SIMPLIFICATION AND MINIMIZATION OF BOOLEAN FUNCTIONS

In this section, we look at how to reduce the Boolean Function expressions into equivalent forms that use ***less logic***. This minimization is the key to reducing the ***complexity*** of the ***logic*** prior to implementing in real circuitry. This reduces the ***amount of gates*** needed, ***placement area***, ***wiring***, and power consumption of the logic circuit.

3.1 Algebraic Simplification

Boolean Functions can be reduced algebraically by applying the ***theorems*** covered in prior sections. This process typically consists of a series of factoring based on the ***distributive*** property followed by ***replacing variables*** with constants (i.e., ***0s*** and ***1s***) using the complements theorem. Finally, constants are removed using the ***identity*** theorem.

Example:

Example: Minimizing a Logic Expression Algebraically

Given: The following truth table.

row	A	B	C	F	minterm
0	0	0	0	1	$m_0 = A'\cdot B'\cdot C'$
1	0	0	1	0	-
2	0	1	0	1	$m_2 = A'\cdot B\cdot C'$
3	0	1	1	1	$m_3 = A'\cdot B\cdot C$
4	1	0	0	0	-
5	1	0	1	0	-
6	1	1	0	1	$m_6 = A\cdot B\cdot C'$
7	1	1	1	1	$m_7 = A\cdot B\cdot C$

Find: A minimized logic expression using algebraic manipulations.

Solution:

$F = A'\cdot B'\cdot C' + A'\cdot B\cdot C' + A'\cdot B\cdot C + A\cdot B\cdot C' + A\cdot B\cdot C$

The first step is to write the canonical sum. The minterms are written in the truth table so this sum can be written directly as:

$F = A'\cdot B'\cdot C' + (A'\cdot \underline{B}\cdot C' + A'\cdot \underline{B}\cdot C + A\cdot \underline{B}\cdot C' + A\cdot \underline{B}\cdot C)$

$F = A'\cdot B'\cdot C' + \underline{B}\cdot(A'\cdot C' + A'\cdot C + A\cdot C' + A\cdot C)$

Next, we notice that B exists in each of these product terms. Let's factor it out using the distributive property.

$F = A'\cdot B'\cdot C' + B\cdot(\underline{A'}\cdot C' + \underline{A'}\cdot C) + (\underline{A}\cdot C' + \underline{A}\cdot C)$

$F = A'\cdot B'\cdot C' + B\cdot(\underline{A'}\cdot(C' + C) + \underline{A}\cdot(C' + C))$

Now we notice that A' and A can be factored out of these product terms using the distributive property.

$F = A'\cdot B'\cdot C' + B\cdot(A'\cdot(\underline{C' + C}) + A\cdot(\underline{C' + C}))$

$F = A'\cdot B'\cdot C' + B\cdot(A'\cdot\underline{1} + A\cdot\underline{1})$

The new expression contains terms that can be minimized using the complements theorem.

$F = A'\cdot B'\cdot C' + B\cdot(A' + A)$

The identity property will get rid of anything AND'd with a 1.

$F = A'\cdot B'\cdot C' + B\cdot(\underline{A' + A})$

$F = A'\cdot B'\cdot C' + B\cdot(\underline{1})$

$F = A'\cdot B'\cdot C' + B$

The complements theorem is again used followed by identity to reduce this term entirely to B.

$F = A'\cdot B'\cdot C' + A'\cdot B\cdot C' + B$

$F = A'\cdot C'\cdot\underline{(B' + B)} + B$

$F = A'\cdot C'\cdot\underline{1} + B$

$F = A'\cdot C' + B$

The next step involves recognizing that one of the eliminated product terms could also have been used to reduce A'·B'·C'. We can write the term back in the expression without impacting the result. We then apply factoring, complements and identity to reduce the expression.

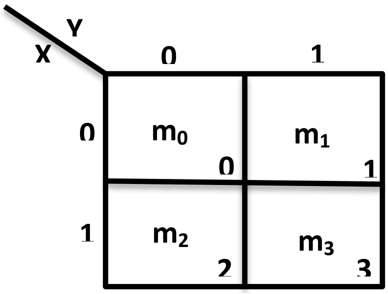
3.2 Simplification Using Karnaugh Maps

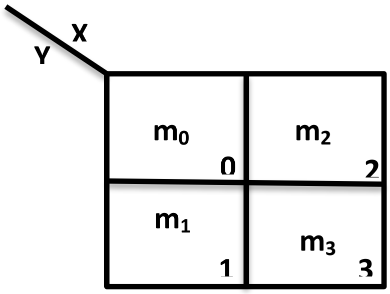
Another method of simplification of Boolean function is ***Karnaugh - Map*** (**K-Map**). This **map** is a diagram made of **squares**, each square represent ***one minterms***, and there are ***several types*** of K- Map depending on the number of **variables** in Boolean function.

1- Formation of a 2-inputs (variables) K-map

A 2-input K-Map will have $\mathbf{2^2}$ cells (or 4 cells) each cell corresponds to a row in the truth table.

row	X	Y	F
0	0	0	$m_0 = \overline{X}\,\overline{Y}$
1	0	1	$m_1 = \overline{X}\,Y$
2	1	0	$m_2 = X\overline{Y}$
3	1	1	$m_3 = X\,Y$

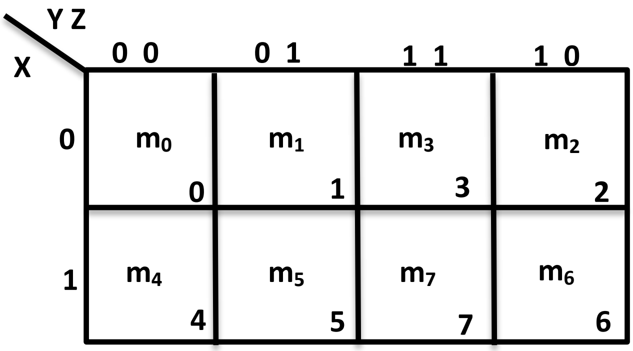




2- Formation of a 3-inputs (variables) K-map

A 3-input K-Map will have $\mathbf{2^3}$ cells (or 8 cells) each cell corresponds to a row in the truth table.

row	X	Y	Z	F
0	0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z} = m_0$
1	0	0	1	$\overline{X}\,\overline{Y}\,Z = m_1$
2	0	1	0	$\overline{X}\,Y\,\overline{Z} = m_2$
3	0	1	1	$\overline{X}\,Y\,Z = m_3$
4	1	0	0	$X\,\overline{Y}\,\overline{Z} = m_4$
5	1	0	1	$X\,\overline{Y}\,Z = m_5$
6	1	1	0	$X\,Y\,\overline{Z} = m_6$
7	1	1	1	$X\,Y\,Z = m_7$



3- Formation of a 4-inputs (variables) K-map

A 4-input K-Map will have $\mathbf{2^4}$ cells (or 16 cells) each cell corresponds to a row in the truth table.

