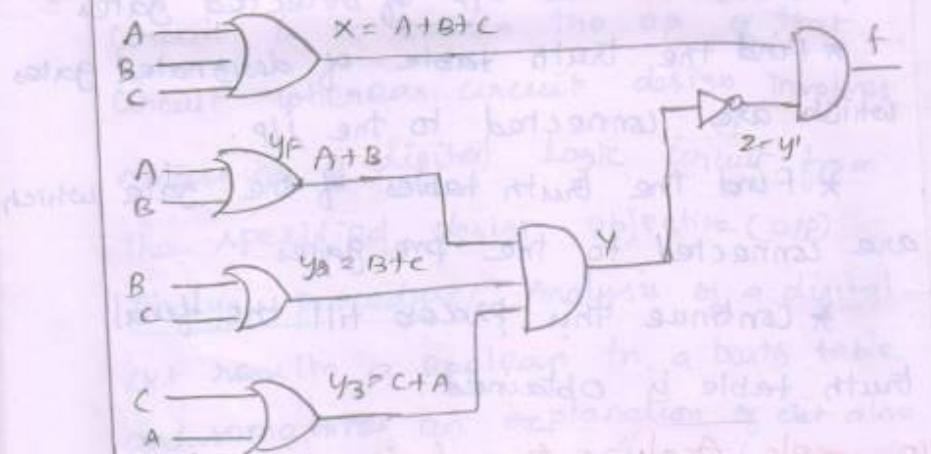
- * List all the binary combinations of i/p variables
- * Designate the o/p of selected gates
- * Find the truth table of designated gates which are connected to the i/p.
- * Find the truth tables of the gate which are connected to the pre. gates
- * Continue this process till the final truth table is obtained.

(Example) Analyze the logic circuit.



Solution:-

Step(1):- Determine the Boolean fn of the
g circuit. Designate all intermediate
gates as 'x', 'y' and 'z'.



Step(2):- Find the o/p x, y , and z .

$$x = A + B + C$$

$$y = y_1 y_2 y_3$$

$$y = (A + B) \cdot (B + C) \cdot (C + A)$$

$$z = y'$$

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

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

$$z = \bar{A} \cdot \bar{B} + \bar{B} \cdot \bar{C} + \bar{C} \cdot \bar{A}$$

Step(3):- Find the final o/p f

$$f = xz$$

Step(4):- x and z are replaced with

$$f = xz$$

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

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

obtain the truth table of the logic!

Step(1):- List all 8 combinations of 3 i/p

Step(2):- Individual truth tables for all designated gates.

Step(3):- Truth table for z is determined

Step(4):- Finally truth table for f is determined as shown

A	B	C	x	y ₁	y ₂	y ₃	y	z	f
0	0	0	0	0	0	0	0	1	0
0	0	1	1	0	1	1	0	1	1
0	1	0	1	1	1	0	0	1	1
0	1	1	1	1	1	1	1	0	0
1	0	0	1	1	0	1	0	1	1
1	0	1	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	0	0

Design procedure

Steps:-

Define i/p and o/p variables with meaningful alpha-numeric symbols

Specify required circuit's fundamental behaviour by means of truth table and

* Simplify required Boolean exp or truth table with the help of any standard technique such as k-map, tabulation or Boolean theorem.

* K-map, tabulation realize the required logic diagram according to the simplified exp.

Example) Design a combinational circuit with 3 i/p A, B, C and one o/p Y. The o/p should be logic 1 when atleast a i/p at logic 1

Step(1) - No of i/p - 3 (A, B, C)

No. of o/p - 1 (Y)

Step(2) - No. of possible combination = 2^3

$$2^3 = 8$$

Inputs			Output
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1

or

standard

From the truth table,

63

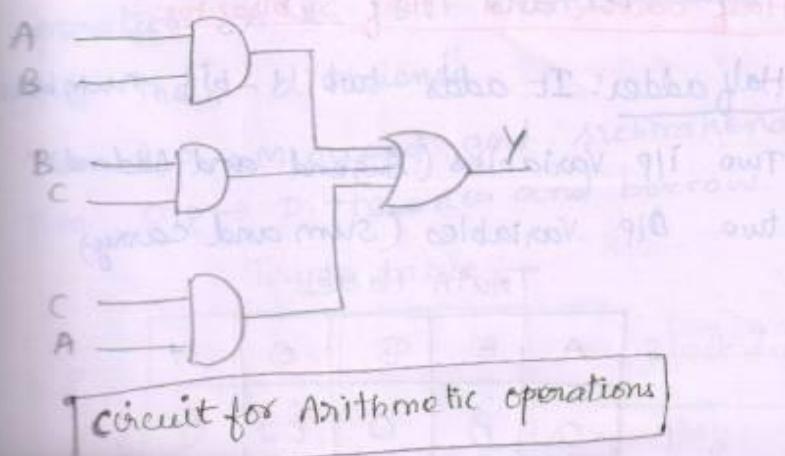
$$Y = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC.$$

Step(3): Desired Boolean fn is simplified using k-map.

		DK			
		00	01	11	10
A	0	0	1	1	0
	1	4	5	7	6

$$Y = AC + AB + BC.$$

Step(4): logic diagram is realized as shown,



Adders/Subtractors:-

Adders/Subtractors perform the basic arithmetic operation of addition and subtraction of 2 binary digits respectively.

Addition	Subtraction
$0+0 = 0$	$1-0=0$
$0+1 = 1$	$1-0=1$ with a produce * borrow of 1
$1+0 = 1$	$1-1=0$
$1+1 = 10$ produce * If minuend bit carry	subtrahend bit then 1 is borrowed from next higher significant bit

Half adder and Half Subtractor

Half adder: It adds two 1-bit numbers.

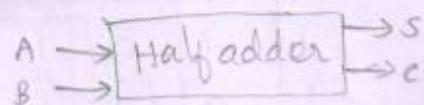
Two 1IP variables (Augend and Addend),

two 0IP Variables (sum and carry)

Truth table

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

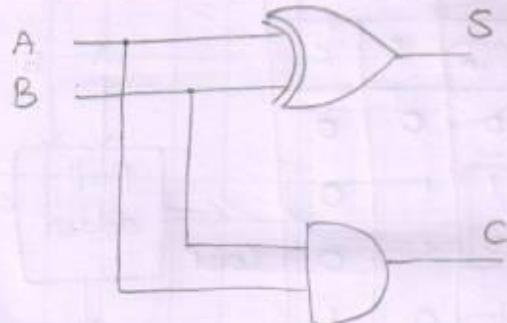
Block Diagram



$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$C = AB$$

Half adder circuit



Half subtractor: It performs subtraction operation on a 1-bit number and gives their difference.

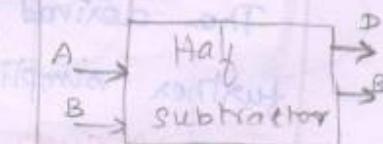
Two I/P \rightarrow Minuend and subtrahend

Two O/P \rightarrow Difference and borrow.

