Sub:	**Digital Design and Computer Organization Answer Key**					Sub Code:	**BCS302 /BAD302**	Branch:		**AINDS/CS(DS)**
Date:	**7/11/2024**	Duration:	**90 minutes**	Max Marks:	**50**	Sem	**III**			**OBE**
	Answer any FIVE Questions							**MARKS**	**CO**	**RBT**

1. **Lists basic theorems of Boolean algebra and its postulates.**
   Prove i) x+x =x ii) x+xy = x   **[10] [1] [L4]**

 Answer: Postulates and Theorems of Boolean Algebra (5 Marks)

       Postulate 2 (a) x + 0 = x   (b) x .1 = x
       Postulate 5 (a) x + x' = 1   (b) x . x' = 0
       Theorem 1 (a) x + x = x   (b) x . x = x
       Theorem 2 (a) x + 1 = 1   (b) x . 0 = 0
       Theorem 3, involution (x')' = x
       Postulate 3, commutative (a) x + y = y + x   (b) xy = yx
       Theorem 4, associative (a) x + (y + z) = (x + y) + z   (b) x(yz) = (xy)z
       Postulate 4, distributive (a) x(y + z) = xy + xz   (b) x + yz = (x + y)(x + z)
       Theorem 5, DeMorgan (a) (x + y)' = x' y '   (b) (xy)' = x' + y'
       Theorem 6, absorption (a) x + xy = x   (b) x(x + y) = x

       i) x+x =x   **(2.5 Marks)**
       x + x = (x + x) .1
       = (x + x)(x + x')
       = x + xx'
       = x + 0
       = x

       ii) x+xy = x   **(2.5 Marks)**
       x + xy = x . 1 + xy
       = x(1 + y)
       = x(y + 1)
       = x . 1
       = x

2. **[A] Explain canonical and standard forms with examples.**   **[5] [1] [L2]**

Answer: A binary variable may appear either in its normal form (*x*) or in its complement form (*x*). Now consider two binary variables *x* and *y* combined with an AND operation. Since each variable may appear in either form, there are four possible combinations: *xy*, *xy*, *xy*, and *xy*. Each of these four AND terms is called a *minterm*, **or a** *standard product.*   **(2 Marks)**
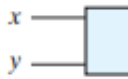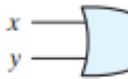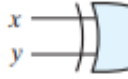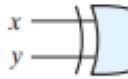
In a similar fashion, *n* variables forming an OR term, with each variable being primed or unprimed, provide 2*n* possible combinations, called **maxterms, or standard sums.** Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form.** **(2 Marks)**

Example - $f1 = x'y'z + xy'z' + xyz$ **(1 Marks)**

**[B] Explain Digital Logic Gates with Graphic symbol, Boolean Function and Truth table.** **[5] [1] [L2]**

Name	Graphic symbol	Algebraic function	Truth table		

AND — Graphic symbol with inputs x, y output F — $F = x \cdot y$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR — Graphic symbol with inputs x, y output F — $F = x + y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Inverter — Graphic symbol with input x output F — $F = x'$

x	F
0	1
1	0

Buffer — Graphic symbol with input x output F — $F = x$

x	F
0	0
1	1

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

NAND $\quad$ $F = (xy)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

NOR $\quad$ $F = (x + y)'$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-OR (XOR) $\quad$ $F = xy' + x'y = x \oplus y$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Exclusive-NOR or equivalence $\quad$ $F = xy + x'y' = (x \oplus y)'$

3. Implement the Logic operations with NAND gates and Implement the following Boolean function with NAND gates: F (x, y, z) = (1, 2, 3, 4, 5, 7) **[10] [1] [L4]**