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

4- Formation of a 5-inputs (variables) K-map

A 5-input K-Map will have $\mathbf{2^5}$ cells (or 32 cells) each cell corresponds to a row in the truth table.

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

5- Formation of a 6-inputs (variables) K-map

A 6-input K-Map will have 2^6 cells (or 64 cells) each cell corresponds to a row in the truth table.

ABC \ DEF	000	001	011	010	110	111	101	100
000	0	1	3	2	7	6	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

Ex: Simply the following Boolean functions using K –Map?

$$1\text{ - } F = \overline{X} Y Z + X \overline{Y}\,\overline{Z} + X \overline{Y} Z + \overline{X} Y \overline{Z}$$

X \ YZ	00	01	11	10
0			1	1
1	1	1		

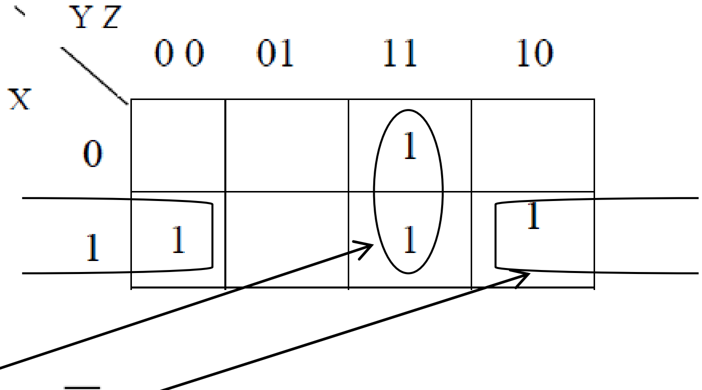
$$F = X \overline{Y} + \overline{X} Y$$

If the function is simplified using Boolean- algebra

$$F = \overline{X} Y Z + X \overline{Y}\,\overline{Z} + X \overline{Y} Z + \overline{X} Y \overline{Z}$$

$$\overline{X} Y (Z + \overline{Z}) + X \overline{Y} (Z + \overline{Z}) = \overline{X} Y + X \overline{Y}$$

2 - $F = \bar{X}YZ + X\bar{Y}\bar{Z} + XYZ + XY\bar{Z}$



$F = YZ + X\bar{Z}$

3 – $F = \bar{A}C + \bar{A}B + A\bar{B}C + BC$

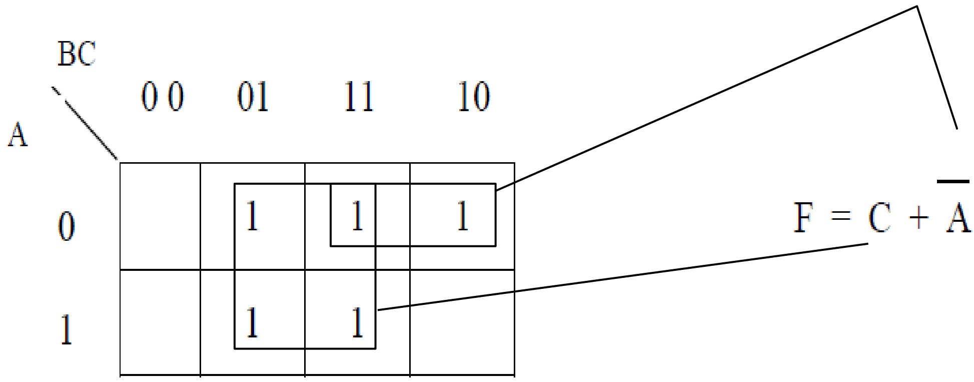
In this function each term must expressed by all variables in the function (A,B,C)

$F(A,B,C) = \bar{A}C.1 + \bar{A}B.1 + A\bar{B}C + BC.1$

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

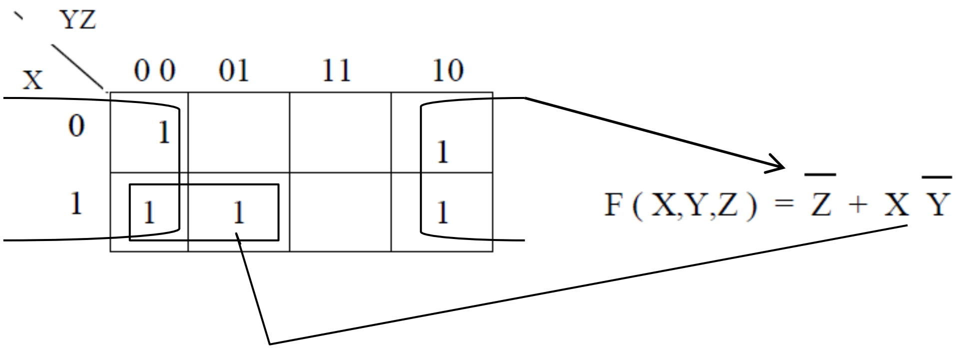
$$= \bar{A}BC + \bar{A}\bar{B}C + \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C + ABC + \bar{A}BC$$

