



CMR INSTITUTE OF TECHNOLOGY		USN <input type="text"/>							
<b>Internal Assessment Test – I</b>									
Sub:	Digital Electronics						Code:	15EC33	
Date:	20 / 09 / 2017	Duration:	90 mins	Max Marks:	50	Sem:	3rd	Branch:	EC/TC
Answer Any FIVE FULL Questions									
							Marks	OBE	
								CO	RBT
1(a)	Define: 1. Literal 2. Canonical SOP 3. Maxterm 4. Combinational Logic 5. Prime Implicant and Essential Prime Implicant						[5]	CO1	L1
(b)	Convert the following Boolean function into: 1. $X=f(a, b, c) = (\bar{a} + b)(b + \bar{c})$ minterm canonical form 2. $Y=f(x, y, z) = x + \bar{x} \bar{z} (y + \bar{z})$ maxterm canonical form						[5]	CO1	L2
2 (a)	Design a three input one output combinational circuit which has output equal to 1 when majority of its inputs are at logic 1 and has output equal to 0 when majority of its inputs are at logic 0.						[7]	CO2	L2
(b)	Explain general approach to combinational logic design.						[3]	CO2	L2

CMR INSTITUTE OF TECHNOLOGY		USN <input type="text"/>							
<b>Internal Assessment Test – I</b>									
Sub:	Digital Electronics						Code:	15EC33	
Date:	20 / 09 / 2017	Duration:	90 mins	Max Marks:	50	Sem:	3rd	Branch:	EC/TC
Answer Any FIVE FULL Questions									
							Marks	OBE	
								CO	RBT
1(a)	Define: 1. Literal 2. Canonical SOP 3. Maxterm 4. Combinational Logic 5. Prime Implicant and Essential Prime Implicant						[5]	CO1	L1
(b)	Convert the following Boolean function into: 1. $X=f(a, b, c) = (\bar{a} + b)(b + \bar{c})$ minterm canonical form 2. $Y=f(x, y, z) = x + \bar{x} \bar{z} (y + \bar{z})$ maxterm canonical form						[5]	CO1	L2
2 (a)	Design a three input one output combinational circuit which has output equal to 1 when majority of its inputs are at logic 1 and has output equal to 0 when majority of its inputs are at logic 0.						[7]	CO2	L2
(b)	Explain general approach to combinational logic design.						[3]	CO2	L2

3	Find the prime implicants and the essential prime implicants of the following Boolean function using K-Map. (a) $Y = \sum (1,3,5,7,8,10,12,13,14) + \sum d (4,6,15)$ (b) $Y = \pi (0,1,4,5,8,9,11) + \pi d (2, 10)$	[5]	CO1	L3
4	Find all the prime implicants and essential prime implicants of the function Y using Quine-McCluskey method and realize the minimal sum using NOR. $Y = \sum (7,9,12,13,14,15) + \sum d (4,11)$	[10]	CO1	L3
5	Explain the working principle of four bit Look Ahead Carry adder with relevant equations and block diagram.	[10]	CO2	L2
6	Design and implement 2-bit magnitude comparator.	[10]	CO1	L2
7 (a)	State any two disadvantages associated with basic encoder circuit	[2]	CO1	L1
(b)	Write a condensed truth table for a 4 to 2 line priority encoder with a valid output, where the highest priority is given to the lowest bit position and obtain the minimal sum expression for the output.	[8]	CO2	L3
8	Implement the following Boolean expression using IC 74138 (a) $A = f(w, x, y, z) = \sum(1,9,11,13)$ obtain POS (b) $B = f(p, q, r, s) = \sum(0,6,12,14)$ obtain SOP	[5 + 5]	CO2	L2

3	Find the prime implicants and the essential prime implicants of the following Boolean function using K-Map. (a) $Y = \sum (1,3,5,7,8,10,12,13,14) + \sum d (4,6,15)$ (b) $Y = \pi (0,1,4,5,8,9,11) + \pi d (2, 10)$	[5]	CO1	L3
4	Find all the prime implicants and essential prime implicants of the function Y using Quine-McCluskey method and realize the minimal sum using NOR. $Y = \sum (7,9,12,13,14,15) + \sum d (4,11)$	[10]	CO1	L3
5	Explain the working principle of four bit Look Ahead Carry adder with relevant equations and block diagram.	[10]	CO2	L2
6	Design and implement 2-bit magnitude comparator.	[10]	CO1	L2
7 (a)	State any two disadvantages associated with basic encoder circuit	[2]	CO1	L1
(b)	Write a condensed truth table for a 4 to 2 line priority encoder with a valid output, where the highest priority is given to the lowest bit position and obtain the minimal sum expression for the output.	[8]	CO2	L3
8	Implement the following Boolean expression using IC 74138 (a) $A = f(w, x, y, z) = \sum(1,9,11,13)$ obtain POS (b) $B = f(p, q, r, s) = \sum(0,6,12,14)$ obtain SOP	[5 + 5]	CO2	L2

## SOLUTIONS TO IAT 1

Define: 1. Literal 2. Canonical SOP 3. Maxterm 4. Combinational Logic

1(a)

5. Prime Implicant and Essential Prime Implicant

5M

Select from below list

**Active high** The "true" status of a switching variable occurs when the variable is a 1, as represented by a positive voltage. Opposite of active low.