Answer: Implement the Logic operations with NAND gates and Implement the following Boolean function with NAND gates: F (x, y, z) = (1, 2, 3, 4, 5, 7)
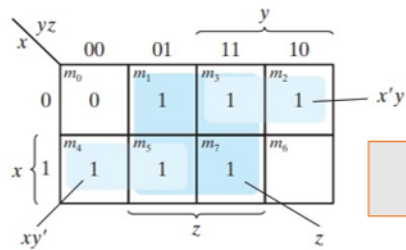
Logic operations with NAND gates **(4 Marks)**
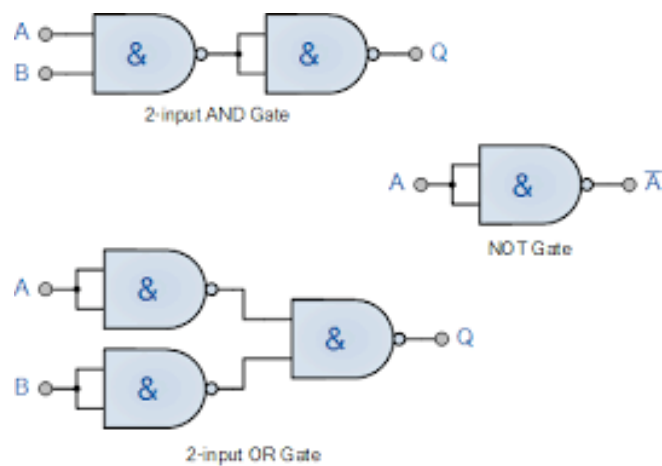
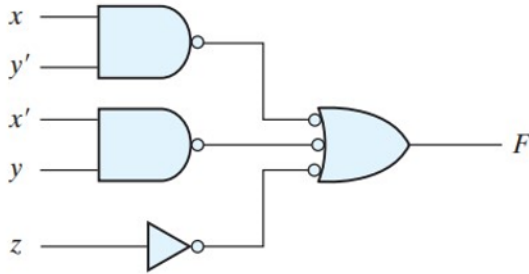**Implement the following Boolean function with NAND gates: F (x, y, z) = (1, 2, 3, 4, 5, 7)** **(6 Marks)**

The first step is to simplify the function into sum-of-products form. This is done by means of the map

Final Expression = $F = xy' + x'y + z$

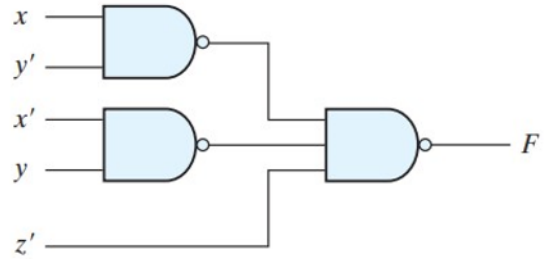Second Step The two-level NAND implementation

$$F = xy' + x'y + z$$

Two-level implementation in mixed notation

Alternative way of drawing the logic diagram

4. **[A]** Simplify the following Boolean functions to a minimum number of literals. **[5] [1] [L3]**
   i) x+x'y
   ii) xy + x'z + yz

Answer: i) $x + x'y = (x + x')(x + y) = 1(x + y) = x + y$ **(2 marks)**

     ii) $xy + x'z + yz = xy + x'z + yz(x + x')$ **(3 marks)**
             $= xy + x'z + xyz + x'yz$
             $= xy(1 + z) + x'z(1 + y)$
             $= xy + x'z.$

     [B] Express the Boolean function F= A + B'C as sum of minterms.   **[5] [1] [L3]**

Answer:

Truth Table for $F = A + B'C$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

     (2 marks)