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



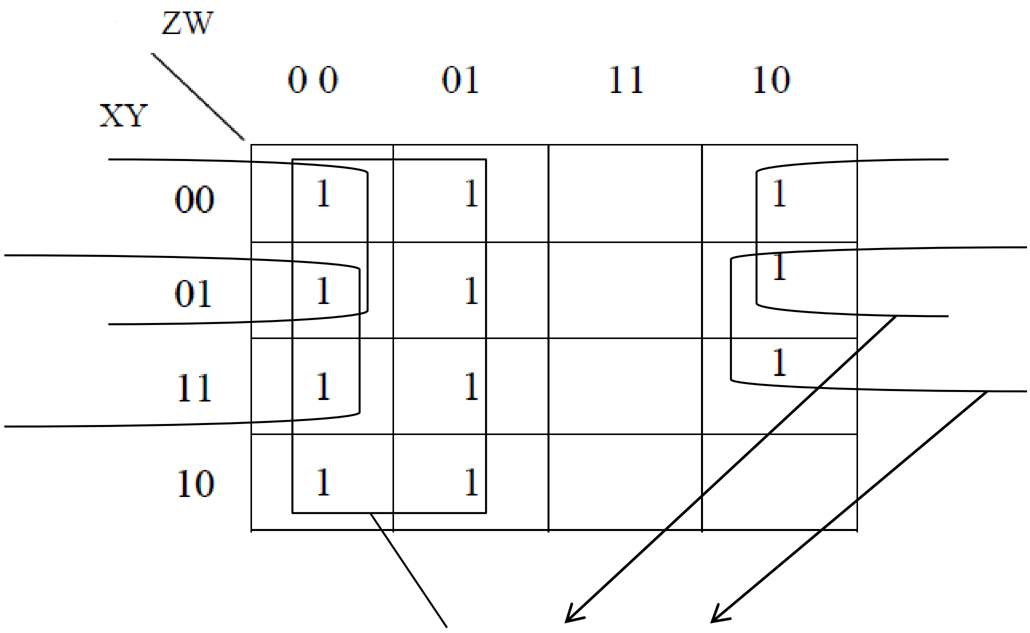
$F = C + \bar{A}B$

4 - $F(X,Y,Z) = \sum(0, 2, 4, 5, 6)$



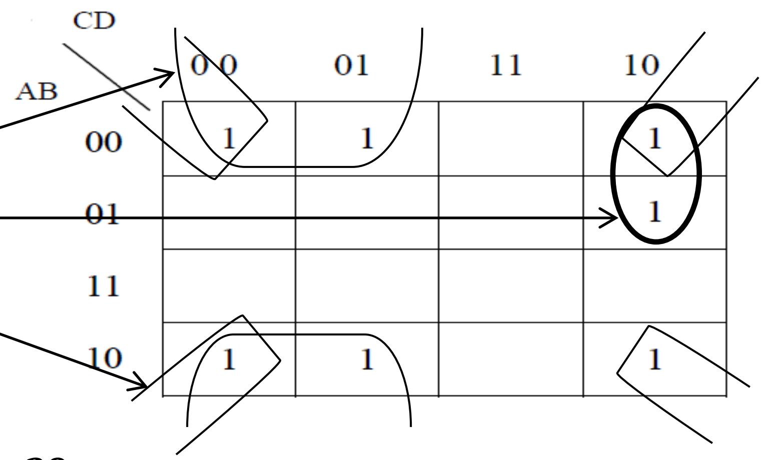
$F(X,Y,Z) = \overline{Z} + X\,\overline{Y}$

5 – $F(X,Y,Z,W) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



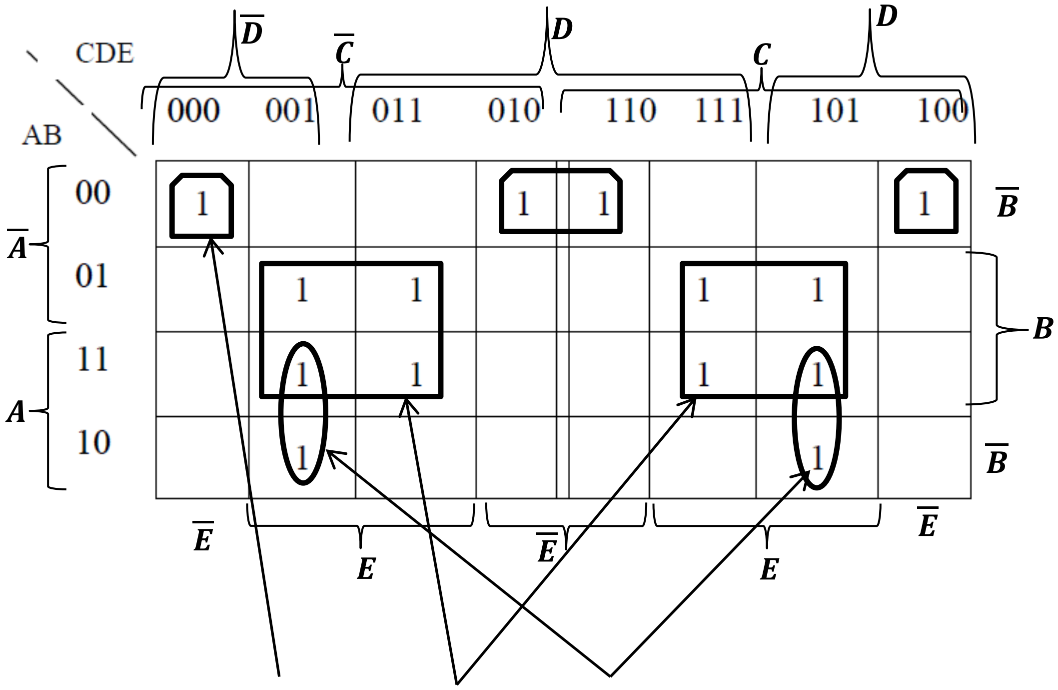
$F(X,Y,Z,W) = \overline{Z} + \overline{X}\,\overline{W} + Y\,\overline{W}$

6 - $F = \overline{A}\,\overline{B}\,\overline{C} + \overline{B}\,C\,\overline{D} + \overline{A}\,B\,C\,\overline{D} + A\,\overline{B}\,\overline{C}$



$F(A,B,C,D) = \overline{B}\,\overline{D} + \overline{B}\,\overline{C} + \overline{A}\,C\,\overline{D}$

7 – F(A,B,C,D.E) = ∑ (0,2,4,6,9,11,13,15,,17,21,25,27,29,31)



$F(A,B,C,D) = \overline{A}\ \overline{B}\ \overline{E} + B\ E + A\ \overline{D}\ E$

H.W

Simplify the following functions in sum of product using K-map

1- $F = \overline{X}\ Y + X\ \overline{Y}\ \overline{W} + W\ (\ \overline{X}\ Y + X\ \overline{Y}\ )$

$2 - F = A\ B\ D + \overline{A}\ \overline{C}\ \overline{D} + \overline{A}\ B + \overline{A}\ C\ \overline{D} + A\ \overline{B}\ \overline{D}$

$3 - F\ (A, B, C, D) = \Pi\ (\ 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14\ )$

Product of Sum simplification

In previous examples the simplification in **Sum of Product** form and each minterms represented by 1 (one) in K-map and each missing term in the function is a complement of the function and represented by 0 (zero) in k-map and the simplified expression obtained F (the complement of the function).