Truth table

A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

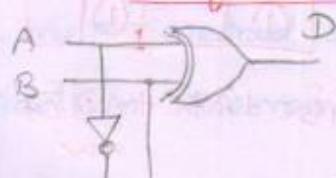
Block diagram



Half adder circuit

$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$D = \bar{A}B + AB$$

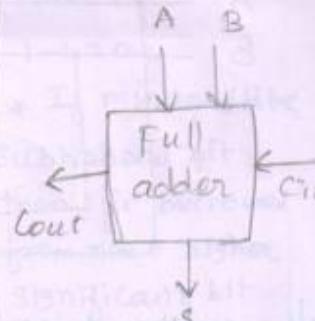


Full adder: It is a combinational circuit that adds three binary bits, and 2 OLP bits.

Truth Table

Inputs			Outputs	
A	B	Cin	S	Cout
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

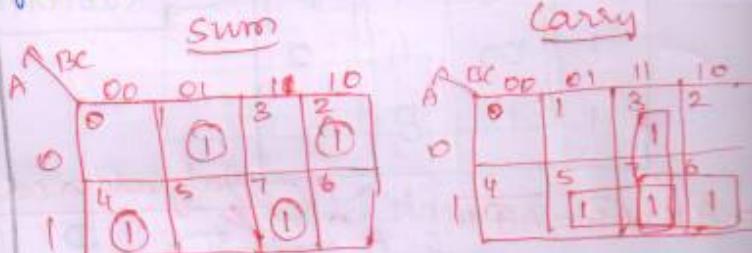
Block diagram



$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Cout = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

The derived Boolean expression can be further simplified using k-map.

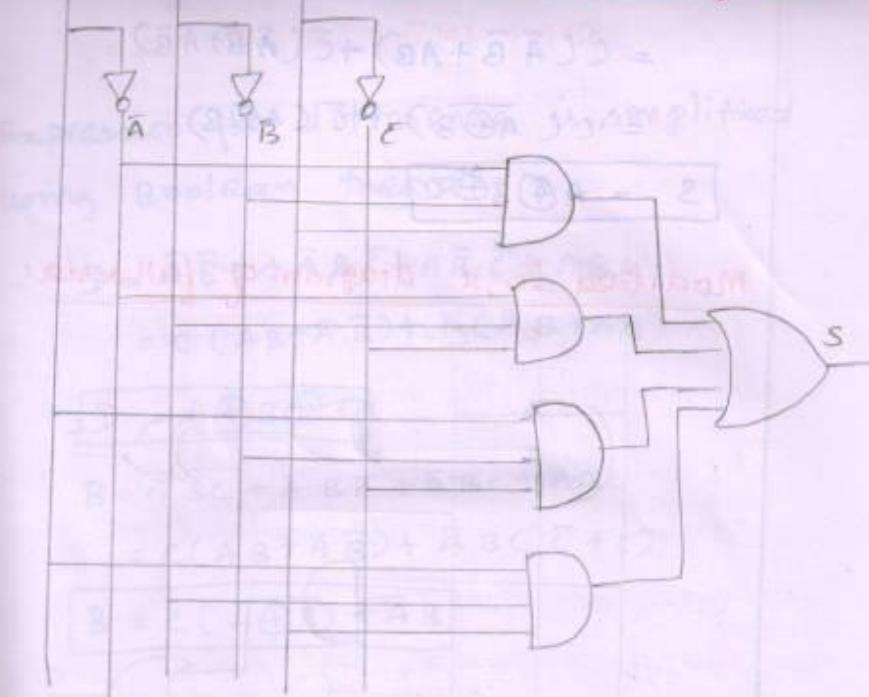


$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

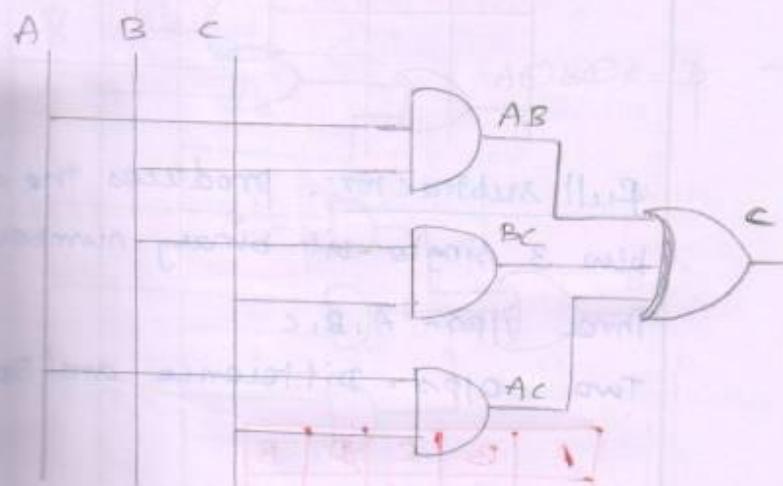
$$C = AB + BC + CA$$

circuit

A B C Logic diagram of sum



Logic diagram of carry



The expression for sum could not be simplified using k-map method. It is simplified using Boolean theorems,

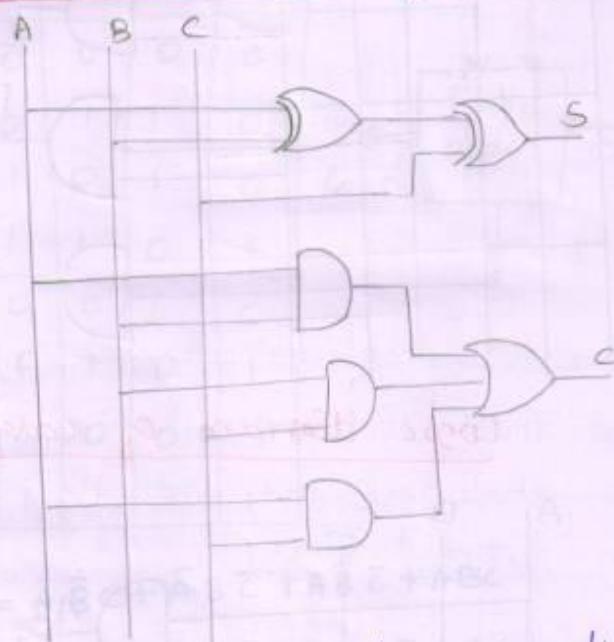
$$S = \bar{A}\bar{B}C + \bar{A}BC\bar{C} + A\bar{B}\bar{C} + ABC$$

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

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

$$\boxed{S = A \oplus B \oplus C}$$

modified logic diagram of full adder:



full subtractor - produces the difference
b/w 3 single-bit binary numbers.

Three O/Ps - A, B, C

Two O/Ps - Difference and Borrow

A	B	C	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

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

$$B = \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC + ABC$$

Expression for difference is simplified using Boolean theorems.