**Active low** The "true" status of a switching variable occurs when the variable is a 0, as represented by a low voltage. Opposite of active high.

**Asserted** The "true" condition of a variable. A variable can be asserted as an active high or an active low signal.

**Bubble logic** Another name for mixed logic, where variables are represented by both active high and active low notations in the same logic circuit.

**Canonical** The basic definition means "conforming to a general rule." The "rule" for switching logic equations is that each minterm or maxterm must contain all of the input variables (or their complements) used in the expression.

**Canonical Sum of Products** A sum (OR function) of products (AND function) is a set of product terms ORed together. For the sum of products to be canonical all of the input variables used in the expression must be contained in each individual product term. For example, if X, Y, and Z are Boolean variables then a canonical sum of products could be  $T = X'YZ + XY'Z + XYZ$ .

**Combinational logic** Combinational logic consists of circuits whose outputs are determined by the present combination of inputs independent of previous input combinations. Combinational logic circuits produce outputs that are specified by a set of Boolean equations.

**Deasserted** The "false" condition of a variable. A deasserted variable can be represented by a high or low voltage signal, depending on its active status. See also active high and active low.

**Don't care terms** Minterms or maxterms that are not used in specifying a logical output function. This occurs when the truth table for an output function is incompletely specified. The leftover terms become don't care terms. Because they are don't care terms, they can be assigned a value of 0 or 1.

**Essential prime implicant** A prime implicant that contains one or more minterms or maxterms that are unique; that is, terms not contained in any other prime implicant. See also prime implicant.

**Karnaugh map** A Karnaugh map is a diagram made up of squares that is used to simplify Boolean equations. Each square represents a minterm or maxterm from an equation. The squares are arranged so that only one bi-

PTO

nary digit changes between adjacent squares, providing visualization of variable redundancies. The number of squares in a Karnaugh map is related to the number of variables in an equation by  $2^x = y$ , where  $x$  is the number of problem variables and  $y$  is the number of map squares.

**Literal** A literal is a Boolean variable or its complement. For instance, if  $X$  is a Boolean variable then both  $X$  and  $X'$  are literal.

**Map entered variables** A variable or expression that is entered into a Karnaugh map square that is not a minterm or maxterm. The use of map entered variables (or expressions) allows the size of a Karnaugh map to be reduced for a given number of input variables. See Karnaugh map.

**Maxterm** A term in a Boolean equation that represents a condition where an output variable is a logical 0 in the output function truth table. A maxterm is the complement of a minterm. Maxterms are expressed as a set of input variables ORed together and ANDed with other maxterms. An example maxterm expression would be  $F = (x + y' + z)(x + y' + z')$ .

**Minterm** A term in a Boolean equation that represents a condition where an output variable is a logical 1 in the output function truth table. A minterm is the complement of a maxterm. Minterms are expressed as a set of input variables ANDed together and ORed with other minterms. An example minterm expression would be  $F = x'yz' + x'yz$ .

**Mixed logic** Mixed logic contains both active high and active low signals in one logic circuit. *Bubble logic* is another name for mixed logic. Mixed logic diagrams match logic levels (active high or low) between inputs and outputs so that an active high output is never connected to an active low input, and so on.

**Multiple output functions** Multiple outputs derived from the same set of input variables are said to be *multiple output functions*. The term is used in conjunction with

simplification of multiple outputs, so that common terms can be shared between outputs.

**POS (product of sums)** Maxterms are used to form POS. The arithmetic equivalent of the logical AND is a product and the equivalent of the logical OR is a sum. A POS term is a logical ANDing of ORed terms. The difference between a POS term and a maxterm is that the POS need not be canonical and the maxterm is canonical.

**Prime implicant** A prime implicant is a permitted group of minterms or maxterms. The grouping of terms requires that the group size be an integer power of 2 (2, 4, 8, 16 . . .) and that symmetry exist for each variable in the group. Variable symmetry means that a variable has an equal number of 0s and 1s for all terms in the group. See also *essential prime implicant*.

**Product term** A product term is a literal or the logical product (AND) of multiple literals. For example, if  $X$ ,  $Y$ , and  $Z$  are Boolean variables then a representative product term could be  $X$ , or  $XY$ , or  $X'YZ'$ .

**Quine-McClusky minimization technique** A Boolean minimization technique that uses a series of ordered tables to determine prime and essential prime implicants instead of a single table, as used in Karnaugh maps. The Q-M algorithm is more structured and lends itself to realization using a computer.

**SOP (sum of products)** Minterms are used to form SOP equations. A sum is the arithmetic equivalent of the logical operation OR. A product is the arithmetic equivalent of the logical operation AND. A sum of products is the logical ORing of ANDed terms. The difference between a SOP term and a minterm is that the SOP term is not canonical and the minterm is canonical.