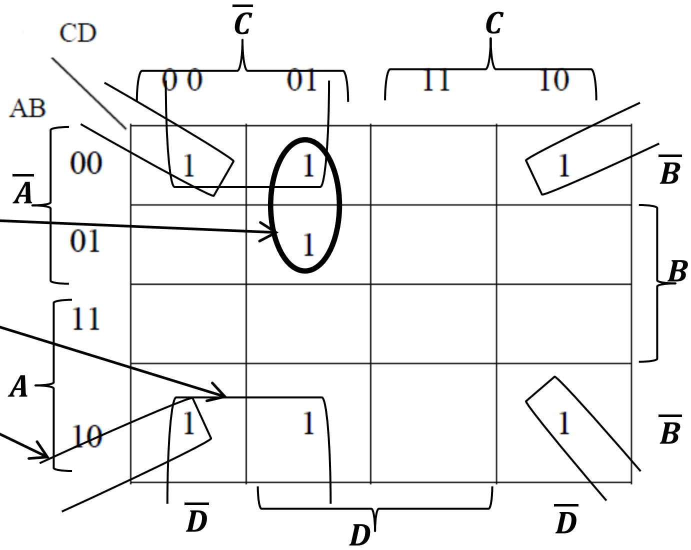
Ex simplify the following function in

1 – Sum of products 2 – product of Sums

F (A,B,C,D) = Σ (0,1, 2, 5, 8, 9, 10)

Sol : 1 – Sum of Products (minterms)

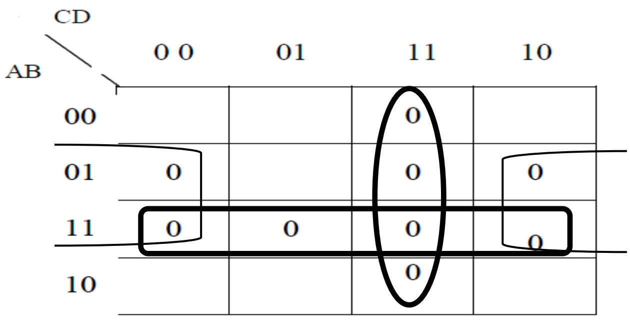
$$F = \overline{B}\,\overline{D} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}\,D$$



2 – Product of Sums

In this case the missing terms is represented by 0 in K-map and simplified to obtain F (complement of the function).

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



And the basic function:

$$F = (\overline{A} + \overline{B})\,(\overline{C} + \overline{D})\,(\overline{B} + D)$$

Ex Simplify the function F in 1 – Sum of Products 2 – Product of Sums

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Note

If the function in **Product of Sums** form then the complement of the function must take first and then the 0 is represented in k-map.

Ex: $(\overline{A} + \overline{B} + C)(B + D)$

The function in Product of Sum form, therefore the complement is taking first

$$\overline{F} = A\,B\,\overline{C} + \overline{B}\,\overline{D}$$

Then these minterms will be assigning in the map by 0 because the function is complement.

Ex : Obtained the simplified expression in Product of Sums

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

Sol

$$\overline{F} = A\,B\,\overline{D} + A\,D + \overline{A}\,\overline{B}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$$

$$\overline{F} = A\,B + \overline{B}\,D + B\,\overline{C}\,\overline{D}$$

AB \ CD	00	01	11	10
00		0	0	
01	0			
11	0	0	0	0
10		0	0	

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

Ex Obtain the simplified expression in Product of Sums

$$F(A, B, C, D) = \Pi(0, 1, 2, 3, 4, 10, 11)$$

$$\overline{F} = \overline{A}\,\overline{B} + \overline{A}\,\overline{C}\,\overline{D} + \overline{B}\,C$$

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

AB \ CD	00	01	11	10
00	0	0	0	0
01	0			
11				
10			0	0

<u>**H.W.**</u>

Obtained the simplified expression of the following functions in

1 – Sum of Products 2 – Product of Sums

$$1 - F = \overline{X}\,\overline{Y} + \overline{Y}\,\overline{Z} + Y\,\overline{Z} + X\,Y\,Z$$

$$2 - F(X, Y, Z, W) = \prod(1, 3, 5, 7, 13, 15)$$

$$3 - F = (A + \overline{B} + D)(\overline{A} + B + D)(C + D)(\overline{C} + \overline{D})$$

Don't- Care Condition

Sometimes a function table or map contains entries for which it is known:

• The input values for the minterm will never occur, or

• The output value for the minterm is not used

In these cases, the output value need not be defined, Instead, the output value is defined as a "don't care" these values are:

1 - Placing "***don't cares***" (an "**X**" entry) in the function table or map,

2 – These values used in simplification with F and F.

3 – These values may be not used in simplification**.**

Ex simplify the Boolean function F in 1.Sum of Products 2.Product of Sums

$$F(X,Y,Z,W) = \Sigma(1, 3, 7, 11, 15) \qquad d(X,Y,Z,W) = \Sigma(0, 2, 5)$$

Sol

1- Sum of Products

XY \ ZW	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

$$F(X,Y,Z,W) = Z\,W + \overline{X}\,\overline{Y}$$

2. Product of Sums

XY \ ZW	00	01	11	10
00	X			X
01	0	X		0
11	0	0		0
10	0	0		0

$$\overline{F}(X,Y,Z,W) = \overline{W} + X\,\overline{Z}$$

$$F(X,Y,Z,W) = W\,(\overline{X} + Z)$$

<u>Ex</u> Simplify the Boolean function F in 1 – Sum of Products 2 Product of Sums using don't care condition

$$F = A\,C\,E + \overline{A}\,C\,\overline{D}\,\overline{E} + \overline{A}\,\overline{C}\,D\,E$$

$$D = D\,\overline{E} + \overline{A}\,\overline{D}\,E + A\,\overline{D}\,\overline{E}$$

<u>Solution</u>:

$$F = A\,C\,E\,.1 + A\,C\,D\,E + A\,C\,D\,E$$

$$= A\,C\,D\,E + A\,C\,\overline{D}\,E + \overline{A}\,C\,\overline{D}\,\overline{E} + \overline{A}\,\overline{C}\,D\,E$$

$$D = D\,\overline{E}\,(A + \overline{A}) + \overline{A}\,\overline{D}\,E\,(C + \overline{C}) + A\,\overline{D}\,\overline{E}\,(C + \overline{C})$$

$$= A\,D\,\overline{E}\,(C+\overline{C}) + \overline{A}\,D\,\overline{E}\,(C + \overline{C}) + \overline{A}\,C\,\overline{D}\,E + \overline{A}\,\overline{C}\,\overline{D}\,E + A\,C\,\overline{D}\,\overline{E} + A\,\overline{C}\,\overline{D}\,\overline{E}$$

$$= A\,C\,D\,\overline{E} + A\,\overline{C}\,D\,\overline{E} + \overline{A}\,C\,D\,\overline{E} + \overline{A}\,\overline{C}\,D\,\overline{E} + \overline{A}\,C\,\overline{D}\,E + \overline{A}\,\overline{C}\,\overline{D}\,E + A\,C\,\overline{D}\,\overline{E} + A\,\overline{C}\,\overline{D}\,\overline{E}$$

