

UNIT - I**BASIC CONCEPTS**

Review of number systems-representation-conversions, Review of Boolean algebra- theorems, sum of product and product of sum simplification, canonical forms min term and max term, Simplification of Boolean expressions-Karnaugh map, completely and incompletely specified functions, Implementation of Boolean expressions using universal gates, Tabulation methods.

Introduction

Basically there are two types of signals in electronics,

- i) Analog - It is a continuous wave that keeps on changing over a time period.
- ii) Digital – It is discrete in nature.

Digital systems:*Advantages:*

- ❖ Digital systems are interfaced well with computers and are easy to control with software.
- ❖ New features can often be added to a digital system without changing hardware.
- ❖ Often this can be done outside of the factory by updating the product's software. So, the product's design errors can be corrected after the product is in a customer's hands.
- ❖ Information storage can be easier in digital systems.
- ❖ The noise-immunity of digital systems permits data to be stored and retrieved without degradation.
- ❖ In a digital system, as long as the total noise is below a certain level, the information can be recovered perfectly.

Disadvantages:

- ❖ In some cases, digital circuits use more energy than analog circuits to accomplish the same tasks, thus producing more heat as well. In portable or battery-powered systems this can limit use of digital systems.
- ❖ Digital circuits are sometimes more expensive, especially in small quantities.
- ❖ For example, light, temperature, sound, electrical conductivity, electric and magnetic fields are analog.

NUMBER SYSTEMS – DECIMAL, BINARY, OCTAL, HEXADECIMAL

❖ Discuss the various number system conversions with examples.

Many number systems are in use in digital technology. The most common are the decimal, binary, octal, and hexadecimal systems.

Types of Number Systems are

- ❖ Decimal Number system
- ❖ Binary Number system
- ❖ Octal Number system
- ❖ Hexadecimal Number system

Table: Types of Number Systems

DECIMAL	BINARY	OCTAL	HEXADECIMAL
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table: Number system and their Base value

Number Systems		
System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Number system Conversions:

- ❖ Converting from one code form to another code form is called code conversion, like converting from binary to decimal or converting from hexadecimal to decimal.

Binary-To-Decimal Conversion:

Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number which contain a 1.

Example: Convert 10110_2 into a decimal number.

The decimal equivalent number is given as:

$$\begin{aligned} & 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ & = 16 + 0 + 4 + 2 + 0 \\ & = 22_{10}. \end{aligned}$$

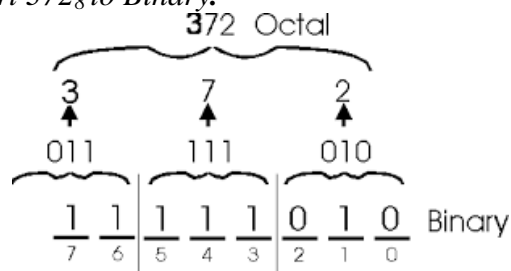
Decimal to binary Conversion: Convert 25_{10} to Binary.

Division	Remainder	Binary
25/2	= 12+ remainder of 1	1 (Least Significant Bit)
12/2	= 6 + remainder of 0	0
6/2	= 3 + remainder of 0	0
3/2	= 1 + remainder of 1	1
1/2	= 0 + remainder of 1	1 (Most Significant Bit)
Result	25_{10}	$= 11001_2$

Binary to octal: Convert $100\ 111\ 010_2$ to octal.

Example: $100\ 111\ 010_2 = (100)\ (111)\ (010)_2 = 4\ 7\ 2_8$

Octal to Binary: Convert 372_8 to Binary.

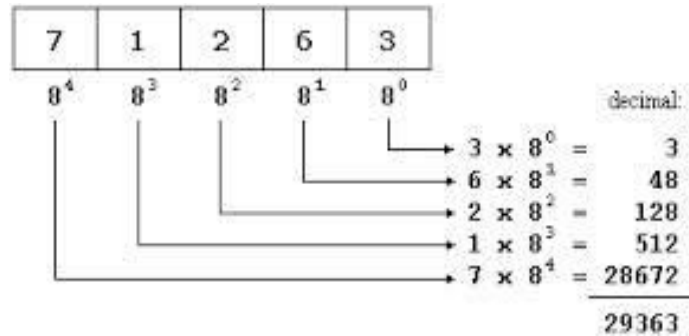


Decimal to octal: Convert 177_{10} to Binary.

Division	Result	Binary
177/8	= 22+ remainder of 1	1 (Least Significant Bit)
22/ 8	= 2 + remainder of 6	6
2 / 8	= 0 + remainder of 2	2 (Most Significant Bit)
Result	177_{10}	$= 261_8$
Binary		$= 010110001_2$

Octal to Decimal: convert $(71263)_8$ to Decimal.

Example:



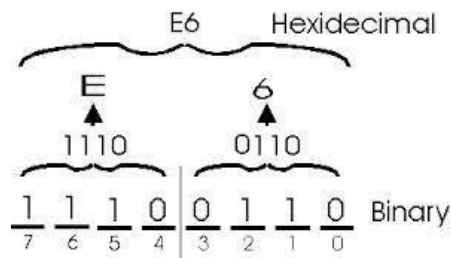
Decimal to Hexadecimal: Convert 378_{10} to Hexadecimal.

Division	Result	Hexadecimal
$378/16$	$= 23 + \text{remainder of } 10$	A (Least Significant Bit)23
$23/16$	$= 1 + \text{remainder of } 7$	7
$1/16$	$= 0 + \text{remainder of } 1$	1 (Most Significant Bit)
Result	378_{10}	$= 17A_{16}$
Binary		$= 0001\ 0111\ 1010_2$

Binary-To-Hexadecimal: Convert $1011\ 0010\ 1111_2$ to Hexadecimal.

Example: $1011\ 0010\ 1111_2 = (1011)(0010)(1111)_2 = B\ 2\ F_{16}$

Hexadecimal to binary: Convert $E6_{16}$ to Binary.



Octal-To-Hexadecimal Hexadecimal-To-Octal Conversion:

- ❖ Convert Octal (Hexadecimal) to Binary first.
- ❖ Regroup the binary number by three bits per group starting from LSB if Octal is required.
- ❖ Regroup the binary number by four bits per group starting from LSB if Hexadecimal is required.

Octal to Hexadecimal: Convert 2650_8 to Hexadecimal.

Octal	Hexadecimal
$= 2\ 6\ 5\ 0$	
$= \mathbf{010\ 110\ 101\ 000}$	$= \mathbf{0101\ 1010\ 1000}$ (Binary)
Result	$= (5A8)_{16}$

Hexadecimal to octal: Convert $(5A8)_{16}$ Hexadecimal to Octal.

Hexadecimal	Octal
$(5A8)_{16}$	= 0101 1010 1000 (Binary)
	= 010 110 101 000 (Binary)
Result	= 2 6 5 0 (Octal)

1's and 2's complement:

❖ Explain about 1's and 2's complement with an example.

- ❖ Complements are used in digital computers to simplify the subtraction operation and for logical manipulation.
- ❖ There are TWO types of complements for each base-r system: the radix complement and the diminished radix complement.
- ❖ The first is referred to as the r's complement and the second as the $(r - 1)$'s complement, when the value of the base r is substituted in the name.
- ❖ The two types are referred to as the 2's complement and 1's complement for binary numbers and the 10's complement and 9's complement for decimal numbers.

Note:

- The 1's complement of a binary number is the number that results when we change all 1's to zeros and the zeros to ones.
- The 2's complement is the binary number that results when we add 1 to the 1's complement.
- It is used to represent negative numbers.

$$2's \text{ complement} = 1's \text{ complement} + 1$$

Example 1) : Find 1's complement of $(1101)_2$

Sol: 1 1 0 1 ← number
 0 0 1 0 ← 1's complement

Example 2) : Find 1's complement of $(1001)_2$

Sol: 1 0 0 1 number
 0 1 1 0 ← 1's complement
 + 1
 ———
 0 1 1 1

Diminished Radix Complement:

Given a number N in base r having n digits, the $(r - 1)$'s complement of N , i.e., its diminished radix complement, is defined as $(r^n - 1) - N$.

The 9's complement of 546700 is $999999 - 546700 = 453299$.

The 9's complement of 012398 is $999999 - 012398 = 987601$.

Radix Complement:

The r 's complement of an n -digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and as 0 for $N = 0$.

For examples:

The 10's complement of 012398 is 987602

The 10's complement of 246700 is 753300

Model 1:*(May 2010)*

Using 10's complement, subtract 72532 - 3250.

$$M = 72532$$

$$10\text{'s complement of } N = + \underline{96750}$$

$$\text{Sum} = 169282$$

$$\text{Discard end carry } 10^5 = - \underline{100000}$$

$$\text{Answer} = 69282$$

Model 2:

Using 10's complement, subtract 3250 - 72532.

$$M = 03250$$

$$10\text{'s complement of } N = + \underline{27468}$$

$$\text{Sum} = 30718$$

Model 3:

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction (a) $X - Y$ and (b) $Y - X$ by using 2's complements.

$$(a) \quad X = 1010100$$

$$2\text{'s complement of } Y = + \underline{0111101}$$

$$\text{Sum} = 10010001$$

$$\text{Discard end carry } 2^7 = - \underline{10000000}$$

$$\text{Answer: } X - Y = 0010001$$

$$(b) \quad Y = 1000011$$

$$2\text{'s complement of } X = \underline{0101100}$$

$$\text{Sum} = 1101111$$

There is no end carry. Therefore, the answer is $Y - X = -(2\text{'s complement of } 1101111) = -0010001$.

Model 4:

Given the two binary numbers $X=1010100$ and $Y=1000011$, perform the subtraction (a) $X-Y$ and (b) $Y-X$ by using 1's complements.

(a) $X - Y = 1010100 - 1000011$

$$\begin{array}{r}
 X = \quad 1010100 \\
 1\text{'s complement of } Y = + \underline{0111100} \\
 \text{Sum} = \quad 10010000 \\
 \text{End around carry} = + \underline{\quad 1} \\
 \text{Answer: } X - Y = \quad 0010001
 \end{array}$$

(b) $Y - X = 1000011 - 1010100$

$$\begin{array}{r}
 Y = \quad 1000011 \\
 1\text{'s complement of } X = + \underline{0101011} \\
 \text{Sum} = \quad 1101110
 \end{array}$$

There is no end carry. Therefore, the answer is $Y - X = -(1\text{'s complement of } 1101110) = -0010001$.

ARITHMETIC OPERATIONS

Binary Addition:

Rules of Binary Addition

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$, and carry 1 to the next more significant bit

Example:

Add: $00011010 + 00001100 = 00100110$

$$\begin{array}{r}
 \quad \quad 1 \quad 1 \\
 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \\
 + 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \\
 \hline
 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0
 \end{array}$$

Binary Subtraction:

Rules of Binary Subtraction

- $0 - 0 = 0$
- $0 - 1 = 1$, and borrow 1 from the next more significant bit
- $1 - 0 = 1$
- $1 - 1 = 0$

Example:

Sub: $00100101 - 00010001 = 00010100$

$$\begin{array}{r} 00100101 \\ - 00010001 \\ \hline 00010100 \end{array}$$

Binary Multiplication:

Rules of Binary Multiplication

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$, and no carry or borrow bits

Example: Multiply the following binary numbers:

(a) 0111 and 1101

(b) 1.011 and 10.01.

(a) 0111×1101

				0	1	1	1	Multiplicand
				×	1	1	0	1
					0	1	1	1
		0			0	0	0	
	0	1	1	1				
0	1	1	1					
1	0	1	1	0	1	1		Final Product

(b) 1.011×10.01

				1.	0	1	1	Multiplicand
				×	1	0.	0	1
					1	0	1	1
		0			0	0	0	
	0	0	0	0				
1	0	1	1					
1	1	.	0	0	0	1	1	Final Product

Binary Division:

Binary division is the repeated process of subtraction, just as in decimal division.

Example : (a) $11001 \div 101$

$$\begin{array}{r} 101 \overline{) 11001} \\ \underline{110} \\ 00100 \\ \underline{0010} \\ 00010 \\ \underline{0000} \\ 00010 \\ \underline{0000} \\ 00000 \end{array}$$

(b) $11110 \div 1001$

$$\begin{array}{r} 1001 \overline{) 11110} \\ \underline{1001} \\ 01100 \\ \underline{0100} \\ 01000 \\ \underline{0100} \\ 00000 \\ \underline{0000} \\ 00000 \\ \underline{0000} \\ 00000 \\ \underline{0000} \\ 00000 \end{array}$$

CODES

Explain the various codes used in digital systems with an example.

- In digital systems a variety of codes are used to serve different purposes, such as data entry, arithmetic operation, error detection and correction, etc.
- Selection of a particular code depends on the requirement.
- Codes can be broadly classified into five groups.
 - (i) Weighted Binary Codes
 - (ii) Non-weighted Codes
 - (iii) Error-detection Codes
 - (iv) Error-correcting Codes
 - (v) Alphanumeric Codes

Weighted Binary Codes

- If each position of a number represents a specific weight then the coding scheme is called weighted binary code.

BCD Code or 8421 Code:

- The full form of BCD is 'Binary-Coded Decimal'. Since this is a coding scheme relating decimal and binary numbers, *four bits are required* to code each decimal number.
- For example, $(35)_{10}$ is represented as 0011 0101 using BCD code, rather than $(100011)_2$
- *Example: Give the BCD equivalent for the decimal number 589.*

The decimal number is 5 8 9

BCD code is 0101 1000 1001

Hence, $(589)_{10} = (010110001001)_{\text{BCD}}$

2421 Code:

- Another weighted code is 2421 code. The weights assigned to the four digits are 2, 4, 2, and 1.
- The 2421 code is the same as that in BCD from 0 to 4. However, it varies from 5 to 9.
- For example, in this case the bit combination 0100 represents decimal 4; whereas the bit combination 1101 is interpreted as the decimal 7, as obtained from $2 \times 1 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 7$.
- This is also a self-complementary code.

BCD Addition:

Examples:

- Consider the addition of $184 + 576 = 760$ in BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

- Add the following BCD numbers: (a) 1001 and 0100, (b) 00011001 and 00010100

Solution

(a)	1 0 0 1		
	+ 0 1 0 0		
	1 1 0 1	→ Invalid BCD number	9
	+ 0 1 1 0	→ Add 6	+4
	0 0 0 1 0 0 1 1	→ Valid BCD number	13 ₁₀
	<u>1</u> <u>3</u>		
(b)	0 0 0 1 1 0 0 1		
	+ 0 0 0 1 0 1 0 0		
	0 0 1 0 1 1 0 1	→ Right group is invalid	19
	+ 0 1 1 0	→ Add 6	+14
	0 0 1 1 0 0 1 1	→ Valid BCD number	33 ₁₀
	<u>3</u> <u>3</u>		

Four Different Binary Codes for the Decimal Digits

Decimal Digt	BCD 8421	2421	Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combinations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Non-weighted Codes

- It basically means that each position of the binary number is not assigned a fixed value.
- Excess-3 codes and Gray codes are such non-weighted codes.

Excess-3 code:

- This code assignment is obtained from the corresponding value of 4-bit binary code after adding 3 to the given decimal digit.
- **Example:** 1000 of 8421 (BCD) = 1011 in Excess-3.

[NOV 2020]

❖ Convert $(367)_{10}$ into its Excess-3 code. and Decimal (643) to Excess - 3 code

Solution.	The decimal number is	3	6	7
	Add 3 to each bit	+3	+3	+3
	Sum	6	9	10

Converting the above sum into 4-bit binary equivalent, we have a

4-bit binary equivalent of 0110 1001 1010

Hence, the Excess-3 code for $(367)_{10}$ = 0110 1001 1010

Excess-3 code for 643 = 1001 0111 0110

Gray code:

- Gray code belongs to a class of code known as minimum change code, in which a number changes by only one bit as it proceeds from one number to the next.
- This code finds use for shift encoders, in some types of analog-to-digital converters, etc.
- The gray code is a reflective digital code which has the special property that any two subsequent numbers codes differ by only one bit. This is also called a *unit-distance code*.

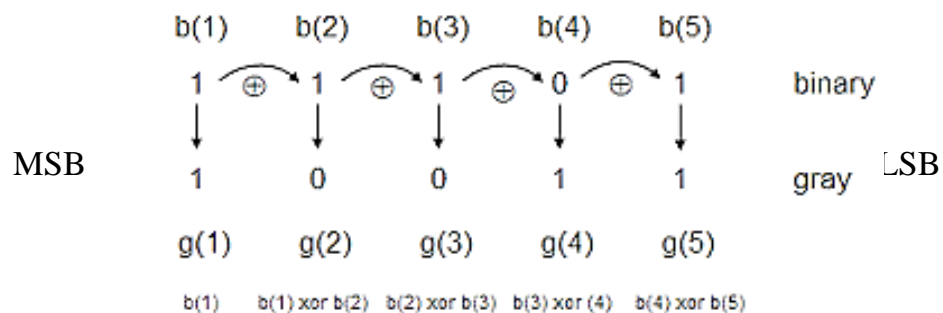
<i>Decimal numbers</i>	<i>Binary code</i>	<i>Gray code</i>
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Binary to Gray Code Conversion:

Any binary number can be converted into equivalent Gray code by the following steps:

- the MSB of the Gray code is the same as the MSB of the binary number;
- the second bit next to the MSB of the Gray code equals the Ex-OR of the MSB and second bit of the binary number; it will be 0 if there are same binary bits or it will be 1 for different binary bits;
- the third bit for Gray code equals the exclusive-OR of the second and third bits of the binary number, and similarly all the next lower order bits follow the same mechanism.

Example:

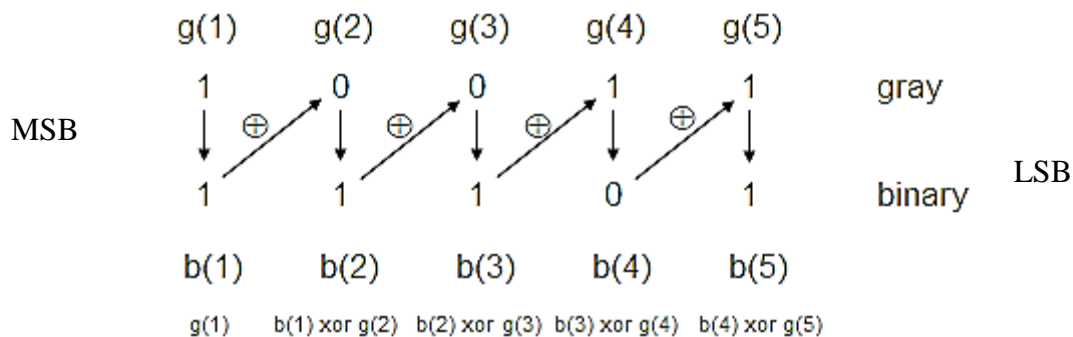


Gray Code to Binary Code Conversion:

Any Gray code can be converted into an equivalent binary number by the following steps:

- i. The MSB of the binary number is the same as the MSB of the Gray code.
- ii. the second bit next to the MSB of the binary number equals the Ex-OR of the MSB of the binary number and second bit of the Gray code; it will be 0 if there are same binary bits or it will be 1 for different binary bits;
- iii. the third bit for the binary number equals the exclusive-OR of the second bit of the binary number and third bit of the Gray code, and similarly all the next lower order bits follow the same mechanism.

Example:



Error detecting codes

- When data is transmitted from one point to another, like in wireless transmission, or it is just stored, like in hard disks and memories, there are chances that data may get corrupted.
- To detect these data errors, we use special codes, which are error detection codes.

Two types of parity

- **Even parity:** Checks if there is an even number of ones; if so, parity bit is zero. When the number of one's is odd then parity bit is set to 1.
- **Odd Parity:** Checks if there is an odd number of ones; if so, parity bit is zero. When the number of one's is even then parity bit is set to 1.

Error correcting code

- Error-correcting codes not only detect errors, but also correct them.
- This is used normally in Satellite communication, where turn-around delay is very high as is the probability of data getting corrupt.

Hamming codes

- Hamming code adds a minimum number of bits to the data transmitted in a noisy channel, to be able to correct every possible one-bit error.
- It can detect (not correct) two-bit errors and cannot distinguish between 1-bit and 2-bits inconsistencies. It can't - in general - detect 3(or more)-bits errors.

Alphanumeric Codes

- An alphanumeric code is a binary code of a group of elements consisting of ten decimal digits, the 26 letters of the alphabet (both in uppercase and lowercase), and a certain number of special symbols such as #, /, &, %, etc.

ASCII (American Standard Code for Information Interchange)

- It is actually a 7-bit code, where a character is represented with seven bits.
- The character is stored as one byte with one bit remaining unused.
- But often the extra bit is used to extend the ASCII to represent an additional 128 characters.

EBCDIC codes

- EBCDIC stands for *Extended Binary Coded Decimal Interchange*.
- It is also an alphanumeric code generally used in IBM equipment and in large computers for communicating alphanumeric data.
- For the different alphanumeric characters the code grouping in this code is different from the ASCII code. It is actually an 8-bit code and a ninth bit is added as the parity bit.

Boolean Algebra and Theorems

Discuss the different postulates of Boolean theorems.

Definition:

- *Boolean algebra is an algebraic structure defined by a set of elements B, together with two binary operators. '+' and '-', provided that the following postulates are satisfied.*
- It can be used to simplify many a complex Boolean expression and also to transform the given expression into more useful and meaningful equivalent expression.

T1: Commutative Law

- (a) $A + B = B + A$
- (b) $A B = B A$

T2: Associative Law

- (a) $(A + B) + C = A + (B + C)$
- (b) $(A B) C = A (B C)$

T3: Distributive Law

- (a) $A (B + C) = A B + A C$
- (b) $A + (B C) = (A + B) (A + C)$

T4: Identity Law

- (a) $A + A = A$
- (b) $A A = A$

T5: Negation Law

$$(\overline{\overline{A}}) = A \quad \text{and} \quad (\overline{\overline{A}}) = A$$

T6: Redundancy

- (a) $A + A B = A$
- (b) $A (A + B) = A$

T7: Operations with '0' & '1'

- (a) $0 + A = A$
- (b) $1 A = A$
- (c) $I + A = I$
- (d) $0 A = 0$

T8 : Complement laws

- (a) $\overline{\overline{A}} + A = 1$
- (b) $\overline{\overline{A}}. A = 0$

- T9 : (a) $A + \overline{A}B = A + B$
- (b) $A. (\overline{A} + B) = A. B$

Postulates of Boolean Algebra:

- The postulates of a mathematical system form the basic assumptions from which it is possible to deduce the rules, theorems, and properties of the system.
- The following are the important postulates of Boolean algebra:
 1. $1 * 1 = 1$ $0 + 0 = 0$.
 2. $1 * 0 = 0 * 1 = 0$ $0 + 1 = 1 + 0 = 1$
 3. $0 * 0 = 0$ $1 + 1 = 1$
 4. $1' = 0$ and $0' = 1$.
- Many theorems of Boolean algebra are based on these postulates, which can be used to simplify Boolean expressions.

The operators and postulates have the following meanings:

- ✓ The binary operator $+$ defines addition.
- ✓ The additive identity is 0.
- ✓ The additive inverse defines subtraction.
- ✓ The binary operator $.$ (dot) defines multiplication.
- ✓ The multiplicative identity is 1.
- ✓ The only distributive law applicable is that of $.$ (dot) over $+$:

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

Two-Valued Boolean Algebra:

A two-valued Boolean algebra is defined on a set of two elements, $B = \{0, 1\}$, with rules for the two binary operators $+$ and $.$ (dot) as shown in the following operator tables.

x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Duality Principle:

- The *duality principle* states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.
- If the *dual* of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

De-Morgan's theorem:

1. The complement of product is equal to the sum of their complements. $(X \cdot Y)' = X' + Y'$
2. The complement of sum is equal to the product of their complements. $(X + Y)' = X' \cdot Y'$

Basic Theorems:*Postulates and Theorems of Boolean Algebra*

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
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)	$x(x + y) = x$

THEOREM 1(a): $x + x = x$.

Statement	Justification
$x + x = (x + x) \cdot 1$	postulate 2(b)
$= (x + x)(x + x')$	5(a)
$= x + xx'$	4(b)
$= x + 0$	5(b)
$= x$	2(a)

THEOREM 1(b): $x \cdot x = x$.

Statement	Justification
$x \cdot x = xx + 0$	postulate 2(a)
$= xx + xx'$	5(b)
$= x(x + x')$	4(a)
$= x \cdot 1$	5(a)
$= x$	2(b)

THEOREM 2(a): $x + 1 = 1$.

Statement	Justification
$x + 1 = 1 \cdot (x + 1)$	postulate 2(b)
$= (x + x')(x + 1)$	5(a)
$= x + x' \cdot 1$	4(b)
$= x + x'$	2(b)
$= 1$	5(a)

THEOREM 2(b): $x \cdot 0 = 0$ by duality.

THEOREM 3: $(x')' = x$. From postulate 5, we have $x + x' = 1$ and $x \cdot x' = 0$, which together define the complement of x . The complement of x' is x and is also $(x')'$.

THEOREM 6(a): $x + xy = x$.

Statement	Justification
$x + xy = x \cdot 1 + xy$	postulate 2(b)
$= x(1 + y)$	4(a)
$= x(y + 1)$	3(a)
$= x \cdot 1$	2(a)
$= x$	2(b)

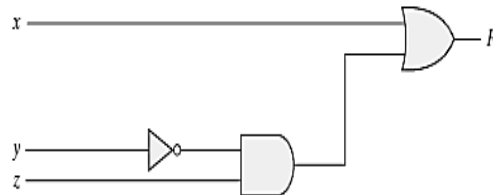
THEOREM 6(b): $x(x + y) = x$ by duality.

Boolean Functions

- ❖ Boolean algebra is an algebra that deals with binary variables and logic operations.
- ❖ A Boolean function described by an algebraic expression consists of binary variables, the constants 0 and 1, and the logic operation symbols.
- ❖ For a given value of the binary variables, the function can be equal to either 1 or 0.

Example: Consider the Boolean function , $F1 = x + y'z$

➤ The gate implementation of F1 is shown below.



Examples:

Simplify the following Boolean functions to a minimum number of literals.

1. $x(x' + y) = xx' + xy = 0 + xy = xy$.
2. $x + x'y = (x + x')(x + y) = 1(x + y) = x + y$.
3. $(x + y)(x + y') = x + xy + xy' + yy' = x(1 + y + y') = x$.
4. $xy + x'z + yz = xy + x'z + yz(x + x')$
 $= xy + x'z + xyz + x'yz$
 $= xy(1 + z) + x'z(1 + y)$
 $= xy + x'z$.
5. $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$, by duality from function 4.

Complement of a function:

- The complement of a function F is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F .

Example:

1. Simplify:-

$$\begin{aligned}(A + B + C)' &= (A + x)' \quad \text{let } B + C = x \\&= A'x' \quad \text{by theorem 5(a) (DeMorgan)} \\&= A'(B + C)' \quad \text{substitute } B + C = x \\&= A'(B'C') \quad \text{by theorem 5(a) (DeMorgan)} \\&= A'B'C' \quad \text{by theorem 4(b) (associative)}\end{aligned}$$

2. Find the complement of the functions $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$.

Solution:

By applying DeMorgan's theorem, the complements are obtained as follows:

$$\begin{aligned}F_1' &= (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z') \\F_2' &= [x(y'z' + yz)]' = x' + (y'z' + yz)' = x' + (y'z')'(yz)' \\&= x' + (y + z)(y' + z') \\&= x' + yz' + y'z\end{aligned}$$

3. Find the complement of the functions $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$ by taking their duals and complementing each literals.

Solution:

1. $F_1 = x'yz' + x'y'z$.

The dual of F_1 is $(x' + y + z')(x' + y' + z)$.

Complement each literal: $(x + y' + z)(x + y + z') = F_1'$.

2. $F_2 = x(y'z' + yz)$.

The dual of F_2 is $x + (y' + z')(y + z)$.

Complement each literal: $x' + (y + z)(y' + z') = F_2'$.

LOGIC GATES

Discuss the various logic gates with its truth table.

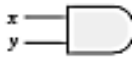

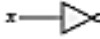
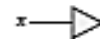


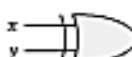
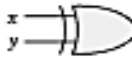
- ❖ A **logic gate** is an idealized or physical device implementing a Boolean function.
- ❖ It performs a logical operation on one or more logical inputs, and produces a single logical output.

Positive and Negative Logic:

- ❖ The binary variables have two states, i.e. the logic '0' state or the logic '1' state.
- ❖ These are represented by two different voltage levels or two different current levels.
- ❖ If the more positive of the two voltage or current levels represents a logic '1' and the less positive of the two levels represents a logic '0', then the logic system is referred to as a **positive logic system**.
- ❖ If the more positive of the two voltage or current levels represents a logic '0' and the less positive of the two levels represents a logic '1', then the logic system is referred to as a **negative logic system**.

Truth Table

A truth table lists all possible combinations of input binary variables and the corresponding outputs of a logic system.

Name	Graphic symbol	Algebraic function	Truth table															
AND		$F = x \cdot y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inverter		$F = x'$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	F	0	1	1	0									
x	F																	
0	1																	
1	0																	
Buffer		$F = x$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	F	0	0	1	1									
x	F																	
0	0																	
1	1																	
NAND		$F = (xy)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Universal Gates(NAND & NOR)

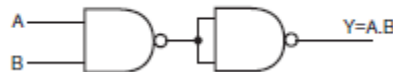
- ❖ The OR, AND and NOT gates are the three basic logic gates , It can be used to construct the logic circuit for any given Boolean expression.
- ❖ The NOR and NAND gate have the property that they individually can be used to hardware-implement a logic circuit corresponding to any given Boolean expression.
- ❖ It is possible to use either only NAND gates or only NOR gates to implement any Boolean expression.
- ❖ This is so because a combination of NAND gates or a combination of NOR gates can be used to perform functions of any of the basic logic gates.
- ❖ It is for this reason that NAND and NOR gates are universal gates.

Implementation of basic gates using NAND gate:

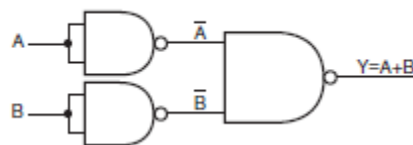
Inverter (NOT gate):



AND gate:



OR gate:

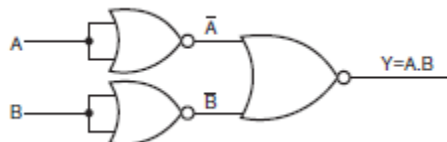


Implementation of basic gates using NOR gate:

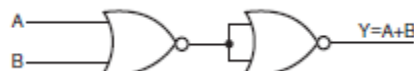
Inverter (NOT gate):



AND gate:



OR gate:

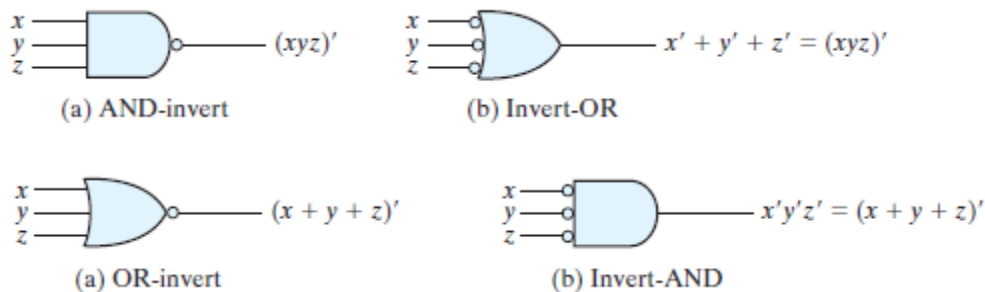


NAND–NOR implementations:

- Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates.
- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.
- Because of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from Boolean functions given in terms of AND, OR, and NOT into equivalent NAND and NOR logic diagrams.

Only NAND/NOR gate circuit:

- A convenient way to implement a Boolean function with NAND/NOR gates is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND/NOR logic.
- The conversion of an algebraic expression from AND, OR, and complement to NAND/NOR can be done by simple circuit manipulation techniques that change AND–OR diagrams to NAND/NOR diagrams.

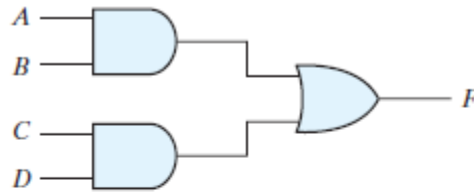


NAND Implementation Procedure:

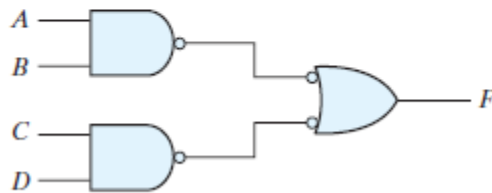
- ✓ Draw the AOI logic of given Boolean expression.
- ✓ Add bubble on input of OR gate & output of AND gate.
- ✓ Add an Inverter on each line that received bubbles.
- ✓ Eliminate double inversions
- ✓ Replace all by NAND gates

Example:

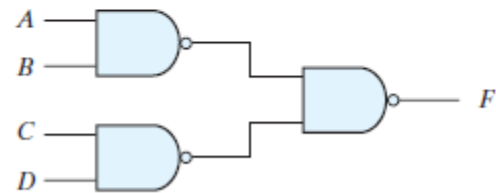
1. Implement $F = AB + CD$ using only NAND gate.