**Sum term** A sum term is a literal or the logical OR of multiple literals. For example, if  $X$ ,  $Y$ , and  $Z$  are Boolean variables then a representative sum term could be  $X$ , or  $X + Y'$  or,  $X + Y' + Z'$ .

Q 1 b. Convert the following Boolean function into:

1.  $X=f(a, b, c) = (\bar{a} + b)(b + \bar{c})$  minterm canonical form

2.  $Y=f(x, y, z) = x + \bar{x}\bar{z}(y + \bar{z})$  maxterm canonical form

5M

1 b) ①  $(\bar{a} + b)(b + \bar{c})$   
 $\Rightarrow \bar{a}b + \bar{a}\bar{c} + bb + b\bar{c}$   
 $\Rightarrow \bar{a}b + b + \bar{a}\bar{c} + b\bar{c}$   
 $= \bar{a}b(c + \bar{c}) + b(a + \bar{a})(c + \bar{c}) + \bar{a}\bar{c}(b + \bar{b}) + b\bar{c}(a + \bar{a})$   
 $= \bar{a}bc + \bar{a}b\bar{c} + (ab + \bar{a}b)(c + \bar{c}) + \bar{a}\bar{c}b + \bar{a}\bar{c}\bar{b}$   
 $+ \bar{a}b\bar{c} + \bar{a}\bar{b}\bar{c} + ab\bar{c} + \bar{a}b\bar{c}$   
 $= \bar{a}bc + \bar{a}b\bar{c} + abc + ab\bar{c} + \bar{a}bc + \bar{a}b\bar{c}$   
 $+ \bar{a}b\bar{c} + \bar{a}\bar{b}\bar{c} + ab\bar{c} + \bar{a}b\bar{c}$   
 $= \Sigma m(0, 3, 4, 7, 6, 3, 2, 2, 0, 6, 2)$   
 $Y = \Sigma m(0, 2, 3, 6, 7) //$

②  $Y = f(x, y, z) = z + \bar{x}\bar{z}(y + \bar{z})$   
 $\therefore (z + \bar{x}\bar{z})(z + y + \bar{z}) \left\{ z + yz = \underline{(z + y)(z + z)} \right\}$   
 $= (z + \bar{x}) (z + \bar{z}) (z + y + \bar{z})$   
 $= (z + \bar{z}) (z + y + \bar{z})$   
 $= (z + \bar{z} + y\bar{y}) (z + y + \bar{z})$   
 $= (z + y + \bar{z}) (z + \bar{y} + \bar{z}) (z + \bar{y} + \bar{z})$   
 $Y = \underline{\underline{\Pi M(1, 3)}}$

Q 2) Design a three input one output combinational circuit which has output equal to 1 when majority of its inputs are at logic 1 and has output equal to 0 when majority of its inputs are at logic 0.

7M

2(a)

Truth table

Cell No	a	b	c	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

K-Map for y

a \ bc	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$y = ab + bc + ca$

Ckt

OR

Design a three-input, one-output minimal two-level gate combinational circuit which has an output equal to 1 when majority of its inputs are at logic 1 and has an output equal to 0 when majority of its inputs are at logic 0.

**Solution:** We need to design a circuit which outputs a 1 when two or more of its three inputs are 1

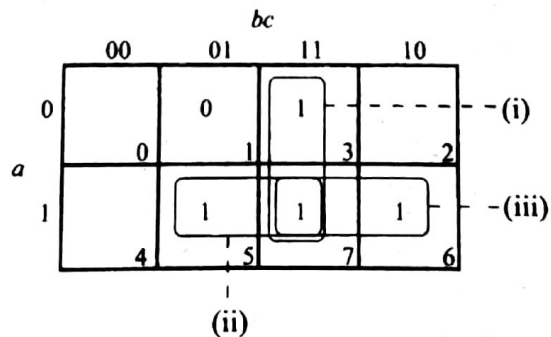
The truth table is shown below.

Cell no.	a	b	c	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

∴

$$f = \Sigma(3, 5, 6, 7)$$

The Karnaugh 1-map of the function is as follows.



Essential 1-cell 3 groups to subcube (i).

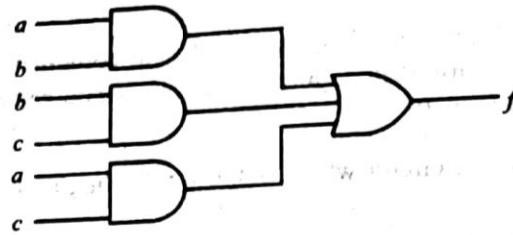
Essential 1-cell 5 groups to subcube (ii).

Essential 1-cell 6 groups to subcube (iii).