1 – Sum of Products

AC \ DE	0 0	01	11	10
00		X	1	X
01	1	X		X
11	X	1	1	X
10	X			X

S.O.P

$$F(A,C,D,E) = A\,C + C\,\overline{E} + \overline{A}\,\overline{C}\,D$$

2 – Product of Sums

AC \ DE	0 0	01	11	10
00	0	X		X
01		X	0	X
11	X			X
10	X	0	0	X

P.O.S.

$$\overline{F}(A,C,D,E) = A\,\overline{C} + \overline{C}\,\overline{D} + \overline{A}\,C\,D$$

$$F(A,C,D,E) = (\overline{A} + C)(C + D)(A + \overline{C} + \overline{D})$$

<u>H.W.</u>

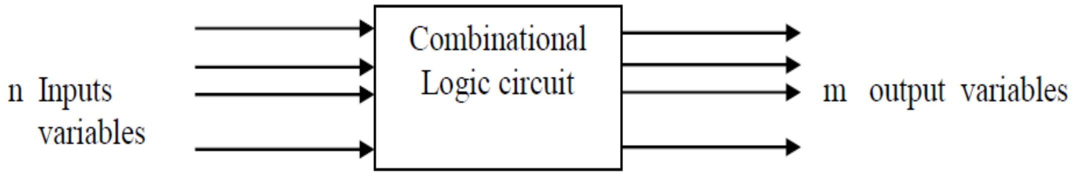
Ex Simplify the Boolean function F in Sum of Products using don't care condition

$$F = \overline{B}\,\overline{C}\,\overline{D} + B\,C\,\overline{D} + A\,B\,\overline{C}\,D$$

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

Combinational Logic Circuit

A combinational circuit consists of inputs variables, logic gates and output variables. The logic gates accepts signal from the inputs and generate signal to the output. A block diagram of a combinational circuit is:



Design Procedure

The design procedure involves the following steps:-

1. The problem is stated.
2. The number of available input variable and required output variable is determined.
3. The input and output variables are assigned letter symbols.
4. The truth table that defines the required relationships between inputs and outputs is derived.
5. The simplified Boolean function for each output is obtained.
6. The logic diagram is drowning.

The ADDERS دوائر الجمع

1. Half Adder

It is a combinational circuit that perform the addition of two bits

0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 0 and carry 1

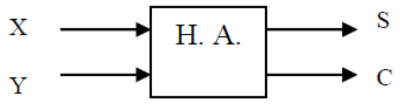
The circuit needs two binary inputs and two binary outputs. The truth table of half adder is:

The circuit needs two binary inputs and two binary outputs. The truth table of half adder is: S= Sum; C= Carry

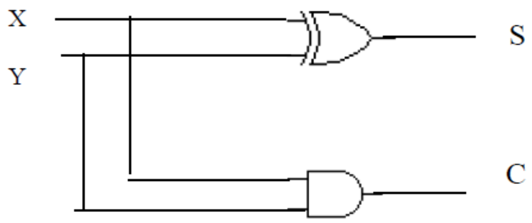
Input		output	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The logic equations $S = \overline{X}Y + X\overline{Y} = X \oplus Y$

$C = X . Y$



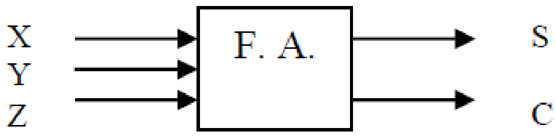
The Block Diagram



Logic Circuit

2– Full Adder

A full adder is a combinational circuit that forms the arithmetic sum of three inputs bits. It consists of three inputs and two outputs. Two of the inputs variables, X and Y, represent the two bits to be added, the third input Z; represent the carry from the previous step. The two output S (for sum) and C (for carry).



Block Diagram

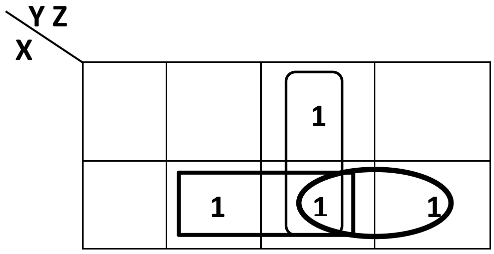
Input			Output	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

To find the logic equations K- map is used:

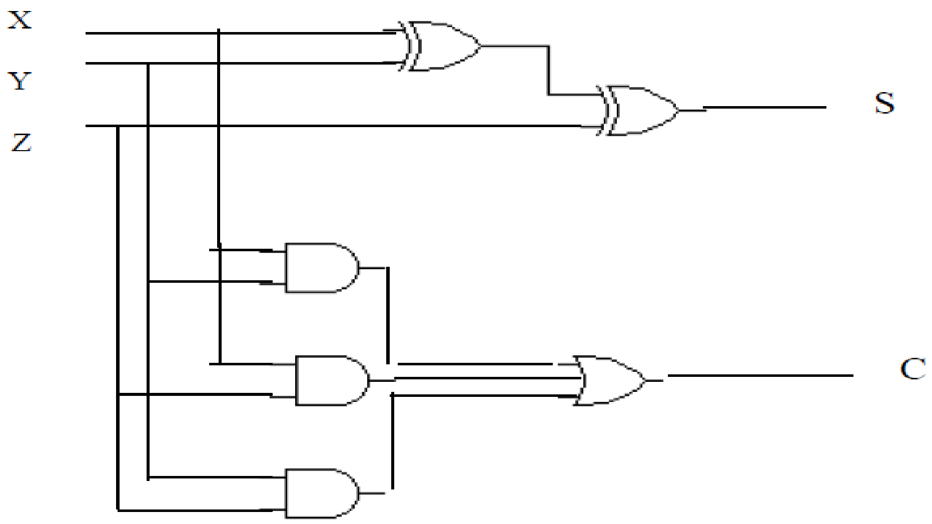
X \ YZ	00	01	11	10
0		1		1
1	1		1	

$$S = \bar{X}\,\bar{Y}\,Z + \bar{X}\,Y\,\bar{Z} + X\,\bar{Y}\,\bar{Z} + X\,Y\,Z$$
$$= Z(\bar{X}\,\bar{Y} + X\,Y) + \bar{Z}(\bar{X}\,Y + X\,\bar{Y})$$
$$= Z(X \odot Y) + \bar{Z}(X \oplus Y)$$
$$= Z(\overline{X \oplus Y}) + \bar{Z}(X \oplus Y)$$
$$= X \oplus Y \oplus Z$$