(a)



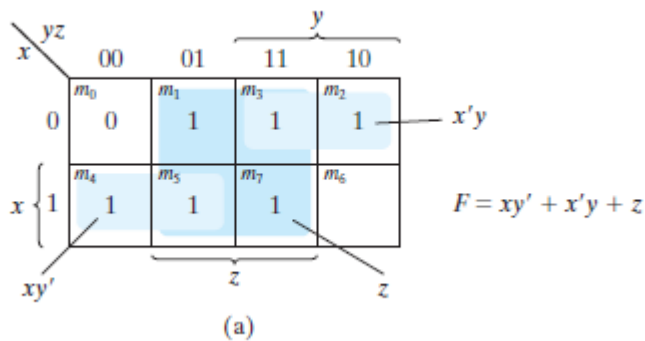
(b)



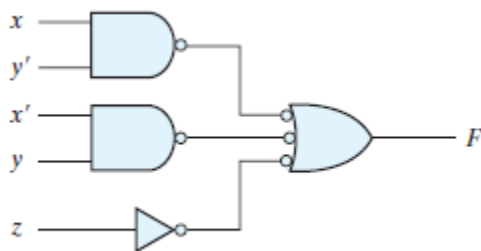
(c)

2. Implement the following Boolean function with NAND gates: $F(x, y, z) = (1, 2, 3, 4, 5, 7)$

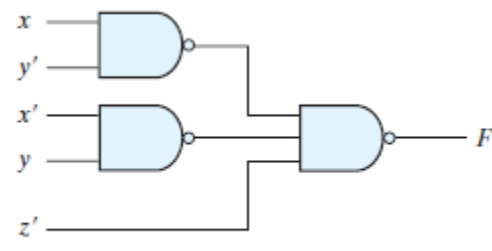
(Dec 2018) (Dec 2014)



(a)



(b)



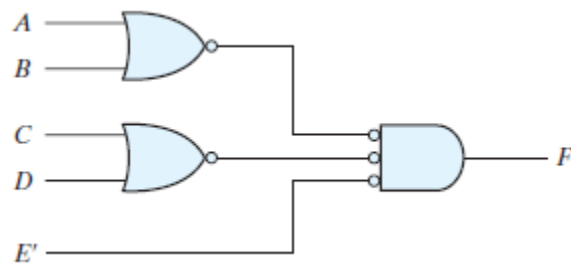
(c)

NOR Implementation Procedure:

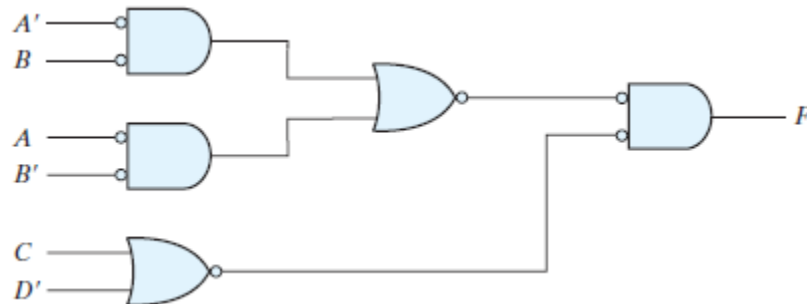
- ✓ Draw the AOI logic of given Boolean expression.
- ✓ Add bubble on input of AND gate & output of OR gate.
- ✓ Add an Inverter on each line that received bubbles.
- ✓ Eliminate double inversions
- ✓ Replace all by NOR gates

Example:

1. Implement $F = (A + B)(C + D)E$ using only NOR gate.



2. Implement $F = (AB' + A'B)(C + D')$ using only NOR gate.



CANONICAL AND STANDARD FORMS

Explain canonical SOP & POS form with suitable example.

- Binary logic values obtained by the logical functions and logic variables are in binary form. An arbitrary logic function can be expressed in the following forms.
 - (i) Sum of the Products (SOP)
 - (ii) Product of the Sums (POS)
- Boolean functions expressed as a sum of minterms or product of maxterms are said to be in *canonical form*.

Product term:

- The AND function is referred to as a product. The variable in a product term can appear either in complementary or uncomplimentary form. **Example: ABC'**

Sum term:

- The OR function is referred to as a Sum. The variable in a sum term can appear either in complementary or uncomplimentary form. **Example: $A+B+C'$**

Sum of Product (SOP):

- The logical sum of two or more logical product terms is called sum of product expression. It is basically an OR operation of AND operated variables. **Example: $Y=AB+BC+CA$**

Product of Sum (POS):

- The logical product of two or more logical sum terms is called product of sum expression. It is basically an AND operation of OR operated variables.
Example: $Y=(A+B).(B+C).(C+A)$

Minterm:

- A product term containing all the K variables of the function in either complementary or uncomplimentary form is called Minterm or standard product.

Maxterm:

- A sum term containing all the K variables of the function in either complementary or uncomplimentary form is called Maxterm or standard sum.

Minterms and Maxterms for Three Binary Variables

<i>x</i>	<i>y</i>	<i>z</i>	Minterms		Maxterms	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical SOP Expression:

- The minterms whose sum defines the Boolean function are those which give the 1's of the function in a truth table.

Procedure for obtaining Canonical SOP expression:

- ✓ Examine each term in a given logic function. Retain if it is a minterm, continue to examine the next term in the same manner.
- ✓ Check for the variables that are missing in each product which is not minterm. Multiply the product by $(X+X')$, for each variable X that is missing.
- ✓ Multiply all the products and omit the redundant terms.

Example 1:

Express the Boolean function $F = A + B'C$ as a sum of minterms.

(Dec 2017)

Solution:

The function has three variables: A , B , and C .

The first term A is missing two variables; therefore,

$$A = A(B + B') = AB + AB'$$

This function is still missing one variable, so

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

The second term $B'C$ is missing one variable; hence,

$$B'C = B'C(A + A') = AB'C + A'B'C$$

Combining all terms, we have

$$F = A + B'C = ABC + ABC' + AB'C + AB'C' + A'B'C$$

But $AB'C$ appears twice, and according to theorem 1 ($x + x = x$), it is possible to remove one of those occurrences.

Rearranging the minterms in ascending order, we finally obtain

$$\begin{aligned} F &= A'B'C + AB'C + AB'C' + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \\ F(A, B, C) &= \sum(1, 4, 5, 6, 7) \end{aligned}$$

Example 2: Obtain the canonical sum of product form of the following function.

(May 2014)

$$F(A, B, C) = A + BC$$

[NOV 2020]

$$\begin{aligned} &= A(B + B')(C + C') + BC(A + A') \\ &= (AB + AB')(C + C') + ABC + A'BC \\ &= ABC + AB'C + ABC' + AB'C' + ABC + A'BC \\ &= ABC + AB'C + ABC' + AB'C' + A'BC \text{ (as } ABC + ABC = ABC) \end{aligned}$$

Hence the canonical sum of the product expression of the given function is

$$F(A, B) = ABC + AB'C + ABC' + AB'C' + A'BC.$$

$$\text{POS Form: } F = (A'+B'+C')(A'+B+C')(A'+B'+C)(A'+B+C)(A+B'+C')$$

Canonical POS Expression:

- The Maxterms whose product defines the Boolean function are those which give the 1's of the function in a truth table.

Procedure for obtaining Canonical POS expression:

- ✓ Examine each term in a given logic function. Retain if it is a maxterm, continue to examine the next term in the same manner.
- ✓ Check for the variables that are missing in each sum which is not maxterm. Add $(X.X')$, for each variable X that is missing.
- ✓ Expand the expression using distributive property eliminate the redundant terms.

Example1:

Express the Boolean function $F = xy + x'z$ as a product of maxterms.

Solution:

First, convert the function into OR terms by using the distributive law:

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

The function has three variables: x, y, and z. Each OR term is missing one variable; therefore,

$$\begin{aligned} x' + y &= x' + y + z.z' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + y.y' = (x + y + z)(x + y' + z) \\ y + z &= y + z + x.x' = (x + y + z)(x' + y + z) \end{aligned}$$

Combining all the terms and removing those which appear more than once, we finally obtain

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ F(x, y, z) &= \pi(0, 2, 4, 5) \end{aligned}$$

Example 2:

Obtain the canonical product of the sum form of the following function.

$$F(A, B, C) = (A + B')(B + C)(A + C') \quad (\text{Dec 2012})$$

Solution:

$$\begin{aligned} F(A, B, C) &= (A + B')(B + C)(A + C') \\ &= (A + B' + 0)(B + C + 0)(A + C' + 0) \\ &= (A + B' + CC')(B + C + AA')(A + C' + BB') \\ &= (A + B' + C)(A + B' + C')(A + B + C)(A' + B + C)(A + B + C') \\ &\quad (A + B' + C') \\ &\quad [\text{using the distributive property, as } X + YZ = (X + Y)(X + Z)] \\ &= (A + B' + C)(A + B' + C')(A + B + C)(A' + B + C)(A + B + C') \\ &\quad [\text{as } (A + B' + C')(A + B' + C) = A + B' + C'] \end{aligned}$$

Hence the canonical product of the sum expression for the given function is

$$F(A, B, C) = (A + B' + C)(A + B' + C')(A + B + C)(A' + B + C)(A + B + C')$$

Karnaugh Map (K-map):

- ❖ Using Boolean algebra to simplify Boolean expressions can be difficult.
- ❖ The Karnaugh map provides a simple and straight-forward method of minimizing boolean expressions which represent combinational logic circuits.
- ❖ *A Karnaugh map is a pictorial method of grouping together expressions with common factors and then eliminating unwanted variables.*
- ❖ A Karnaugh map is a two-dimensional truth-table. Note that the squares are numbered so that the binary representations for the numbers of two adjacent squares differ in exactly one position.

Rules for Grouping together adjacent cells containing 1's:

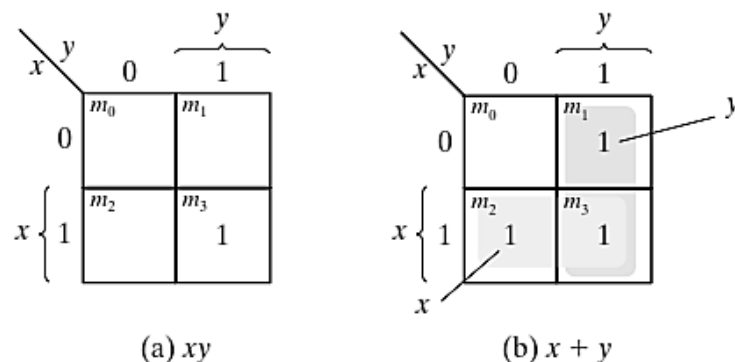
- Groups must contain 1, 2, 4, 8, 16 (2^n) cells.
- Groups must contain only 1 (and X if don't care is allowed).
- Groups may be horizontal or vertical, but not diagonal.
- Groups should be as large as possible.
- Each cell containing a 1 must be in at least one group.
- Groups may overlap.
- Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.
- There should be as few groups as possible.

Obtaining Product Terms

- If A is a variable that has value 0 in all of the squares in the grouping, then the complemented form A is in the product term. „
- If A is a variable that has value 1 in all of the squares in the grouping, then the true form A is in the product term.
- If A is a variable that has value 0 for some squares in the grouping and value 1 for others, then it is not in the product term

The Format of K-Maps:

K-Maps of 2 Variables:



K-Maps of 3 Variables:

- ❖ Simplify the boolean function

$$F(x, y, z) = \sum(2, 3, 4, 5)$$

		y			
		00	01	11	10
x \ yz	0	m_0	m_1	m_3 1	m_2 1
	1	m_4 1	m_5 1	m_7	m_6

Annotations: $x'y$ points to m_2 , xy' points to m_4 , z points to the bottom row.

$$F(x, y, z) = \sum(2, 3, 4, 5) = x'y + xy'$$

- ❖ Simplify the boolean function

$$F(x, y, z) = \sum(3, 4, 6, 7)$$

		y			
		00	01	11	10
x \ yz	0	m_0	m_1	m_3 1	m_2
	1	m_4 1	m_5	m_7 1	m_6 1

Annotations: yz points to m_3 , $xy'z'$ points to m_4 , xyz' points to m_6 , z points to the bottom row.

$$\text{Note: } xy'z' + xyz' = xz'$$

$$F(x, y, z) = \sum(3, 4, 6, 7) = yz + xz'$$

K-Maps of 4 Variables:

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)

		y			
		00	01	11	10
wx \ yz	00	m_0 $w'x'y'z'$	m_1 $w'x'y'z$	m_3 $w'x'yz$	m_2 $w'x'yz'$
	01	m_4 $w'xy'z'$	m_5 $w'xy'z$	m_7 $w'xyz$	m_6 $w'xyz'$
w \ x	11	m_{12} $wxy'z'$	m_{13} $wxy'z$	m_{15} $wxyz$	m_{14} $wxyz'$
	10	m_8 $wx'y'z'$	m_9 $wx'y'z$	m_{11} $wx'yz$	m_{10} $wx'yz'$

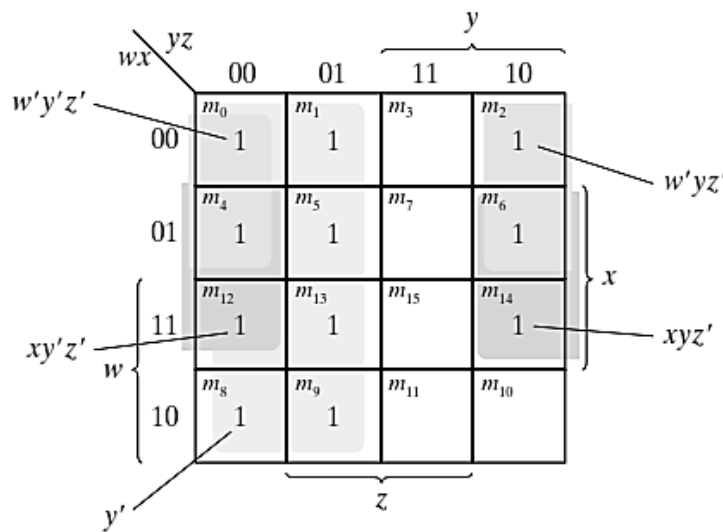
Annotations: x points to the right half (columns 11, 10), z points to the bottom half (rows 11, 10).

(b)

- ❖ Simplify the boolean function

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

(May 2011, 2012, 2014, 2017), (Dec 2014, Dec 2015)



$$\text{Note: } w'y'z' + w'yz' = w'z'$$

$$xy'z' + xyz' = xz'$$

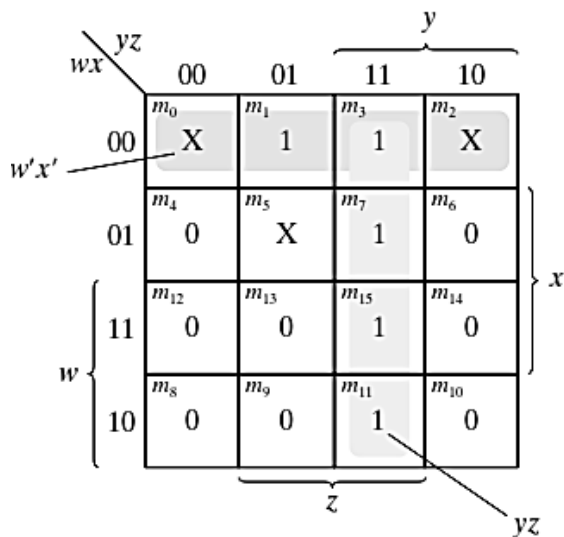
$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$$

- ❖ Simplify the Boolean function

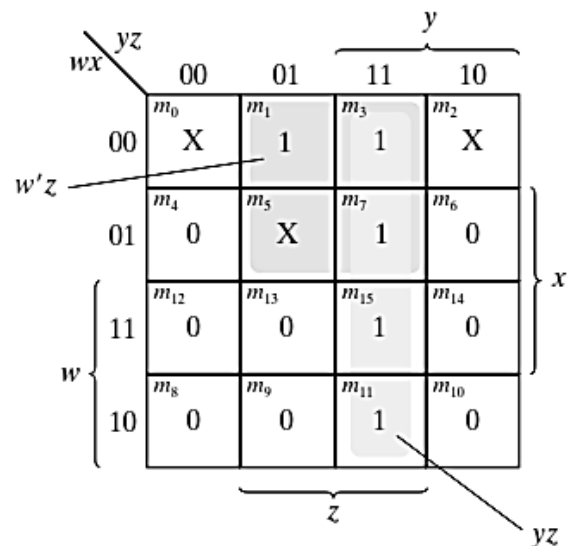
$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

which has the don't-care conditions

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$



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



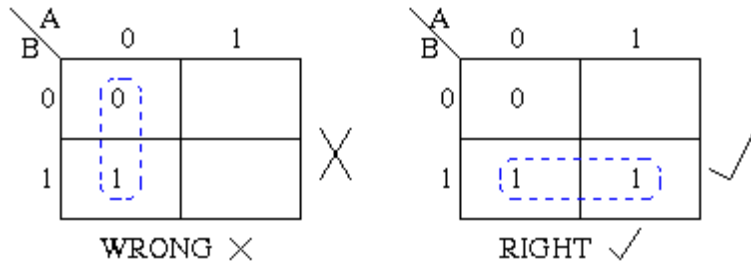
$$(b) F = yz + w'z$$

Note:

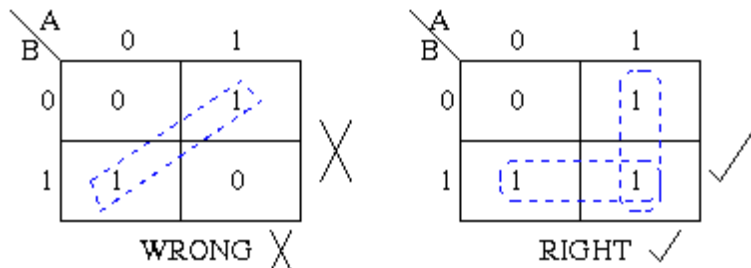
Karnaugh Maps - Rules of Simplification

The Karnaugh map uses the following rules for the simplification of expressions by *grouping* together adjacent cells containing *ones*

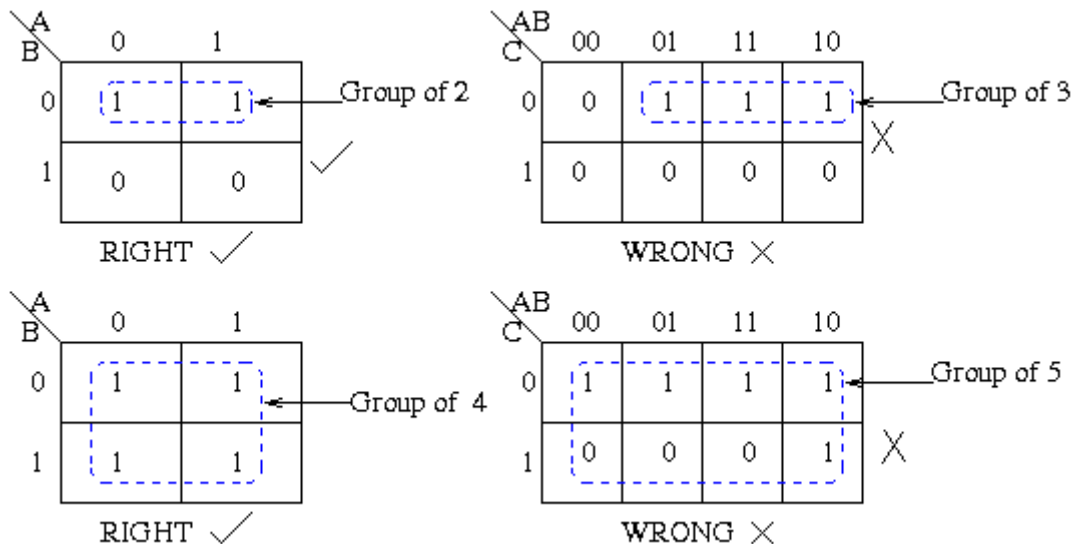
- Groups may not include any cell containing a zero



- Groups may be horizontal or vertical, but not diagonal.



- Groups must contain 1, 2, 4, 8, or in general 2^n cells. That is if $n = 1$, a group will contain two 1's since $2^1 = 2$. If $n = 2$, a group will contain four 1's since $2^2 = 4$.



- Each group should be as large as possible.

$\backslash AB$ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

RIGHT ✓

$\backslash AB$ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

WRONG ✗

(Note that no Boolean laws broken, but not sufficiently minimal)

- Each cell containing a *one* must be in at least one group.

$\backslash AB$ C	00	01	11	10
0	0	0	1	1
1	0	0	0	1

Group I
Group II
1 present in at least one group.

- Groups may overlap.

$\backslash AB$ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups overlapping.
RIGHT ✓

$\backslash AB$ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups not overlapping.
WRONG ✗

- There should be as few groups as possible, as long as this does not contradict any of the previous rules.

$\backslash AB$ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

RIGHT ✓

$\backslash AB$ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

WRONG ✗

Summary:

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

Don't care combination:

In certain digital systems, some input combinations never occur during the process of normal operation because those input conditions are guaranteed never to occur. Such input combinations are don't care conditions.

Completely specified functions:

If a function is completely specified, it assumes the value 1 for some input combinations and the value 0 for others.

Incompletely specified functions:

There are functions which assume the value 1 for some combinations and 0 for some other and either 0 or 1 for the remaining combinations. Such a functions are called incompletely specified .

Prime Implicants:

A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map. If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be essential.

Quine-McCluskey (or) Tabulation Method

Minimization of Logic functions:

Steps:

- ✓ A set of all prime implicants of the function must be obtained.
- ✓ From the set of prime implicants, a set of essential implicants must be determined by preparing a prime implicant chart.
- ✓ The minterm which are not covered by the essential implicants are taken into consideration and a minimum cover is obtained from thr remaining prime implicants.

Example:

Simplify the boolean function $F(A,B,C,D) = \sum m(1,3,6,7,8,9,10,12,14,15) + \sum m(11,13)$ using Quine McClusky method.
(May 2018, Dec 2017, Dec2016, May 2016)

Step:1

Minterms	Binary representation	Minterms	Binary representation
m_1	0 0 0 1	m_1	0 0 0 1 ✓
m_3	0 0 1 1	m_8	1 0 0 0 ✓
m_6	0 1 1 0	m_3	0 0 1 1 ✓
m_7	0 1 1 1	m_6	0 1 1 0 ✓
m_8	1 0 0 0	m_9	1 0 0 1 ✓
m_9	1 0 0 1	m_{10}	1 0 1 0 ✓
m_{10}	1 0 1 0	m_{12}	1 1 0 0 ✓
m_{12}	1 1 0 0	m_7	0 1 1 1 ✓
m_{14}	1 1 1 0	m_{14}	1 1 1 0 ✓
m_{15}	1 1 1 1	dm_{11}	1 0 1 1 ✓
dm_{11}	1 0 1 1	dm_{13}	1 1 0 1 ✓
dm_{13}	1 1 0 1	m_{15}	1 1 1 1 ✓

Step:2

Minterms	Binary representation	Minterms	Binary representation
1, 3	0 0 - 1 ✓	1, 3, 9, 11	- 0 - 1
1, 9	- 0 0 1 ✓	8, 9, 10, 11 ✓	1 0 - -
8, 9	1 0 0 - ✓	8, 10, 12, 14	1 - - 0
8, 10	1 0 - 0 ✓		
8, 12	1 - 0 0 ✓	6, 7, 14, 15 ✓	- 1 1 -
3, 7	0 - 1 1 ✓		
3, 11	- 0 1 1 ✓	12, 13, 14, 15	1 1 - -
6, 7	0 1 1 - ✓		
6, 14	- 1 1 0 ✓		
9, 11	1 0 - 1 ✓		
9, 13	1 - 0 1 ✓		
10, 14	1 - 1 0 ✓		
10, 11	1 0 1 - ✓		
12, 14	1 1 - 0 ✓		
12, 13	1 1 0 - ✓		
7, 15	- 1 1 1 ✓		
14, 15	1 1 1 - ✓		

Step:3

Prime implicants	Binary representation
1, 3, 9, 11 ($\bar{B}D$)	- 0 - 1
8, 9, 10, 11, 12, 13, 14, 15 (A)	1 - - -
6, 7, 14, 15 (BC)	- 1 1 -

Step:4

Prime implicants	m ₁	m ₃	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₂	m ₁₄	m ₁₅	dm ₁₁	dm ₁₃
1, 3, 9, 11 ($\bar{B}D$)	⊙	⊙				⊙					⊙	
8, 9, 10, 11, 12, 13, 14, 15					⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙
6, 7, 14, 15 (BC)			⊙	⊙					⊙	⊙		

$$\therefore F(A, B, C, D) = \bar{B}D + A + BC$$

[NOV/DEC 2021]

* Simplify the Boolean expression, $xy + x(wz + wz')$ to minimum number of literals. [NOV/DEC - 2021]

$$\begin{aligned}
 & xy + x(wz + wz') \\
 &= xy + xw(z + z') \quad \left\{ \begin{array}{l} z + z' = 1 \end{array} \right. \\
 &= xy + xw \\
 &= x(y + w)
 \end{aligned}$$

TWO MARKS

1. Define Digital Systems. Give an example.

A system which is processing discrete or digital signal is called as Digital System. Digital computer is the best example of a digital system.

2. What is meant by (i) bit, (ii) byte, (iii) Nibble?

- (i) A binary digit is called bit.
- (ii) A group of 8 bits are called Byte
- (iii) In binary number a group of four bits called Nibble.

3. Define Radix.

Radix specifies the number of symbols used for the corresponding number system. .

4. List the number systems used in digital systems.

- i) Decimal Number system
- ii) Binary Number system
- iii) Octal Number system
- iv) Hexadecimal Number system

5. Why is a hexadecimal number system called as an alphanumeric number system?

Hexadecimal number system has the base as 16 and therefore it requires 16 distinct symbols to represent the numbers. These are numerals 0 to 9 and alphabets A to F. Since both numeric digits and alphabets are used to represent the digits in hexadecimal number system, it is called as an alphanumeric number system.

6. What is 1's and 2's complement?

- ✓ The 1's complement of a binary number is the number that results when we change all 1's to zeros and the zeros to ones.
- ✓ The 2's complement is the binary number that results when we add 1 to the 1's complement.. It is used to represent negative numbers.

7. Convert Binary to hex decimal.

$$1011\ 0010\ 1111_2 = (1011)\ (0010)\ (1111)_2 = B\ 2\ F_{16}$$

8. Convert hex decimal to octal.

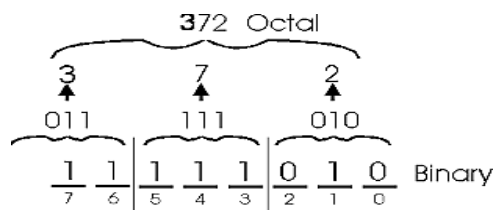
Hexadecimal	Octal
(5A8) ₁₆	= 0101 1010 1000 (Binary)
	= 010 110 101 000 (Binary)
Result	= 2 6 5 0 (Octal)

9. Convert binary to octal and vice versa with an example.

Binary to octal:

Example: $100\ 111\ 010_2 = (100)(111)(010)_2 = 4\ 7\ 2_8$

Octal to Binary:



10. Find 2's complement of $(1001)_2$.

Sol: 1 0 0 1 number
 0 1 1 0 ← 1's complement
 + 1 add 1
 —————
 0 1 1 1

11. Illustrate Diminished Radix Complement with an example?

(Dec- 2009)

Given a number N in base r having n digits, the $(r-1)$'s complement of N , i.e., its diminished radix complement, is defined as $(m-1) - N$.

The 9's complement of 546700 is $999999 - 546700 = 453299$.

The 9's complement of 012398 is $999999 - 012398 = 987601$.

12. What is BCD code (8421)?

A decimal number in BCD(8421) is the same as its equivalent binary number only when the number is between 0 and 9. A BCD number greater than 10 looks different from its equivalent binary number, even though both contain 1's and 0's. Moreover, the binary combinations 1010 through 1111 are not used and have no meaning in BCD.

13. BCD addition: $184 + 576 = ?$

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

14. What is excess-3 code?

This code assignment is obtained from the corresponding value of 4-bit binary code after adding 3 to the given decimal digit.

Example: 1000 of 8421 (BCD) = 1011 in Excess-3.

15. What is gray code?

The gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next.

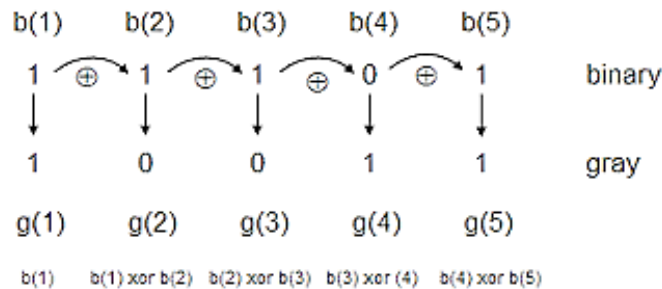
16. Write the applications of gray code.

(May 2012)

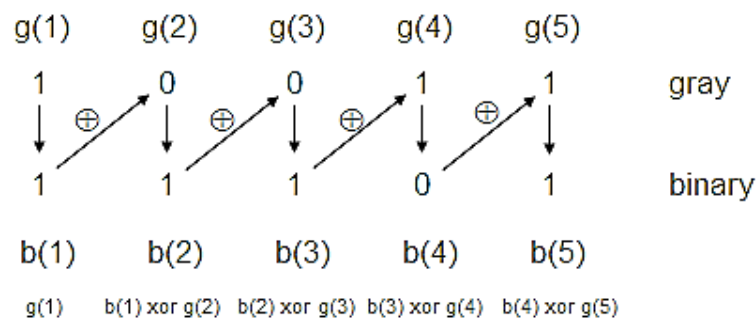
- ✓ Used in telegraphy
- ✓ Robust communication
- ✓ Error detection and correction

17. How to convert binary to gray code with an example?

(May 2017)



18. How to convert gray to binary code with an example?



19. What is Error detecting codes?

When data is transmitted from one point to another, like in wireless transmission, or it is just stored, like in hard disks and memories, there are chances that data may get corrupted. To detect these data errors, we use special codes, which are error detection codes.

20. What is Error correcting code

Error-correcting codes not only detect errors, but also correct them. This is used normally in Satellite communication, where turn-around delay is very high as is the probability of data getting corrupt.

21. What is Hamming codes?

Hamming code adds a minimum number of bits to the data transmitted in a noisy channel, to be able to correct every possible one-bit error.

It can detect (not correct) two-bit errors and cannot distinguish between 1-bit and 2-bits inconsistencies. It can't - in general - detect 3(or more)-bits errors.

22. What are the Two types of parity?

-Even parity: Checks if there is an even number of ones; if so, parity bit is zero. When the number of one's is odd then parity bit is set to 1.

-Odd Parity: Checks if there is an odd number of ones; if so, parity bit is zero. When the number of one's is even then parity bit is set to 1.

23. What is Parity Checker / Generator?

(Dec 2018)

The circuit that generates the parity bit in the transmitter is called a parity generator. The circuit that checks the parity in the receiver is called a parity checker.

24. What is EBCDIC codes?

- EBCDIC stands for *Extended Binary Coded Decimal Interchange*.
- For the different alphanumeric characters the code grouping in this code is different from the ASCII code. It is actually an 8-bit code and a ninth bit is added as the parity bit.

25. What is meant by Boolean algebra & Boolean Expression?

A system of algebra that operates on Boolean variables are called Boolean algebra.

The binary nature of Boolean algebra makes it useful for analysis, simplification and design of logic circuits are called Boolean Expression.

26. What are basic properties of Boolean algebra?

(Dec 2016)

The basic properties of Boolean algebra are commutative property, associative property and distributive Property

27. State the associative property of Boolean algebra.

The associative property of Boolean algebra states that the OR ing of several variables results in the same regardless of the grouping of the variables. The associative property is stated as follows:

$$A + (B + C) = (A + B) + C, \text{ ii). } A (B C) = (A B) C$$

28. State the commutative property of Boolean algebra.

The commutative property states that the order in which the variables are OR ed makes no difference. The commutative property is i). $A + B = B + A$ ii). $AB = BA$

29. State the distributive property of Boolean algebra.

The distributive property states that AND ing several variables and OR ing the result with a single variable is equivalent to OR ing the single variable with each of the several variables and then AND ing the sums.

The distributive property is

$$\text{i). } A + BC = (A + B)(A + C) \quad \text{ii). } A (B + C) = AB + AC$$

30. State De Morgan's theorem. (Dec 2019) (April/May 2011,2010,2013) Dec- 2017 ,(May 2018)

De Morgan suggested two theorems that form important part of Boolean algebra. They are

- 1) The complement of a product is equal to the sum of the complements. $(A \cdot B)' = A' + B'$
- 2) The complement of a sum term is equal to the product of the complements. $(A + B)' = A'B'$

31. Define Duality Theorem.

Duality property states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged. If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

32. List the important postulates of Boolean theorems.

The following are the important postulates of Boolean algebra:

1. $1 \cdot 1 = 1$, $0 + 0 = 0$.
2. $1 \cdot 0 = 0 \cdot 1 = 0$, $0 + 1 = 1 + 0 = 1$.
3. $0 \cdot 0 = 0$, $1 + 1 = 1$.
4. $1' = 0$ and $0' = 1$.

33. What is Boolean algebra?

Boolean algebra is an algebra that deals with binary variables and logic operations. A Boolean function described by an algebraic expression consists of binary variables, the constants 0 and 1, and the logic operation symbols.

34. What are the two forms of Boolean expression?

The two forms of Boolean expressions are:

- i). Sum of Products Form
- ii). Product of Sum Form

35. Define Minterm & Maxterm.

(May 2018)

The products of Boolean expression where all possible variables appear once in complement or un complement variables are called Minterm.

A sum terms in a Boolean expression where all possible variables appear once, in complement or un complement form are called Maxterm.

36. Define Product term.

The AND function is referred to as a product. The variable in a product term can appear either in complementary or uncomplimentary form. **Example: ABC'**

37. Define Sum term.

The OR function is referred to as a Sum. The variable in a sum term can appear either in complementary or uncomplimentary form. **Example: $A+B+C'$**

38. Define Sum of Product (SOP).

The logical sum of two or more logical product terms is called sum of product expression. It is basically an OR operation of AND operated variables. **Example: $Y=AB+BC+CA$**

39. Define Product of Sum (POS).**(May 2014)**

The logical product of two or more logical sum terms is called product of sum expression. It is basically an AND operation of OR operated variables. **Example:** $Y=(A+B).(B+C).(C+A)$

40. What is Canonical form?

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in *canonical form*.

41. What is Canonical SOP Expression?

The minterms whose sum defines the Boolean function are those which give the 1's of the function in a truth table.

42. What is Canonical POS Expression?

The Maxterms whose product defines the Boolean function are those which give the 1's of the function in a truth table.