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

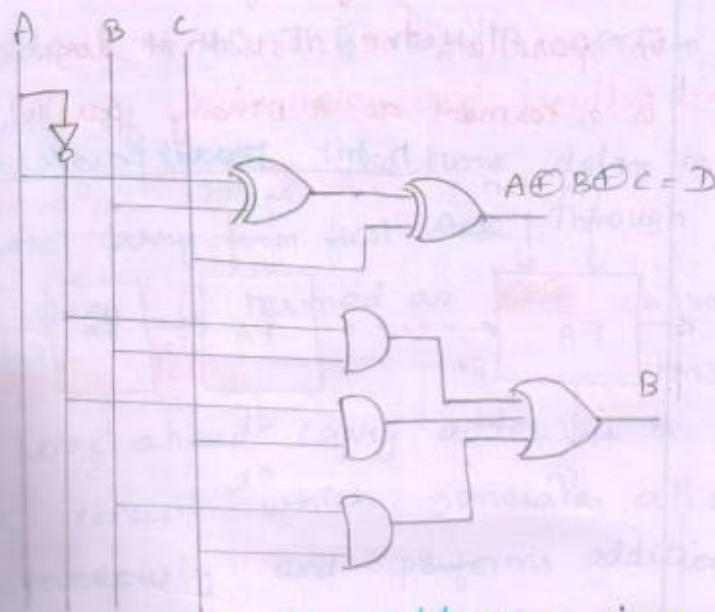
$$= C(A\bar{B} + \bar{A}\bar{B}) + \bar{C}(\bar{A}B + A\bar{B})$$

$$\boxed{D = A \oplus B \oplus C}$$

$$B = \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC + ABC$$

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

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



binary parallel bit adder! A binary parallel adder is a combinational circuit which produces the sum of 2 binary numbers.

consider 2 binary numbers.

1 1 1

1 1 1

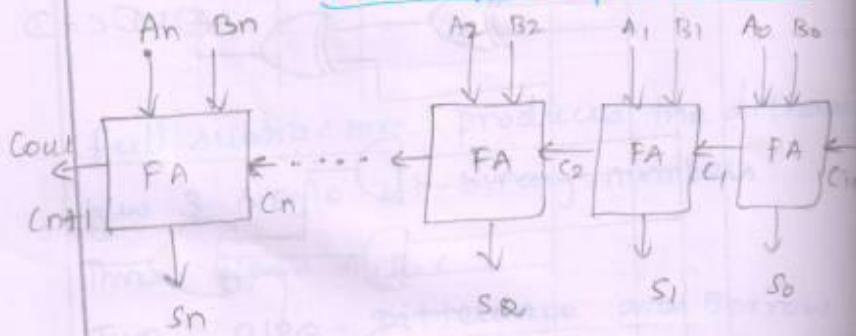
1 0 0

1 0 0

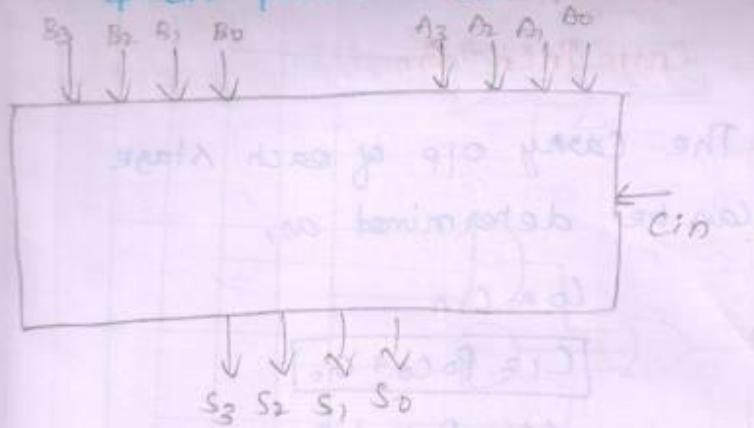
For each column, a full adder is needed for summing the bits. This is achieved by cascading the full adders. The output carry of a full-adder forms the input carry for the next full adder.

When the full adders are connected in parallel, the resultant logic circuit is termed as a binary parallel adder.

n-bit parallel parallel adder



4-bit parallel adder.



Carry-Look ahead adder!

Carry propagation:- In parallel adder, each full adder had to wait for the carry for the next full adder except first full adder (A_0, B_0). This results certain amount of time delay and limits the speed of addition. The time delay to propagate carry from first stage through last stage is termed as ~~propagation delay~~ propagation delay.

Look ahead carry adder is a logic circuit which generates all carry simultaneously and performs addition operation with increased speed.

Scan exp for carry propagate P_n ,
generate G_n , sum S_n , carry C_{n+1} .

$$P_n = A_n \oplus B_n$$

$$S_n = P_n \oplus C_n$$

$$C_{n+1} = P_n C_n + G_n.$$

The carry out of each stage
can be determined as,

$$C_0 = C_{in}$$

$$C_1 = P_0 C_0 + G_0$$

$$C_2 = P_1 C_1 + G_1.$$

sub C_1 in C_2 . eam

$$C_2 = P_1 (P_0 C_0 + G_0) + G_1.$$

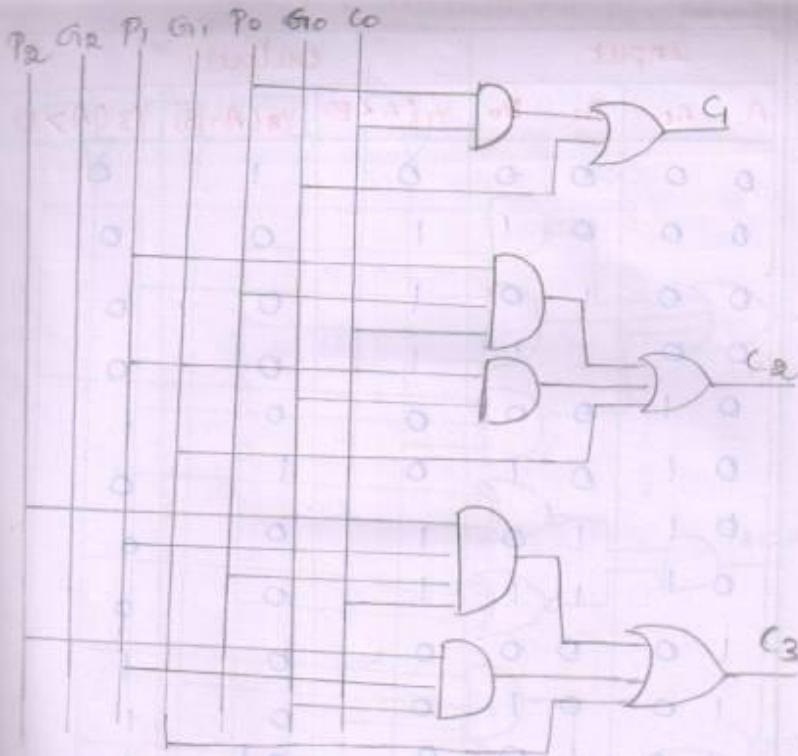
$$C_2 = P_1 P_0 C_0 + P_1 G_0 + G_1.$$

$$C_3 = P_2 C_2 + G_2.$$

sub C_2 in C_3 eam.

$$C_3 = P_2 (P_1 P_0 C_0 + P_1 G_0 + G_1) + G_2$$

$$C_3 = P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2$$

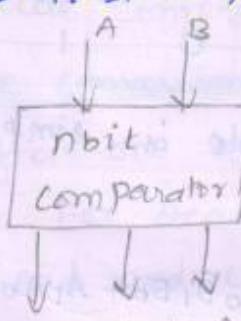


Magnitude comparator: It compares the magnitudes of 2 binary numbers.

Two (ip) \rightarrow A, B

Three o/p \rightarrow A > B, A = B, A < B.