C = XY + XZ + YZ



The logic circuit



The Subtractors

1 – Half Subtractor

A half subtractor is a digital logic circuit that performs binary subtraction of two single-bit binary numbers. It has **two** inputs, X and Y, and **two** outputs, ***DIFFERENCE*** and ***BORROW***. To perform (X-Y) the truth table is:

Input		output	
X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

D= Difference

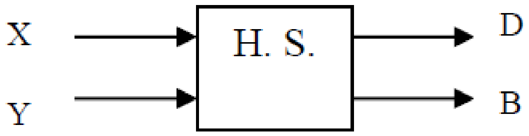
B = Borrow

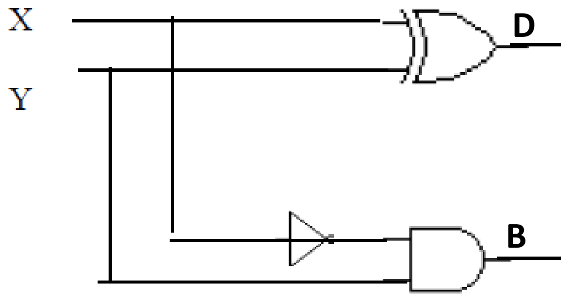
The logic equations

$$D = \overline{X}\,Y + X\,\overline{Y} = X \oplus Y$$

$$B = \overline{X}\,Y$$

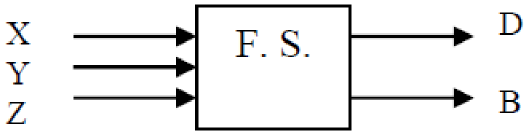
The Block Diagram





2 – Full – Subtractor

A full subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed. This circuit **has three inputs and two outputs**. The truth table:



the block diagram

Input			Output	
X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

To find the logic equations K- map is used

X \ YZ	00	01	11	10
0		1		1
1	1		1	

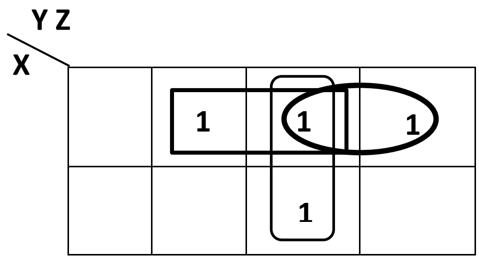
$$D = X\,\overline{Y}\,\overline{Z} + \overline{X}\,\overline{Y}\,Z + X\,Y\,Z + \overline{X}\,Y\,\overline{Z}$$

$$= \overline{Z}(X\overline{Y} + \overline{X}Y) + Z(\overline{X}\,\overline{Y} + XY)$$

$$= \overline{Z}(X \oplus Y) + Z(X \odot Y)$$

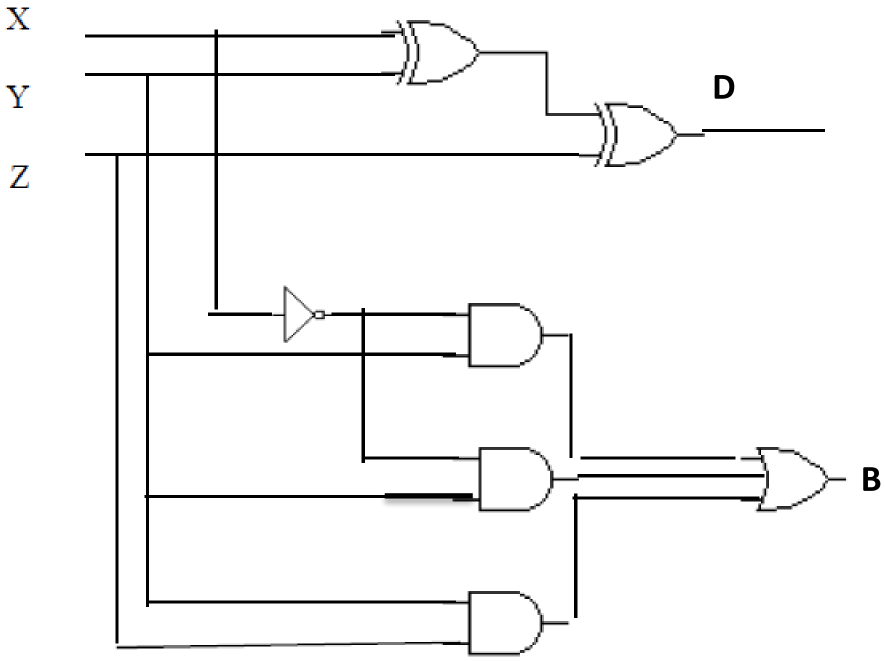
$$= \overline{Z}(X \oplus Y) + Z(\overline{X \oplus Y})$$

$$= X \oplus Y \oplus Z$$



$$B = \overline{X}Y + \overline{X}Z + YZ$$

The logic circuit



<u>Code Conversion</u>

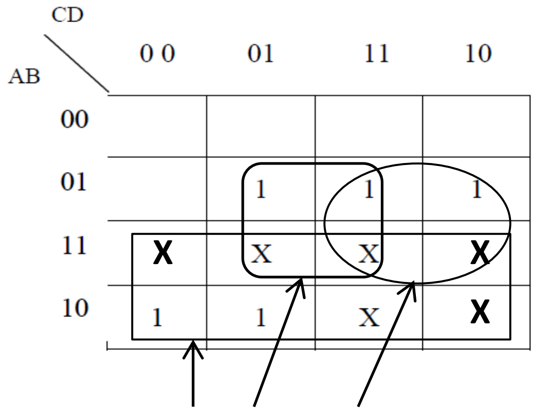
To convert from binary code to another code, a combinational circuit performs this transformation by means of logic gates.

<u>Ex</u> Design a combinational circuit that convert a BCD code to Excess-3 code.