43. What is meant by karnaugh map or K-Map method?**(May 2016)**

A karnaugh map or k map is a pictorial form of truth table, in which the map diagram is made up of cells, with each cell representing one minterm or maxterm of the function. This method provides a simple straight forward procedure for minimizing Boolean function.

44. Define Cell.

The smallest unit of a karnaugh map, corresponding to one rows of a truth table. The input variables are the cells coordinates and the output variable is the cells contents.

45. Define Pair, Quad, and Octet.

- i). Pair: A group of two adjacent cells in a karnaugh map. A pair cancels one variable in a K-Map simplification.
- ii). Quad: A group of four adjacent cells in a karnaugh map. A quad cancels two variables in a K-Map simplification.
- iii). Octet: A group of eight adjacent cells in a karnaugh map. A pair cancels three variable in a K-Map simplification.

46. What are called don't care conditions?**(April/May 2013)**

In some logic circuits certain input conditions never occur, therefore the corresponding output never appears. In such cases the output level is not defined, it can be either high or low.

These output levels are indicated by 'X' or 'd' in the truth tables and are called don't care conditions or incompletely specified functions.

47. State the limitations of karnaugh map.**(Dec 2017)**

- i. It is limited to six variable maps (i.e.) more than six variable involving expressions are not reduced.
- ii. The map method is restricted in its capability since they are useful for simplifying only Boolean expression represented in standard form.

48. list out the advantages and disadvantages of K-map method.

The advantages of the K-map method are:

- i). It is a fast method for simplifying expression up to four variables
- ii). It gives a visual method of logic simplification.
- iii). Prime implicants and essential prime implicants are identified fast.
- iv). Suitable for both SOP and POS forms of reduction
- v). It is more suitable for class room teachings on logic simplification.

The disadvantages of the K-map method are:

- i). It is not suitable for computer reduction .
- ii). K-maps are not suitable when the number of variables involved exceed four
- iii). Care must be taken to fill in every cell with the relevant entry, such as a 0, 1 (or) don't care terms.

49. What is tabulation method?

A method involving an exhaustive tabular search method for the minimum expression to solve a Boolean equation for more variables is called as a tabulation method.

50. What is a prime implicant?

A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map. They cannot be reduced further. (Or) A prime implicant is a group of minterms which cannot be combined with any other minterm or groups.

51. What is an essential prime implicant?

The Essential Prime Implicant is a prime implicant in which one or more minterms are unique, it contains at least one minterm which is not contained in any other prime implicant.

52. List out the advantages and disadvantages of Quine-Mc Cluskey method? (May 2015)

The advantages of Quine-Mc Cluskey method are:

- i). This is suitable when the number of variables exceed four.
- ii). Digital computers can be used to obtain the solution fast.
- iii). Essential prime implicants, which are not evident in K-map, can be clearly seen in the final results.

The disadvantages are:

- i). Lengthy procedure than K-map.
- ii). Requires several grouping and steps as compared to K-map.
- iii). It is much slower.
- iv). No visual identification of reduction process.
- v). The Quine Mc Cluskey method is essentially a computer reduction method.

53. What is a Logic gate?

Logic gates are the basic elements that make up a digital system. The electronic gate is a circuit that is able to operate on a number of binary inputs in order to perform a particular logical function.

54. Which gates are called as the universal gates? What are its advantages?

The NAND and NOR gates are called as the universal gates. These gates are used to perform any type of logic application.

55. Write the applications of gray code.

Used in telegraphy, robust communication and error detection & correction.

56. Show that the logical sum of all minterms of a Boolean function of 2 variables is 1

The combinations of minterms are $A'B' + A'B + AB' + AB$ (Nov/Dec 2009)

$$= A'(B + B') + A(B + B')$$

$$= A + A'$$

57. Find the complement of the functions $F1 = x'yz' + x'y'z$ and $F2 = x(y'z' + yz)$. (Dec 2015)

By applying De-Morgan's theorem.

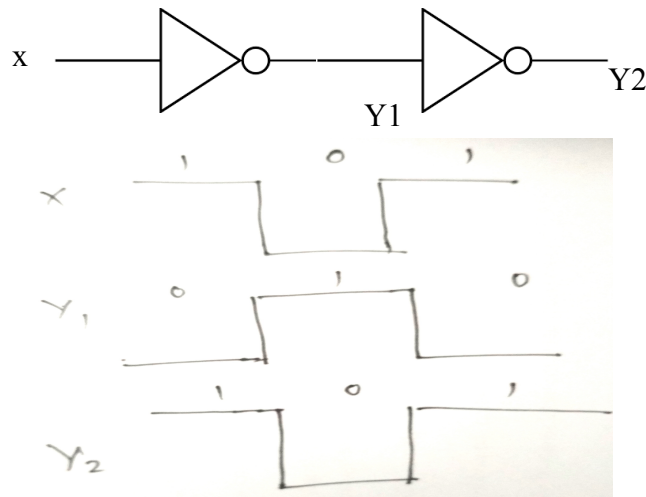
$$F1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$$

$$F2' = [x(y'z' + yz)]' = x' + (y'z' + yz)'$$

$$= x' + (y'z')'(yz)'$$

$$= x' + (y + z)(y' + z')$$

58. Sketch the the waveform of each inverter output in the given diagram. (Dec 2017)



59. Convert decimal 8723 to both BCD and ASCII code for ASCII an even parity bit is to be appended at left. (Dec 2019)

Decimal - 8723

BCD - 1000011100100011

ASCII- 11000011100100011

UNIT-2

[MAY 2019]

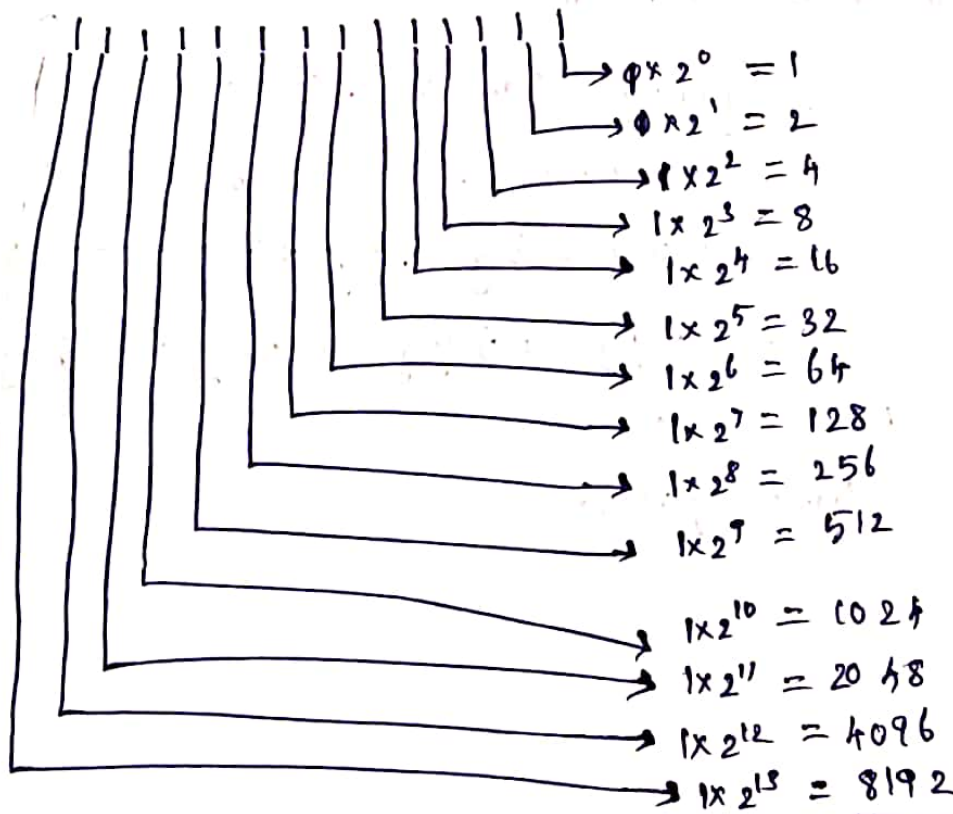
59. What is the largest binary number that can be expressed with 14 bits? Determine the equivalent decimal and hexadecimal numbers. [NOV / DEC 2021]

Sol:-

Largest Binary number in 14 bits.

11111111111111

Decimal:-



16383

Hexa Decimal:

00 1111 1111 1111 11
3 F F F

The answer is : 3FFF

2. Find the complement of $F = wx + yz$ and then show that $FF' = 0$

Sol:-

$$F = wx + yz$$

$$F' = \overline{wx + yz}$$

\therefore complement

$$= \overline{wx} + \overline{yz}$$

$$F = (\bar{w} + \bar{x}) \cdot (\bar{y} + \bar{z}) //$$

show that $FF' = 0$

$$= (wx + yz)(\bar{w} + \bar{x}) \cdot (\bar{y} + \bar{z})$$

$$= (w\bar{w}x + w\bar{x}\bar{x} + \bar{w}yz + \bar{x}yz) \cdot (\bar{y} + \bar{z})$$

$$\therefore A \cdot \bar{A} = 0$$

$$= (\bar{w}yz + \bar{x}yz) \cdot (\bar{y} + \bar{z})$$

$$= \bar{w}y\bar{y}z + \bar{x}y\bar{y}z + \bar{w}yz\bar{z} + \bar{x}yz\bar{z}$$

$$F\bar{F} = 0$$

hence proved

UNIT-1

NUMBER CONVERSION

Problems to be discussed:-

I. ① Decimal to Binary

$$* (48)_{10} = ()_2$$

$$\begin{array}{r} 2 \overline{) 48} \\ 2 \overline{) 24} - 0 \\ 2 \overline{) 12} - 0 \\ 2 \overline{) 6} - 0 \\ 2 \overline{) 3} - 0 \\ 1 - 1 \end{array}$$

$$\therefore (48)_{10} = (110000)_2 //$$

$$* (48.16)_{10} = ()_2$$

$$\begin{array}{r} 2 \overline{) 48} \\ 2 \overline{) 24} - 0 \\ 2 \overline{) 12} - 0 \\ 2 \overline{) 6} - 0 \\ 2 \overline{) 3} - 0 \\ 1 - 1 \end{array}$$

$$0.16 \times 2 = 0.32 \Rightarrow 0$$

$$0.32 \times 2 = 0.64 \Rightarrow 0$$

$$0.64 \times 2 = 1.28 \Rightarrow 1$$

$$0.28 \times 2 = 0.56 \Rightarrow 0$$

$$\therefore (48.16)_{10} = (110000.0010)_2 //$$

② Decimal to Octal

$$* (40.72)_{10} = ()_8$$

$$\begin{array}{r} 8 \overline{) 40} \\ 5 - 0 \end{array}$$

$$0.72 \times 8 = 5.76 \Rightarrow 5$$

$$0.76 \times 8 = 6.08 \Rightarrow 6$$

$$0.08 \times 8 = 0.64 \Rightarrow 0$$

$$0.64 \times 8 = 5.12 \Rightarrow 5$$

$$\therefore (40.72)_{10} = (50.5605)_8 //$$

③ Decimal to hexadecimal

$$* (32.15)_{10} = ()_{16}$$

$$\begin{array}{r} 16 \overline{) 32} \\ 2 - 0 \end{array}$$

$$0.15 \times 16 = 2.40 \Rightarrow 2$$

$$0.40 \times 16 = 6.40 \Rightarrow 6$$

$$0.40 \times 16 = 6.40 \Rightarrow 6$$

$$0.40 \times 16 = 6.40 \Rightarrow 6$$

$$\therefore (32.15)_{10} = (20.2666)_{16} //$$

④ Binary to Decimal

$$* (100110.10)_2 = ()_{10}$$

$$\begin{array}{rcl}
 1 & 0 & 0 & 1 & 1 & 0 \\
 | & | & | & | & | & | \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 1 \times 2^5 & \Rightarrow & 1 \times 32 = 32 \\
 0 \times 2^4 & \Rightarrow & 0 \times 16 = 0 \\
 0 \times 2^3 & \Rightarrow & 0 \times 8 = 0 \\
 1 \times 2^2 & \Rightarrow & 1 \times 4 = 4 \\
 1 \times 2^1 & \Rightarrow & 1 \times 2 = 2 \\
 0 \times 2^0 & \Rightarrow & 0 \times 1 = 0
 \end{array}$$

$$38$$

$$\begin{array}{rcl}
 1 & 0 \\
 | & | \\
 \downarrow & \downarrow \\
 1 \times 2^{-1} & \Rightarrow & 1 \times \frac{1}{2} = 0.5 \\
 0 \times 2^{-2} & \Rightarrow & 0 \times \frac{1}{4} = 0
 \end{array}$$

$$0.5$$

$$\therefore (100110.10)_2 = (38.5)_{10} //$$

⑤ Binary to octal

$$* (010110.01)_2 = ()_8$$

$$\begin{array}{c}
 010 / 110 . 010 \\
 \hline 2 \quad \quad \hline 6 \quad \quad \hline 2
 \end{array}
 \rightarrow \text{In binary to octal, we can separate the numbers - 3}$$

$$(010110.01)_2 = (26.2)_8 //$$

⑥ Binary to hexa decimal

$$* (100110.010)_2 = ()_{16}$$

$$\begin{array}{c}
 0010 / 0110 . 0100 \\
 \hline 2 \quad \quad \hline 6 \quad \quad \hline 4
 \end{array}
 \rightarrow \text{In binary to hexa decimal, we can separate the numbers - 4}$$

$$(100110.010)_2 = (26.4)_{16} //$$

⑦. Hexa decimal to Binary

$$* (17B.C2)_{16} = ()_2$$

1	7	B	.	C	2
↓	↓	↓		↓	↓
0001	0111	1011		1100	0010

$$(17B.C2)_{16} = (0001\ 0111\ 1011.\ 1100\ 0010)_2$$

⑧. Hexadecimal to octal

$$* (17B.C2)_{16} = ()_8$$

1	7	B	.	C	2
↓	↓	↓		↓	↓
0001	0111	1011		1100	0010
0	5	7		6	0

→ In octal we can separate the numbers-3

$$(17B.C2)_{16} = (0573.60A)_8 //$$

⑨. Hexadecimal to Decimal

$$* (17B.C2)_{16} = ()_{10}$$

$$\begin{aligned} (17B.C2)_{16} &= 1 \times 16^2 + 7 \times 16^1 + B \times 16^0 + C \times 16^{-1} + 2 \times 16^{-2} \\ &= (1 \times 256) + (7 \times 16) + 11 + \frac{12}{16} + 2 \times \frac{1}{256} \\ &= 256 + 112 + 11 + 0.75 + 0.0078 \end{aligned}$$

$$= 379.7578$$

$$= 379.7$$

$$\therefore (17B.C2)_{16} = (379.7)_{10} //$$

⑩ octal to Decimal

* $(64.5)_8 = ()_{10}$

$$\begin{array}{l} 64.5 \\ \swarrow \quad \downarrow \quad \searrow \\ 6 \times 8^1 \Rightarrow 48 \\ 4 \times 8^0 \Rightarrow 4 \\ 5 \times 8^{-1} \Rightarrow 0.625 \\ \hline 54.625 \end{array}$$

$(64.5)_8 = (54.625)_{10}$

⑪ Octal to Binary

* $(645.4)_8 = ()_2$

$$\begin{array}{ccccccc} 6 & 4 & 5 & . & 4 \\ \downarrow & \downarrow & \downarrow & & \downarrow \\ 110 & 100 & 101 & & 100 \end{array}$$

$\therefore (645.4)_8 = (110100101.100)_2$

⑫ octal to hexadecimal

* $(7810.4)_8 = ()_{16}$

$$\begin{array}{ccccccc} 7 & 8 & 1 & 0 & . & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow \\ 0111 & 1000 & 0001 & 0000 & & 0100 \end{array}$$

$\therefore (7810.4)_8 = (0111100000010000.0100)_{16}$

$\Rightarrow (F08.8)_{16}$

* II. SIGNED BINARY NUMBER:

3 methods

↳ First method (or) Sign Magnitude

↳ One's Complement

↳ Two's Complement

SIGN- MAGNITUDE:-

* use one bit to represent the sign

0 → positive (first bit)

1 → negative (first bit)

* Remaining bits are used to represent the magnitude.

Example:-

* Find decimal equivalent of following binary numbers assuming sign-magnitude representation of the binary num.

1) $(101100)_2$

$$(\boxed{1} \ 0 \ 1 \ 1 \ 0 \ 0)_2 = (-12)_{10}$$

sign bit is 1, which means the number is negative

2) $(1111)_2$

$$(\boxed{1} \ 1 \ 1 \ 1)_2 = (-7)_{10}$$

3) $(001000)_2$

$$(\boxed{0} \ 0 \ 1 \ 0 \ 0 \ 0) = (8)_{10} \therefore \text{sign bit} = 0$$

No. is positive.

* ONE'S COMPLEMENT REPRESENTATION:-

* The one's complement of a number is found by changing all the 0's (zero's) of the word to (one's) and all the 1's (one's) to 0's (zero's).

Example:-

* Find the one's complement of the following binary numbers.

① 0 100 11 00 1

to change the one's complement

\therefore 1 0 11 000 110.

② 110 110 10

To change the one's complement

\therefore 00100 101.

* TWO'S COMPLEMENT REPRESENTATION:-

* If 1 (one) is added to 1's complement of a binary number, the resulting number is known as 2's complement.

Example:-

* Find 2's complement:-

① 0 100 111 0

Number = 0 1 0 0 1 1 1 0

1's complement = 1 0 1 1 0 0 0 1

Add 1 (or)

2's complement = $\begin{array}{r} 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \\ +\ 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \end{array}$

III. ARITHMETIC OPERATION:-

Arithmetic operations in a computer are done using binary numbers and not decimal numbers and these take place in its arithmetic unit.

Addition

$$\begin{aligned} 0+0 &= 0 \\ 0+1 &= 0 \\ 1+0 &= 1 \\ 1+1 &= 10 \end{aligned}$$

Subtraction

$$\begin{aligned} 0-0 &= 0 \\ 1-0 &= 1 \\ 1-1 &= 0 \\ 10-1 &= 1 \end{aligned}$$

Multiplication

$$\begin{aligned} 0 \times 0 &= 0 \\ 0 \times 1 &= 0 \\ 1 \times 0 &= 0 \\ 1 \times 1 &= 1 \end{aligned}$$

Division

$$\begin{aligned} 0 \div 1 &= 0 \\ 1 \div 1 &= 1 \end{aligned}$$

Binary Addition:-

Example:-

$$\begin{array}{r} \text{MSB} \quad 1 \quad 1 \quad 1 \quad 1 \quad \text{LSB} \\ 1 \quad 0 \quad 1 \quad 0 \\ \hline 1 \quad 1 \quad 0 \quad 0 \quad 1 \end{array}$$

$$\begin{array}{r|l} \text{Decimal} & 2) \\ \hline 15 & 10.001 \\ 10 & 11.110 \\ \hline 25 & 101.111 \end{array}$$

Binary Subtraction:-

Example:-

$$\begin{array}{r} \text{MSB} \quad 1 \quad 1 \quad 0 \quad 1 \quad \text{LSB} \\ (-) \quad 1 \quad 0 \quad 0 \quad 1 \\ \hline 0 \quad 1 \quad 0 \quad 0 \end{array}$$

$$\begin{array}{r|l} \text{Decimal} & 2) \\ \hline 13 & 1001 \\ 9 & (-) \quad 101 \\ \hline 4 & 0100 \end{array}$$

$$\begin{array}{r} 3) \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\ (-) \quad \quad \quad 1 \quad 1 \\ \hline 1 \quad 1 \quad 0 \quad 1 \end{array}$$

$$\begin{array}{r} 4) \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \\ (-) \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \\ \hline 1 \quad 1 \quad 1 \end{array}$$

Binary Multiplication:

* Multiply

$$\begin{array}{r}
 1) \quad 1011 \times 1101 \\
 \hline
 \begin{array}{r}
 1011 \\
 00000 \\
 1011 \\
 1011 \\
 \hline
 10001111
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 2) \quad 1.01 \times 10.1 \\
 \hline
 \begin{array}{r}
 101 \\
 000 \\
 101 \\
 \hline
 11.001
 \end{array}
 \end{array}$$

Binary Division:

* Divide

$$\begin{array}{r}
 1) \quad 101 \overline{) 11001} \\
 \underline{101} \\
 00101 \\
 \underline{101} \\
 0
 \end{array}$$

$$\begin{array}{r}
 2) \quad 10010 \overline{) 11010.0000} \\
 \underline{10010} \\
 0100000 \\
 \underline{10010} \\
 011000 \\
 \underline{10010} \\
 010100 \\
 \underline{10010} \\
 0001000
 \end{array}$$

Home Work:-

1) $(1111)_2 + (1010)_2 = ?$

2) $(1010)_2 \text{ from } (1111)_2 = ?$

$$\begin{array}{r}
 111 \\
 1010 \\
 \hline
 ?
 \end{array}$$

3) $100110 \times 1001 = ?$

4) $11101 \div 1101$

Number systems:-

Problems: [Home work]

* DECIMAL NUMBER SYSTEMS:

↳ Decimal to Binary

1) $(120)_{10} = ()_2$

2) $(0.85)_{10} = ()_2$

3) $(108.3125)_{10} = ()_2$

* ↳ Decimal to octal

1) $(416.12)_{10} = ()_8$

(Next to radix point)

* multiply by 8

2) $(3964.63)_{10} = ()_8$

↳ Decimal to hexadecimal

1) $(106.0664)_{10} = ()_{16}$

* multiply by 16

2) $(4019.257)_{10} = ()_{16}$

* BINARY NUMBER SYSTEMS:

↳ Binary to Decimal

1) $(110111.1101)_2 = ()_{10}$

2) $(01010110.0110)_2 = ()_{10}$

↳ Binary to octal

1) $(11001.101011)_2 = ()_8$

2) $(11101101101)_2 = ()_8$

↳ Binary to Hexadecimal

$$1) (111101110111.111011)_2 = ()_{16}$$

$$2) (1011100011010111.1101101)_2 = ()_{16}$$

* HEXA DECIMAL NUMBER SYSTEMS:

↳ Hexa decimal to Binary

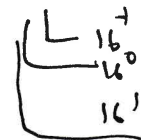
$$1) (ECF.EEE)_{16} = ()_2$$

$$2) (74F4.BAD)_{16} = ()_2$$

↳ Hexa decimal to Decimal

$$1) (2A.8)_{16} = ()_{10}$$

$$2) (AF.2F)_{16} = ()_{10}$$



↳ Hexa decimal to octal conversion

$$1) (2AB.9)_{16} = ()_8$$

$$2) (3FC.82)_{16} = ()_8$$

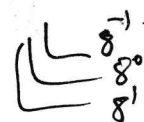
4bit → 1bit

* OCTAL NUMBER SYSTEMS:

↳ octal to decimal

$$1) (6327.45)_8 = ()_{10}$$

$$2) (326.216)_8 = ()_{10}$$



2) octal to Binary

$$1) (246.71)_8 = ()_2$$

$$2) (34.321)_8 = ()_2$$

SUBTRACTION USING COMPLEMENTS:-

* 1's complement Subtraction:

1) Subtract 11011_2 from 11101_2 using 1's complement.

$$\begin{array}{r} 11101 \\ (+) 00100 \rightarrow 1's \text{ complement of } 11011_2 \\ \hline \text{Carry } 1 \quad 00001 \\ \quad \quad \quad \rightarrow +1 \\ \hline 00010 \\ \hline \end{array}$$

Direct sub

$$\begin{array}{r} 11101 \\ - 11011 \\ \hline 00010 \\ \hline \end{array}$$

2) Subtract 1100_2 from 1011_2

$$\begin{array}{r} 1011 \\ (+) 0011 \rightarrow 1's \text{ comp of } 1100 \\ \hline \text{No carry } 1110 \rightarrow \text{Answer in } 1's \text{ complement in opposite sign} \\ \hline \text{Ans } \boxed{-0001} \end{array}$$

Home Work:-

* Subtract 101101_2 from 110011_2 using 1's complement.

* Subtract 1101_2 from 1010_2 using 1's complement.

* 2's Complement Subtraction:

1) Subtract $(1010)_2$ from $(1111)_2$ Using 2's complement.

Sol.

$$1010$$

2's complement $\rightarrow 1 + 1$

$$0101 + 1 \Rightarrow 0110$$

$$1111$$

$0110 \rightarrow 2's \text{ comp.}$

$$\begin{array}{r} \text{Discard} \swarrow \\ \text{carry} \swarrow \\ \hline 10101 \\ \hline \end{array}$$

The answer is 0101_{11}

2) Subtract $(1010)_2$ from $(1000)_2$ using 2's complement. (Dec 2018)

$$1010$$

2's complement $\rightarrow 0101 + 1 \Rightarrow 0110$

$$1000$$

$0110 \rightarrow 2's \text{ complement}$

$$\begin{array}{r} \text{no} \\ \text{carry} \\ \hline 1110 \\ \hline \end{array}$$

The answer is 1110_{11}

$$\begin{array}{r} 2's \text{ comp.} \\ 1110 \\ - 0001 + 1 \Rightarrow \\ - 0010 \\ \hline \end{array}$$

* Addition of two positive numbers:

$$1) \quad 8 + 9 = 17$$

* Find no. of bits required to represent the number.

$$2^{n-1} - 1 \geq \max(8, 9, 17) \therefore n$$

$$2^{6-1} - 1 \geq 17 \therefore n = 6$$

$$* (+8) \quad 0 \ 0 \ 1 \ 0 \ 0 \ 0 \quad n=6$$

$$(+9) \quad 0 \ 0 \ 1 \ 0 \ 0 \ 1$$

$$\underline{\underline{0 \ 1 \ 0 \ 0 \ 0 \ 1}}$$

* MSB = 0, \therefore The sign of number is positive.

* Result, $+(010001) \Rightarrow +17$

* Addition of Positive & Negative Numbers

$$2) \quad 8 + (-9) = -1$$

* Find the no. of bits required

$$2^{n-1} - 1 > \max(8, 9) \quad \therefore n=5$$

* Binary ~~Sub.~~ Sub.

$$\begin{array}{r} (+) \ 01000 \\ (-) \ 01001 \\ \hline \end{array}$$

$$\begin{array}{r} \therefore 01001 \\ \xrightarrow{2's \text{ comp}} 10111 \end{array}$$

*

$$\begin{array}{r} \therefore \ 01000 \\ (-) \ 10111 \\ \hline \underline{\underline{01111}} \end{array}$$

The MSB is 1, hence answer is negative.

Result = $-2's \text{ comp of } (11111)$

$$\Rightarrow -00001 \Rightarrow -1$$

* Addition of two Negative numbers:

$$(-8) + (-9) = -17$$

Step 1: Find the no. of bits required, $2^{n-1} - 1 > \max(8, 9, 17)$
 $n = 6$.

Step 2: 1st complement of (-8) \rightarrow 001000 \rightarrow 110111
 2nd comp of 110111 \rightarrow 001000

Step 3: 1st comp. of $(-9) \rightarrow$ 001001 \rightarrow 110110
 2nd comp \rightarrow 110110 + 1 \rightarrow 110111

Step 4: Binary addition

$$\begin{array}{r} 111000 \quad (-8) \\ 110111 \quad (-9) \\ \hline 1 \quad 101111 \end{array}$$

The MSB is 1, the answer is negative

Result = -2nd comp. of (101111)

$$\Rightarrow -(010000) + 1 \Rightarrow -(010001) \Rightarrow -17$$

home work:

* Consider the following examples

$$9 - 8 = 1$$

$$42 - 22 = ?$$

9's Complement:

decimal	0	1	2	3	4	5	6	7	8	9
9's comp	9	8	7	6	5	4	3	2	1	0

* Find 9's Complement of the following decimal numbers.

1) 19

2) 24

* sub. each digit in the number from 9 to get 9's comp.

1) 99
(-) 19

80 → 9's comp. of 19

2) 99

(-) 24

75 → 9's comp. of 24

9's Complement Subtraction:

1) subtract 06 from 18

(+) 18
93 → 9's comp. of 6

carry → 11
+ 1 → Add carry to Result.
12

home Work

* Sub. 49 from 34

* Sub. 23 from 39

10's Complement:

10's comp = 9's comp + 1

1) Find 10's complement of decimal numbers

a) 9

sol,

9 → 9's comp. of 9

0

+ 1

Add 1

→ 10's comp.

546 / 99

(-) 46 → 9's comp. of 46

53

add 1

1

54

→ 10's comp. 59

10's Complement subtraction:

* Sub - 4 from 9

$$\begin{array}{r} 9 \\ 6 \rightarrow 10's \text{ comp. of } 4 \\ \hline \text{drop} \quad \textcircled{1} \quad 5 \\ \text{arry.} \quad \hline \end{array}$$

* Sub - 32 from 69

$$\begin{array}{r} 69 \\ 68 \\ \hline \text{drop} \quad \textcircled{1} \quad 37 \\ \text{arry.} \quad \hline \end{array}$$

BINARY CODES:

* Excess-3 code:

convert $(643)_{10}$ to excess-3 code.

$$\begin{array}{r} \begin{array}{ccc} 6 & 4 & 3 \end{array} \\ \text{Add 3 to each bit} \quad \begin{array}{ccc} +3 & +3 & +3 \end{array} \\ \hline \text{Sum} \rightarrow \begin{array}{ccc} 9 & 7 & 6 \end{array} \\ \hline \end{array}$$

* Binary to gray code:

* convert $(10110)_2$

$$\begin{array}{ccccccc} 1 & \oplus & 0 & \oplus & 1 & \oplus & 1 & \oplus & 0 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1 & & 1 & & 1 & & 0 & & 0 \end{array}$$

the gray code is.
 $(1100)_2$

* Gray to Binary code:

* convert $(110101)_2$

$$\begin{array}{ccccccc} 1 & & 1 & & 0 & & 1 & & 0 & & 1 \\ \downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \downarrow \\ 1 & & 0 & & 0 & & 1 & & 1 & & 0 \end{array}$$

the binary code
is
 $(100110)_2$

Home Work Problems:

* Convert Binary to gray; $(10101101)_2$, $(1111)_2$

* Convert Gray to Binary, $(1010111)_2$, $(1011)_2$

* BCD addition;

$$184 + 576$$

Sol.

$$\begin{array}{r} 0001 \\ 0101 \\ \hline 0111 \end{array} \quad \begin{array}{r} 1000 \\ 0111 \\ \hline 10000 \end{array} \quad \begin{array}{r} 0100 \\ 0110 \\ \hline 1010 \end{array} \rightarrow \text{above 9}$$

$$\begin{array}{r} 0110 \\ \hline 0111 \end{array} \quad \begin{array}{r} 0110 \\ \hline 0110 \end{array} \quad \begin{array}{r} 0110 \\ \hline 0000 \end{array} \quad \text{add 6.}$$

The answers: 760

home work

BCD addition

$$* \cdot 1001 + 0100 \quad / \quad * \quad 00011001 + 00010100$$

IV. PROPERTIES OF BOOLEAN ALGEBRA:

* Commutative Property:-

Boolean addition is commutative is given by,

$$A+B = B+A, \quad A \cdot B = B \cdot A$$

* Associative Property:-

The associative property of addition is given by,

$$A+(B+C) = (A+B)+C$$

The associative law of multiplication is given by,

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

* Distributive Property:-

The boolean addition is distributive over boolean multiplication,

$$A+BC = (A+B)(A+C)$$

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

Proof:-

$$A+BC = A \cdot 1 + BC \quad \therefore (A \cdot 1 = A) \quad \text{by T1}$$

$$\Rightarrow A(1+B) + BC \quad \therefore (1+B=1)$$

$$\Rightarrow A \cdot 1 + AB + BC$$

$$\Rightarrow A \cdot (1+C) + AB + BC \quad \therefore (1+C=1)$$

$$\Rightarrow \underline{A \cdot 1} + A \cdot C + AB + BC$$

$$\Rightarrow A \cdot A + AC + AB + BC \quad \therefore (A \cdot A = A)$$

$$\Rightarrow A(A+C) + B(A+C)$$

$$A+BC \Rightarrow (A+B)(A+C)$$

hence proved.

* Absorption law: It is given by,

i) $A + AB = A$

Proof: $A + AB \Rightarrow A \cdot 1 + AB$

$\Rightarrow A(1+B)$

$\Rightarrow A \cdot 1 \quad \therefore [1+B = 1]$

$\boxed{A + AB = A}$

hence proved.

ii) $A \cdot (A+B) = A$

Proof: $A \cdot (A+B) = A \cdot A + AB$

$\Rightarrow A + AB$

$\Rightarrow A(1+B)$

$\Rightarrow A \cdot 1$

$\boxed{A \cdot (A+B) \Rightarrow A}$

hence proved.

$\therefore [A \cdot A = A]$

$\therefore [1+B = 1]$

$\therefore [A \cdot 1 = A]$

iii) $A + \bar{A}B = A+B$

Proof:-

$A + \bar{A}B \Rightarrow (A+\bar{A})(A+B) \rightarrow$ (use distributive law)

$\hookrightarrow A+BC = (A+B)(A+C)$

Proof :- $A + \bar{A}B \Rightarrow (A+\bar{A})(A+B)$

$\Rightarrow 1 \cdot (A+B)$

$\therefore [\bar{A}+A = 1]$ complement law.

$\boxed{A + \bar{A}B \Rightarrow A+B}$

hence proved.

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

Proof:-

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

$\Rightarrow 0 + AB$

$\therefore [A \cdot \bar{A} = 0]$

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

hence proved.

* Consensus Law:-

[MAX-2017]

i, $AB + \bar{A}C + BC = AB + \bar{A}C$

Proof:-

$$AB + \bar{A}C + BC \Rightarrow AB + \bar{A}C + BC \cdot 1$$

$$\Rightarrow AB + \bar{A}C + BC(A + \bar{A}) \quad \therefore [A + \bar{A} = 1]$$

$$\Rightarrow AB + \bar{A}C + ABC + \bar{A}BC$$

$$\Rightarrow AB(1 + C) + \bar{A}C(1 + B)$$

$$\boxed{AB + \bar{A}C + BC \Rightarrow AB + \bar{A}C} \quad \therefore [1 + B = 1, 1 + C = 1]$$

hence proved.