Block diagram



2-bit magnitude comparator:

Two 2-bit numbers A = A₁A₀, B = B₁B₀.

produce three o/p, Y₁, Y₂, and Y₃.

$$Y_1 \Rightarrow A_1A_0 < B_1B_0$$

Input		Output		
A ₁	A ₀	B ₁	B ₀	Y ₁ (A < B) Y ₂ (A = B) Y ₃ (A > B)
0	0	0	0	0 1 0
0	0	0	1	1 0 0
0	0	1	0	0 0 0
0	0	1	1	0 0 0
0	1	0	0	0 0 1
0	1	0	1	1 0 0
0	1	1	0	0 0 0
0	1	1	1	0 0 0
1	0	0	0	0 0 1
1	0	0	1	0 0 1
1	0	1	0	0 1 0
1	0	1	1	0 1 0
1	1	0	0	0 0 0
1	1	0	1	0 0 1
1	1	1	0	0 0 1
1	1	1	1	0 0 0

The truth table is simplified using
the k-map.

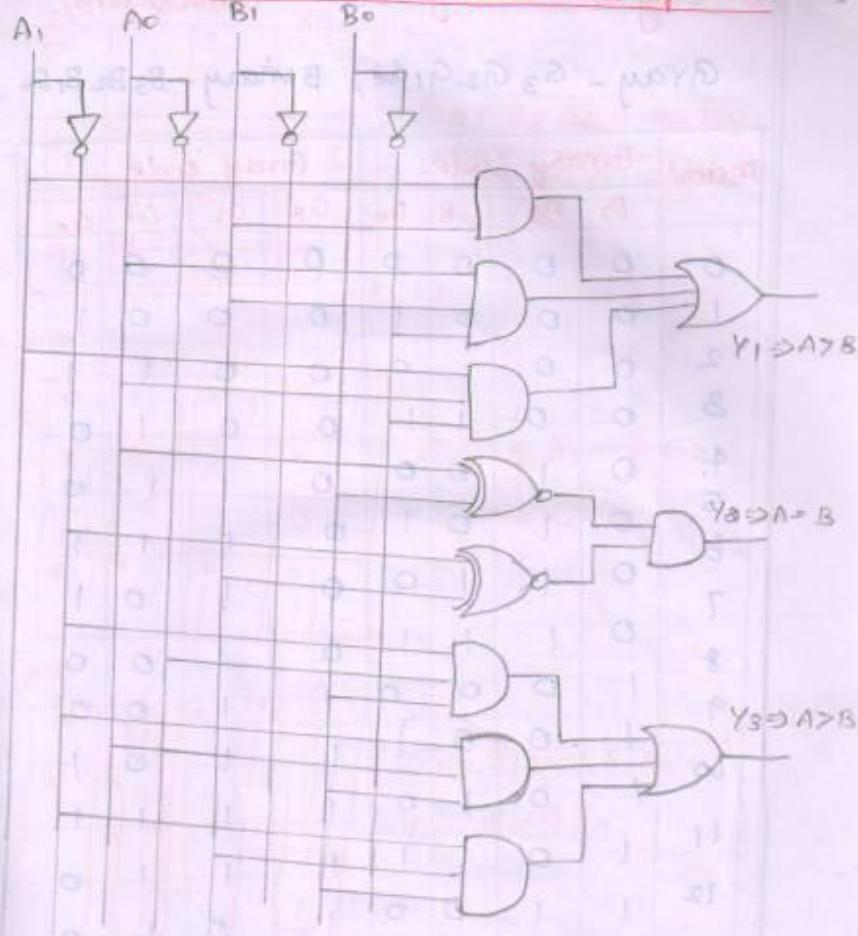
$$Y_1 = A_1 \bar{B}_1 + A_0 \bar{B}_1 B_0 + A_1 A_0 \bar{B}_0$$

$$Y_2 = (A_0 B_0 + \bar{A}_0 \bar{B}_0)(A_1 B_1 + \bar{A}_1 \bar{B}_1)$$

$$= (\overline{A_0 \oplus B_0})(\overline{A_1 \oplus B_1})$$

$$Y_3 = \bar{A}_1 B_1 + \bar{A}_0 \bar{A}_1 B_0 + \bar{A}_0 B_1 B_0$$

Two-bit magnitude comparator



Code Conversions

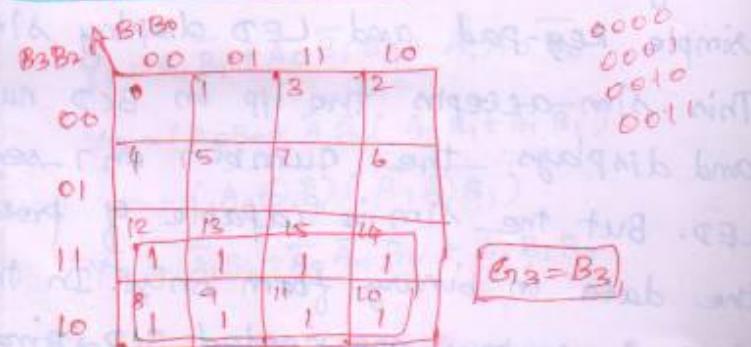
Need for code conversion: It is necessary to convert from one binary code to another binary code. For example, consider a simple key-pad and LED display 81m. This 81m accepts the i/p in BCD number and displays the number on 7-segment LED. But the 81m is capable of processing the data in binary form only. In this case, code conversions are needed sometimes.

Binary to Gray code conversion:

Gray - $G_3 G_2 G_1 G_0$, Binary - $B_3 B_2 B_1 B_0$.

Decimal	Binary Code				Gray code			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	0	0	0
10	1	0	0	1	1	1	0	1
11	1	0	1	0	1	1	1	1
12	1	1	0	0	1	0	1	0
13	1	1	0	1	0	1	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	0	0	0	0

k-map simplification shows



B_3B_2	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$$G_3 = \overline{B_3}B_2 + B_3\overline{B_2}$$

$$G_{10} = B_3 \oplus B_2$$

B_3B_2	00	01	11	10
00	0	1	3	1
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$$G_1 = B_2\overline{B}_1 + \overline{B}_1B_2$$

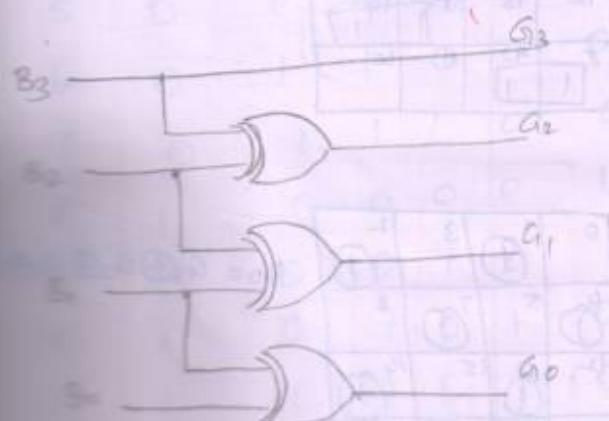
$$G_1 = B_1 \oplus B_2$$

B_3B_2	00	01	11	10
00	0	1	3	1
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$$G_{10} = \overline{B}_1B_0 + B_1\overline{B}_0$$

$$G_{10} = B_0 \oplus B_1$$

Logic diagram :- $G_3 = B_3$, $G_{10} = B_3 \oplus B_2$,
 $G_1 = B_2 \oplus B_1$, $G_{10} = B_1 \oplus B_0$.