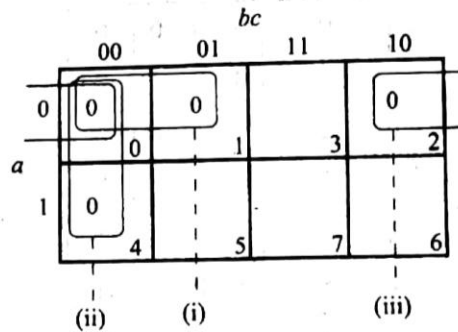
∴ Minimal sum is

$$f = bc + ac + ab$$

The two level minimal sum implementation is shown below.



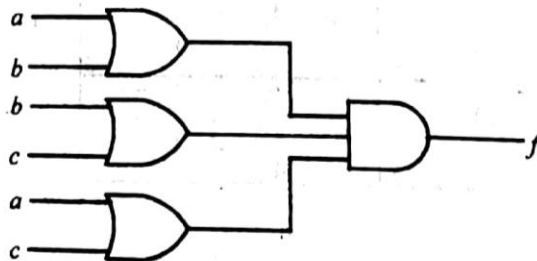
If the cost is taken as the number of gate inputs, the cost of the implementation is 9.  
 Next, let us look at the minimal product implementation by grouping 0-cells.



Essential 0-cell 1 groups to subcube (i).  
 Essential 0-cell 4 groups to subcube (ii).  
 Essential 0-cell 2 groups to subcube (iii).  
 $\therefore$  Minimal product is

$$f = (a + b)(b + c)(a + c)$$

The two level minimal product implementation is shown below.



Here, the cost works out to 9.  
 Thus either one of these would be a minimal circuit if the number of gate inputs is the minimization criterion.



Q 2 b) Explain the general approach to combinational logic design. 3M

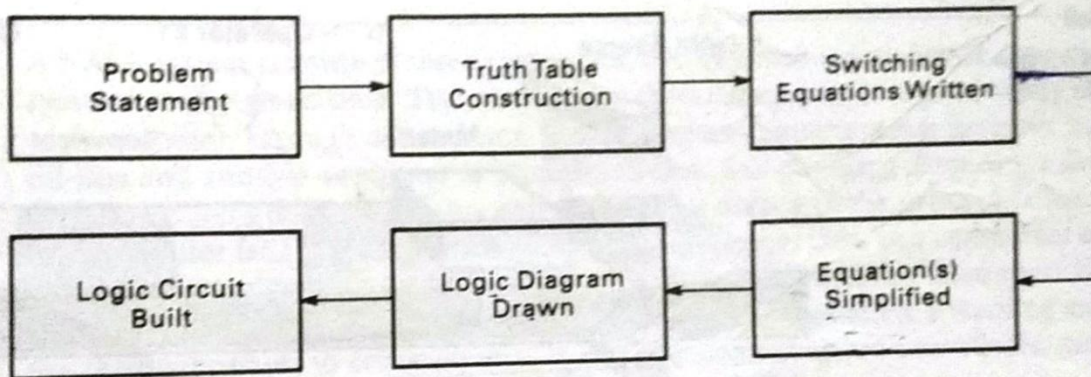
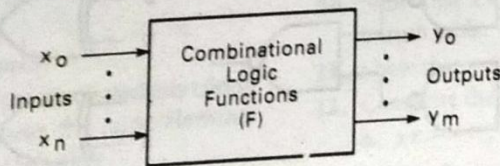
Combinational logic deals with the techniques of "combining" the basic gates, mentioned previously, into circuits that perform some desired function. Examples of useful combinational logic functions are adders, subtractors, decoders, encoders, multipliers, dividers, display drivers, and keyboard encoders.

Logic circuits without feedback from output to the input, constructed from a functionally complete gate set, are said to be *combinational*. Logic circuits that contain no memory (ability to store information) are combinational; those that contain memory, including flip-flops, are said to be *sequential*. We will study sequential logic in later chapters; for now our task is to develop skills in dealing with combinational logic. Combinational logic can be modeled as illustrated in Figure 3.1.

Let  $X$  be the set of all input variables  $\{x_0, x_1, \dots, x_n\}$ , and  $Y$  be the set of all output variables  $\{y_0, y_1, \dots, y_m\}$ . The combinational function,  $F$ , operates on the input variable set  $X$ , to produce the output variable set,  $Y$ . Notice that the output variables  $y_0$  through  $y_m$  are not fed back to the input. The output is related to the input as

$$Y = F(X)$$

Figure 3.1  
Combinational logic model

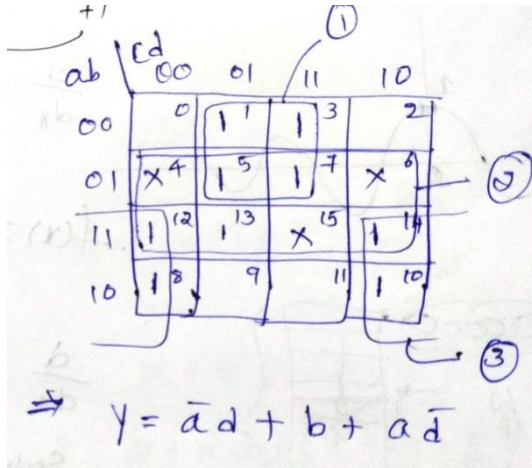


The logic circuits developed in Chapter 2, illustrating various switching properties and theorems, were all combinational. The relationship between the input and output variables can be expressed in equations, logic diagrams, or truth tables. A truth table specifies the input conditions under which the outputs are true or false (1 or 0). Switching equations are then derived from the truth tables and realized (constructed) using gates. This chapter deals with developing truth tables and graphically simplifying equations using several techniques.