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

Proof:-

$$(A+B)(\bar{A}+C)(B+C) \Rightarrow (A+B)(\bar{A}+C)(B+C+0)$$

$$\Rightarrow (A+B)(\bar{A}+C)(B+C+A\bar{A}) \quad [\because A \cdot \bar{A} = 0]$$

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

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

$$\boxed{(A+B)(\bar{A}+C)(B+C) \Rightarrow (A+B)(\bar{A}+C)} \quad \therefore [A(A+B) = A]$$

by absorption law

hence proved.

NOTE:- Boolean Laws

1) $A+0 = A$, $A \cdot 1 = A$

2) $A+1 = 1$, $A \cdot 0 = 0$

3) $A+A = A$, $A \cdot A = A$

4) $A+\bar{A} = 1$, $A \cdot \bar{A} = 0$

5) $\bar{\bar{A}} = A$, $\bar{\bar{\bar{A}}} = \bar{A}$

* Principles of Duality:

one expression can be obtained from the other in each pair by replacing every '0' with '1' and every '1' with '0', every (+) with (.), and ~~every~~ (+). Any pair of expression satisfying this property is called dual expression. This characteristics of Boolean algebra is called principle of duality.

* DeMorgan's Theorems:

- 1) the complement of product is equal to the sum of the complement.

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

- 2) the complement of sum is equal to the product of the complement.

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

Proof:-

A	B	\bar{A}	\bar{B}	$A + B$	$A \cdot B$	$\overline{A + B}$	$\overline{A \cdot B}$	$\bar{A} \cdot \bar{B}$	$\bar{A} + \bar{B}$
0	0	1	1	0	0	1	1	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	1	0	0	0	0

hence proved.

BOOLEAN EXPRESSIONS:

Prob Lens:

1) Prove that $AB + BC + \bar{B}C = AB + C$

Sol:

$$AB + BC + \bar{B}C \Rightarrow AB + C(B + \bar{B})$$

$$\Rightarrow AB + C \cdot 1 \quad \therefore [B + \bar{B} = 1]$$

$$\boxed{AB + BC + \bar{B}C = AB + C}$$

hence proved.

2) Simplify the expression;

Sol:

$$\bar{A} \cdot B + A \cdot B + \bar{A} \cdot \bar{B} \Rightarrow (\bar{A} + A) \cdot B + \bar{A} \cdot \bar{B}$$

$$\Rightarrow 1 \cdot B + \bar{A} \cdot \bar{B} \quad \therefore [\bar{A} + A = 1]$$

$$\Rightarrow B + \bar{A} \cdot \bar{B}$$

$$\Rightarrow B + \bar{A} \quad \therefore [A + \bar{A} \cdot B = \bar{A} + B] \text{ absorption law.}$$

3) Simplify the expression, $A + A \cdot \bar{B} + \bar{A} \cdot B$.

Sol

$$A + A \cdot \bar{B} + \bar{A} \cdot B = A(1 + \bar{B}) + \bar{A} \cdot B$$

$$\Rightarrow A(1) + \bar{A} \cdot B$$

$$\Rightarrow A + B \quad \text{by absorption law.}$$

4) Complement the expression, $\bar{A}B + C\bar{D}$

Sol:

$$\overline{\bar{A}B + C\bar{D}} \Rightarrow \overline{\bar{A}B} + \overline{C\bar{D}}$$

$$\Rightarrow (\overline{\bar{A}}) + (\overline{C\bar{D}}) \Rightarrow (\bar{A} + \bar{B}) (\bar{C} + \bar{D})$$

$$\Rightarrow (A + \bar{B}) (\bar{C} + \bar{D}) //$$

5) Simplify the expression, $Y = (\bar{A} + B)(A + B)$

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

$$\Rightarrow B + A\bar{A}$$

$$\Rightarrow B + 0$$

$$Y = B //$$

$$[\therefore A + BC = (A + B)(A + C)]$$

$$[\therefore A - \bar{A} = 0]$$

6) Simplify the expression, $AB + \overline{AC} + A\overline{B}C (AB + C)$

(Dec 2017)

Sol:

$$AB + \overline{AC} + A\overline{B}C (AB + C) \Rightarrow AB + \overline{AC} + \overline{A\overline{B}C} \cdot AB + A\overline{B}C \cdot C$$

$$\Rightarrow AB + \overline{AC} + A\overline{B}C \quad [\because B \cdot \overline{B} = 0, C \cdot C = C]$$

$$\Rightarrow AB + \overline{A} + \overline{C} + A\overline{B}C$$

$$\Rightarrow \overline{A} + \overline{A}B + \overline{C} + A\overline{B}C \quad [\because A + \overline{A}B = A + B]$$

$$\Rightarrow \overline{A} + A(B + \overline{B}) + \overline{C}$$

$$\Rightarrow \overline{A} + A(1) + \overline{C} \quad \therefore [B + \overline{B} = 1]$$

$$\Rightarrow 1 + \overline{C}$$

$$\therefore [A + \overline{A} = 1]$$

$$\Rightarrow 1$$

$$\therefore [1 + \overline{C} = 1]$$

7) Simplify the expression,

$$\overline{A\overline{B}} + ABC + A(B + A\overline{B}) \Rightarrow \overline{A(\overline{B} + B)} + A(B + A) \quad \therefore [B + \overline{B} = 1]$$

$$\Rightarrow \overline{A(\overline{B} + B)} + AB + A \cdot A$$

$$\Rightarrow \overline{A\overline{B}} + \overline{AC} + AB + A$$

$$\Rightarrow \overline{A\overline{B}} + \overline{AC} + A(B + 1)$$

$$\Rightarrow \overline{A\overline{B}} + \overline{AC} + A \cdot 1 \quad [\because B + 1 = 1]$$

$$\Rightarrow (\overline{A\overline{B}}) \cdot (\overline{AC}) + A \quad \therefore [\overline{A+B} = \overline{A} \cdot \overline{B}]$$

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

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

$$\therefore [(A+B)(A+C) = A + BC]$$

$$\Rightarrow 1 + B\overline{C}$$

$$\therefore [A + \overline{A} = 1]$$

$$\Rightarrow 1$$

$$\therefore [1 + B\overline{C} = 1]$$

$$\Rightarrow 1$$

Home Inlook problems:

1) Simplify $Y = ABC + A\bar{B}C + AB\bar{C}$ to $Y = A(B+C)$

2) " $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$ ans: \bar{C}

3) $Y = \overline{(AB+C)}(\overline{A+B+C})$ ans: $(\bar{A}+\bar{B}+\bar{C})(A+B+C)$

4) $Y = \bar{A}C[\bar{A}BD] + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$ ans: $\bar{B}C + \bar{A}\bar{D}(B+C)$

5) Prove, $(A+B)(\bar{A}\bar{C}+C)(\bar{B}+AC) = \bar{A}B$

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

7) Simplify, $Y = ABC + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}B\bar{C}$ ans: $(\bar{A}+\bar{B})(C+\bar{C})$

8) Prove, $BCD + A\bar{C}\bar{D} + ABD = BCD + A\bar{C}\bar{D} + AB\bar{C}$

BOOLEAN FUNCTIONS:

* SOP :- eg. $Y = AB + BC + AC$

* POS :- eg. $Y = (A+B)(B+C)(C+A)$

* Minterm \rightarrow Product term \rightarrow eg. $\bar{A}\bar{B}\bar{C} \rightarrow 000$

	A	B	C	minterm		
	A	B	C	\bar{A}	\bar{B}	\bar{C}
m_0	0	0	0	\bar{A}	\bar{B}	\bar{C}
m_1	0	0	1	\bar{A}	\bar{B}	C
m_2	0	1	0	\bar{A}	B	\bar{C}
m_3	0	1	1	\bar{A}	B	C
m_4	1	0	0	A	\bar{B}	\bar{C}
m_5	1	0	1	A	\bar{B}	C
m_6	1	1	0	A	B	\bar{C}
m_7	1	1	1	A	B	C

Canonical SOP expression:-

$$\text{eg: } Y = \sum_m (0, 5, 6)$$

$$\Rightarrow m_0 + m_5 + m_6 \Rightarrow \bar{A}\bar{B}\bar{C} + A\bar{B}C + ABC$$

Procedure for obtaining canonical form:-

- 1) Examine each term in given logic fn. Retain if it is in min term.
- 2) Check missing variables, if not minterm, multiply by $(x + \bar{x})$, missing variable.
- 3) Multiply all the products & omit redundant terms.

Problems:-

- 1) Obtain the canonical SOP form of fn.

$$Y(A, B) = A + B$$

$$\Rightarrow A(B + \bar{B}) + B(A + \bar{A})$$

$$\Rightarrow AB + A\bar{B} + AB + \bar{A}B$$

$$Y(A, B) \Rightarrow \underline{AB + A\bar{B} + \bar{A}B} \quad \because [AB + AB = AB]$$

- 2) Obtain canonical SOP of the fn.

$$Y(A, B, C) = A + BC$$

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

$$\Rightarrow (A\bar{B} + A\bar{B} + AB + AB)(C + \bar{C}) + ABC + \bar{A}BC$$

$$Y(A, B, C) \Rightarrow \underline{A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC}$$

max term:

	A	B	C	max term
m_0	0	0	0	$A + B + C$
m_1	0	0	1	$A + B + \bar{C}$
m_2	0	1	0	$A + \bar{B} + C$
m_3	0	1	1	$A + \bar{B} + \bar{C}$
m_4	1	0	0	$\bar{A} + B + C$
m_5	1	0	1	$\bar{A} + B + \bar{C}$
m_6	1	1	0	$\bar{A} + \bar{B} + C$
m_7	1	1	1	$\bar{A} + \bar{B} + \bar{C}$

* Canonical POS expression

eg... $Y = \pi (0, 2, 4, 7)$

$\Rightarrow m_0 \cdot m_2 \cdot m_4 \cdot m_7$

$\Rightarrow (A+B+C)(A+\bar{B}+C)(\bar{A}+B+C)(\bar{A}+\bar{B}+\bar{C}) //$

* Procedure for obtaining canonical form:

step 1: Retain if it is in max term

step 2: missing variable in each term,
Add $(X + \bar{X})$, missing

step 3: expand using distributive property
& eliminate the redundant terms.

Problems:

1) obtain pos of expression.

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

$$\Rightarrow (A + \bar{B} + \underline{C \cdot \bar{C}})(\underline{A \cdot \bar{A}} + B + C)(\underline{A + B \cdot \bar{B}} + \bar{C})$$

$$\Rightarrow (A + \bar{B} + C)(\underline{A + \bar{B} + \bar{C}})(A + B + C)(\underline{\bar{A} + B + C})$$

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

$$Y \Rightarrow (A + \bar{B} + C)(A + \bar{B} + \bar{C})(A + B + C)(\bar{A} + B + C)(A + B + \bar{C})$$

2) Express the function $Y = A + \bar{B}C$ as (Dec 2018)

i, canonical sop and ii, canonical pos form

i) canonical sop

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

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

$$\Rightarrow ABC + AB\bar{C} + \underline{A\bar{B}C} + A\bar{B}\bar{C} + \underline{A\bar{B}C} + \bar{A}\bar{B}C$$

$$\Rightarrow ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}C$$

$$Y \Rightarrow m_7 + m_6 + m_5 + m_4 + m_1$$

$$Y = \sum m(1, 4, 5, 6, 7)$$

ii) canonical pos form

$$Y = A + \bar{B}C \Rightarrow$$

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

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

$$\Rightarrow (A + \bar{B} + C \cdot \bar{C})(A + \bar{B} + C)$$

$$\Rightarrow (A + \bar{B} + C)(A + \bar{B} + \bar{C})(A + B + C)(\underline{A + \bar{B} + C})$$

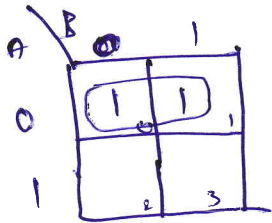
$$\Rightarrow m_2 \cdot m_3 \cdot m_0$$

$$\therefore Y = \prod (0, 2, 3)$$

KARNAUGH MAP:-

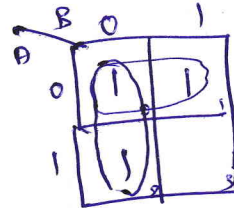
* 2 variable k-map (Sum of product)

1) $\Sigma(A, B) = \{0, 1\}$



$\therefore \Sigma(A, B) = \bar{A}$

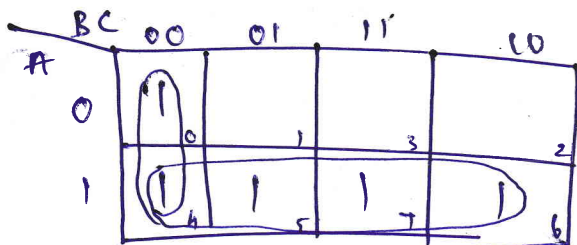
2) $\Sigma(A, B) = \{0, 1, 2\}$



$\Sigma(A, B) = \bar{B} + \bar{A}$

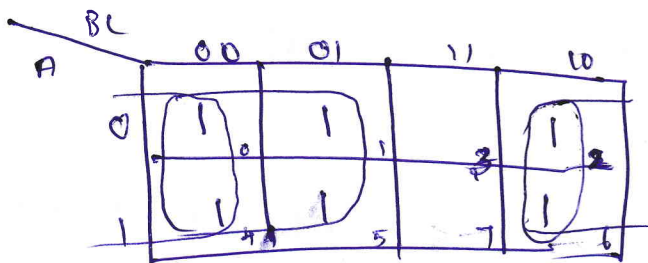
* 3 variable k-map (SOP)

1) $f(A, B, C) = \Sigma m(0, 4, 5, 6, 7)$



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

2) $f(A, B, C) = \Sigma m(0, 1, 2, 4, 5, 6)$



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

Home Work

1) $\gamma(A, B, C) = \Sigma m(1, 3, 5, 7)$

2) $\gamma(A, B, C) = \Sigma m(0, 1, 4, 5)$

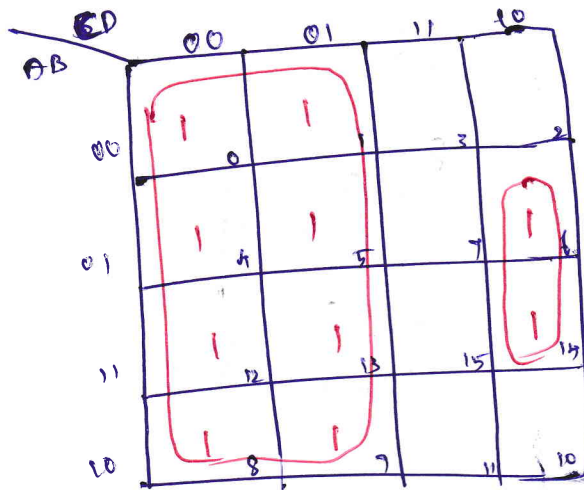
3) $\gamma(A, B, C) = \Sigma m(0, 2, 4, 6)$

Four Variable K-map (SOP) :-

[MAY-2017]
model

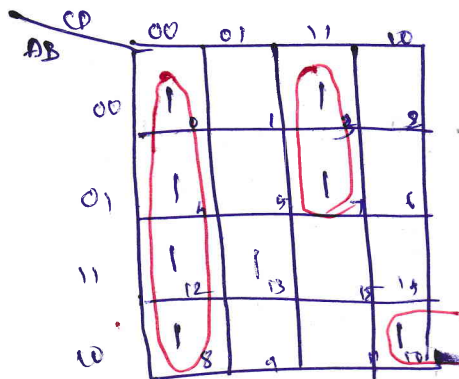
(Dec 2017)

1) $Y = \sum m(0, 1, 4, 5, 6, 8, 9, 12, 13, 14)$



$$Y = \bar{C} + BCD$$

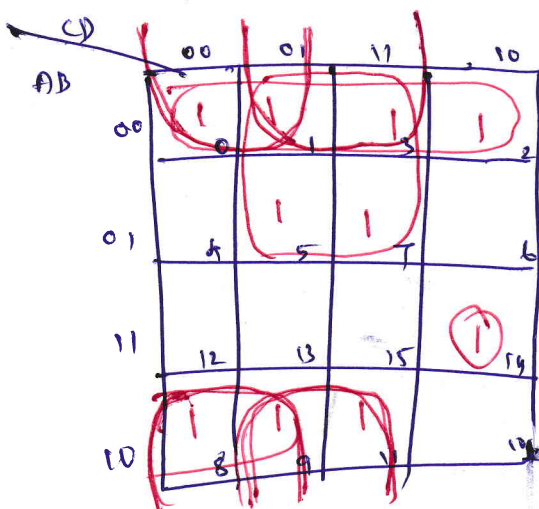
2) $Y(A, B, C, D) = \sum m(0, 3, 4, 7, 8, 10, 12, 13)$



$$Y = \bar{C}\bar{D} + \bar{B}CD + A\bar{B}\bar{D}$$

3) Minimize the following logic fn. using K map.

$$Y(A, B, C, D) = \sum m(0, 1, 2, 3, 5, 7, 8, 9, 11, 14)$$



$$Y(A, B, C, D) = \bar{A}\bar{B} + \bar{A}D + ABC\bar{D} + \bar{B}D + \bar{B}\bar{C}$$

* home work

simplify the following Boolean expression

$$Y(A, B, C, D) = \sum m(1, 2, 5, 6, 8, 9)$$

$$Y(A, B, C, D) = \sum m(0, 1, 2, 3, 4, 5, 6, 11)$$

Five Variable K-map (SOP)

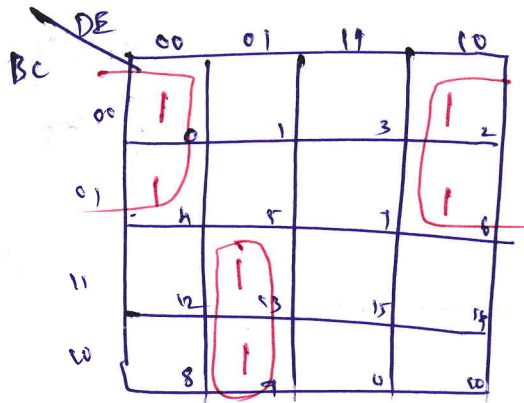
(Dec 2018)

[DEC-2014]

* Simplify the expression using K-map.

$$F(A, B, C, D, E) = (0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$$

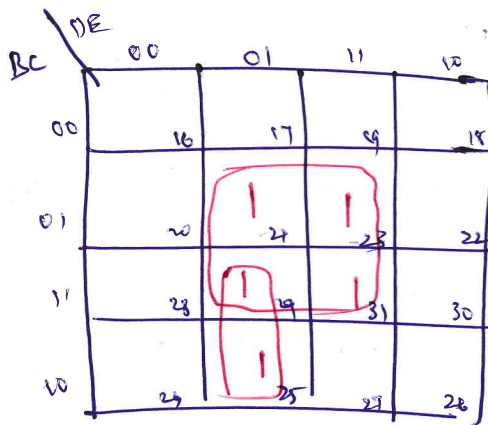
For \bar{A} :



$$\Rightarrow \bar{B}\bar{E} + B\bar{D}E$$

$$\Rightarrow \bar{A}\bar{B}\bar{E} + \bar{A}B\bar{D}E$$

For A



$$\Rightarrow CE + B\bar{D}E$$

$$\Rightarrow ACE + AB\bar{D}E$$

$$\therefore F(A, B, C, D, E) = \bar{A}\bar{B}\bar{E} + \bar{A}B\bar{D}E + ACE + AB\bar{D}E$$

Prob Cons:-

1) Minimize the following Boolean expressions using K-map.

i) $Y(A, B, C, D) = AB + C\bar{B} + C$

ii) $Y(A, B, C, D) = AB\bar{C} + BCD + B\bar{C}\bar{D}$

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

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

$$\Rightarrow ABC + AB\bar{C} + A\bar{B}C + \bar{A}\bar{B}C + (A + \bar{A})(BC + \bar{B}C)$$

$$\Rightarrow ABC + AB\bar{C} + A\bar{B}C + \bar{A}\bar{B}C + \underline{ABC} + \underline{A\bar{B}C} + \underline{\bar{A}BC} + \underline{\bar{A}\bar{B}C}$$

$$\Rightarrow m_7 + m_6 + m_5 + m_1 + m_3$$

$$\therefore Y(A, B, C) = \sum m(1, 3, 5, 6, 7)$$

	BC			
	00	01	11	10
A				
0	0	1	1	2
1	4	5	7	6

$$\therefore Y(A, B, C) = C + AB$$

2) Simplify using K-map.

[max-2017] model

$$Y = ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C + AB$$

Sol

$$Y = ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C(D + \bar{D}) + AB[C + \bar{C}](CD + \bar{D})$$

$$\Rightarrow ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + [ABC + A\bar{B}C](D + \bar{D})$$

$$\Rightarrow ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + \underline{ABCD} + \underline{AB\bar{C}\bar{D}} + \underline{AB\bar{C}D} + \underline{ABC\bar{D}}$$

$$\Rightarrow m_{15} + m_8 + m_{11} + m_{10} + m_{14} + m_{13} + m_{12}$$

$$\therefore Y = \sum m(8, 10, 11, 12, 13, 14, 15)$$

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

$$Y = AB + A\bar{D} + AC$$

* 3 Variable k-map (POS)

1) $f(A,B,C) = \Pi_m(1,5,6,7,0,11)$

	BC	BC	B+C	B+C	B+C
	00	01	11	10	
A	0	1	3	2	
A	4	5	7	6	

$\Rightarrow B \cdot \bar{A}$

$\bar{B} + A = f(A,B,C)$
 $\Rightarrow \bar{A} \cdot B = f(A,B,C)$

2) $f(A,B,C) = \Pi_m(1,3,5,7,6)$

	BC	B+C	B+C	B+C
	00	01	11	10
A	0	1	3	2
A	4	5	7	6

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

$\Rightarrow \dots$

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

Four Variable k-map (POS)

3) Reduce the following using k-map.

$Y(A,B,C,D) = \Pi_m(0,2,8,9,12,13,15)$

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

$\bar{Y} = \bar{A}\bar{C} + ABD + \bar{A}\bar{B}\bar{D}$

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

Prob 6m: minimize the expression using k-map.

Home Work

$Y(A,B,C,D) = (A+B+C)(\bar{A} + B + D)$ (MAY 2016)

$Y(A,B,C,D) = (A+B+C+D \cdot \bar{D})(\bar{A} + B + D + C \cdot \bar{C})$

$\Rightarrow (A+B+C+D)(\bar{A} + B + D)(\bar{A} + B + C + D)(\bar{A} + B + \bar{C} + D)$

$Y(A,B,C,D) = m_0 \cdot m_1 \cdot m_8 \cdot m_{10} \Rightarrow \Pi_m(0,1,8,10)$

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

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

$Y(A,B,C,D) = (A+B+C) \cdot (\bar{A} + B + D)$

Don't care conditions:

May 2019

1) Simplify the Boolean expressions.

i) $Y = \sum_m (1, 3, 7, 11, 15) + d(0, 2, 5)$ (SOP) \rightarrow minterm

CD \ AB	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

$Y = \bar{A}\bar{B} + CD$

ii) $Y = \prod M (4, 5, 6, 8, 12) \cdot d(1, 2, 3, 9, 11, 14)$ (POS) \rightarrow maxterm

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

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

$Y = (A + \bar{B}) \cdot (\bar{A} + C + D)$ (POS)

Home Work:

$Y(A, B, C, D) = \sum_m (1, 3, 5, 8, 9, 11, 15) + d(2, 13)$ [DEC-2014]

$Y(A, B, C, D) = \prod M (1, 2, 3, 8, 9, 10, 11, 14) \cdot d(7, 15)$

2) Simplify the Boolean fn. having don't care conditions in (i) SOP (ii) POS.

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

$d(A, B, C, D) = \bar{A}B\bar{C}D + ACD + A\bar{B}\bar{D}$

Sol.

$F(A, B, C, D) = \bar{A}\bar{B}\bar{D}(C + \bar{C}) + \bar{A}C\bar{D}(B + \bar{B}) + \bar{A}BC(D + \bar{D}) \rightarrow$ SOP

$\Rightarrow \bar{A}\bar{B}\bar{D}C + \bar{A}\bar{B}\bar{D}\bar{C} + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D}$

$\Rightarrow m_2 + m_0 + m_4 + m_3 + m_6$

$\therefore \sum_m (0, 2, 3, 6, 7)$

$d(A, B, C, D) = \bar{A}B\bar{C}D + ACD + A\bar{B}\bar{D} \Rightarrow \bar{A}B\bar{C}D + ACD(B + \bar{B}) + A\bar{B}\bar{D}(C + \bar{C})$

$\Rightarrow \bar{A}B\bar{C}D + ABCD + A\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}C\bar{D}$

$\Rightarrow m_5 + m_{15} + m_4 + m_{10} + m_8$

$\therefore \sum d(5, 8, 10, 11, 15)$

i) SOP

AB \ CD	CD			
	00	01	11	10
00	1	0	1	1
01	0	X	1	1
11	0	0	X	0
10	X	0	X	X

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

ii) POS

AB \ CD	CD			
	00	01	11	10
00	1	0	1	1
01	0	X	1	1
11	0	0	X	0
10	X	0	X	X

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

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

Home Work:

* Using the K-map. Simplify the Boolean expression.

i) minimal SOP ii) minimal POS expressions.

$$Y = \sum_m (0, 2, 3, 6, 7) + \sum_d (8, 10, 11, 15)$$

* Quine-McCluskey or Tabulation method of minimization

1) Find the minimal sop for the Boolean expression.

$$f = \sum (1, 2, 3, 7, 8, 9, 10, 11, 14, 15) \quad [\text{MAX. 2013, 15, 17}] \quad (\text{Dec 2019})$$

[NOV 2020]

sol.:

* Binary Representation of min terms.

min term	variable			
	A	B	C	D
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
7	0	1	0	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0
15	1	1	1	1

* Group of min terms for different number of 1's

No. of 1's	Min term	variable			
		A	B	C	D
1	1	0	0	0	1
	2	0	0	1	0
	8	1	0	0	0
2	3	0	0	1	1
	9	1	0	0	1
	10	1	0	1	0
3	7	0	1	1	1
	11	1	0	1	1
	14	1	1	1	0
4	15	1	1	1	1

*

2. cell combination:-

combination	A	B	C	D
(1, 3)	0	0	—	1
(1, 9)	—	0	0	1
(2, 3)	0	0	1	—
(2, 10)	—	0	1	0
(8, 9)	1	0	0	—
(8, 10)	1	0	—	0
(8, 7)	0	—	1	1
(3, 11)	—	0	1	1
(9, 11)	1	0	—	1
(10, 11)	1	0	1	—
(10, 14)	1	—	1	0
(7, 15)	—	1	1	1
(11, 15)	1	—	1	1
(14, 15)	1	1	1	—

* 4- cell combination:-

Combination	A	B	C	D
(1, 3, 9, 11)	—	0	—	1
(2, 3, 10, 11)	—	0	1	—
(8, 9, 10, 11)	1	0	—	—
(8, 7, 11, 15)	—	—	1	1
(10, 11, 14, 15)	1	—	1	—

* Prime Implicants Table:

Prime Implicants	Minterms									
	1	2	3	7	8	9	10	11	14	15
$(1, 3, 9, 11)^*$	x		x			x		x		
$(2, 3, 10, 11)^*$		x	x				x	x		
$(8, 9, 10, 11)$					x	x	x	x		
$(3, 7, 11, 15)$			x	x				x		x
$(10, 11, 14, 15)$							x	x	x	x
	✓	✓		✓	✓				✓	

Answer: $\therefore Y = \bar{B}D + \bar{B}C + CD + A\bar{B} + AC$

2) Find the minimal sop for the Boolean expression.

$F(A, B, C, D) = \sum m(0, 1, 2, 5, 6, 7, 8, 9, 10, 14)$

[MAIS 2017]
model.

Sol.

* Binary Representation of minterms.

minterm	variable			
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
14	1	1	1	0

* group of minterms of diff. no. of 1's.

No of 1's	minterm	variable			
		A	B	C	D
0	0	0	0	0	0
1	1	0	0	0	1
	2	0	0	1	0
	3	1	0	0	0
2	5	0	1	0	1
	6	0	1	1	0
	9	1	0	0	1
	10	1	0	1	0
3	7	0	1	1	1
	14	1	1	1	0

* 2-cell combination:

combination	A	B	C	D
(0,1) ✓	0	0	0	—
(0,2) ✓	0	0	—	0
(0,8) ✓	—	0	0	0
(1,5)	0	—	0	1
(1,9) ✓	—	0	0	1
(2,6) ✓	0	—	1	0
(2,10) ✓	—	0	1	0
(8,9) ✓	1	0	0	—
(8,10) ✓	1	0	—	0
(5,7)	0	1	—	1
(6,7)	0	1	1	—
(6,14) ✓	—	1	1	0
(10,14) ✓	1	—	1	0

* 4-cell combination:-

Combination	A	B	C	D
(0,1,8,9)	—	0	0	—
(0,2,8,10)	—	0	—	0
(0,8,1,9)	—	0	0	—
(0,8,2,10)	—	0	—	0
<hr/>				
(2,6,10,14)	—	—	1	0
(2,10,6,14)	—	—	1	0

* Prime Implicant Table.

Prime Implicants	Min Term									
	0	1	2	5	6	7	8	9	10	14
(0,1,8,9)*	x	x					x	x		
(0,2,8,10)	x		x				x		x	
(0,8,1,9)	x	x					x	x		
(0,8,2,10)	x		x				x		x	
(2,6,10,14)*			x		x				x	x
(2,10,6,14)			x		x				x	x
		✓			✓			✓		✓

* Prime Implicant Table.

Min terms.

Prime Implicants	0	1	2	5	6	7	8	9	10	14
✓ (0, 1, 8, 9)	x	x					x	x		
(0, 2, 8, 10)	x		x				x		x	
✓ (2, 6, 10, 14)			x		x				x	x
(1, 5)		x		x						
✓ (5, 7)				x		x				
(6, 7)					x	x				

$$Y = \bar{A}BD + C\bar{D} + \bar{B}\bar{C}$$

Simplify the following 5 variable Boolean expression using
mечusky method [DEC-2014, 2016]

$$F = \sum m(0, 1, 9, 15, 24, 29, 30) + d(8, 11, 31)$$

sol.

min term Binary Representation.

0	_____	00000
1	_____	00001
9	_____	01001
15	_____	01111
24	_____	11000
29	_____	11101
30	_____	11110
d8	_____	01000
d11	_____	01011
d31	_____	11111

No. of 1's	Min term	A	B	C	D	E
0	0	0	0	0	0	0
1	1	0	0	0	0	1
	8	0	1	0	0	0
2	9	0	1	0	0	1
	24	1	1	0	0	0
3	11	0	1	0	1	1
4	15	0	1	1	1	1
	29	1	1	1	0	1
	30	1	1	1	1	0
5	31	1	1	1	1	1

* 2 cell combination.

combi	A	B	C	D	E
(0,1)	0	0	0	0	-
(0,8) ✓	0	-	0	0	0
(1,9) ✓	0	-	0	0	1
(8,24)	-	1	0	0	0
(9,11)	0	1	0	-	1
(11,15)	0	1	-	1	1
(15,31)	-	1	1	1	1
(29,31)	1	1	1	-	1
(30,31)	1	1	1	1	-

* A cell combination

combination	A	B	C	D	E
(0,8,1,9)	0	-	0	0	-

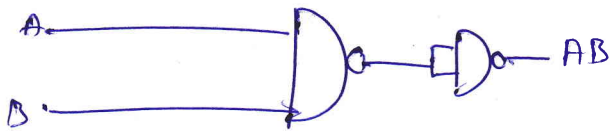
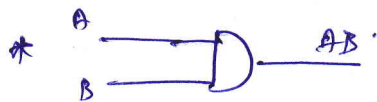
* Prime Implication Table.

Prime Implicant	0	1	9	15	24	29	30	31
(0, 8, 16)	x	x	x					
(0, 1)	x	x						
(8, 24) ✓					x			
(9, 11)			x					
(11, 15) ✓				x				
(15, 31) ✓				x				x
(29, 31) ✓						x		x
(30, 31) ✓							x	x

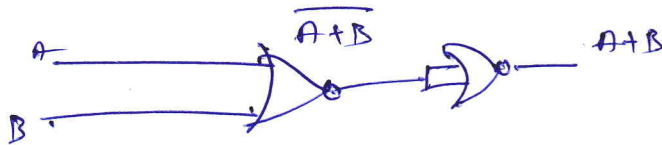
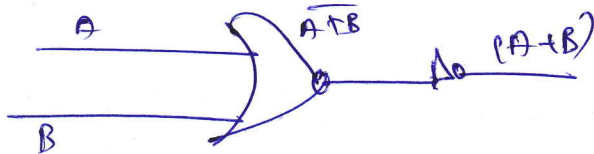
$$F \Rightarrow \overline{B}\overline{C}\overline{D}\overline{E} + \overline{A}BDE + B\overline{C}DE + ABCE + ABCD + \overline{A}\overline{C}\overline{D}$$

_____ x _____

* Convert AND to NAND gate:



* Convert OR gate to NOR gate:



1) draw the AOB logic

2) add bubble on ip of OR gate
to ip side of AND gate.

3) Add an inverter on each line
that received bubble (step 2)

4) eliminate double inversions

5) draw all NAND symbols

1) draw AOB logic.

2) Add bubble ip of AND gate
to ip of OR gate.

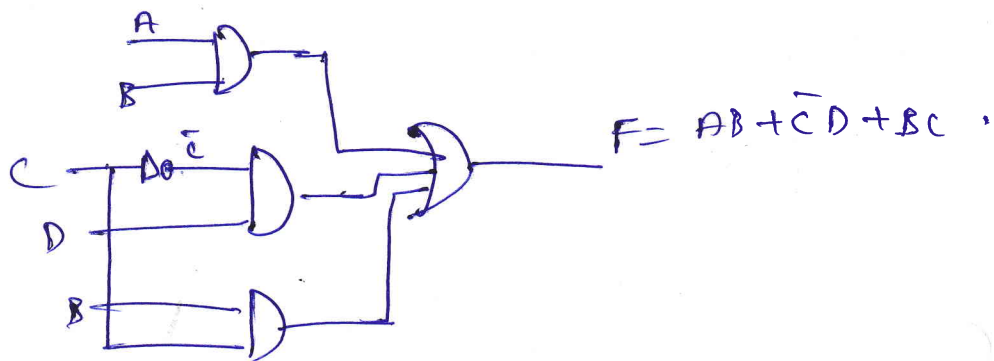
3) After subtract one inverter.
C receives bubble)

4) draw all NOR symbols.