Gray to Binary :-

K-map simplification :-

		G ₁ G ₀	00	01	11	10
		G ₃ G ₂	00	01	11	10
			0	1	3	2
00						
01			4	5	7	6
11			12	13	15	14
10			8	9	11	10
			1	1	1	1

$$B_3 = G_3$$

		G ₁ G ₀	00	01	11	10
		G ₃ G ₂	00	01	11	10
			0	1	3	2
00						
01			4	5	7	6
11			12	13	15	14
10			8	9	11	10
			1	1	1	1

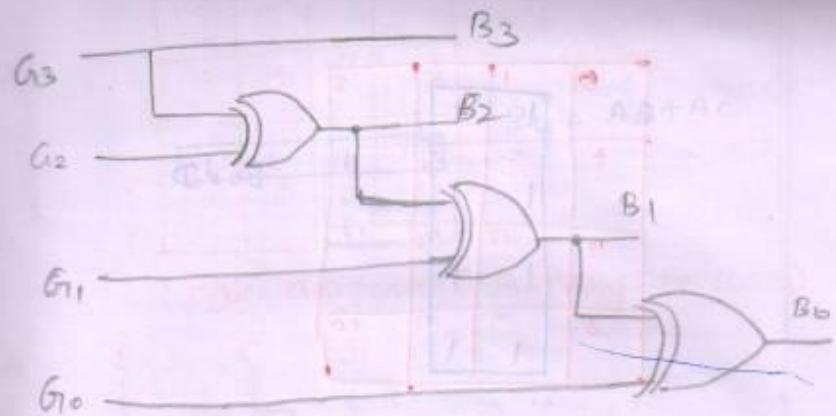
$$B_2 = G_3 \oplus G_2$$

		G ₁ G ₀	00	01	11	10
		G ₃ G ₂	00	01	11	10
			0	1	3	2
00						
01			4	5	7	6
11			12	13	15	14
10			8	9	11	10
			1	1	1	1

$$B_1 = G_3 \oplus G_2 \oplus G_1$$

0	1	3	2
4	5	7	6
12	13	15	14
8	9	11	10
1	1	1	1

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$



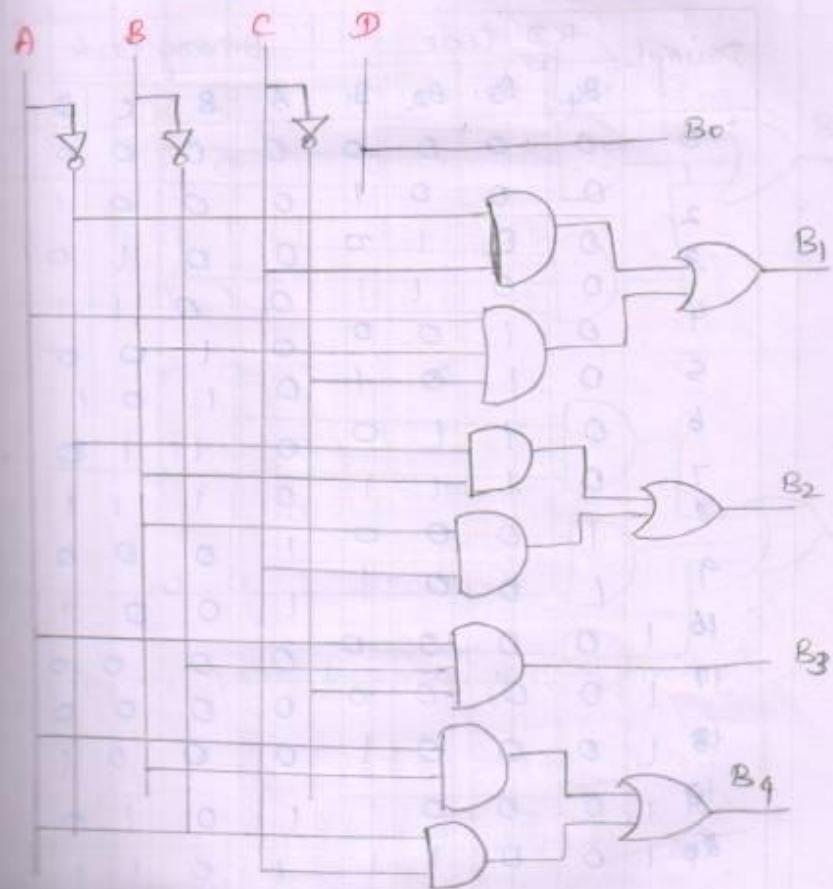
binary to BCD: Converts Binary number into its equivalent BCD Code.

Decimal	Binary code				BCD Code			
	A	B	C	D	B ₄	B ₃	B ₂	B ₁ , B ₀
0	0	0	0	0	0	0	0	0 0 0
1	0	0	0	1	0	0	0	0 0 1
2	0	0	1	0	0	0	0	1 0
3	0	0	1	1	0	0	0	1 1
4	0	1	0	0	0	0	0	1 0 0
5	0	1	0	1	0	0	0	1 0 1
6	0	1	1	0	0	0	1	1 1 0
7	0	1	1	1	0	0	1	1 1 1
8	1	0	0	0	1	0	0	0 0 0
9	1	0	0	1	1	0	0	0 0 1
10	1	0	1	0	1	0	0	0 0 0
11	1	0	1	1	1	0	0	0 0 1
12	1	1	0	0	0	0	0	1 0
13	1	1	0	1	0	0	0	1 1