Q 3 Find the prime implicants and the essential prime implicants of the following Boolean function using K-Map. 5M + 5M

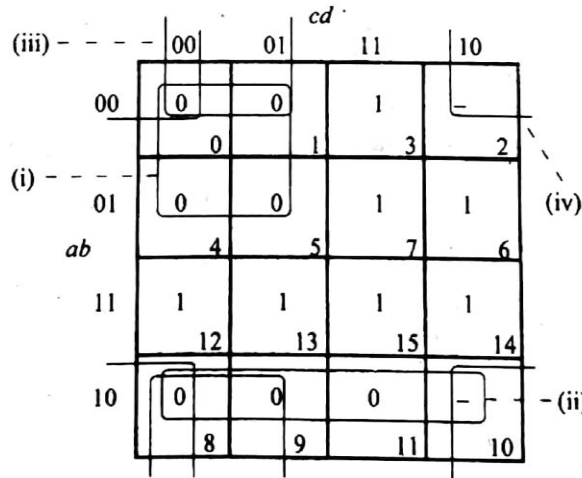
(a)  $Y = \sum (1, 3, 5, 7, 8, 10, 12, 13, 14) + \sum d (4, 6, 15)$

(b)  $Y = \pi (0, 1, 4, 5, 8, 9, 11) + \pi d (2, 10)$



a) Y contains only EPIs.

b.  $f = \Pi (0, 1, 4, 5, 8, 9, 11) + \Pi d (2, 10)$   
Plot the Karnaugh map of the function.



The prime implicants and essential prime implicants are tabulated now.

Subcube no.	Essential 0 -cell no.	Prime implicants	Essential prime implicants
(i)	4 and 5	$a + c$	$a + c$
(ii)	11	$\bar{a} + b$	$\bar{a} + b$
(iii)	none	$b + c$	-
(iv)	none	$b + d$	-

Q 4) Find all the prime implicants and essential prime implicants of the function Y using Quine-McCluskey method and realize the minimal sum using NOR. **10M**

$$Y = \sum (7, 9, 12, 13, 14, 15) + \sum d (4, 11)$$

Find all the prime implicants of the function

$$f(a, b, c, d) = \sum (7, 9, 12, 13, 14, 15) + \sum d (4, 11)$$

using Quine-McCluskey algorithm

**Solution:** We saw in Section 1.5.5 that don't care cells are considered as 1-cells to determine prime implicants. The given function thus becomes

$$f(a, b, c, d) = \sum (4, 7, 9, 11, 12, 13, 14, 15)$$

**Step 1:** Represent each minterms in its 1/0 notation.

No.	Minterm	1/0 notation	Index
7	$\bar{a}bcd$	0111	3
9	$a\bar{b}\bar{c}d$	1001	2
12	$ab\bar{c}\bar{d}$	1100	2
13	$ab\bar{c}d$	1101	3
14	$abc\bar{d}$	1110	3
15	$abcd$	1111	4
4	$\bar{a}\bar{b}\bar{c}\bar{d}$	0100	1
11	$a\bar{b}cd$	1011	3

**Step 2:** List the minterms in increasing order of their index.

4	0	1	0	0	} index 1
9	1	0	0	1	
12	1	1	0	0	} index 2
7	0	1	1	1	
11	1	0	1	1	} index 3
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	index 4

Fig. 2.9 Steps 2 and 3 of Quine-McCluskey algorithm

All further steps are shown in Fig. 2.10. The minterms combining at various iterations are shown with an offset in the '✓' marks

4	0	1	0	0	✓	(4, 12)	-	1	0	0		(9, 11) (13, 15)	1	-	-	1
9	1	0	0	1	✓	(9, 11)	1	0	-	1	✓	(12, 13) (14, 15)	1	1	-	-
12	1	1	0	0	✓	(9, 13)	1	-	0	1	✓					
7	0	1	1	1	✓	(12, 13)	1	1	0	-	✓					
11	1	0	1	1	✓	(12, 14)	1	1	-	0	✓					
13	1	1	0	1	✓	(7, 15)	-	1	1	1						
14	1	1	1	0	✓	(11, 15)	1	-	1	1	✓					
15	1	1	1	1	✓	(13, 15)	1	1	-	1	✓					
						(14, 15)	1	1	1	-	✓					