Problems:

1) draw the logic circuit using gate. $F = AB + \bar{C}D + BC$.

sol.

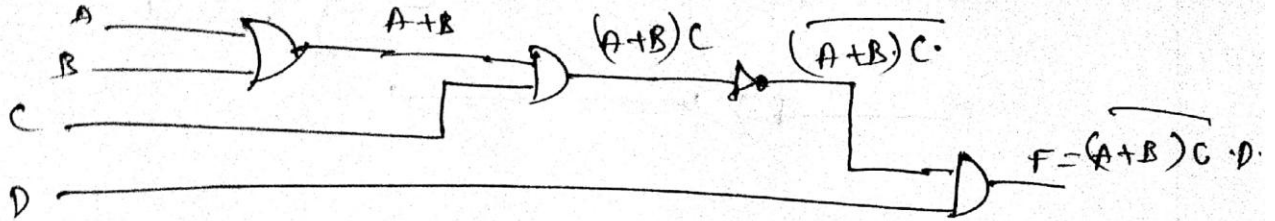


* Draw the logic circuit using only NOR gates. May 2019

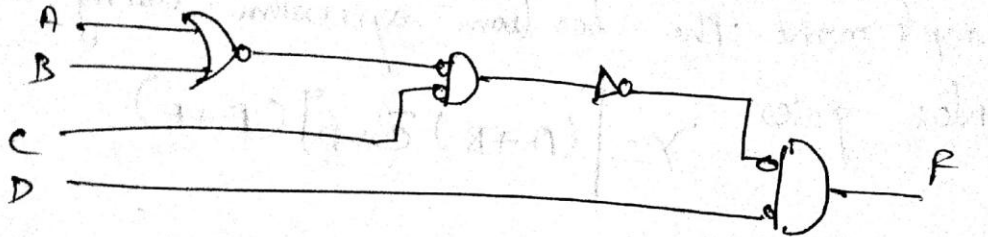
$$Y = \overline{(A+B)} C \cdot D$$

Show that if all the gate in a two - level OR-AND gate network are replaced by NOR gate, the output function does not change. [NOV 2020]

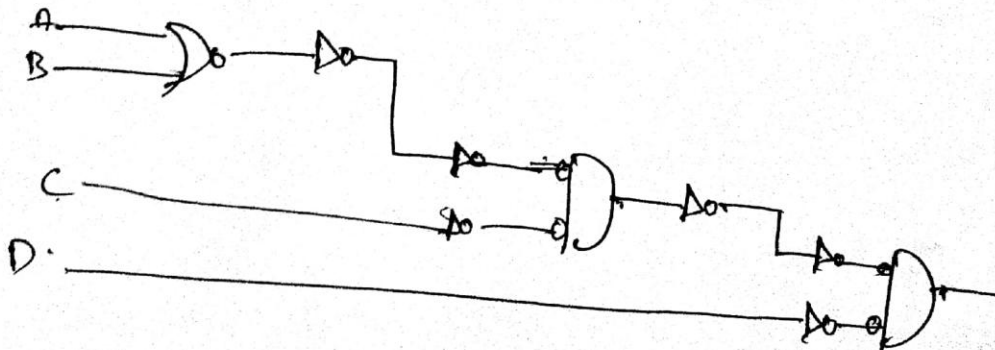
Step 1:- AOB logic.



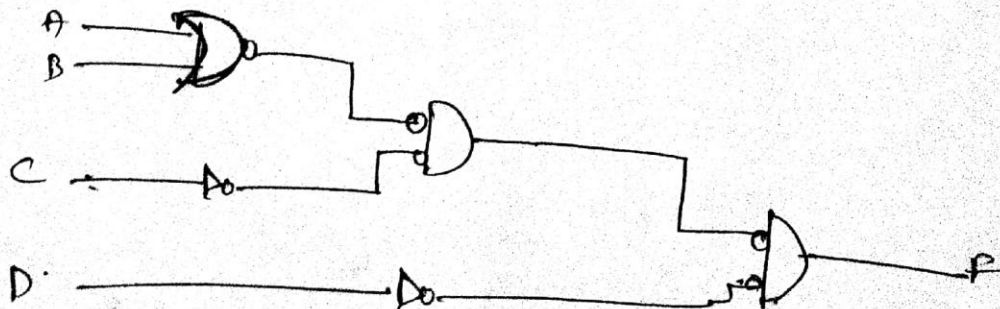
Step 2:- Add bubbles · o/p of OR gates, i/p of AND gates.



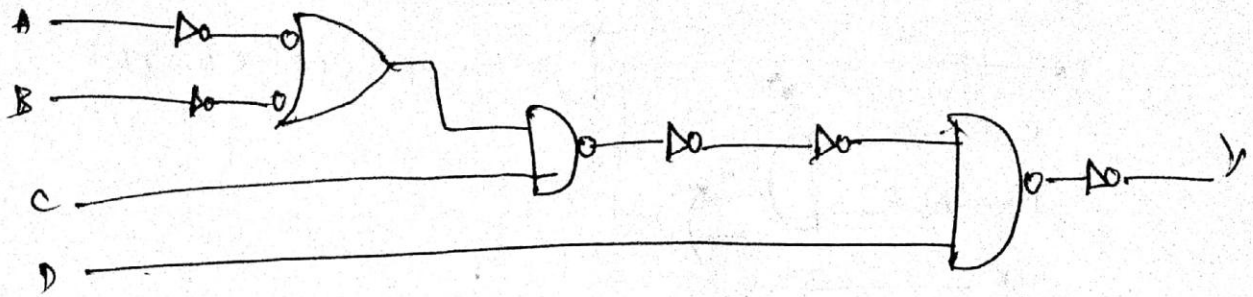
Step 3:- Add inverters. each line that received bubbles.



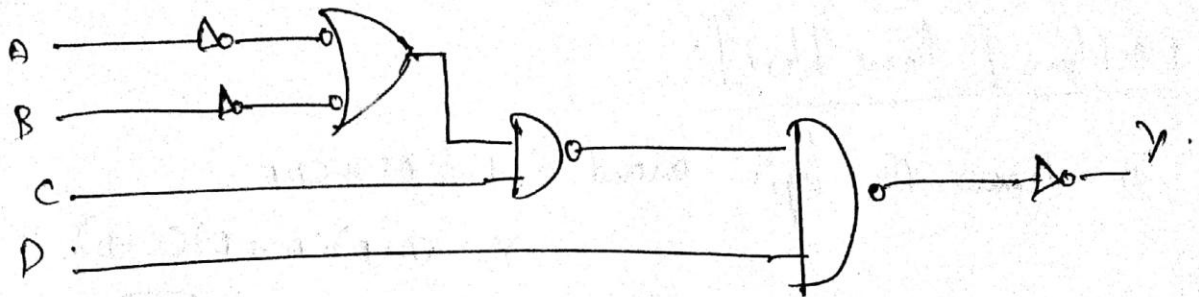
Step 4:- eliminate double inversions.



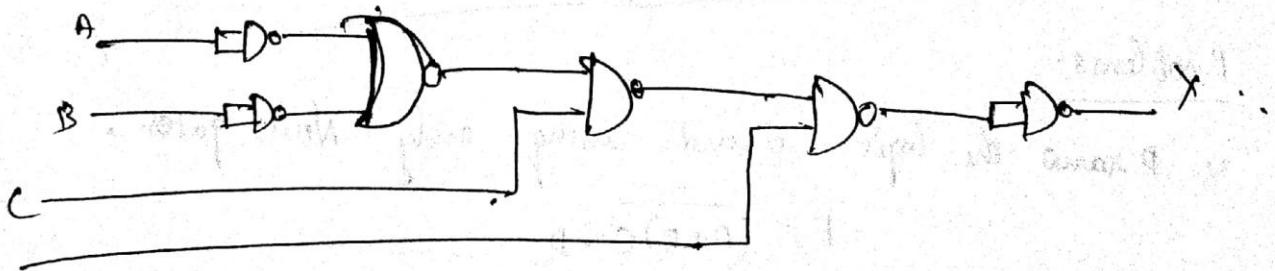
step 3:- Add inverters on each line that received bubble.



step 4:- Eliminate double inversions.



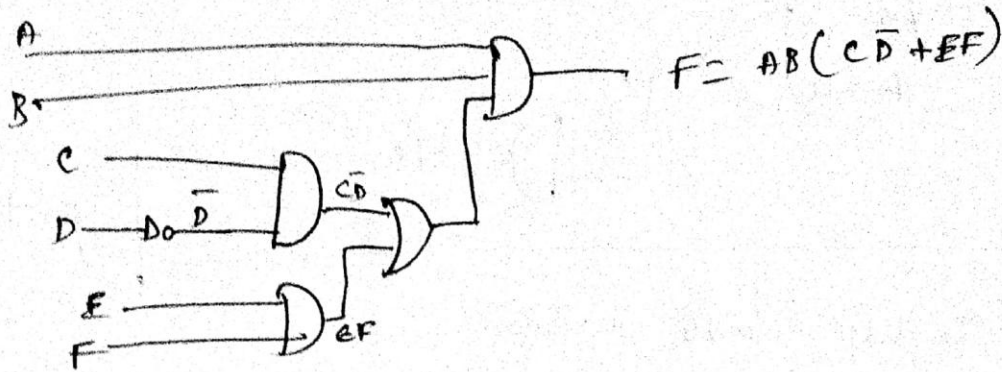
step 5:- Replace by NAND gates.



Home lab

Implement using NAND gates, $Y = (A + B + \bar{C}) D + EF$

2) $F = AB(C\bar{D} + EF)$



Problems [home work] :-

1) Draw the logic circuit. $F = AB + CDE$

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

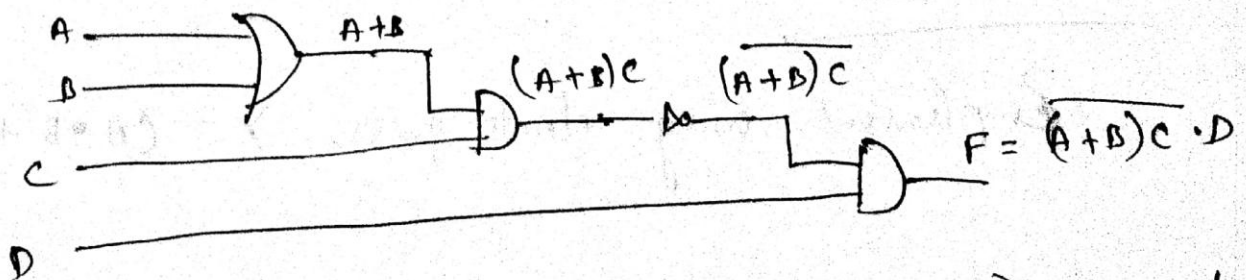
$$Y = ABC + A\bar{B}(\bar{A}\bar{C})$$

Problems :-

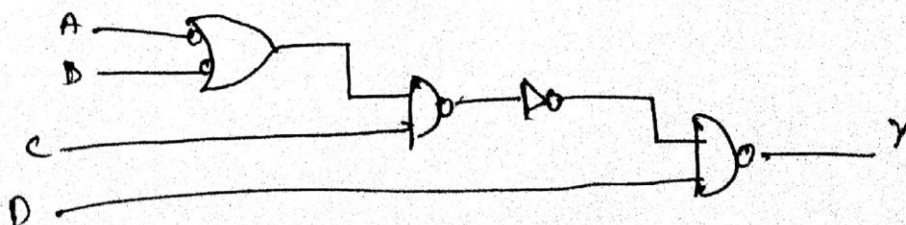
1) Draw the logic circuit using only NAND gates.

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

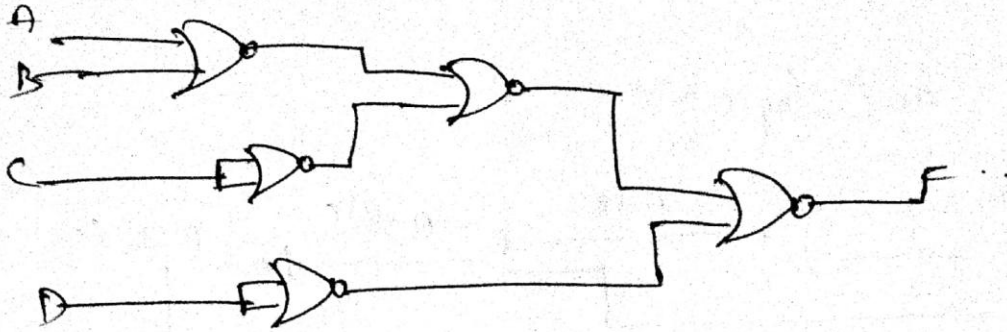
Step 1:- A OR logic.



Step 2:- Add bubbles on the o/p of the AND gates & i/p of OR gate.



steps draw nor diagram.



home labork:

Complement the boolean expression using only nor gates.

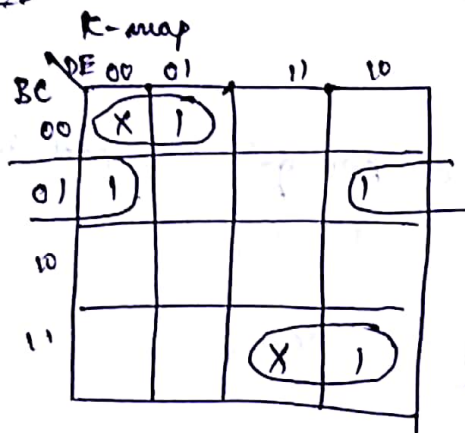
$$Y = [(A+B)(C+D)](E+F)$$

* Find the MSOP Representation for

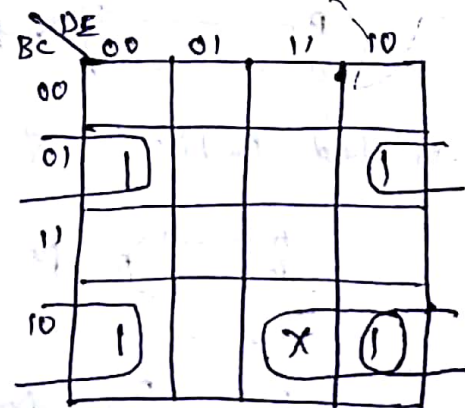
$$F(A, B, C, D, E) = \sum_m(1, 4, 6, 10, 20, 22, 24, 26) + \sum_d(0, 11, 16, 27)$$

using K-map. Draw the circuit of the minimal expression using NAND gates.

Sol:-



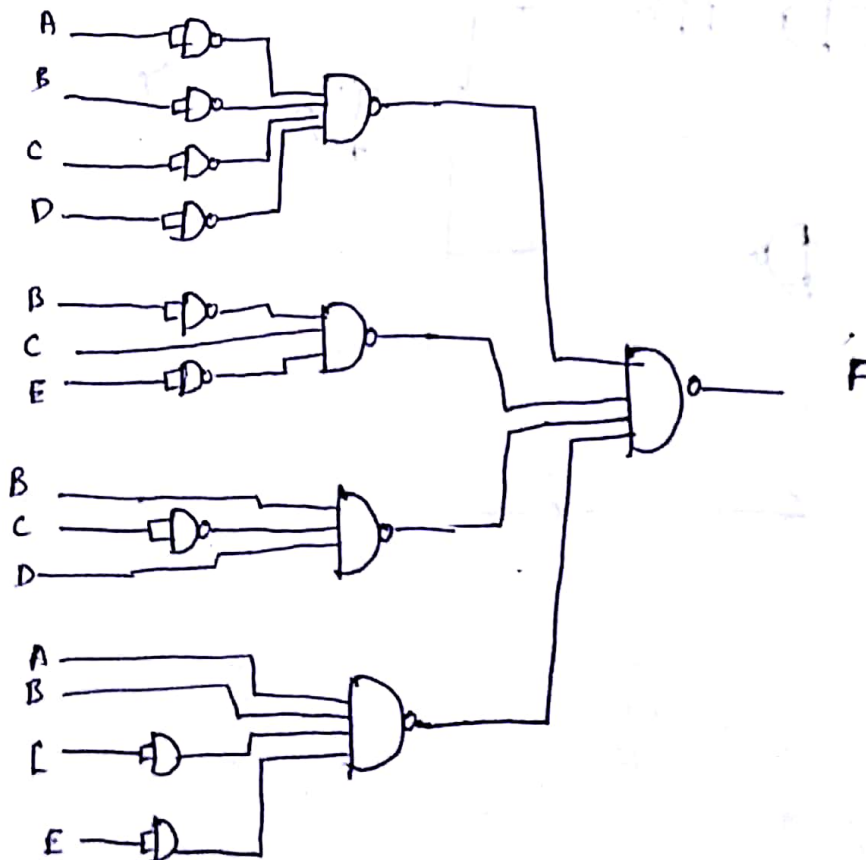
A=0



A=1

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

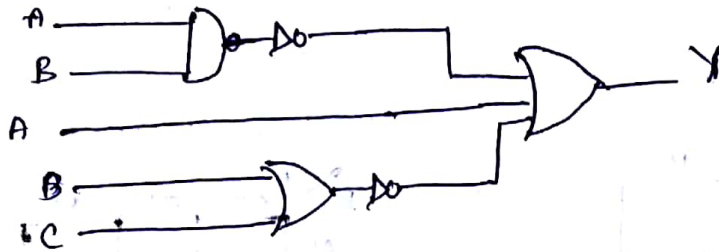
Logic Diagram:-



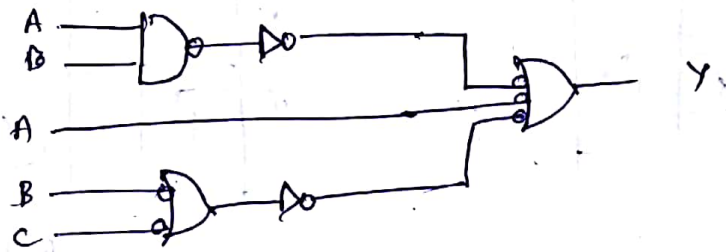
* Implement $Y = \overline{AB} + A + \overline{(B+C)}$ using NAND gates [DEC-2013]

Sol.

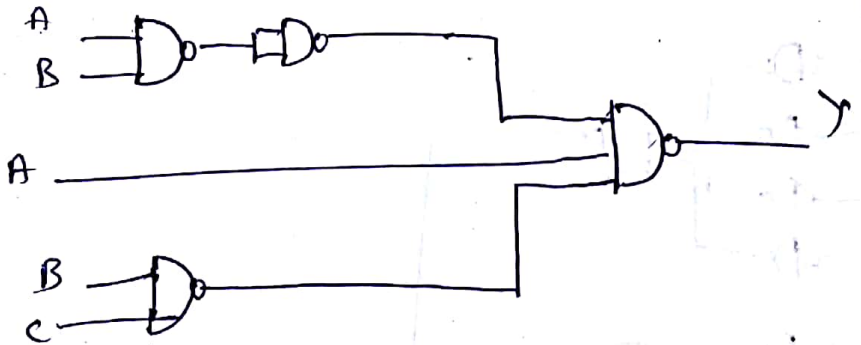
1) Draw AOE logic.



2) Add Bubble Tip of OR & o/p of AND gate & add inverters.



3) NAND Implementation.



————— X —————

* Determine the M SOP of Boolean expression.

[DEC-2018]

$$f(A, B, C, D) = \sum_m (1, 2, 3, 9, 12, 13, 14) + \sum_d (0, 7, 10, 15)$$

using Quine MC-cluskey method.

Step 1: Bit Representation.

Minterms	A	B	C	D	E
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
9	1	0	0	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
d0	0	0	0	0	
d7	0	1	1	1	
d10	1	0	1	0	
d15	1	1	1	1	

Step 2: 2' cell combination.

	Minterms	A	B	C	D
0	0	0	0	0	0
	1	0	0	0	1
1	2	0	0	1	0
	3	0	0	1	1
	9	1	0	0	1
2	12	1	1	0	0
	d10	1	0	1	0

min-terms	A	B	C	D
13	1	1	0	1
14	1	1	1	0
15	0	1	1	1
15	1	1	1	1

step 3: 2 cell combination

A-cell	variable			
	A	B	C	D
(0,1)✓	0	0	0	—
(0,2)✓	0	0	—	0
(1,3)✓	0	0	—	1
(1,9)	—	0	0	1
(2,3)✓	0	0	1	—
(2,10)	—	0	1	0
(13,15)	1	1	—	1
(14,15)	1	1	1	—
(7,15)	—	1	1	1

Step 4: A cell comb.

A-cell	variable			
	A	B	C	D
(0,1,2,3)	0	0	—	—
⊕				

Step 5: Primary Simplification Table.

Cell	minutems														
(0,1,2,3)✓	1	2	3	9	12	13,	14	0	7	10	15				
	x	x	x					x							
(1,9)✓	x			x											
(2,10)															
(13,15)✓															
(14,15)✓															
(7,15)✓															
12 ✓															

$$F = \overline{C}\overline{A}\overline{B} + \overline{B}\overline{C}D + \overline{A}\overline{B}D + ABD + ABC + BCD + AB\overline{C}D$$

SOLVED EXAMPLES

Example 1.52 Implement the following Boolean function with NAND-NAND logic

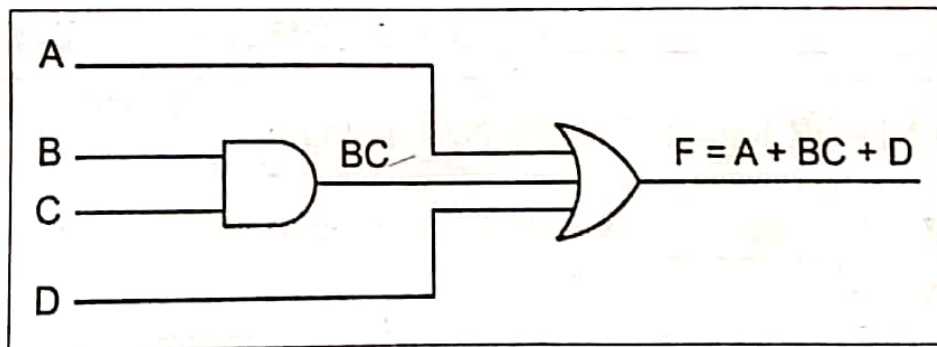
$$F = AB + ABC + \bar{A}BC + A\bar{B} + D \text{ (using only NAND gates).}$$

☺ **Solution:**

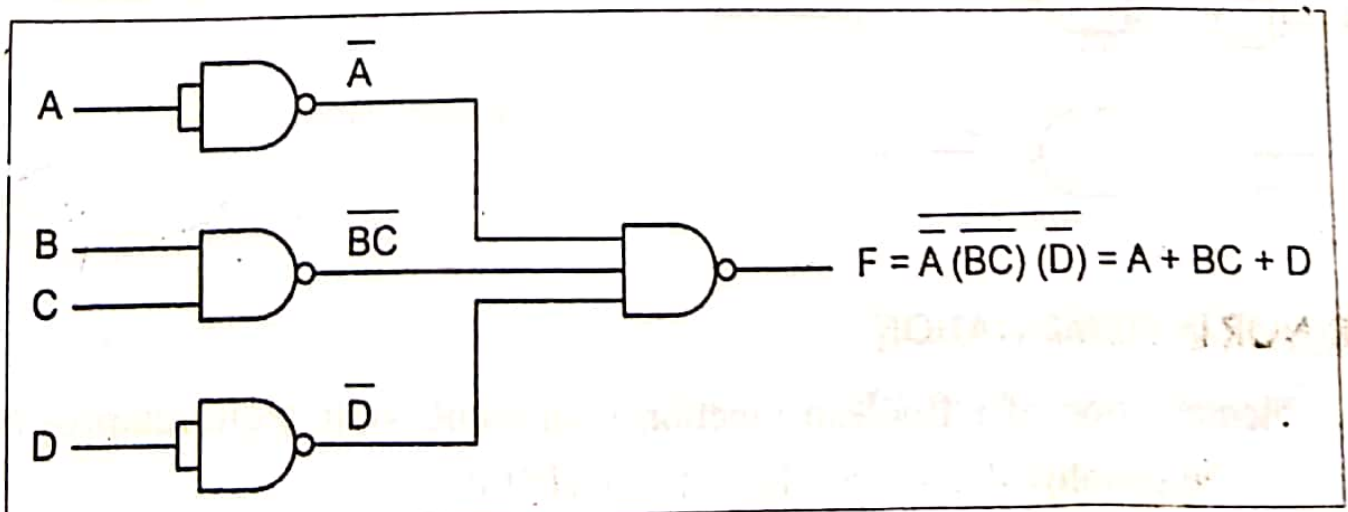
Step 1: Simplify the given Boolean function to get minimum number of literals.

$$\begin{aligned} F &= AB + ABC + \bar{A}BC + A\bar{B} + D \\ &= AB + BC(A + \bar{A}) + A\bar{B} + D \quad [A + \bar{A} = 1] \\ &= A(B + \bar{B}) + BC(A + \bar{A}) + D \\ \boxed{F} &= \boxed{A + BC + D} \end{aligned}$$

Step 2: Draw the AND-OR logic gate



Step 3: Convert AND-OR logic to NAND-NAND logic



Example 1.53 Implement the following Boolean function with NAND-NAND logic

$F = (A, B, C) = \sum m(0, 1, 3, 5, 6, 7)$ (using only NAND gates).

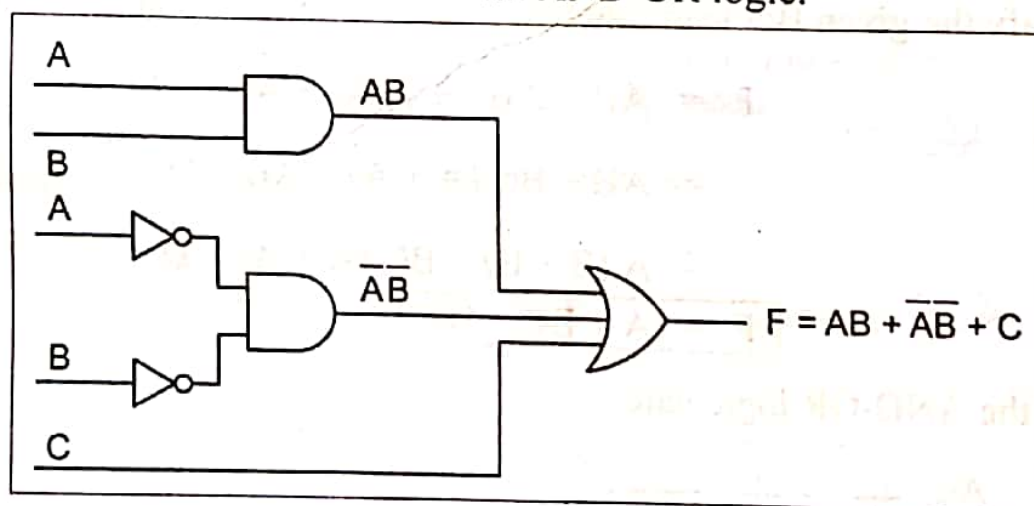
☺ **Solution:**

Step 1: Simplify the given Boolean function.

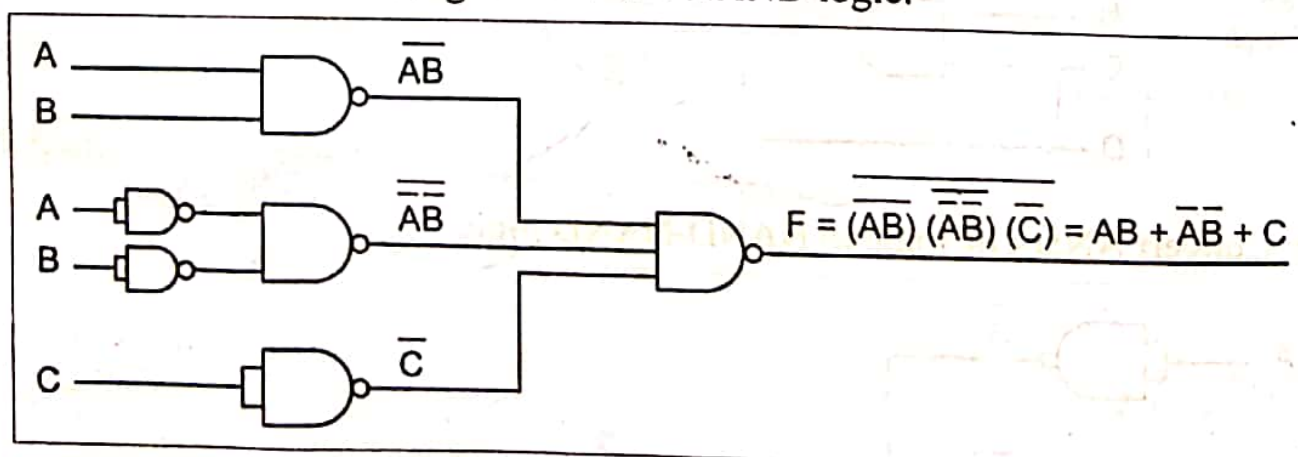
BC					
A		00	01	11	10
0		1	1	1	0
1		0	1	1	1

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

Step 2: Implement Boolean function with AND-OR logic.



Step 3: Convert AND-OR logic to NAND-NAND logic.



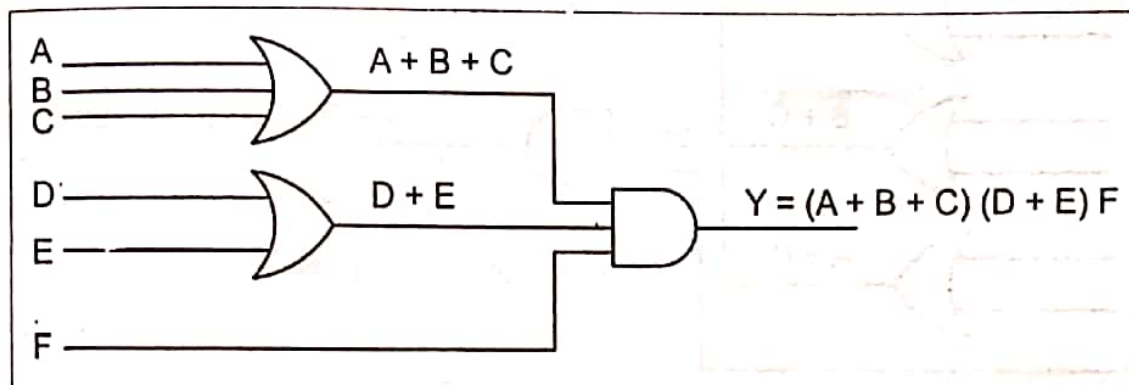
1.7.2. NOR-NOR IMPLEMENTATION

- ❖ The implementation of a Boolean function with NOR-NOR logic requires that the function to be simplified in the product of sum (POS) form.

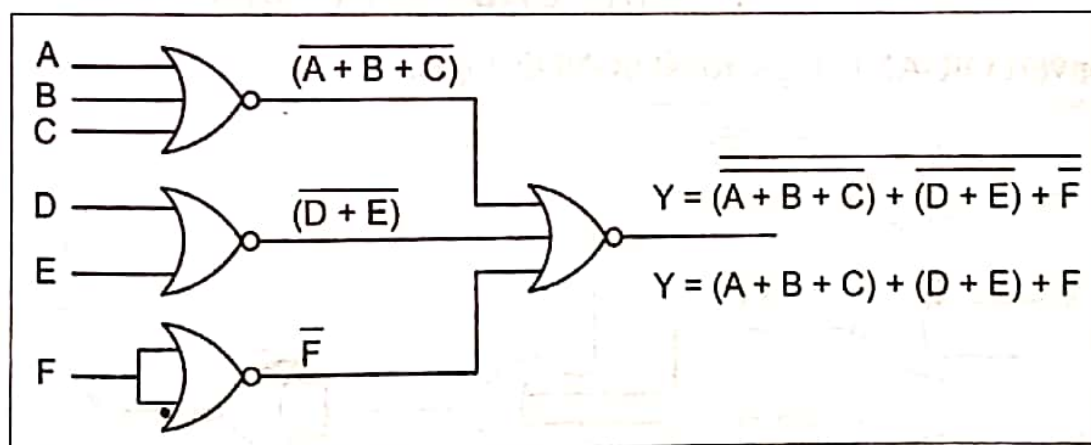
❖ Lets consider POS form (OR-AND logic) and how it will be represented in NOR-NOR logic.

Consider the Boolean function $Y = (A + B + C)(D + E)F$

OR-AND Logic Gate



NOR-NOR Implementation



Procedure for obtaining NOR-NOR Logic Diagram

1. Simplify the given Boolean function and express it in POS form.
2. Draw a NOR gate for each sum term of the function.
3. If Boolean function includes any single literal, draw NOR gate for each single literal.
4. Draw a single NOR gate in the second level, with inputs coming from outputs of first level gates.