0	1	3	2
4	5	7	6
12	13	15	14
1	1	1	1

$$B_4 = AB + AC$$

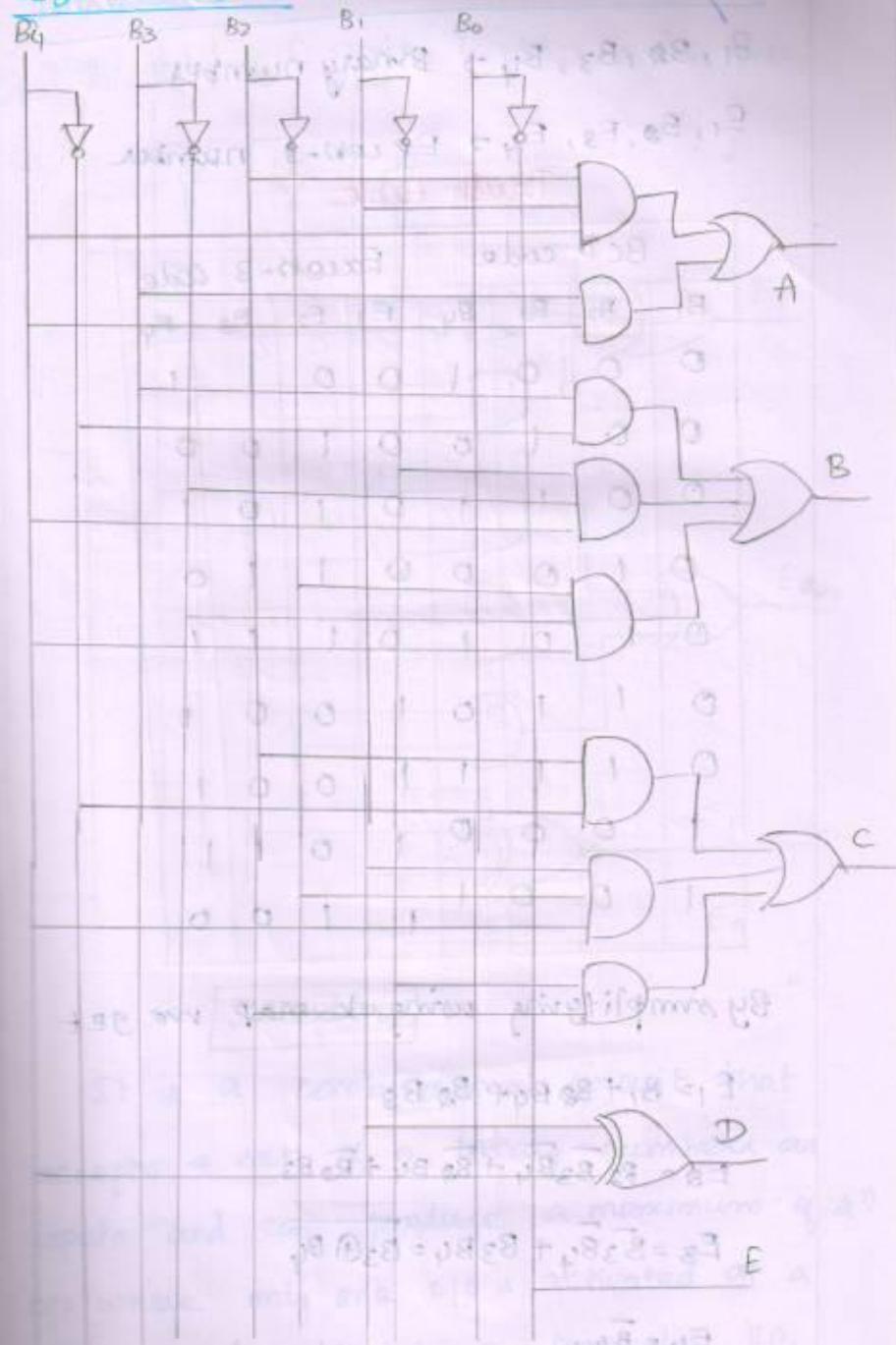
Logic diagram (Binary to BCD)



BCD to Binary conversion:- It is known that the binary and BCD digit are the same from decimal numbers 0 through 9. Hence, a minimum of five I/P Variables are required to design BCD to Binary converter.

Decimal	BCD Code				Binary code			
	B ₄	B ₃	B ₂	B ₁	A	B	C	D
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	0	1	0	1
6	0	1	1	0	0	1	1	0
7	0	1	1	1	0	1	1	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	1	0	0	1
10	1	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0
12	1	0	0	1	0	0	0	1
13	1	0	0	1	1	0	1	0
14	1	0	0	1	1	0	1	1
15	1	0	0	1	1	1	0	0

Logic diagram:-



BCD to Excess-3 code conversions

$B_1, B_2, B_3, B_4 \rightarrow$ Binary number

$E_1, E_2, E_3, E_4 \rightarrow$ Excess-3 number

Truth table

BCD code				Excess-3 code			
B_1	B_2	B_3	B_4	E_1	E_2	E_3	E_4
0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	1	0	0	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

By simplifying using k-map, we get

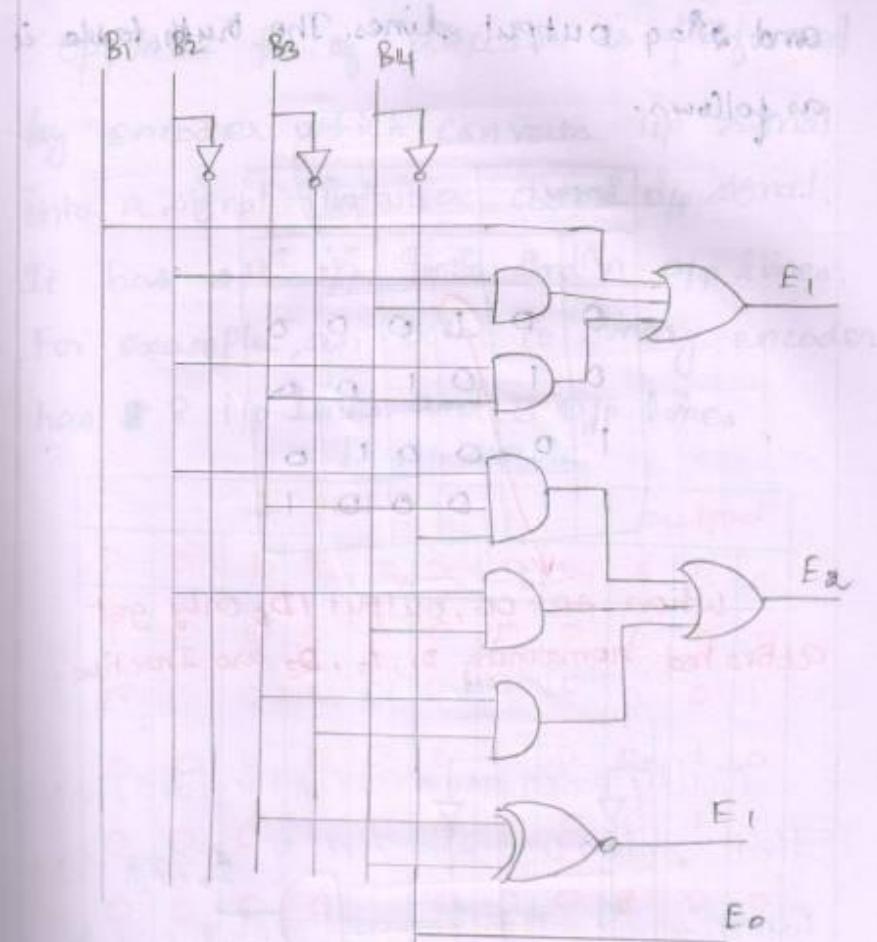
$$E_1 = B_1 + B_2 B_4 + B_2 B_3$$

$$E_2 = B_2 \overline{B_3} \overline{B_4} + \overline{B_2} B_4 + \overline{B_2} B_3$$

$$E_3 = \overline{B_3} \overline{B_4} + B_3 B_4 = \overline{B_3} \oplus B_4$$

$$E_4 = \overline{B_4}$$

Logic diagram for BCD to Excess-3



Decoders

It is a combinational circuit that accepts a set of n binary numbers as inputs and can produce a maximum of 2^n o/p where only one o/p is activated at a time corresponding to a particular i/p. If a decoder has m o/p lines then $m \leq 2^n$.