Fig. 2.10 The Quine-McCluskey algorithm

The terms without a ✓ besides them constitute the prime implicants.  
The prime implicants are

$$b\bar{c}\bar{d}, bcd, ad \text{ and } ab$$

$$Y = b\bar{c}\bar{d} + bcd + ad + ab$$

**Implement Y equation using only NOR gates.**

Q 5) Explain the working principle of four bit Look Ahead Carry adder with relevant equations and block diagram. **10M**

#### **4.1.5 LOOK AHEAD CARRY ADDER**

The parallel adders and subtractors seen earlier are essentially ripple configurations. The effect of propagation delay would be more and more prominent as the number of bits increase. The carry at any given stage would be available only after the carry of the previous stage has been generated. The carry generated by the full adder in the least significant bit stage must propagate through all the intermediate adders till it reaches the most significant bit adder. Observe from Equation 4.1 and 4.2 that apart from the input bits to be added, the sum and carry out of any stage depends on the carry output of the previous stage. The carry out of the previous stage depends on the carry out of the stage prior to that and so on. If we could ensure that the sum and carry output of any stage is not dependent on the results of any previous stages, then the ripple effect is eliminated and speed of addition remarkably increases.

Consider the carry Equation 4.2

$$\begin{aligned} c_{i+1} &= a_i b_i + b_i c_i + a_i c_i \\ c_{i+1} &= a_i b_i + c_i (a_i + b_i) \end{aligned} \quad (4.5)$$

The term  $a_i b_i$  relates to the carry formed at the  $i^{\text{th}}$  stage and is referred to as the carry generate function  $g_i$ .

*i.e.*, 
$$g_i = a_i b_i \tag{4.6}$$

The term  $(a_i + b_i)$  relates to the carry  $c_i$  generated at the previous stage and thus  $(a_i + b_i)$  is referred to as the carry propagate function  $p_i$ .

*i.e.*, 
$$p_i = a_i + b_i \tag{4.7}$$

Observe that both  $p_i$  and  $g_i$  are functions of only the parallel inputs  $a_i$  and  $b_i$ . Comparing Equations 4.5, 4.6 and 4.7 we can write

$$c_{i+1} = g_i + p_i c_i \tag{4.8}$$

Now, let us look at the carry generated at every stage of the 4-bit parallel adder. Setting  $i = 0$  in Equation 4.8, we get

$$c_1 = g_0 + p_0 c_0 \tag{4.9}$$

and setting  $i = 1$ , in Equation 4.8, we get

$$c_2 = g_1 + p_1 c_1$$

substitute for  $c_1$  from Equation 4.9

$$\begin{aligned} \therefore c_2 &= g_1 + p_1 (g_0 + p_0 c_0) \\ c_2 &= g_1 + p_1 g_0 + p_1 p_0 c_0 \end{aligned} \tag{4.10}$$

According to Equations 4.9 and 4.10,  $c_1$  and  $c_2$  are functions of only the parallel inputs and  $c_0$ . Similarly,

$$c_3 = g_2 + p_2 c_2$$

Substitute for  $c_2$  from Equation 4.10

$$\begin{aligned} c_3 &= g_2 + p_2 (g_1 + p_1 g_0 + p_1 p_0 c_0) \\ c_3 &= g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0 \end{aligned} \tag{4.11}$$

and

$$\begin{aligned} c_3 &= g_2 + p_2 (g_1 + p_1 g_0 + p_1 p_0 c_0) \\ c_4 &= g_3 + p_3 c_3 \\ &= g_3 + p_3 (g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0) \\ c_4 &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0 \end{aligned} \tag{4.12}$$

and so on.

Equation 4.11 and 4.12 also indicate that  $c_3$  and  $c_4$  are functions of only the parallel inputs and  $c_0$ .

The Boolean expression for  $c_1, c_2, c_3$  etc., can themselves be implemented using gates and the carry required at each stage of the parallel adder can be made available simultaneously, thereby increasing the speed of addition. The look ahead carry generators are available in IC form in

several bit sizes. A four-bit fast parallel adder with lookahead carry is shown in Fig. 4.19. The lookahead carry block is an implementation of Equations 4.9, 4.10, 4.11 and 4.12 whereas the adder blocks are an implementation of Equations 4.1, 4.6 and 4.7.