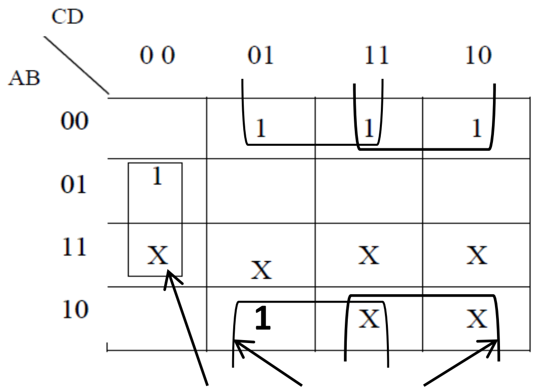
<u>Solution</u>

The truth table consists of 4 inputs and 4 outputs

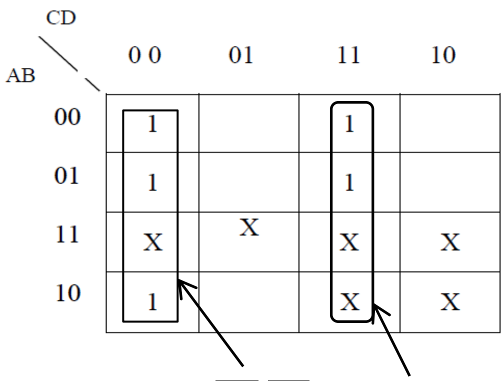
Input				Output			
A	B	C	D	X	Y	Z	W
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x



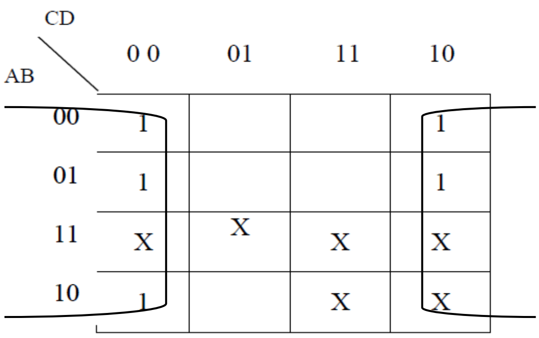
$$X = A + BD + BC$$
$$= A + B(D + C)$$



$$Y = B\overline{C}\,\overline{D} + \overline{B}D + \overline{B}C$$
$$= B\overline{C}\,\overline{D} + \overline{B}(D + C)$$

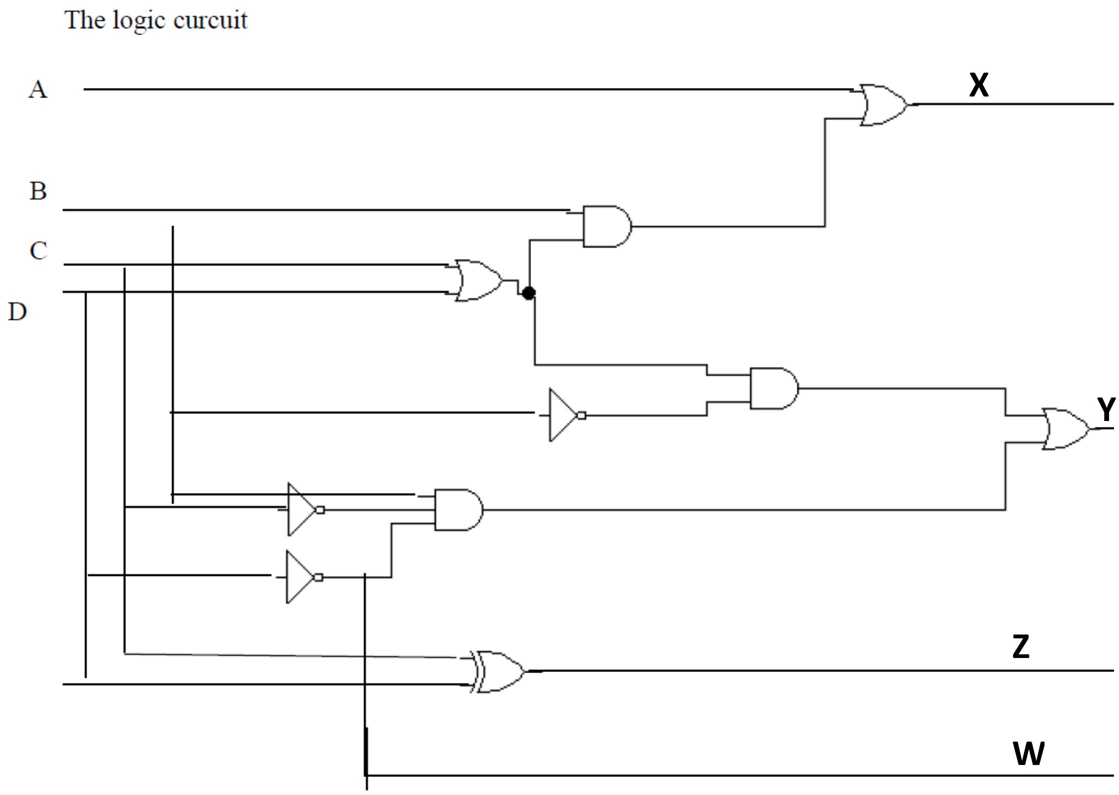


$$Z = \overline{C}\,\overline{D} + C\,D$$
$$= C \odot D$$



$$W = \overline{D}$$

Logic Circuit



<u>Ex</u> A combinational circuit has four inputs and one output, the output equal **1** when:

1 – all the inputs are equal to 1 or

2 – non of the inputs are equal to 1 or

3 – an odd number of inputs are equal to 1.

Design the logic circuit?

Sol

Input				Output
X	Y	Z	W	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

XY \ ZW	00	01	11	10
00	1	1		1
01	1		1	
11		1	1	1
10	1		1	

$$F = \overline{Y}\,\overline{Z}\,\overline{W} + \overline{X}\,\overline{Y}\,\overline{W} + \overline{X}\,\overline{Y}\,\overline{Z} + X Z W$$

$$+ Y Z W + X Y W + X Y Z + \overline{Y}\,\overline{Z}\,\overline{W}$$

Ex Design a combinational circuit that inputs is three – bit numbers and the output is equal to the squared of the input numbers in binary?