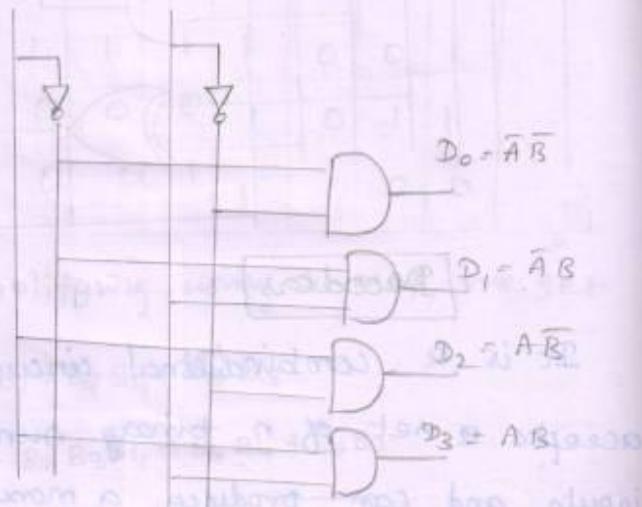
$$\xrightarrow{\text{Decoder}} \left\{ \begin{array}{l} \xrightarrow{\text{Decoder}} \\ \xrightarrow{\text{Decoder}} \end{array} \right\} m \leq 2^n \text{ o/p}$$

8 to 4 line decoder has 8 input lines

and $2^3=8$ output lines. The truth table is as follows:

Input		Output			
A	B	D ₀	D ₁	D ₂	D ₃
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

when AB=00, output D₀ only get activated. Remaining D₁, D₂, D₃ are inactive.



Logic diagram of 8 to 4 decoder

Encoder

87

Opposite fn of decoder is performed by encoder which converts i/p signal into a signal into a coded o/p signal. It has n i/p lines and m o/p lines. For example, an octal to binary encoder has 8 i/p lines and 3 o/p lines.

Truth Table

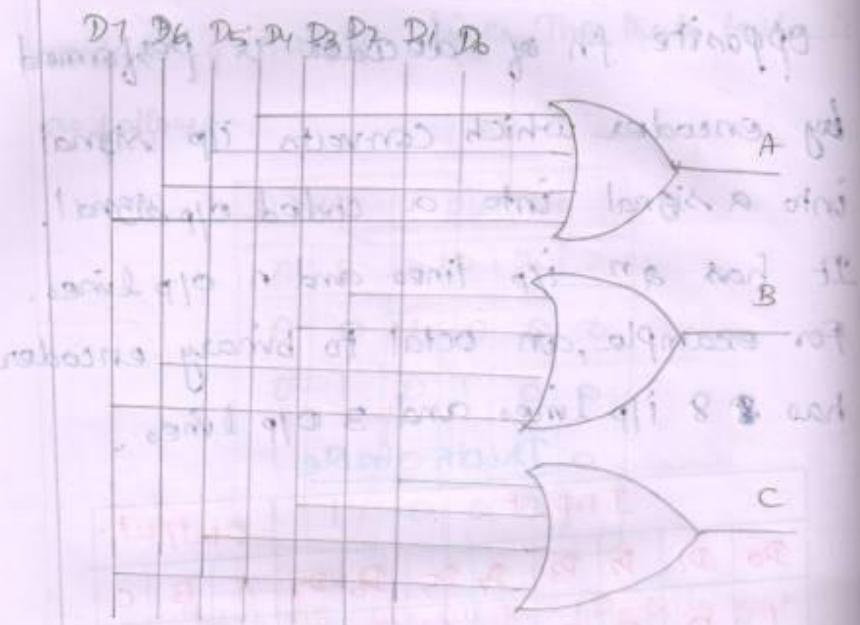
Input								Output		
D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Logical expressions:-

$$A = D_4 + D_5 + D_6 + D_7 \text{ (MSD)}$$

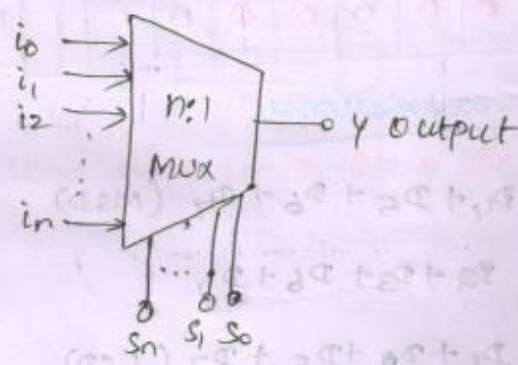
$$B = D_2 + D_3 + D_6 + D_7$$

$$C = D_1 + D_2 + D_5 + D_7 \text{ (LSD)}$$



Multiplexer

Multiplexer is a Combinational Circuit that selects one out of many i/p's normally 2^n inputs and only one o/p. The selection of i/p is done by n select lines.



4:1 MUX: four i/p lines - i_0, i_1, i_2, i_3

two select lines - s_0, s_1

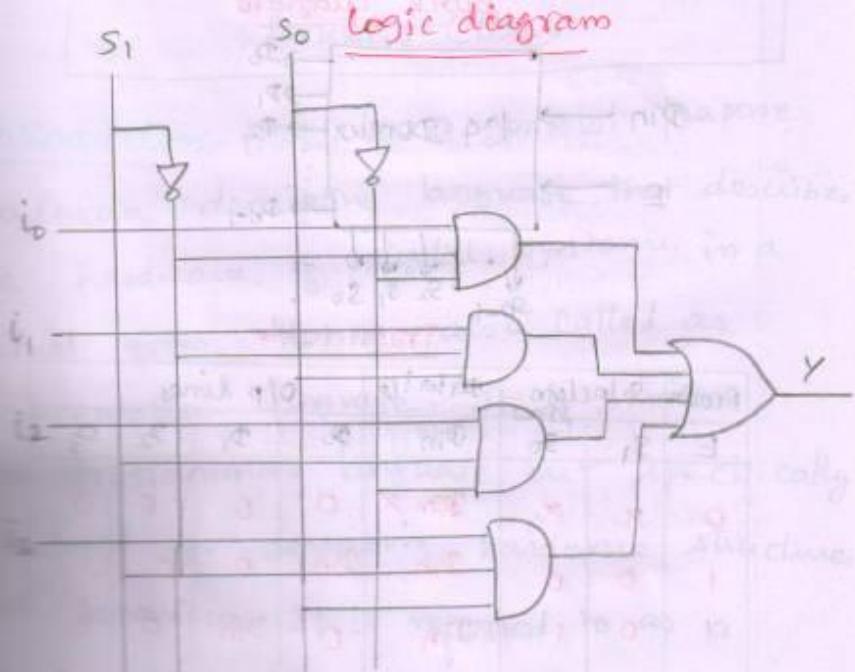
Based on selection information, the MUX selects one i/p and sends it to o/p y .

Truth table

select i/p		output
s_1	s_0	y
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

$$\text{Boolean expression } y = \bar{s}_0 \bar{s}_1 i_0 + \bar{s}_1 s_0 i_1 + s_0 s_1 i_2 + s_1 s_0 i_3$$

logic diagram



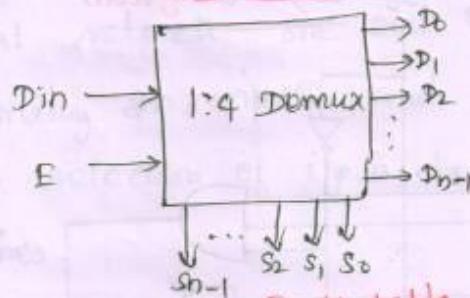
Demultiplexers

A demultiplexer is a combinational circuit that performs the reverse operation of mux. It receives the information on a single i/p and distributes it over many o/p lines. The selection lines specify one of several o/p lines where the information has to be transmitted.