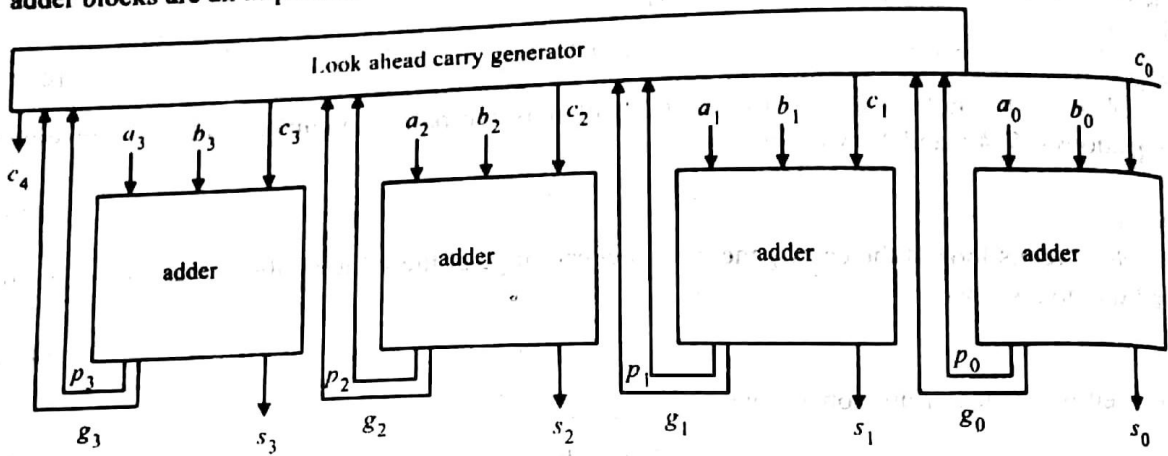


Fig. 4.19 Four-bit parallel fast look ahead carry adder

The implementation of the adder blocks of Fig. 4.19 is shown in Fig. 4.20.

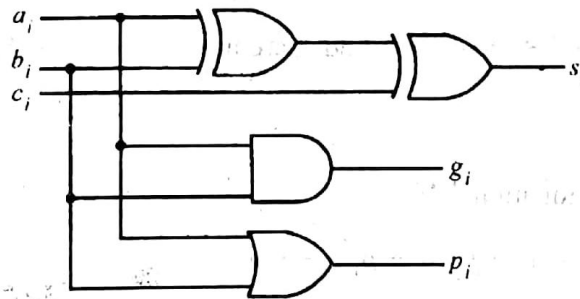
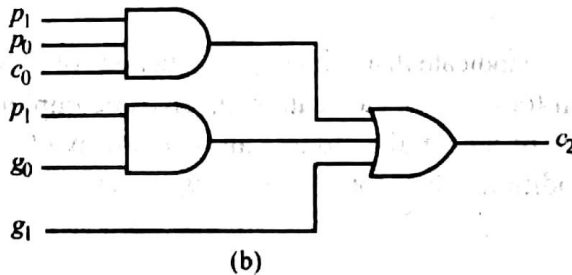
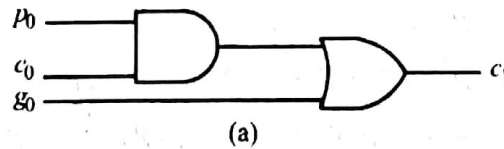
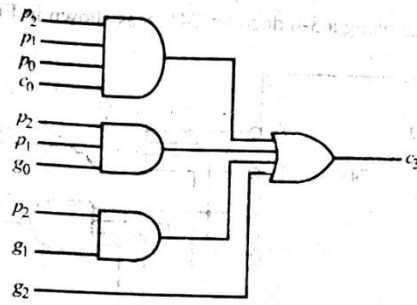


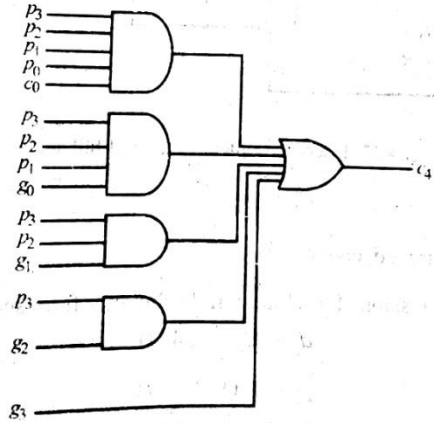
Fig. 4.20 Adder block implementation

The four-bit lookahead carry generator can be implemented as shown in Fig. 4.21.





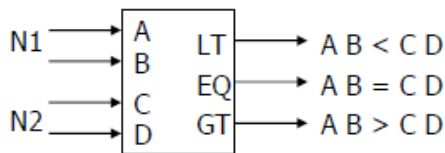
(c)



(d)

**Fig. 4.21** Four-bit lookahead carry generator

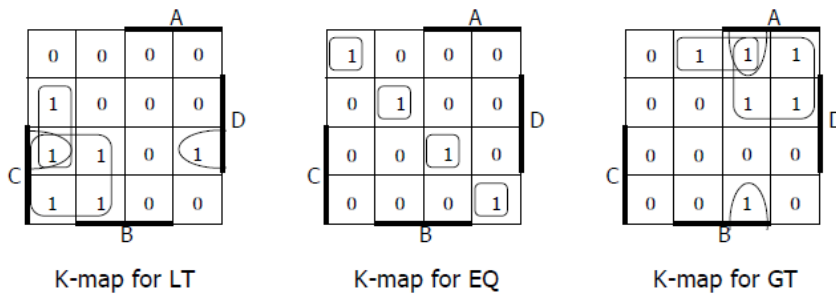
**Q 6. Design and implement 2-bit magnitude comparator. 10M**