SOLVED EXAMPLES

Example 1.54 Implement the following Boolean function with only NOR gates.

$$Y = AC + BC + AB + D$$

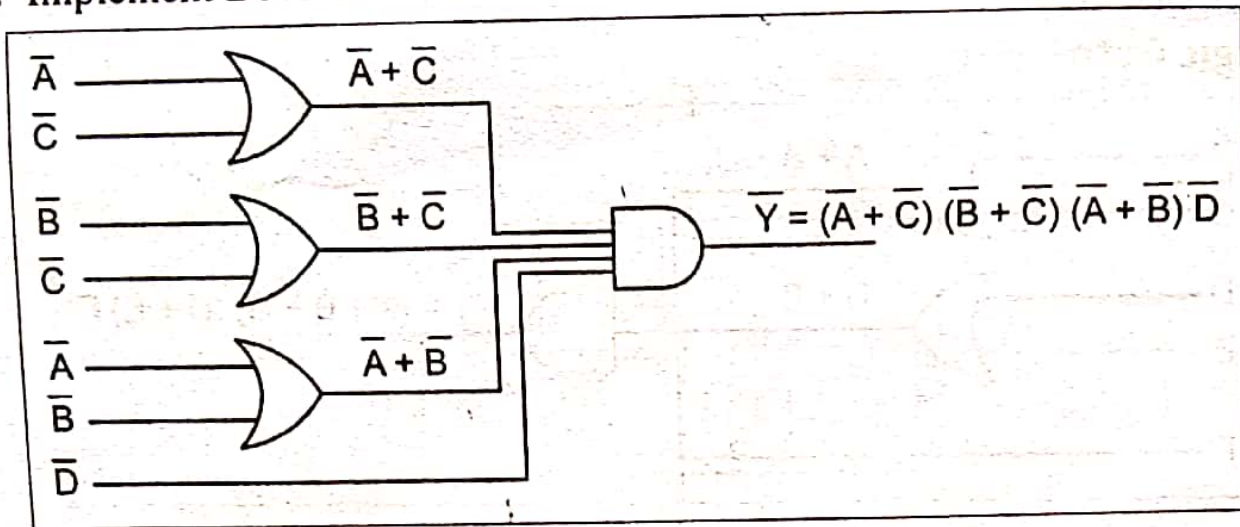
☺ **Solution:**

Step 1: Here the given function is in SOP format. So first we convert it into POS form, using duality theorem, we get

$$Y = AC + BC + AB + D$$

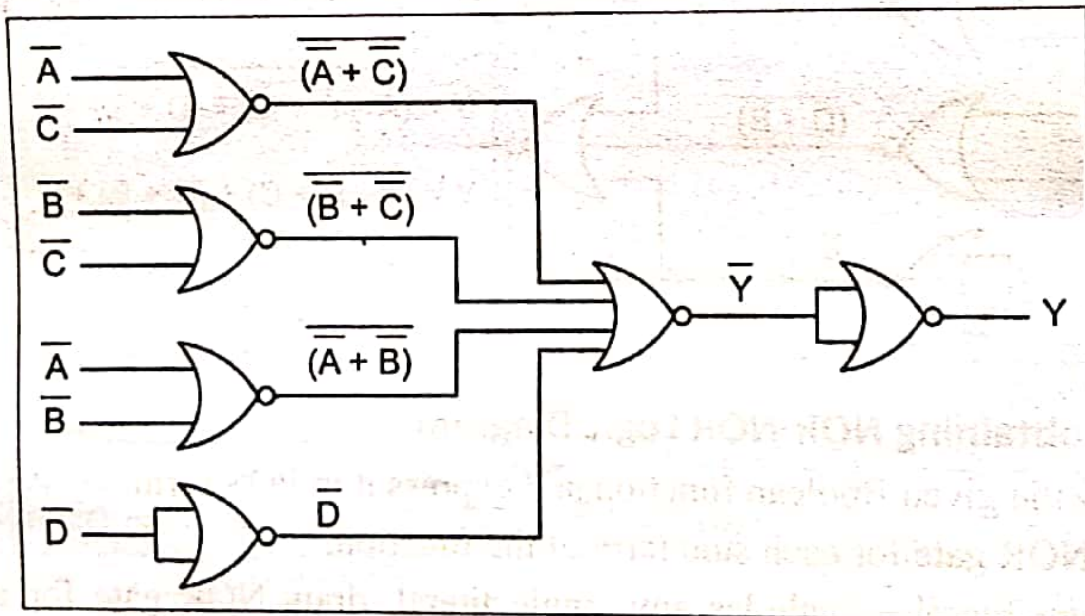
$$\bar{Y} = (\bar{A} + \bar{C})(\bar{B} + \bar{C})(\bar{A} + \bar{B})\bar{D}$$

Step 2: Implement Boolean function with OR-AND logic.



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

Step 3: Convert OR-AND logic to NOR-NOR logic.



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

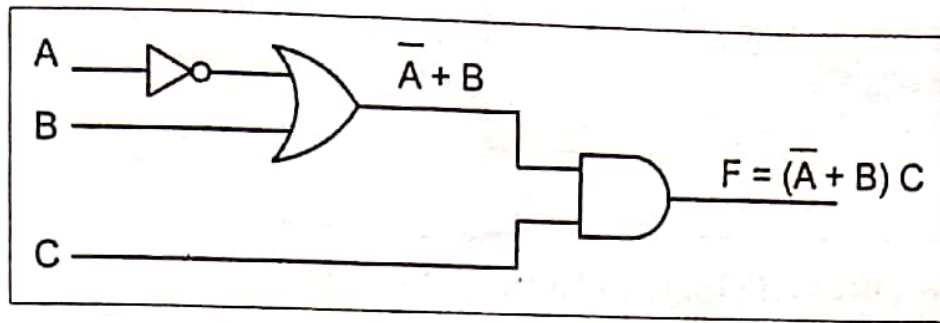
$$\bar{Y} = (\bar{A} + \bar{C})(\bar{B} + \bar{C})(\bar{A} + \bar{B})\bar{D}$$

$$Y = AC + BC + AB + D$$

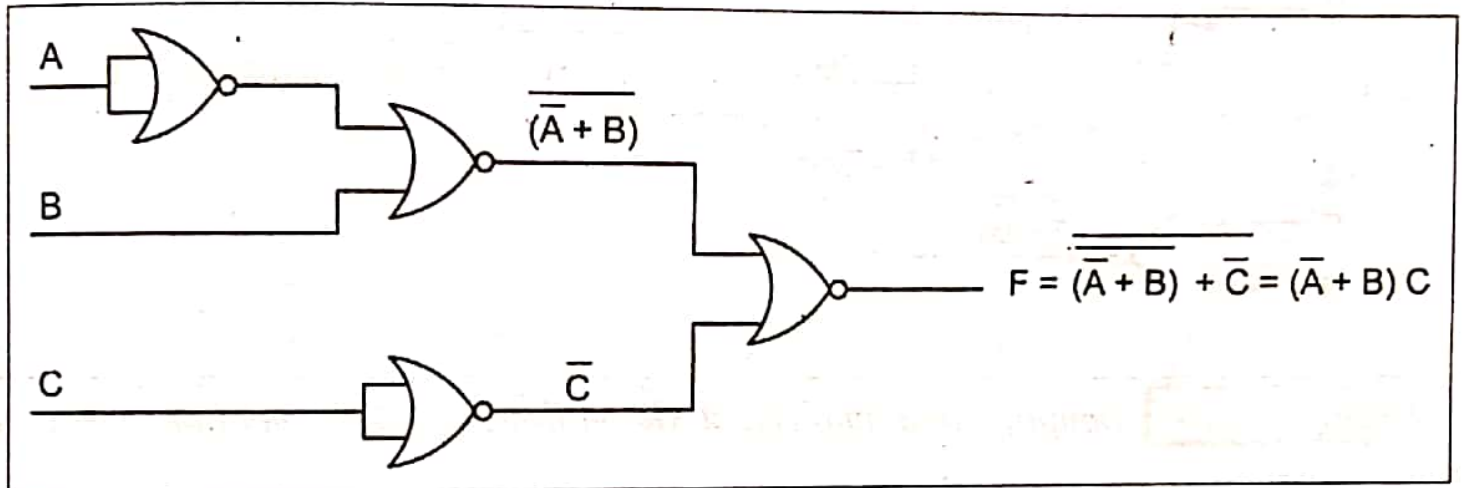
Example 1.55 Implement the following Boolean function with NOR-NOR logic (using only NOR gates) $F = (\bar{A} + B)C$.

☺ **Solution:**

Step 1: Implement Boolean function with OR-AND logic.



Step 2: Convert OR-AND logic to NOR-NOR logic.



Example 1.56 Simplify and implement the following POS function using NOR gates

$f(A, B, C, D) = \prod M(0, 1, 2, 3, 12, 13, 14, 15)$.

(Dec 2019)

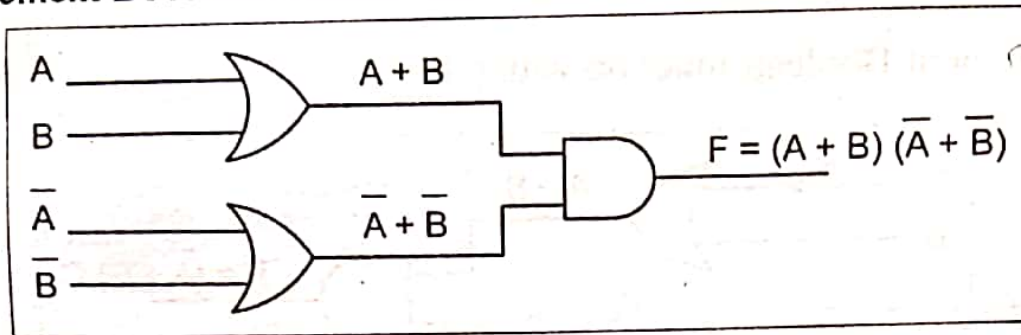
☺ **Solution:**

Step 1: Simplify the given Boolean function.

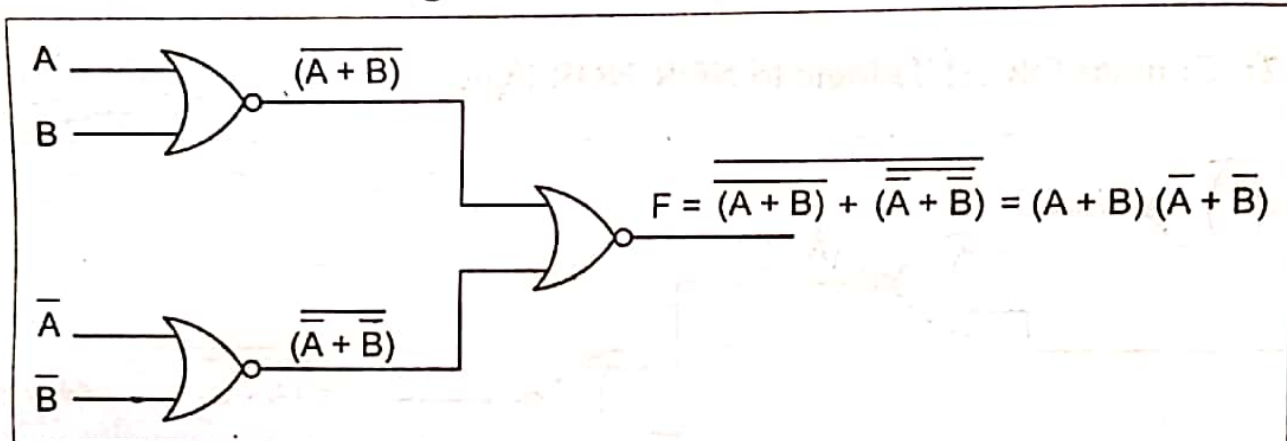
AB \ CD				
	00	01	11	10
00	0	0	0	0
01				
11	0	0	0	0
10				

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

Step 2: Implement Boolean function with OR-AND logic.



Step 3: Convert OR-AND logic to NOR-NOR logic.



Example 1.57 Simplify and implement the following SOP function using NOR gates.

(Dec 2019)

$$f(A, B, C, D) = \sum m(0, 1, 4, 5, 10, 11, 14, 15)$$

☺ **Solution:**

Step 1: Given function is SOP, so convert into its equivalent POS function.

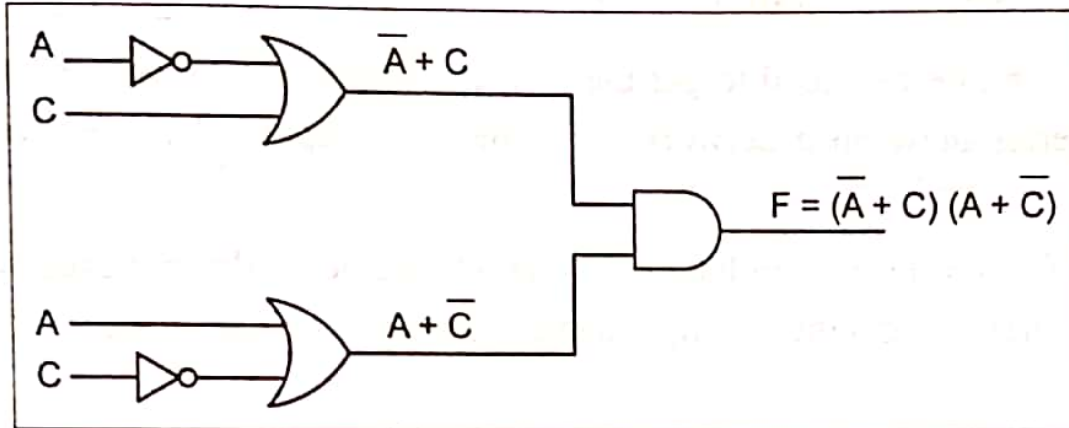
$$\therefore \sum m(0, 1, 4, 5, 10, 11, 14, 15) = \prod M(2, 3, 6, 7, 8, 9, 12, 13)$$

Step 2: Simplify the given Boolean function using K-Map simplification.

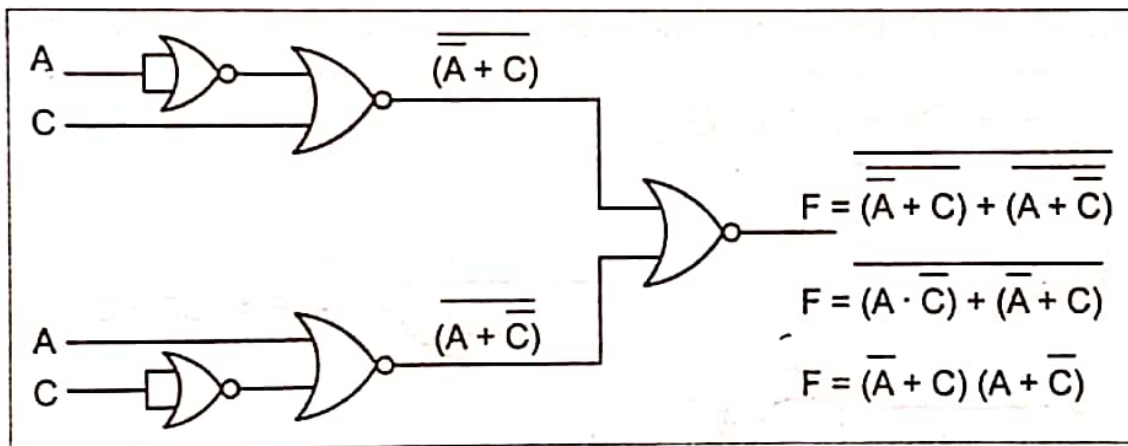
AB \ CD	CD			
	00	01	11	10
00			0	0
01			0	0
11	0	0		
10	0	0		

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

Step 3: Implement Boolean function with OR-AND logic.



Step 4: Convert OR-AND logic to NOR-NOR logic.



Example 1.68 Implement the Boolean expression using gates.

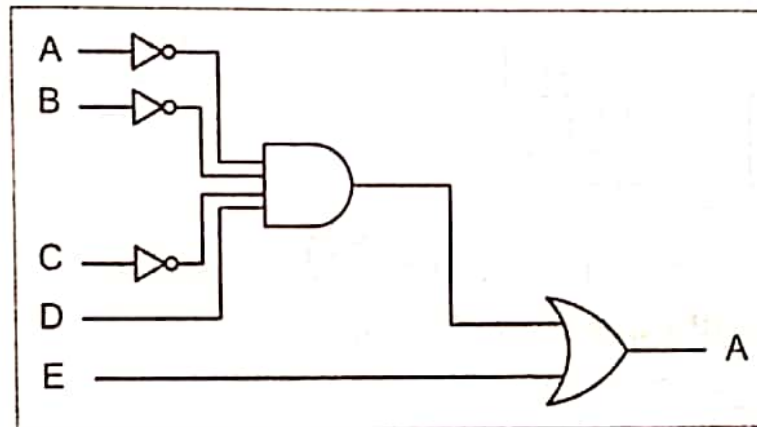
$$X = \overline{(AB + C)} D + E$$

(Nov/Dec 07)

☺ **Solution:**

$$X = \overline{(AB + C)} D + E = (\overline{AB} \cdot \overline{C}) D + E$$

$$X = \overline{A} \overline{B} \overline{C} D + E$$



Example 1.69 Sketch a NAND-NAND logic circuit for the Boolean expression.

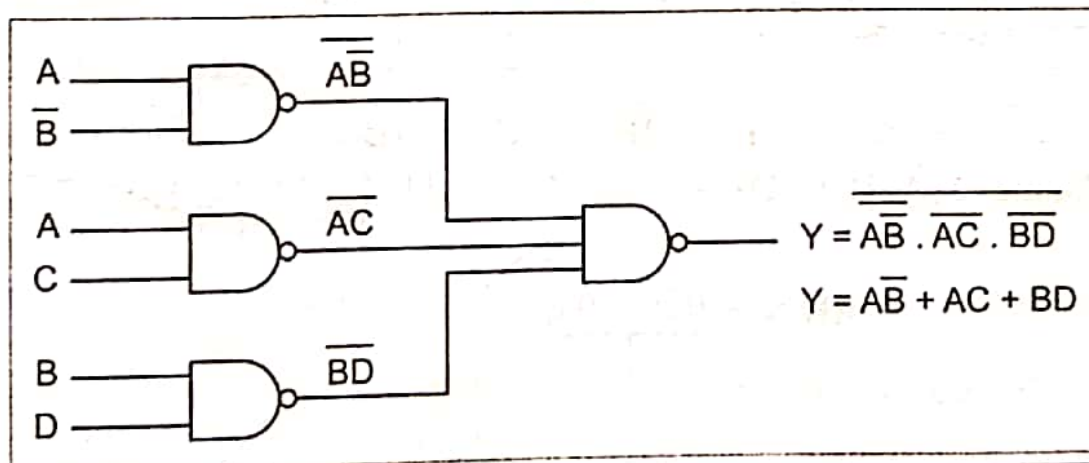
$$Y = \overline{A} \overline{B} + AC + BD$$

(Nov/Dec 07)

☺ **Solution:**

$$Y = \overline{A} \overline{B} + AC + BD$$

Given function is a SOP form, so we can implement NAND-NAND logic directly.



Example 1.70 Simplify the given Boolean function into:

(i) Sum of products form

(ii) Product of sum form and implement if using basic gates.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

(May/June 13)

☺ **Solution:**

(i) SOP form

Given function $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$

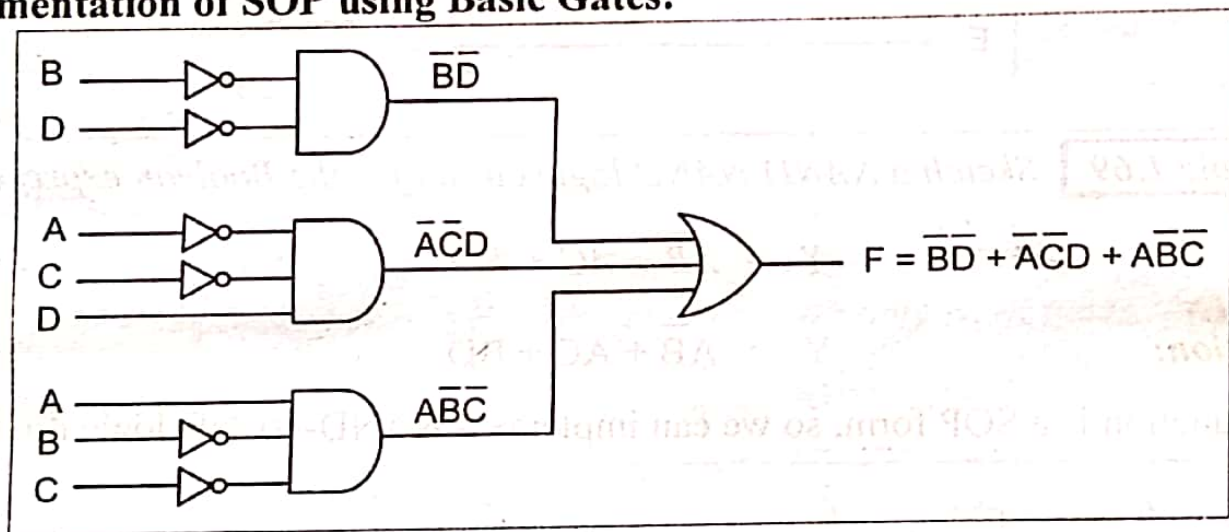
It is a SOP form, so we can simplify it directly using K-Map. Then we get

CD \ AB	00	01	11	10
00	1	1		1
01		1		
11				
10	1	1		1

[In SOP, 0 = \bar{A} , 1 = A]

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

Implementation of SOP using Basic Gates:



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

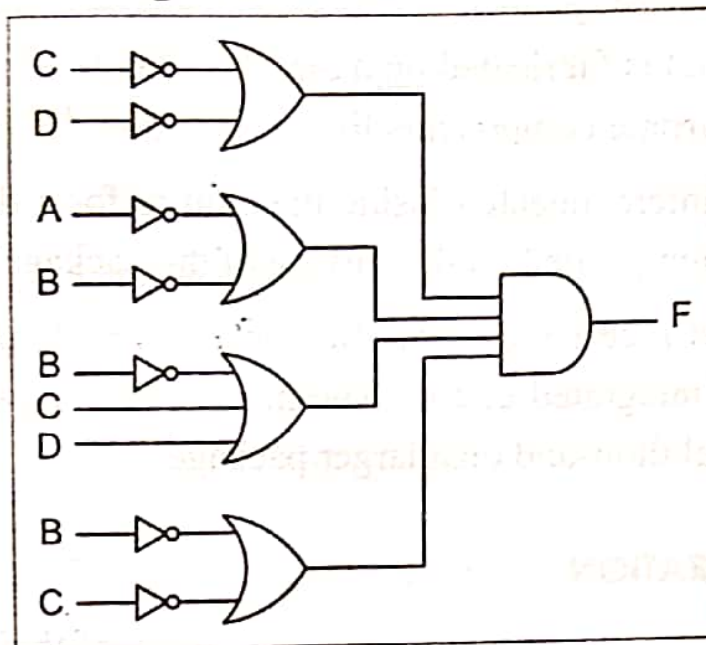
(ii) POS Form: $F(A, B, C, D) = \Pi(3, 4, 6, 7, 11, 12, 13, 14, 15)$

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

[In POS, 0 = A, 1 = \bar{A}]

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

Implementation of POS using Basic Gates:



Example 1.71 Implement the given function using NAND gates

$$F(x, y, z) = \sum m(0, 6)$$

(Nov/Dec 12)

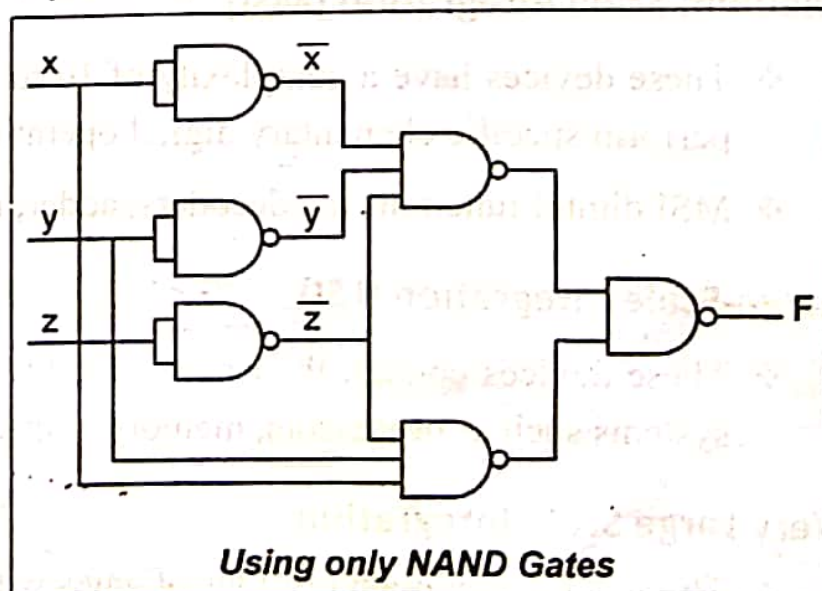
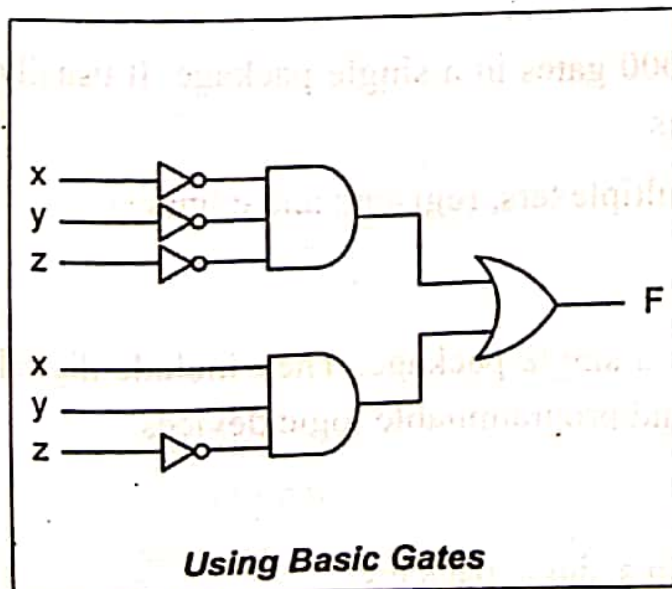
© **Solution:** Given function $F(x, y, z) = \sum m(0, 6)$ is a SOP form. Simplify it using K-Map.

$x \backslash yz$	00	01	11	10
0	1			
1				1

$$F(x, y, z) = \bar{x} \bar{y} \bar{z} + x y \bar{z}$$

SOP format can be implemented directly using NAND gates.

$$F(x, y, z) = \bar{x} \bar{y} \bar{z} + x y \bar{z}$$



* Determine all the prime implicants of the function $f(a, b, c, d) = \sum m(0, 2, 3, 4, 8, 10, 12, 13, 14)$ using Quine-McCluskey method. Verify the same using k-map Technique. [NOV/DEC-2021]

Step:1 Binary representation of min terms

Min terms	Variable			
	A	B	C	D
0	0	0	0	0
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
8	1	0	0	0
10	1	0	1	0
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0

Step:2 Group of Minterms of different number of 1's

No. of 1's	Min terms	Variable A B C D
0	0	0 0 0 0
1	2	0 0 1 0
	4	0 1 0 0
	8	1 0 0 0
2	3	0 0 1 1
	10	1 0 1 0
	12	1 1 0 0
3	13	1 1 0 1
	14	1 1 1 0

Step:3 2-Cell Combination

Combination	A B C D
(0, 2)	0 0 - 0
(0, 4)	0 - 0 0
(0, 8)	- 0 0 0
(2, 3)	0 0 1 -
(2, 10)	- 0 1 0
(4, 12)	- 1 0 0
(8, 10)	1 0 - 0
(8, 12)	1 - 0 0
(10, 14)	1 - 1 0
(12, 13)	1 1 0 -
(12, 14)	1 1 - 0

Step:4 4-Cell Combination

Combination	A	B	C	D
(0, 8, 2, 10)	—	0	—	0
(0, 8, 4, 12)	—	—	0	0
(8, 12, 10, 14)	1	—	—	0

Step:5 Prime Implicant table

Prime Implicants	0	2	3	4	8	10	12	13	14
(0, 8, 2, 10) $\overline{B}\overline{D}$	*	*			*	*			
(0, 8, 4, 12) $\overline{C}\overline{D}$	*			*	*		*		
(8, 12, 10, 14) $A\overline{D}$					*	*	*		*
(2, 3)* $\overline{A}\overline{B}C$		*	*						
(12, 13)* $AB\overline{C}$							*	*	

∴ The output expression is,

$$f(a, b, c, d) = \overline{A}\overline{B}C + \overline{C}\overline{D} + AB\overline{C} + A\overline{D}$$

k-Map:-

	CD			
	00	01	11	10
AB	00	1	1	1
	01	1		
	11	1	1	1
	10	1		1

$$f(a, b, c, d) = \bar{A}\bar{B}C + \bar{C}\bar{D} + AB\bar{C} + A\bar{D}$$

*.) With the use of maps, - find the simplest sum-of-products form of the function $F = fg$ where $f = abc' + c'd + a'cd' + b'cd'$ and $g = (a+b+c'+d')(b'+c'+d)(a'+c+d')$ [Nov/Dec-2021]

$$\begin{aligned} g &= (a+b+c'+d')(b'+c'+d)(a'+c+d') \\ &= (a+b+c'+d')(aa'+b'+c'+d)(a'+bb'+c+d') \\ &\quad \text{(By distributive property)} \end{aligned}$$

$$\begin{aligned} g &= (a+b+c'+d')(a+b'+c'+d)(a'+b'+c'+d) \\ &\quad (a'+b+c+d')(a'+b'+c+d') \longrightarrow \textcircled{1} \text{ [Pos]} \end{aligned}$$

$$\begin{aligned} g &= (a'b'cd) + (a'bcd') + (abc'd') + (ab'c'd) \\ &\quad + (abc'd) \\ &\quad \hookrightarrow \textcircled{2} \text{ [Sop]} \end{aligned}$$

$$f = abc' + c'd + a'cd' + b'cd'$$

$$= [abc'(d+d')] + [(a+a')(b+b')c'd] \\ + [a'cd'(b+b')] + [(a+a')b'cd']$$

$$= \underline{abc'd} + abc'd' + \underline{abc'd} + ab'c'd + a'b'cd \\ + a'b'c'd + a'bcd' + \underline{a'b'cd'} + ab'cd' + \\ \underline{a'b'cd'}$$

$$f = abc'd + abc'd' + ab'c'd + a'b'c'd + \\ a'b'c'd + a'bcd' + a'b'cd' + ab'cd'$$

↳ ③ [SOP]

$$f = (a'+b'+c+d')(a'+b'+c+d)(a'+b+c+d') \\ (a+b'+c+d')(a+b+c+d')(a+b'+c'+d)$$

$$(a+b+c'+d)(a'+b+c'+d) \rightarrow \textcircled{4} [POS]$$

Pos

$$F = fg$$

$$F = (\underline{a'+b'+c+d'}) (a'+b'+c+d) (\underline{a'+b+c+d'}) \\ (a+b'+c+d') (a+b+c+d') (\underline{a+b'+c'+d}) \\ (a+b+c'+d) (a'+b+c'+d) \cdot (a+b+c'+d') \\ (\underline{a+b'+c'+d}) (a'+b'+c'+d) (\underline{a'+b+c+d'}) \\ (\underline{a'+b'+c+d'})$$

$$F = (a' + b' + c + d') (a' + b' + c + d) (a' + b + c + d') \\ (a + b' + c + d') (a + b + c + d') (a + b' + c' + d) \\ (a + b + c' + d) (a' + b + c' + d) (a + b + c' + d') \\ (a' + b' + c' + d) \rightarrow (5) [pos]$$

Sop

$$F = (a' b' c' d) + (a' b' c' d') + (a' b' c d) + (a' b' c d') \\ + (a' b' c' d) + (a' b' c d') + (a' b' c d') + (a' b' c d') \\ + (a' b' c d) + (a' b' c d') \\ 0011 \quad 1110$$

$$= m_{13} + m_{12} + m_9 + m_5 + m_1 + m_6 + m_2 + m_{10} \\ + m_3 + m_{14}$$

$$= \sum m(1, 2, 3, 5, 6, 9, 10, 12, 13, 14)$$

$$F = \sum m(1, 2, 3, 5, 6, 9, 10, 12, 13, 14)$$

K-Map

ab \ cd	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

Sop

$$F = c'd + cd' + abc' + a'b'c$$

UNIT II COMBINATIONAL LOGIC CIRCUITS

Problem formulation and design of combinational circuits - Code-Converters, Half and Full Adders, Binary Parallel Adder – Carry look ahead Adder, BCD Adder, Magnitude Comparator, Decoder, Encoder, Priority Encoder, Mux/Demux, Case study: Digital trans-receiver / 8 bit Arithmetic and logic unit, Parity Generator/Checker, Seven Segment display decoder

COMBINATIONAL CIRCUITS

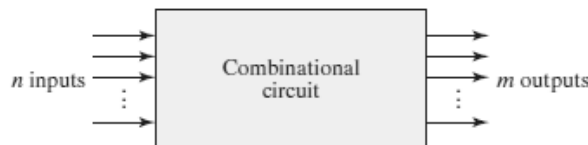
❖ The digital system consists of two types of circuits, namely

- (i) Combinational circuits and
- (ii) Sequential circuits

Combinational circuits

❖ *A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs without regard to previous inputs or previous state of outputs..*

❖ A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.



Sequential circuits:

❖ *Sequential circuits contain logic gates as well as memory cells. Their outputs depend on the present inputs and also on the states of memory elements.*

❖ Since the outputs of sequential circuits depend not only on the present inputs but also on past inputs.

❖ The circuit behavior must be specified by a time sequence of inputs and memory states.

DESIGN PROCEDURE

Explain the procedure involved in designing combinational circuits.

(May 2015)

Any combinational circuit can be designed by the following steps of design procedure.

1. The problem is stated.
2. Identify the input variables and output functions.
3. The input and output variables are assigned letter symbols.
4. The truth table is prepared that completely defines the relationship between the input variables and output functions.
5. The simplified Boolean expression is obtained by any method of minimization algebraic method, Karnaugh map method, or tabulation method.
6. A logic diagram is realized from the simplified expression using logic gates.

HALF ADDER

Construct a half adder with necessary diagrams.