4:1 Demux: 1:4 demux accepts data

on a single i/p variable D_{in} (Data i/p) and distributes over four o/p's, D_0, D_1, D_2, D_3 . Two selection lines S_1 and S_0 .

Block diagram



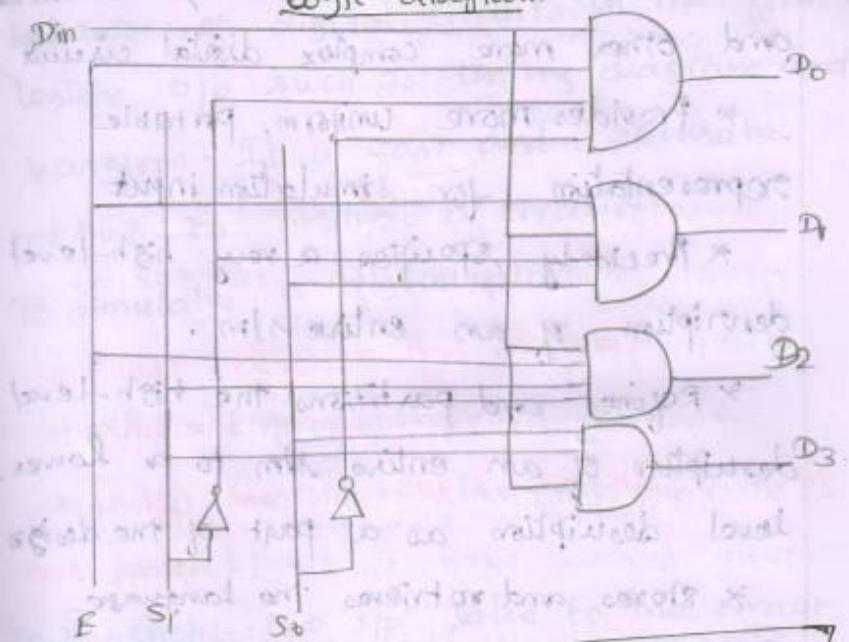
Truth table

Enable	Selection lines		Data i/p	o/p lines			
	S_1	S_0		D_{in}	D_0	D_1	D_2
0	x	x	D_{in}	0	0	0	0
1	0	0	D_{in}	D_{in}	0	0	0
1	0	1	D_{in}	0	D_{in}	0	0
1	1	0	D_{in}	0	0	D_{in}	0

$$D_0 = \bar{S}_1 \bar{S}_0 D_{in}, D_1 = \bar{S}_1 S_0 D_{in}, D_2 = S_1 \bar{S}_0 D_{in}, D_3 = S_1 S_0 D_{in}$$

$$D_3 = S_1 S_0 D_{in}$$

logic diagram



Introduction to Hardware Description language (HDL)

Introduction: HDL is a general purpose hardware descriptive language that describes the hardware of digital systems in a textual form, hence also called as documentation language. It is similar to a programming language, but specifically developed for describing hardware structures and behaviours. It is referred to as a structural description for describing an

Features:-

- * Represents logic diagrams, Boolean exp and other more complex digital circuits
- * Provides more uniform, portable representation for simulation input
- * Precisely specifies a very high-level description of an entire s/m.
- * Refines and partitions the high-level description of an entire s/m to a lower-level description as a part of the design.
- * Stores and retrieves the language

Content:

major design steps:-

* Design entry

* Simulation

* Synthesis and timing verification

* Fault simulation

Design entry:-

It writes description of a hardware functionality. The description could be

* Boolean expression

* Truth table

* Interconnection of gates

Logic simulation:- is the application of simulation software that displays the behavior of digital circuits in the form of legible o/p such as timing diagrams and waveform. It is fast and accurate method to analyze a circuit.

To simulate a digital design

* First describe the design

* Simulate the design

* Verify and check the design using

test bench

Test bench:- the i/p value to the circuit

that tests operation of a design is

known as test bench.

Logic synthesis & timing verification:-

Logic synthesis is a process by which an abstract form of desired circuit (netlist in terms of logic gates) is described in HDL. It generates data base that describes the elements and structure of a circuit which in turn helps to fabricate IC.

Netlist:- physical components and their interconnections

Faming verification: checks each signal

path for any propagation delay and rate.

Confirms the operation of fabricated IC.

Types of HDL:

XVHDL (Very High speed Integrated Circuits HDL), sometimes referred as VHSC HDL.

A verilog HDL has a syntax that describes precisely the legal constructs that can be used in the language.

Starting with verilog- module and some important keywords

module: is the basic building block in verilog. It is denoted by "module"

and "endmodule" that contains HDL text.

Identifier-list of variables.

Input and output indicates whether the variable is i/p or o/p.

Important data type:

Wire - declares interconnection, if any it

Integer - used to manipulate quantities.

reg - computer variable in a memory.

Predefined logic value :-

95

* D, 1, X and Z

* X → uninitialized or unknown value

* Z → high impedance value

Bitwise operations:

* Bitwise NOT ~

* " AND &

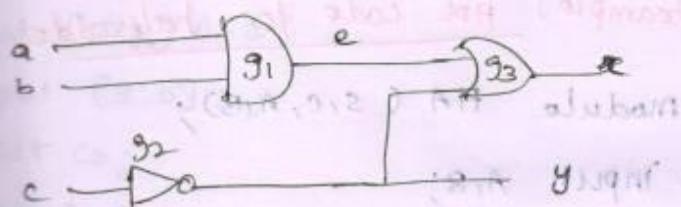
* " OR |

* " XOR ^

* " XNOR ~^ or ^~

* " NOR ~^ or ^~

Example HDL model for simple circuit



module simple_circuit (a,b,c, z,y);

input a,b,c;

output z,y;

wire e

and g1(e,a,b) // first letter o/p

not g2(y,c) and then i/p

or g3(z,e).

endmodule.

HDL for combinational circuits

* Gate level or structural modeling

(circuit is constructed in terms of gates)

* Data flow modeling

(It uses 'assign' keyword followed by predictive o/p and equal sign)

* Behavioral modeling

(describes the behaviour of the circuit,
mostly used for sequential circuits, It
uses the key word 'always' followed by
expression).

(Example) HDL code for half adder

```
module HA ( S,C,A,B );  
    input A,B;  
    output S,C;  
    XOR ( S,A,B );  
    AND ( C,A,B );  
endmodule.
```

HDL code for full adder

97

```
module FA(S, Cout, A, B, Cin);  
    input A, B, Cin;  
    output S, Cout;  
    wire S1, C1, C2;  
    halfadder HA1(S1, C1, A, B);  
    halfadder HA2(S, C2, S1, C1);  
    OR G1(Cout, C2, C1);  
endmodule
```

HDL code for 4 bit adder

```
module 4-bit-adder (S, C4, A, B, C0);  
    input [3:0] A, B;  
    output [3:0] S;  
    input C0;  
    output C4;  
    wire C1, C2, C3;  
    fulladder FA0([S[0], C1, A[0], B[0], C0]);  
    fulladder FA1([S[1], C2, A[1], B[1], C1]);  
    fulladder FA2([S[2], C3, A[2], B[2], C2]);  
    fulladder FA3([S[3], C4, A[3], B[3], C3]);  
endmodule.
```