block diagram  
and  
truth table

A	B	C	D	LT	EQ	GT
0	0	0	0	0	1	0
		0	1	1	0	0
		1	0	1	0	0
		1	1	1	0	0
0	1	0	0	0	0	1
		0	1	0	1	0
		1	0	1	0	0
		1	1	1	0	0
1	0	0	0	0	0	1
		0	1	0	0	1
		1	0	0	1	0
		1	1	1	0	0
1	1	0	0	0	0	1
		0	1	0	0	1
		1	0	0	0	1
		1	1	0	1	0

we'll need a 4-variable Karnaugh map  
for each of the 3 output functions



$$LT = A' B' D + A' C + B' C D$$

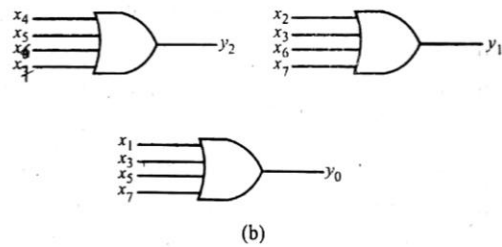
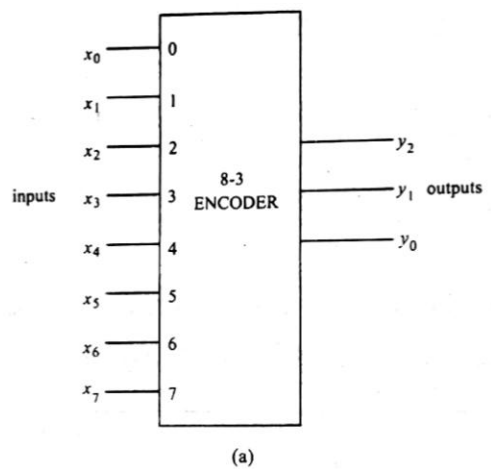
$$EQ = A' B' C' D' + A' B C' D + A B C D + A B' C D' = (A \text{ xnor } C) \cdot (B \text{ xnor } D)$$

$$GT = B C' D' + A C' + A B D'$$

LT and GT are similar (flip A/C and B/D)

Draw the Logic Diagram for LT, EQ and GT.

Q 7. State any two disadvantages associated with basic encoder circuit (Consider 4:2 or 8:3 decoders as example). 2M



**Fig. 3.34** (a) Block diagram of a 8 to 3 line encoder (b) Implementation of a 8 to 3 line encoder

There are some problems associated with this implementation. It is possible that when more than one inputs are at logic 1 there may be an error in the output code. For example, if  $x_3 = x_4 = 1$ , then  $y_2 y_1 y_0 = 111$ , whereas this output should have resulted for only  $x_7 = 1$ . The other problem is that the output is 000 when all the inputs are at logic 0 as well as when  $x_0 = 1$ . The schematic in Fig. 3.34 does not distinguish between these two conditions of inputs.

These problems can be overcome by the priority encoder with a validity indicator. The truth table of a 8 to 3 line priority encoder is shown in Fig. 3.35.



Q 7b) Write a condensed truth table for a 4 to 2 line priority encoder with a valid output, where the highest priority is given to the lowest bit position and obtain the minimal sum expression for the output. 8M

Cell no.	$x_0$	$x_1$	$x_2$	$x_3$	$y_1$	$y_0$	Valid
0	0	0	0	0	0	0	0
8, 9, 10, 11, 12, 13, 14, 15	1	×	×	×	0	0	1
4, 5, 6, 7	0	1	×	×	0	1	1
2, 3	0	0	1	×	1	0	1
1	0	0	0	1	1	1	1

The Karnaugh maps for the outputs are shown below.

		$x_2 x_3$			
		00	01	11	10
00	0	1	1	1	
	0	1	3	2	
01	0	0	0	0	
	4	5	7	6	
11	0	0	0	0	
	12	13	15	14	
10	0	0	0	0	
	8	9	11	10	

$y_1$ -map

$$y_1 = \bar{x}_0 \bar{x}_1 x_2 + \bar{x}_0 \bar{x}_1 x_3$$

		$x_2 x_3$			
		00	01	11	10
00	0	1	0	0	
	0	1	3	2	
01	1	1	1	1	
	4	5	7	6	
11	0	0	0	0	
	12	13	15	14	
10	0	0	0	0	
	8	9	11	10	

$y_0$ -map

$$y_0 = \bar{x}_0 x_1 + \bar{x}_0 \bar{x}_2 x_3$$

		$x_2 x_3$			
		00	01	11	10
00	0	1	1	1	
	0	1	3	2	
01	1	1	1	1	
	4	5	7	6	
11	1	1	1	1	
	12	13	15	14	
10	1	1	1	1	
	8	9	11	10	

Valid-map

$$\text{Valid} = x_0 + x_1 + x_2 + x_3$$

Q 8. Implement the following Boolean expression using IC 74138

5M + 5M

(a)  $A = f(w, x, y, z) = \sum(1,9,11,13)$  obtain POS

(b)  $B = f(p, q, r, s) = \sum(0,6,12,14)$  obtain SOP