(Nov-06, May- 07)

- ❖ A half-adder is an arithmetic circuit block that can be used to add two bits and produce two outputs such as SUM and CARRY.
- ❖ The Boolean expressions for the SUM and CARRY outputs are given by the equations

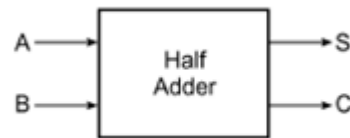
$$S = A'B + AB'$$

$$C = AB$$

Truth Table:

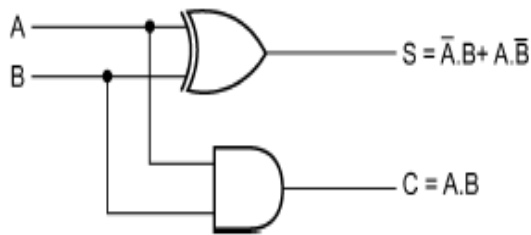
Input variables		Output variables	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Symbol:

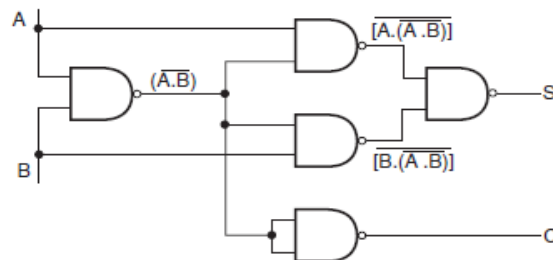


- ❖ The outputs S and C functions are similar to Exclusive-OR and AND functions respectively.
- ❖ Below Figure shows the logic diagram to implement the half-adder circuit.

Logic Diagram:



Half adder using NAND gate:



FULL ADDER

Design a full adder using NAND and NOR gates respectively.

(Nov -10)

- ❖ A Full-adder is an arithmetic circuit block that can be used to add three bits and produce two outputs such as SUM and CARRY.
- ❖ Let us consider the input variables augend as A, addend as B, and previous carry as X, and outputs sum as S and carry as C.
- ❖ As there are three input variables, eight different input combinations are possible.

Truth table:

Input variables			Outputs	
X	A	B	S	C
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

Karnaugh map:

	A'B'	A'B	AB	AB'
X'		1		1
X	1		1	

K-Map for Sum

	A'B'	A'B	AB	AB'
X'			1	
X		1	1	1

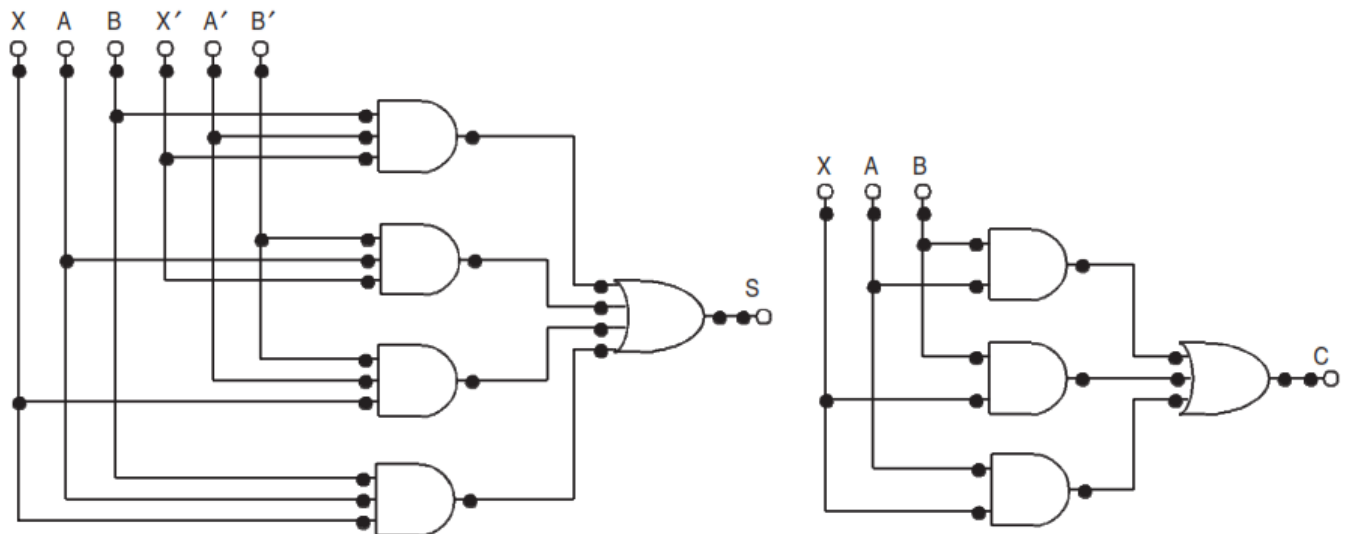
K-Map for Carry

❖ The simplified Boolean expressions of the outputs are

$$S = X'A'B + X'AB' + XA'B' + XAB$$

$$C = AB + BX + AX$$

Logic diagram:

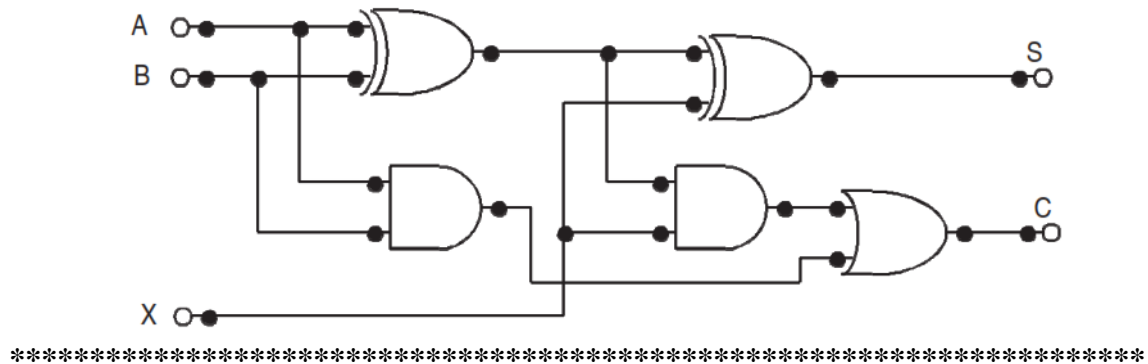


❖ The Boolean expressions of S and C are modified as follows

$$\begin{aligned}
 S &= X'A'B + X'AB' + XA'B' + XAB \\
 &= X' (A'B + AB') + X (A'B' + AB) \\
 &= X' (A \oplus B) + X (A \oplus B)' \\
 &= X \oplus A \oplus B \\
 C &= AB + BX + AX = AB + X (A + B) \\
 &= AB + X (AB + AB' + AB + A'B) \\
 &= AB + X (AB + AB' + A'B) \\
 &= AB + XAB + X (AB' + A'B) \\
 &= AB + X (A \oplus B)
 \end{aligned}$$

Full adder using Two half adder:

❖ Logic diagram according to the modified expression is shown Figure.



HALF SUBTRACTOR

Design a half subtractor circuit.

(Nov-2009)

❖ A half-subtractor is a combinational circuit that can be used to subtract one binary digit from another to produce a DIFFERENCE output and a BORROW output.

Truth table:

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

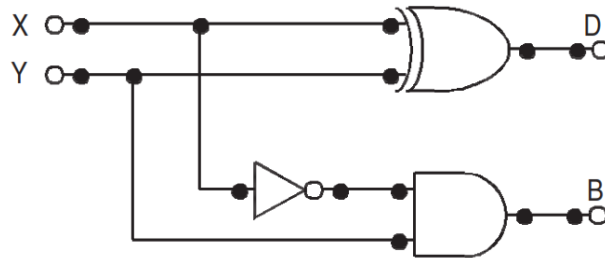
❖ **Boolean expressions** of the outputs D and B functions can be written as

$$D = X'Y + XY'$$

$$B = X'Y$$

Logic diagram:

- ❖ Figure shows the logic diagram to realize the half-subtractor circuit\



FULL SUBTRACTOR

Design a full subtractor.

(Nov-2009,07)

- ❖ A combinational circuit of full-subtractor performs the operation of subtraction of three bits such as the minuend, subtrahend, and borrow generated from the subtraction operation of previous significant digits and produces the outputs difference and borrow.

Truth table:

Input variables			Outputs	
X	Y	Z	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	0
1	1	0	0	0
1	1	1	1	1

Karnaugh map:

	Y'Z'	Y'Z	YZ	YZ'
X'		1		1
X	1		1	

K-Map for D

	Y'Z'	Y'Z	YZ	YZ'
X'		1	1	1
X			1	

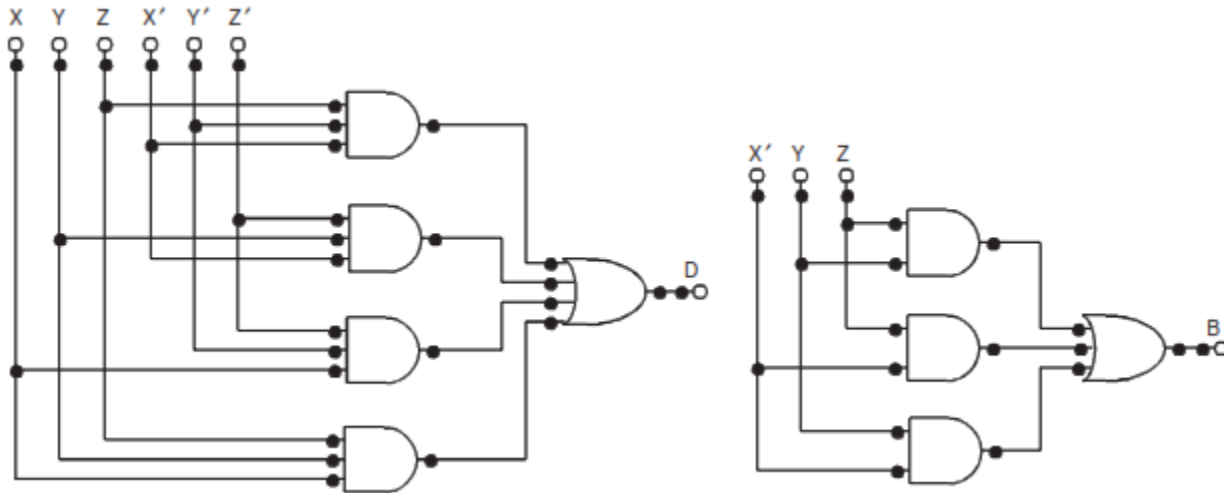
K-Map for B

- ❖ The simplified **Boolean expressions** of the outputs are

$$S = X'Y'Z + X'YZ' + XY'Z' + XYZ$$

$$C = X'Z + X'Y + YZ$$

Logic diagram:

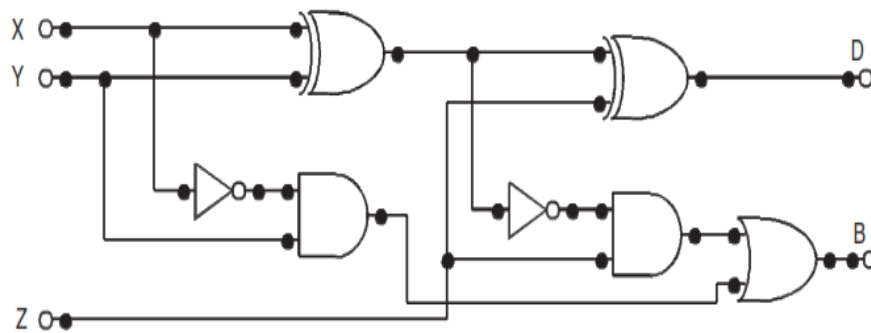


❖ The Boolean expressions of D and B are modified as follows

$$\begin{aligned} D &= X'Y'Z + X'YZ' + XY'Z' + XYZ \\ &= X' (Y'Z + YZ') + X (Y'Z' + YZ) \\ &= X' (Y \oplus Z) + X (Y \oplus Z)' \\ &= X \oplus Y \oplus Z \end{aligned}$$

$$\begin{aligned} B &= X'Z + X'Y + YZ = X'Y + Z (X' + Y) \\ &= X'Y + Z(X'Y + X'Y' + XY + X'Y) \\ &= X'Y + Z(X'Y + X'Y' + XY) \\ &= X'Y + X'YZ + Z(X'Y' + XY) \\ &= X'Y + Z(X \oplus Y)' \end{aligned}$$

❖ Logic diagram according to the modified expression is shown Figure.



PARALLEL BINARY ADDER: (RIPPLE CARRY ADDER)

Explain about four bit adder with neat diagram.

- ❖ Two binary bits can be added and the addition of two binary bits with a carry.
- ❖ In practical situations it is required to add two data each containing more than one bit.
- ❖ Two binary numbers each of n bits can be added by means of a full adder circuit.
- ❖ Consider the example that two 4-bit binary numbers $B_4 B_3 B_2 B_1$ and $A_4 A_3 A_2 A_1$ are to be added with a carry input C_1 .
- ❖ This can be done by cascading four full adder circuits as shown in Figure.
- ❖ The least significant bits A_1, B_1 , and C_1 are added to produce sum output S_1 and carry output C_2 .
- ❖ Carry output C_2 is then added to the next significant bits A_2 and B_2 producing sum output S_2 and carry output C_3 .
- ❖ C_3 is then added to A_3 and B_3 and so on. Thus finally producing the four-bit sum output $S_4 S_3 S_2 S_1$ and final carry output C_{out} .
- ❖ Such type of four-bit binary adder is commercially available in an IC package.

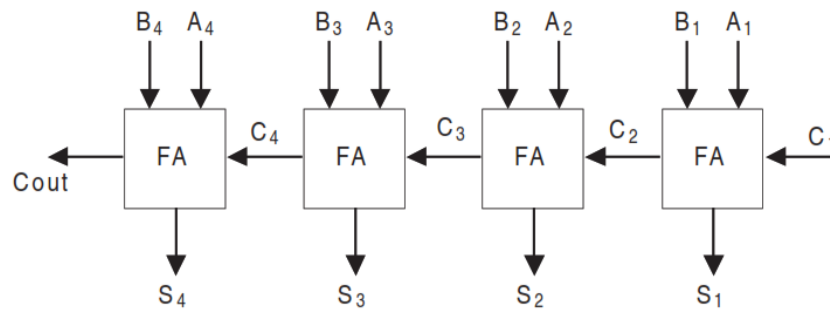


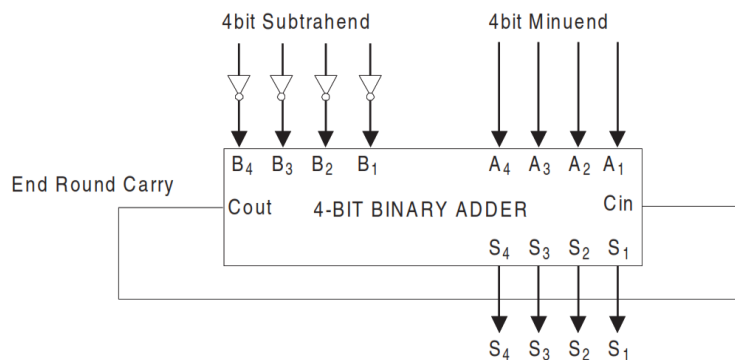
Fig: 4 bit adder

- ❖ For the addition of two n bits of data, n numbers of full adders can be cascaded as demonstrated in figure.
- ❖ It can be constructed with 4-bit, 2-bit, and 1-bit full adder IC packages.
- ❖ The carry output of one package must be connected to the carry input of the next higher order bit IC package of higher order bits.
- ❖ The addition technique adopted here is a parallel type as all the bit addition operations are performed in parallel. Therefore, this type of adder is called a parallel adder.
- ❖ The 4-bit parallel binary adder IC package is useful to develop combinational circuits.

PARALLEL BINARY SUBTRACTOR

Explain about four bit subtractor with neat diagram.

- ❖ By 1's complement method, the bits of subtrahend are complemented and added to the minuend. If any carry is generated it is added to the sum output.
- ❖ Below Figure demonstrates the subtraction of $B_4 B_3 B_2 B_1$ from $A_4 A_3 A_2 A_1$.
- ❖ Each bit of $B_4 B_3 B_2 B_1$ is first complemented by using INVERTER gates and added to $A_4 A_3 A_2 A_1$ by a 4-bit binary adder.
- ❖ End round carry is again added using the C_{in} pin of the IC.



Direct Subtraction	1's complement method
1 1 0 1	1 1 0 1 (+)
- 1 0 0 1	1's complement → 0 1 1 0
0 1 0 0	Carry → 1 0 0 1 1
	Add Carry → 1
	0 1 0 0

Fig: 4 bit subtractor

Example

Fast adder (or) Carry Look Ahead adder

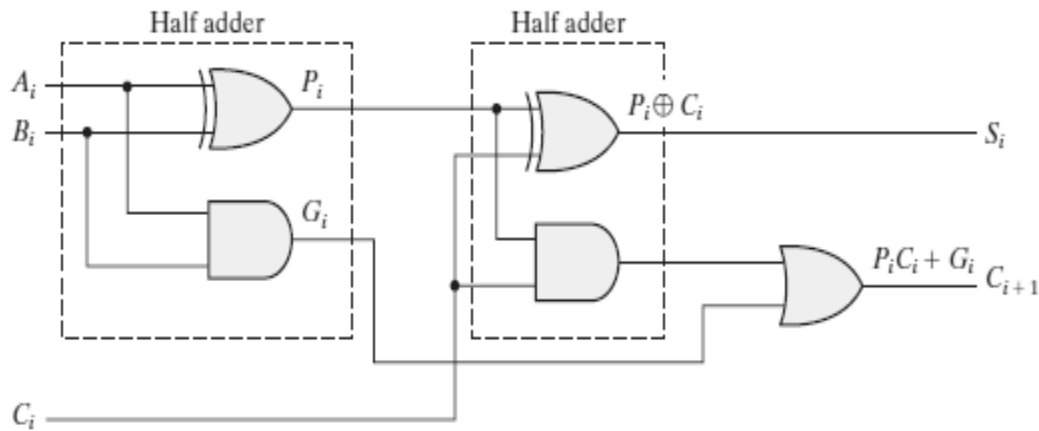
Design a carry look ahead adder circuit.

(Nov-2010),(May 2018)

Fast Adder:

- ❖ Every logic gate offers some delay when the signal passes from its input to output, which is called the propagation delay of the logic gate.
- ❖ So every combinational circuit takes some time to produce its correct output after the arrival of all the input, which is called total propagation time.
- ❖ Total propagation time is equal to the propagation delay of individual gates times the number of gate levels in the circuit.
- ❖ For a 4-bit parallel binary adder, carry propagation takes the longest propagation time.
- ❖ One method to reduce the propagation delay time is to use faster gates.

- ❖ Another technique is to employ a little more complex combinational circuit, which can reduce the carry propagation delay time.
- ❖ The most widely used method employs the principle of look ahead carry generation, which is illustrated below.
- ❖ *The carry look ahead adder is based on the principle of looking at the lower order bits of the augend and addend to see if a higher order carry is to be generated.*
- ❖ It uses two functions carry generate and carry propagate.



- ❖ Consider the circuit of the full adder shown in Fig. It defines two new binary variables

$$P_i = A_i \oplus B_i \quad \text{and} \quad G_i = A_i B_i$$
- ❖ The output sum and carry can respectively be expressed as

$$S_i = P_i \oplus C_i \quad \text{and} \quad C_{i+1} = G_i + P_i C_i$$
- ❖ G_i is called a carry generate, and it generates an output carry if both the inputs A_i and B_i are logic 1, regardless of the input carry.
- ❖ P_i is called the *carry propagate* because it is the term associated with the propagation of the carry from C_i to C_{i+1} .

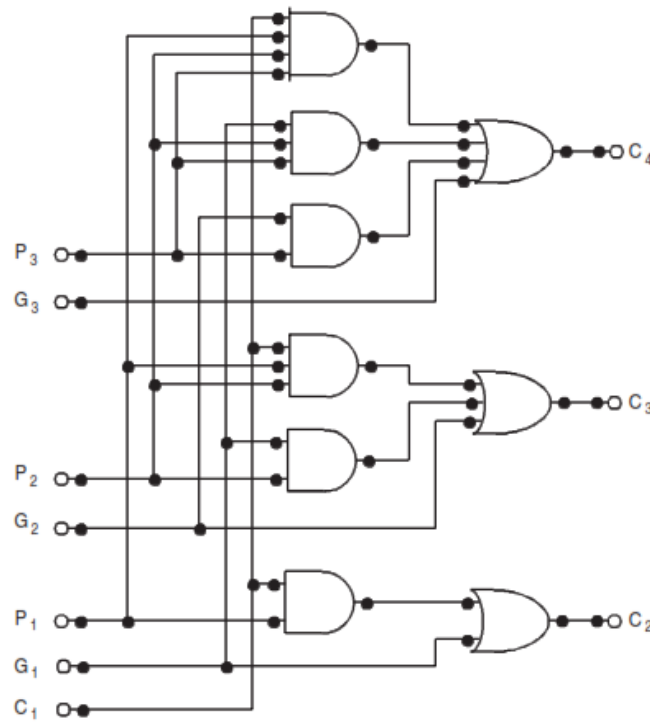


Fig: Logic diagram

- ❖ Boolean expressions for the carry output of each stage can be written after substituting C_i and C_{i+1} as

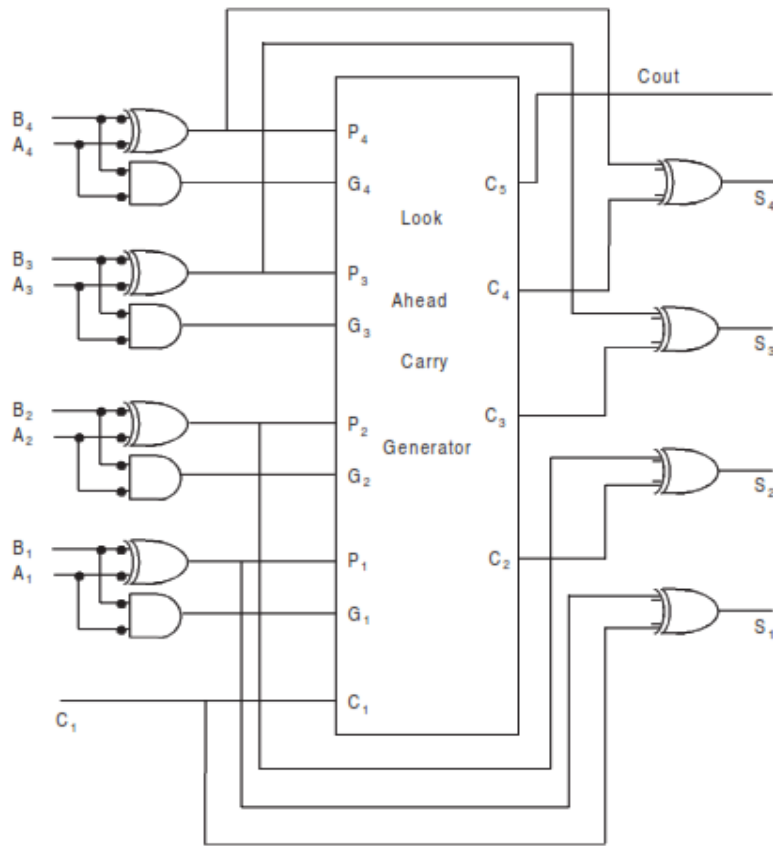
$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

$$C_5 = G_4 + P_4 C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_1$$

- ❖ In fact, all the intermediate carry as well as the final carry C_2 , C_3 , C_4 , and C_5 can be implemented by only two levels of gates and available at the same time.
- ❖ The final carry C_5 need not have to wait for the intermediate carry to propagate.



BCD Adder

Design to perform BCD addition.

(May -08)(Dec 2017)

- ❖ Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage.
- ❖ Since each input cannot exceed 9, the output sum must not exceed $9 + 9 + 1 = 19$ (1 in the sum is input carry from a previous stage).
- ❖ If a four-bit binary adder is used, the normal sum output will be of binary form and may exceed 9 or carry may be generated. So the sum output must be converted to BCD form.
- ❖ Suppose we apply two BCD digits to a four-bit binary adder. The adder will form the sum in binary and produce a result that ranges from 0 through 19.
- ❖ These binary numbers are listed in table and are labeled by symbols K, Z8, Z4, Z2, and Z1.
- ❖ K is the carry, and the subscripts under the letter Z represent the weights 8, 4, 2, and 1 that can be assigned to the four bits in the BCD code.

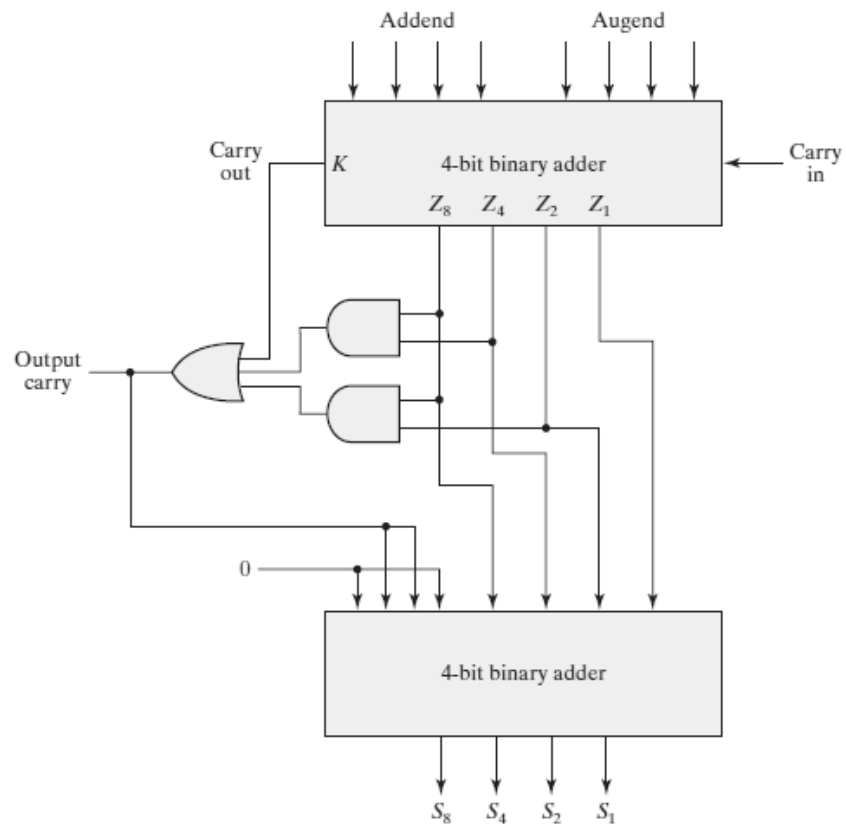


Fig: BCD Adder

Derivation of BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z ₈	Z ₄	Z ₂	Z ₁	C	S ₈	S ₄	S ₂	S ₁	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Karnaugh map:

		$Z_2 Z_1$			
		00	01	11	10
$Z_8 Z_4$	00				
	01				
	11	1	1	1	1
	10			1	1

- ❖ A BCD adder that adds two BCD digits and produces a sum digit in BCD is shown in Fig.
- ❖ The two decimal digits, together with the input carry, are first added in the top four-bit adder to produce the binary sum.
- ❖ When the output carry is equal to 0, nothing is added to the binary sum. When it is equal to 1, binary 0110 is added to the binary sum through the bottom four-bit adder.
- ❖ The condition for a correction and an output carry can be expressed by the Boolean function

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

MULTIPLEXERS AND DEMULTIPLEXERS

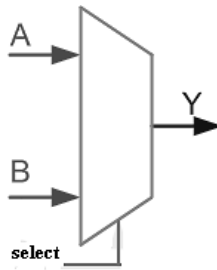
Multiplexer: (MUX)

Design a 2:1 and 4:1 multiplexer.

- ❖ A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- ❖ The selection of a particular input line is controlled by a set of selection lines.
- ❖ Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

2 to 1 MUX:

- ❖ A 2 to 1 line multiplexer is shown in figure below, each 2 input lines A to B is applied to one input of an AND gate.
- ❖ Selection lines S are decoded to select a particular AND gate.



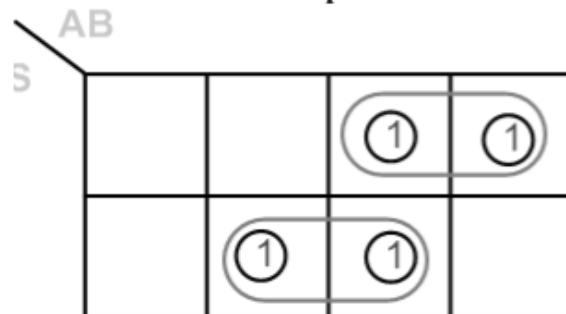
- ❖ To derive the gate level implementation of 2:1 MUX we need to have truth table as shown in figure.
- ❖ Boolean expression for output Y,

$$Y = A.S' + B.S$$

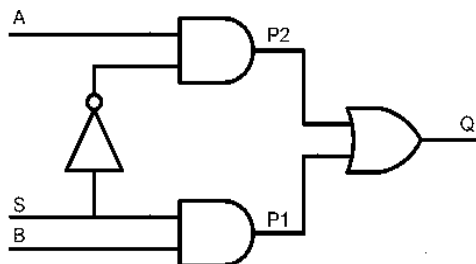
Truth table

B	A	S	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

K-map



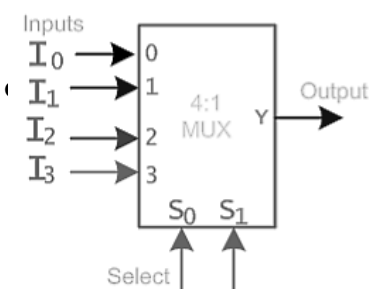
Logic Diagram:



4 to 1 MUX: (Illustrate the concept of basic 4 –input Multiplexer)

(Dec2018)

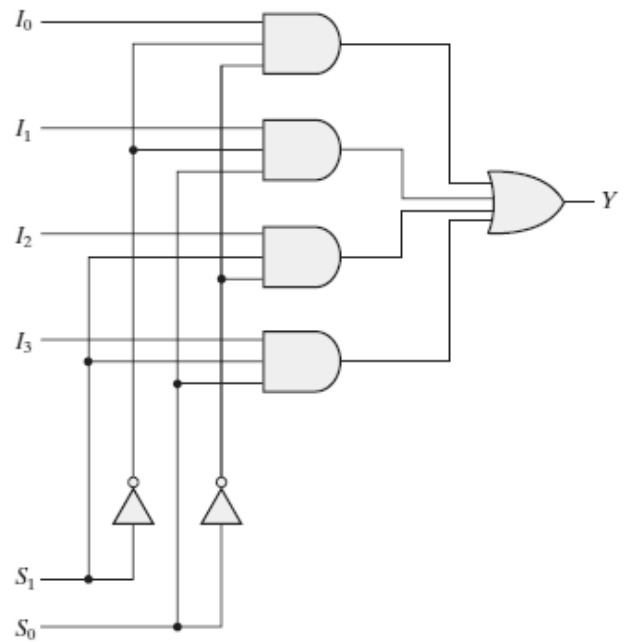
- ❖ A 4 to 1 line multiplexer is shown in figure below, each of 4 input lines I_0 to I_3 is applied to one input of an AND gate.
- ❖ Selection lines S_0 and S_1 are decoded to select a particular AND gate.
- ❖ The truth table for the 4:1 mux is given in the table below.



Logic Diagram

Truth Table:

SELECT LINES		OUTPUT
S1	S0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



- ❖ Let us consider that select input combination $S_1 S_0$ is 01
- ❖ The AND gate associated with I_1 will have two of its inputs equal to logic 1 and a third input is connected to I_1 .
- ❖ Therefore, output of this AND gate is according to the information provided by channel I_1 .
- ❖ Other three AND gates have logic 0 to at least one of their inputs which makes their outputs to logic 0.
- ❖ Hence, OR output (Y) is equal to the data provided by the channel I_1 .
- ❖ Thus, information from I_1 is available at Y. Normally a multiplexer has an ENABLE input to also control its operation.