Sol

Input			Output					
X	Y	Z	F_5	F_4	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

X \ YZ	00	01	11	10
0		1	1	
1		1	1	

$$F_0 = Z$$

X \ YZ	00	01	11	10
0				1
1				1

$F_2 = Y\,\overline{Z}$

X \ YZ	00	01	11	10
0			1	
1		1		

$F_3 = \overline{X}\,Y\,Z + X\,\overline{Y}\,Z$

X \ YZ	00	01	11	10
0				
1	1	1	1	

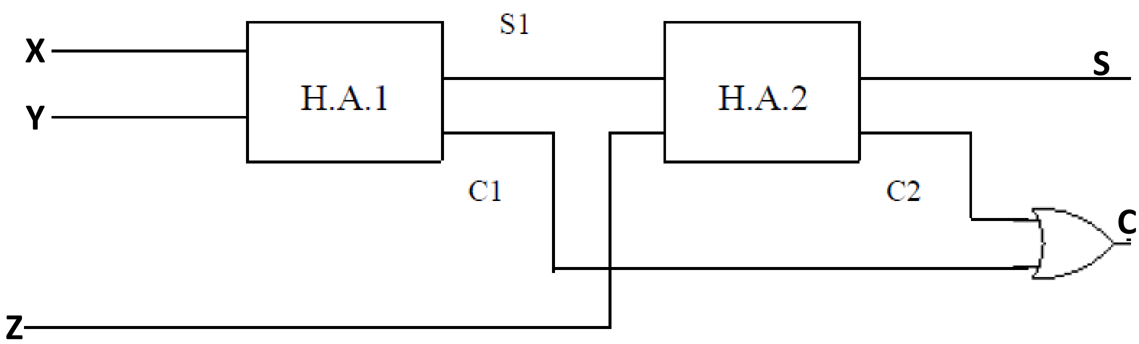
$F_4 = X\,\overline{Y} + X\,Z$

X \ YZ	00	01	11	10
0				
1			1	1

$F_5 = X\,Y$

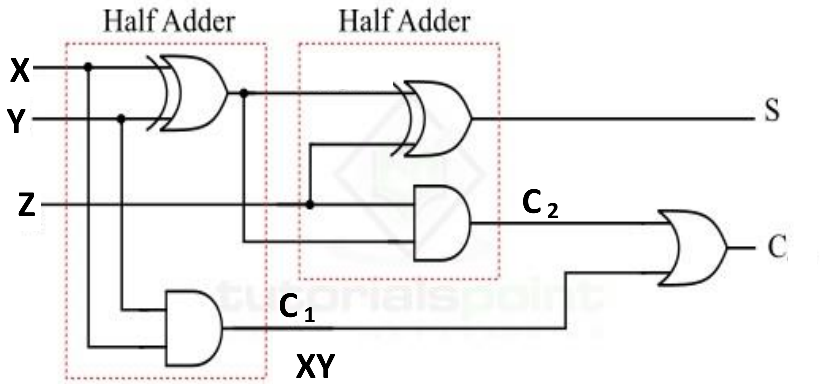
<u>Ex</u>: Design a Full – Adder using two Half - Adder and OR gate, draw the Block diagram and logic circuit?

The block diagram



Block Diagram

The logic circuit:

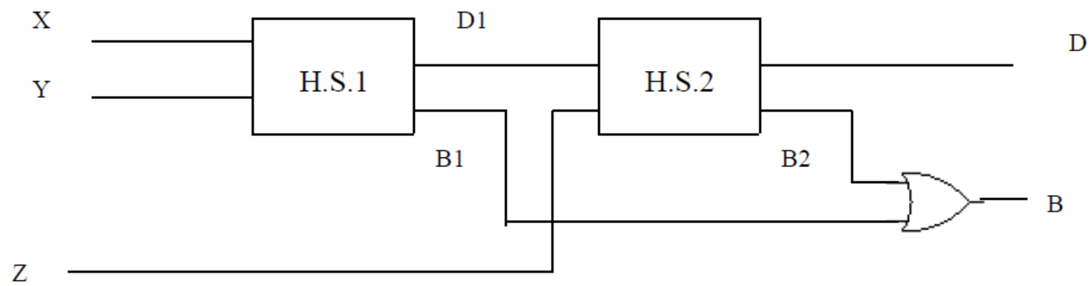


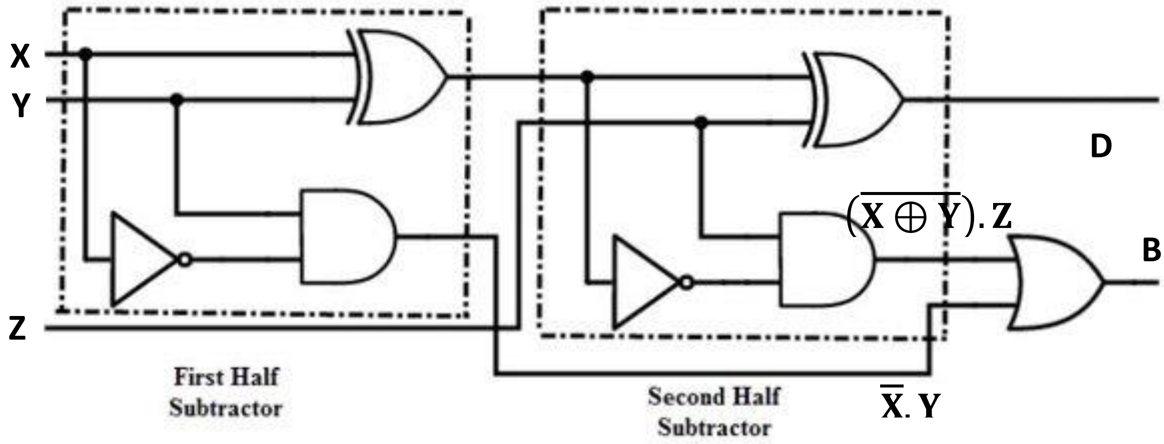
Logic Diagram of Full Adder using Half Adder

$$C = C1 + C2 = X\,Y + (X \oplus Y)\,.\,Z$$
$$= X\,Y + (X\,\overline{Y} + \overline{X}\,Y)\,.\,Z$$
$$= XY(\overline{Z} + Z) + (X\overline{Y} + \overline{X}Y).Z$$
$$= XYZ + XY\overline{Z} + X\overline{Y}Z + \overline{X}YZ = \sum(3,5,6,7)$$

$$S = S1 \oplus Z = X \oplus Y \oplus Z$$

<u>Ex</u> Design Full- Subtractor using two Half – Subtractor and OR gate, draw the Block diagram and logic circuit?





$$D = X \oplus Y \oplus Z$$

$$B = (\overline{X \oplus Y}).Z + \overline{X}.Y = (\overline{X}.\overline{Y} + X.Y)Z + \overline{X}Y$$

$$= \overline{X}.\overline{Y}Z + XYZ + \overline{X}Y = \overline{X}(Y + \overline{Y}Z) + XYZ$$

$$= \overline{X}\,(Y + \overline{Y})(Y + Z) + XYZ = \overline{X}\,Y + \overline{X}Z + XYZ$$

$$= Y(\overline{X} + XZ) + \overline{X}Z = Y(\overline{X} + X)(\overline{X} + Z) + \overline{X}Z$$

$$= \overline{X}Y + YZ + \overline{X}Z$$

<u>Ex</u>: Show that a Full-Subtractor can be obtained from a Full – adder and one inverter?