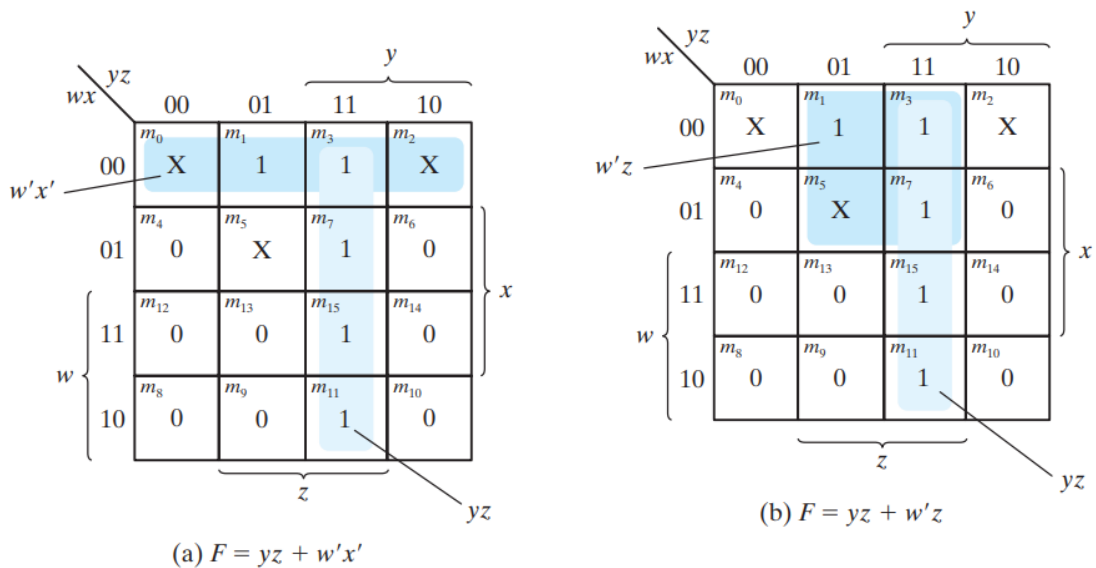$$F = A'B'C + AB'C' + AB'C + ABC' + ABC$$
$$= m1 + m4 + m5 + m6 + m7$$
$$= \Sigma(1,4,5,6,7)$$

     (3 marks)

5. Simplify the Boolean function F (w, x, y, z) = $\sum$(1, 3, 7, 11, 15) which has the don't-care conditions d (w, x, y, z) = $\sum$(0, 2, 5) **[10] [1] [L3]**

Answer: **(4 marks for K-map)**

(a) $F = yz + w'x'$

(b) $F = yz + w'z$

F (w, x, y, z) = yz + w'x' = (0, 1, 2, 3, 7, 11, 15)   **(3 marks)**
F (w, x, y, z) = yz + w'z = (1, 3, 5, 7, 11, 15)   **(3 marks)**

Both expressions include minterms 1, 3, 7, 11, and 15 that make the function F equal to 1.
The don't-care minterms 0, 2, and 5 are treated differently in each expression.
The simplified expression in product-of-sums form:
F (w, x, y, z) = z(w + y) = (1, 3, 5, 7, 11, 15)
In this case, we include minterms 0 and 2 with the 0's and minterm 5 with the 1's.

6. Explain 4 levels of programming abstractions provided by Verilog HDL. Use OR gate realization as an example at Gate Level, Data Flow Level and Behavioral Level respectively.
   [10] [1] [L3]

Answer: There are four level in Verilog HDL –
Switch Level, Gate Level, Data Flow Level and Behavioral Level. **(4 marks)**

1. Switch Level – At this level a module can be implemented in terms of switches. Here nmos and pmos are used as switches for the design.

2. Gate Level – At this level module is implemented in terms of logic gates. Gate level is the lowest of abstraction. Basic logic gates are available as predefined primitives.

3. Data Flow Level – Also called Register Transfer level. At this level, module is designed by specifying the data flow. Signals are assigned by the data manipulating equations. Design is implemented using continuous assignments. All such assignments are concurrent in nature.

4. Behavioral Level – This is the highest level of abstraction provided by HDL. Behavioral level describes the system by its behavior. Different elements like function, task, and blocks can be used. Two important constructs under this level are initial and always.

Example of Gate Level: **(2 marks)**

```
module ORgate ( c, a, b);
output c;
input a,b;
or(c,a,b);
endmodule
```

Example of Data Flow Level: **(2 marks)**

```
module ORgate ( c, a, b);
output c;
input a,b;
assign c = a|b;
endmodule
```

Example of Behavioral Level: **(2 marks)**

```
module ORgate ( c, a, b);
output c;
input a,b;
always@*begin
c = a|b;
end
endmodule
```