Problems :

Example: Implement the Boolean expression using MUX

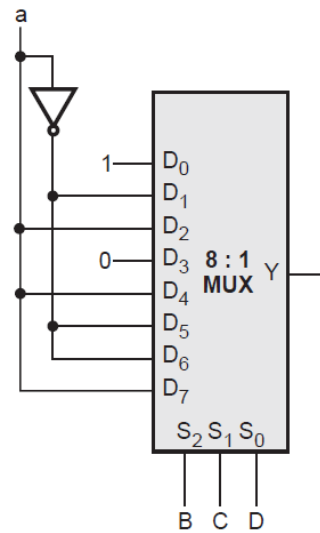
$$F(A,B,C,D) = \sum m(0,1,5,6,8,10,12,15)$$

(May 2018)

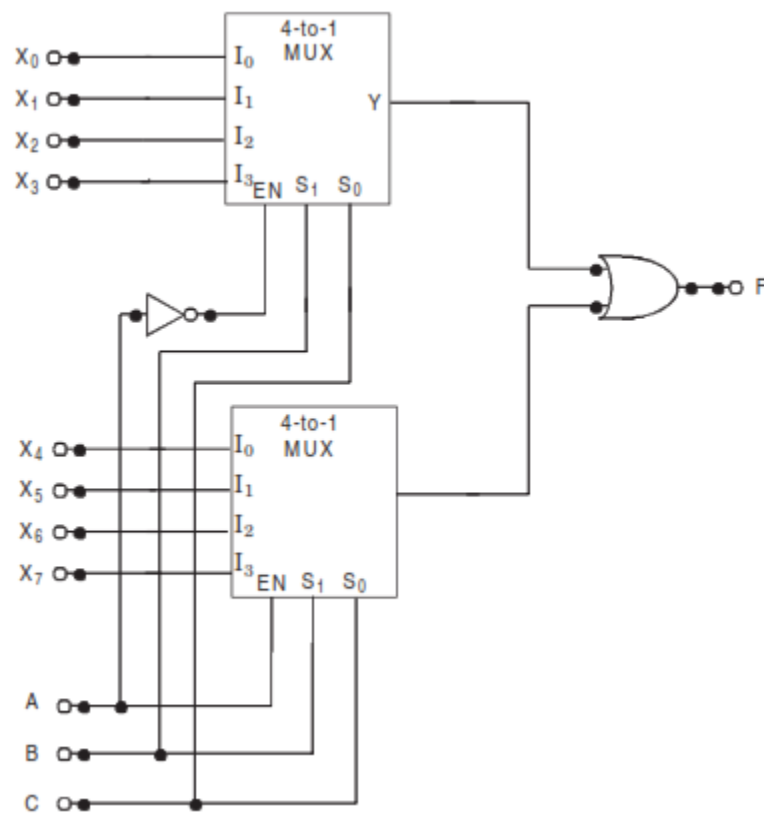
Solution : Implementation table :

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
\bar{a}	0	1	2	3	4	5	6	7
a	8	9	10	11	12	13	14	15
	1	\bar{a}	a	0	a	\bar{a}	\bar{a}	a

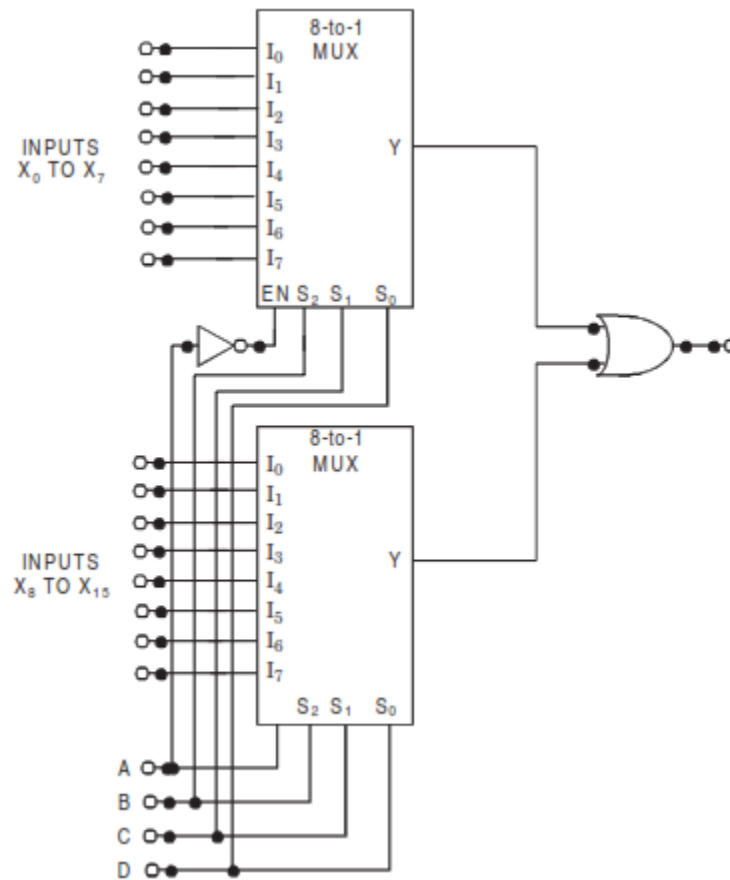
Logic Diagram:



Design 8-to-1 line multiplexer is realized by two 4-to-1 line multiplexers.



Design 16-to-1 multiplexer can be realized with five 4-to-1 multiplexers.



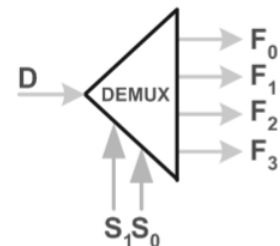
DEMULTIPLEXERS:

Explain about demultiplexers.

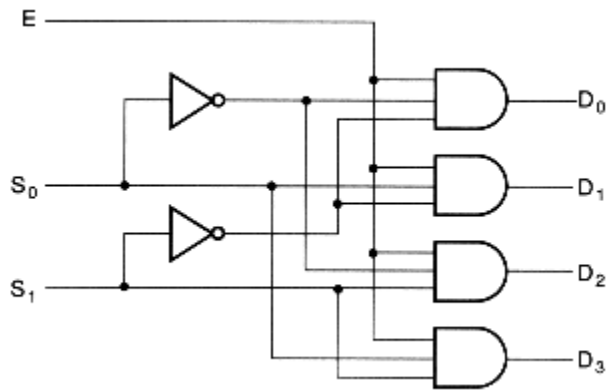
- ❖ Demultiplexing is the process that receives information from one channel and distributes the data over several channels.
- ❖ It is the reverse operation of the multiplexer.
- ❖ A demultiplexer is the logic circuit that receives information through a single input line and transmits the same information over one of the possible 2^n output lines.
- ❖ The selection of a specific output line is controlled by the bit combinations of the selection lines.
- ❖ Example: 1-to-4 De-multiplexer

Truth table

S1	S0	F0	F1	F2	F3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D



Logic Diagram:



Truth Table:

INPUT				OUTPUT			
E	D	S0	S1	Y0	Y1	Y2	Y3
1	1	0	0	1	0	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Design and explain the 1 to 8 Demultiplexer. [NOV 2020]

❖ Example: *1-to-8 De-multiplexer*

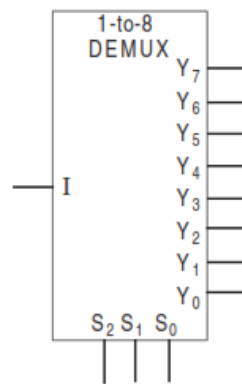
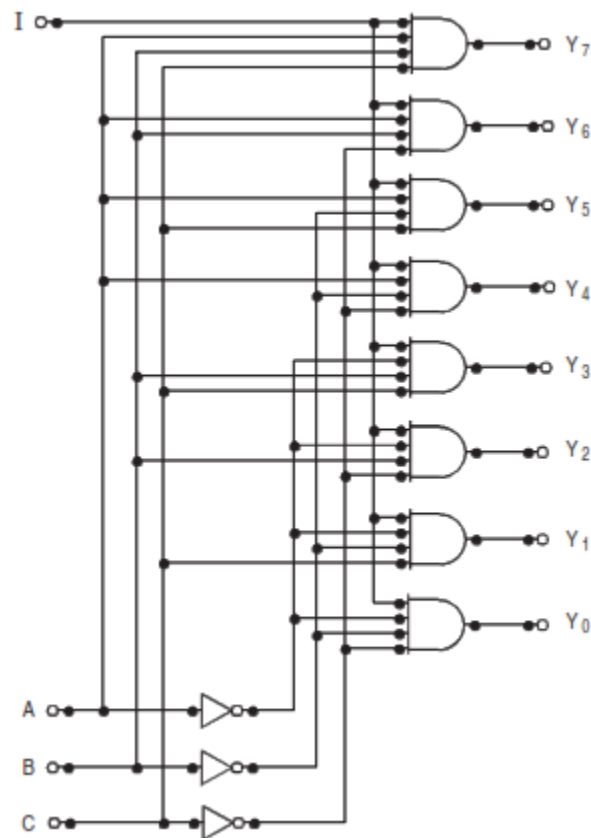


Figure 5.95

Truth Table:

Selection Inputs			Outputs							
A	B	C	Y ₀	Y ₁	Y ₂	Y ₃	Y ₄	Y ₅	Y ₆	Y ₇
0	0	0	Y ₀ = I	0	0	0	0	0	0	0
0	0	1	0	Y ₁ = I	0	0	0	0	0	0
0	1	0	0	0	Y ₂ = I	0	0	0	0	0
0	1	1	0	0	0	Y ₃ = I	0	0	0	0
1	0	0	0	0	0	0	Y ₄ = I	0	0	0
1	0	1	0	0	0	0	0	Y ₅ = I	0	0
1	1	0	0	0	0	0	0	0	Y ₆ = I	0
1	1	1	0	0	0	0	0	0	0	Y ₇ = I

Logic diagram:



Example:

1. Implement full adder using De-multiplexer.

Solution :

Step 1 : Truth table

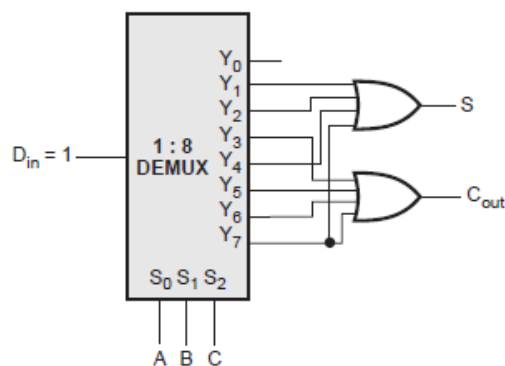
Inputs			Outputs	
A	B	C _{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Step 2 : For full adder

$$\text{Carry} = C_{\text{out}} = \sum m(3, 5, 6, 7)$$

and $\text{Sum} = S = \sum m(1, 2, 4, 7)$

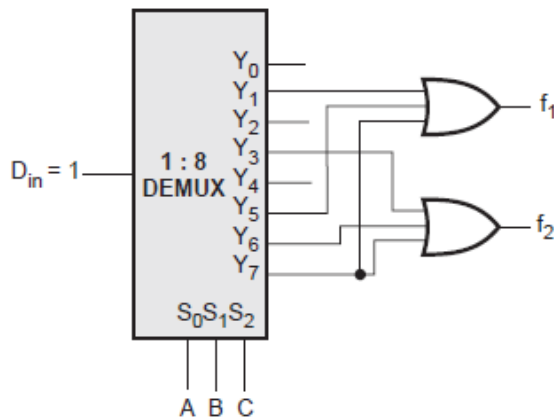
Step 3 : When $D_{\text{in}} = 1$, the demultiplexer gives minterms at the output.



2. Implement the following functions using de-multiplexer.

$$f_1(A,B,C) = \sum m(1,5,7), f_2(A,B,C) = \sum m(3,6,7)$$

Solution:



COMPARATORS

Design a 2 bit magnitude comparator.

(May 2006)

It is a combinational circuit that compares two numbers and determine their relative magnitude. The output of comparator is usually 3 binary variables indicating:

$$A < B, A = B, A > B$$

1-bit comparator: Let's begin with 1 bit comparator and from the name we can easily make out that this circuit would be used to compare 1 bit binary numbers.

A	B	A > B	A = B	A < B
0	0	0	1	0
1	0	1	0	0
0	1	0	0	1
1	1	0	1	0

For a 2-bit comparator we have four inputs A_1A_0 and B_1B_0 and three output E (is 1 if two numbers are equal) G (is 1 when $A > B$) and L (is 1 when $A < B$) If we use truth table and K-map the result is

		A>B	
A	B	0	1
	0	0	0
1	1	1	0

Equation is $A>B = A.\bar{B}$

		A<B	
A	B	0	1
	0	0	1
1	1	0	0

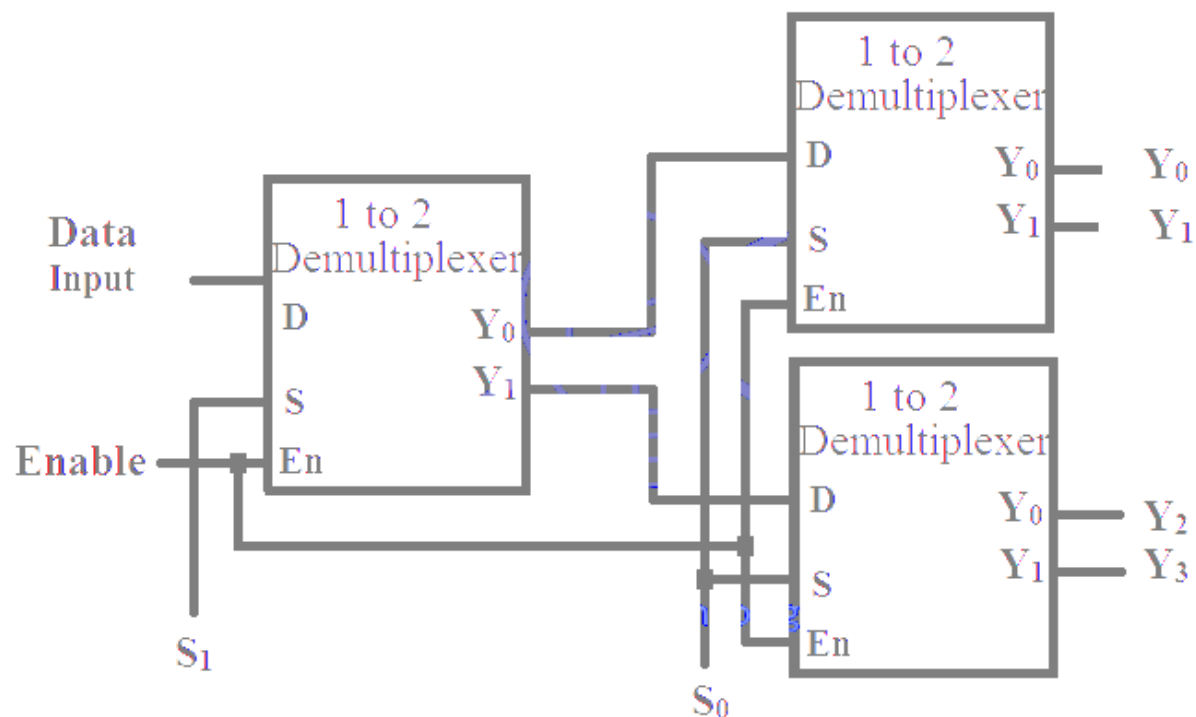
Equation is $A<B = \bar{A}.B$

		A=B	
A	B	0	1
	0	1	0
1	1	0	1

The equation is $f(A=B) = \bar{A}.\bar{B} + A.B$
 $= A \text{ XNOR } B$

Convert 2 to 4 line decoder with enable input 1:4 Demux.

(Dec 2019)



Design of 2 – bit Magnitude Comparator.

The truth table of 2-bit comparator is given in table below

Truth table:

Inputs				Outputs		
A ₃	A ₂	A ₁	A ₀	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K-Map:

For A>B

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

$$A > B = A_0 B_1' B_0' + A_1 B_1' + A_1 A_0 B_0'$$

For A=B

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

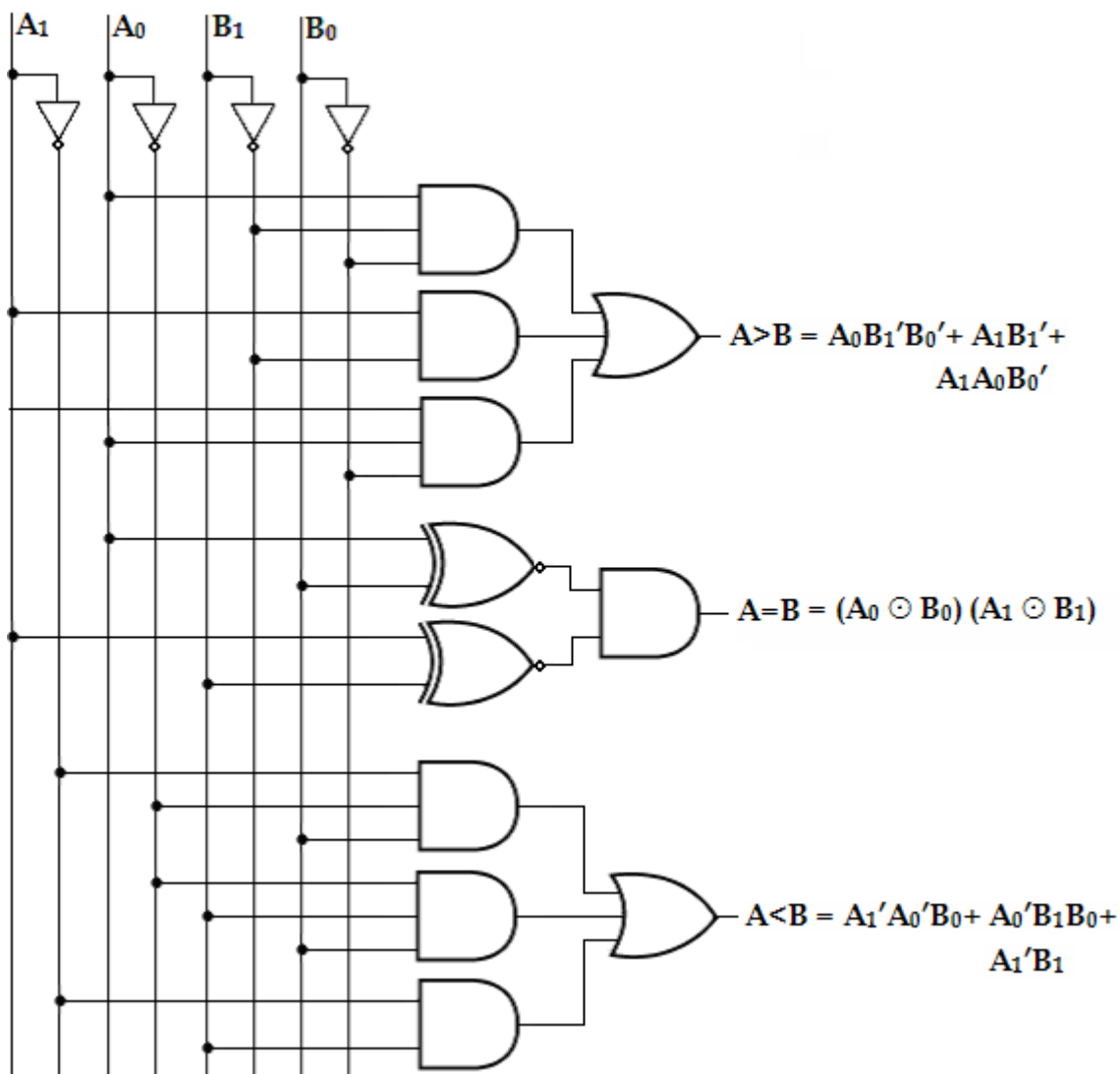
$$\begin{aligned}
 A = B &= A_1' A_0' B_1' B_0' + A_1' A_0 B_1' B_0 + \\
 &\quad A_1 A_0 B_1 B_0 + A_1 A_0' B_1 B_0' \\
 &= A_1' B_1' (A_0' B_0' + A_0 B_0) + A_1 B_1 (A_0 B_0 + A_0' B_0') \\
 &= (A_0 \odot B_0) (A_1 \odot B_1)
 \end{aligned}$$

For A<B

$A_1 A_0 \backslash B_1 B_0$	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$$A < B = A_1' A_0' B_0 + A_0' B_1 B_0 + A_1' B_1$$

Logic Diagram:



4 bit magnitude comparator:

Design a 4 bit magnitude comparators. (Apr – 2019) [NOV 2020]

Input

$$A = A_3 A_2 A_1 A_0$$

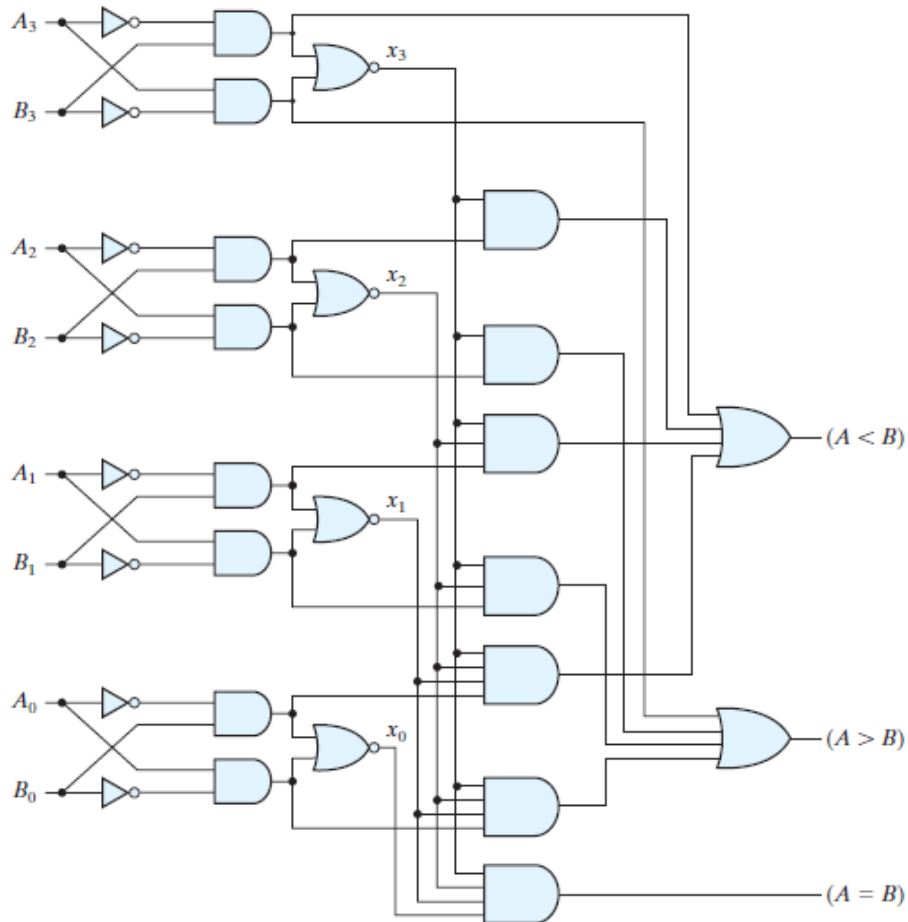
$$B = B_3 B_2 B_1 B_0$$

Function Equation

$$(A = B) = x_3 x_2 x_1 x_0$$

$$(A > B) = A_3 B'_3 + x_3 A_2 B'_2 + x_3 x_2 A_1 B'_1 + x_3 x_2 x_1 A_0 B'_0$$

$$(A < B) = A'_3 B_3 + x_3 A'_2 B_2 + x_3 x_2 A'_1 B_1 + x_3 x_2 x_1 A'_0 B_0$$



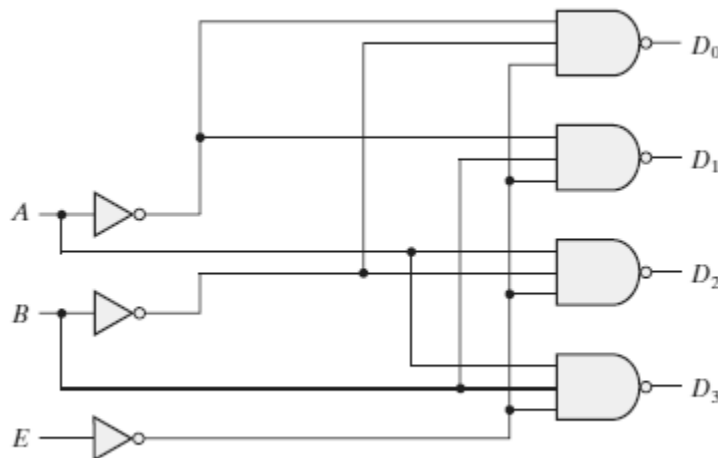
Four-bit magnitude comparator

DECODERS AND ENCODERS

Decoder:

Explain about decoders with necessary diagrams.

- ❖ A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.
- ❖ Decoders have a wide variety of applications in digital systems such as data demultiplexing, digital display, digital to analog converting, memory addressing, etc.
- ❖ **2 to 4 decoder:**



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	1

(b) Truth table

3 to 8 Decoder:

Design 3 to 8 line decoder with necessary diagram.

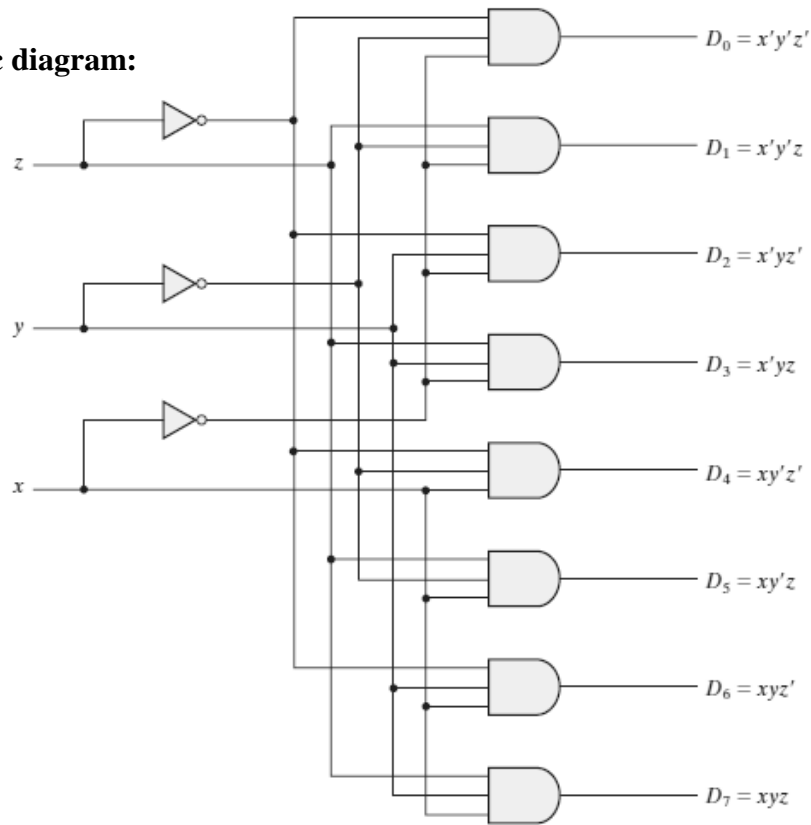
(May -10)

- ❖ The 3-to-8 line decoder consists of three input variables and eight output lines.
- ❖ Each of the output lines represents one of the minterms generated from three variables.
- ❖ The internal combinational circuit is realized with the help of INVERTER gates and AND gates.

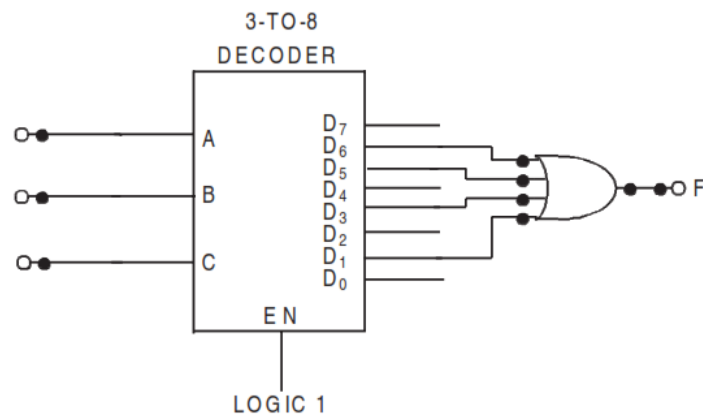
Truth table:

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

Logic diagram:



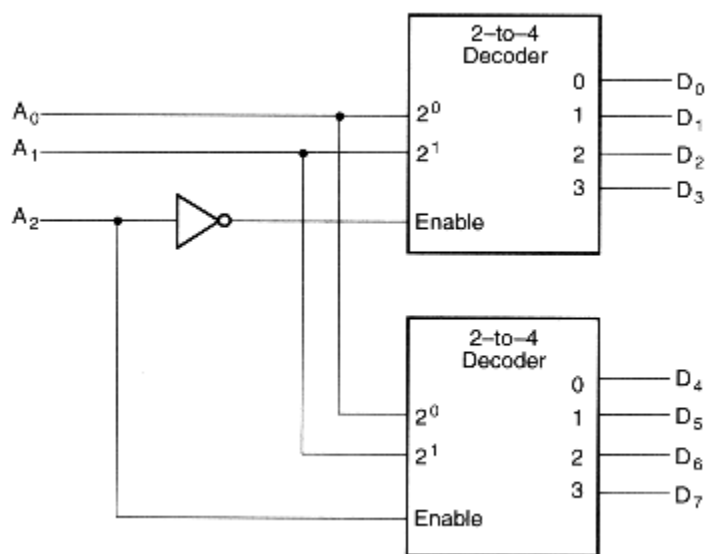
Example: *Implement the function $F(A,B,C) = \Sigma(1,3,5,6)$.*



Design for 3 to 8 decoder with 2 to 4 decoder:

- ❖ Its *enable* inputs can be used to build a three to eight decoder as follows.

- ❖ When A_2 is logic 0, a lower decoder is activated and gives output D_0 to D_3 and an upper decoder is activated for A_2 is logic 1, output D_4 to D_7 are available this time.

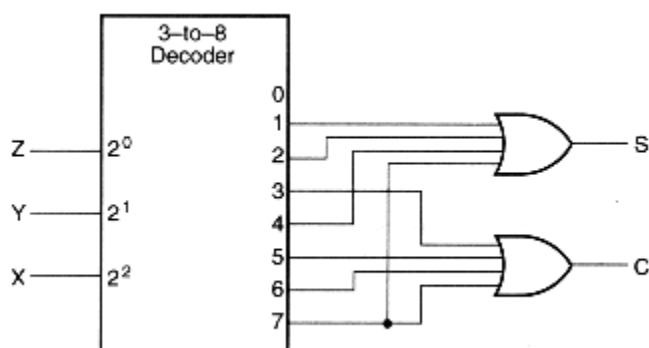


Implementation of Boolean function using decoder:

- ❖ Since the three to eight decoder provides all the minterms of three variables, the realization of function in terms of the sum of products can be achieved using a decoder and OR gates as follows.

Example: Implement full adder using decoder.

Sum is given by $\sum m(1, 2, 4, 7)$ while *Carry* is given by $\sum m(3, 5, 6, 7)$ as given by the minterms each of the OR gates are connected to.



$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz$$

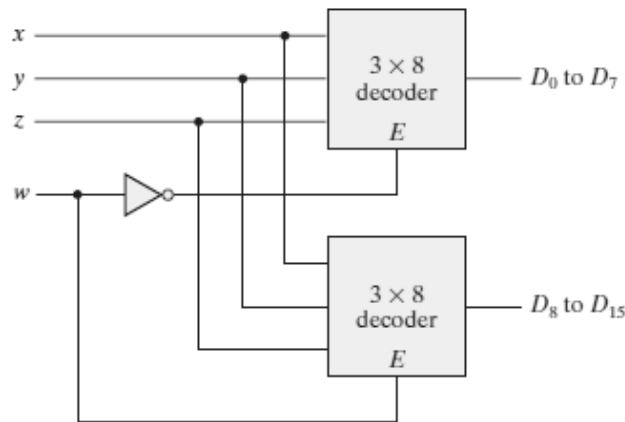
Full Adder

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

Design for 4 to 16 decoder using 3 to 8 decoder:

- A 4-to-16 line decoder has four input variables and sixteen outputs, whereas a 3-to-8 line decoder consists of three input variables and eight outputs.
- Input variables are designated as W, X, Y, and Z.

- W input is used as the ENABLE input of the upper 3-to-8 line decoder, which provides D_8 to D_{15} outputs depending on other input variables X, Y, and Z.
- W is also used as an ENABLE input at inverted mode to a lower decoder, which provides D_0 to D_7 outputs.



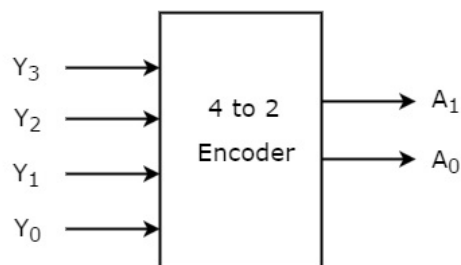
ENCODERS

Explain about encoders.

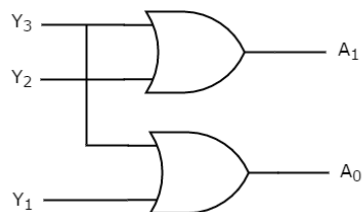
- ❖ An encoder is a digital circuit that performs the inverse operation of a decoder.
- ❖ An encoder has 2^n (or fewer) input lines and n output lines.

4 to 2 Encoder

Let 4 to 2 Encoder has four inputs Y_3 , Y_2 , Y_1 & Y_0 and two outputs A_1 & A_0 .
The block diagram of 4 to 2 Encoder is shown in the following figure.



Logic Diagram:



$$A_1 = Y_2 + Y_3$$

$$A_0 = Y_3 + Y_1$$

- ❖ The output lines, as an aggregate, generate the binary code corresponding to the input value.

interpret octal to Binary Encoder in brief. [NOV 2020]

Octal to Binary Encoder: 8 TO 3 ENCODER

- ❖ The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.
- ❖ Output z is equal to 1 when the input octal digit is 1, 3, 5, or 7.
- ❖ Output y is 1 for octal digits 2, 3, 6, or 7, and output x is 1 for digits 4, 5, 6, or 7.
- ❖ These conditions can be expressed by the following Boolean output functions:

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

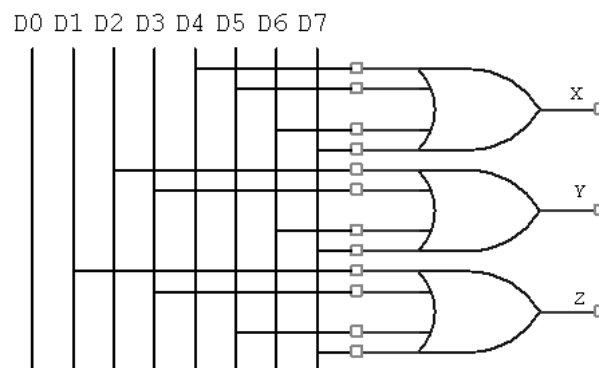
$$x = D_4 + D_5 + D_6 + D_7$$

The encoder can be implemented with three OR gates.

Truth table:

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
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	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	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

Logic Diagram:



Priority Encoder:

Design a priority encoder with logic diagram.

(May 2017, May 2019)

- Special type of encoder that senses when two or more inputs are activated simultaneously and then generates a code corresponding to the highest numbered input

Truth table:

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
0	1	0	0	0	1	1
1	1	0	0			
0	0	1	0			
0	1	1	0	1	0	1
1	0	1	0			
1	1	1	0			
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1	1	1	1
1	0	0	1			
1	0	1	1			
1	1	0	1			
1	1	1	1			
1	1	1	1			

K-Map:

D_0D_1		D_2D_3 For X			
		00	01	11	10
00	x	1	1	1	1
01	0	1	1	1	1
11	0	1	1	1	1
10	0	1	1	1	1

$$x = D_2 + D_3$$

D_0D_1		D_2D_3 For Y			
		00	01	11	10
00	x	1	1	0	0
01	1	1	1	0	0
11	1	1	1	0	0
10	0	1	1	0	0

$$y = D_3 + D_1D_2$$

D_0D_1		D_2D_3 For V			
		00	01	11	10
00	0	1	1	1	1
01	1	1	1	1	1
11	1	1	1	1	1
10	1	1	1	1	1

$$V = D_0 + D_1 + D_2 + D_3$$

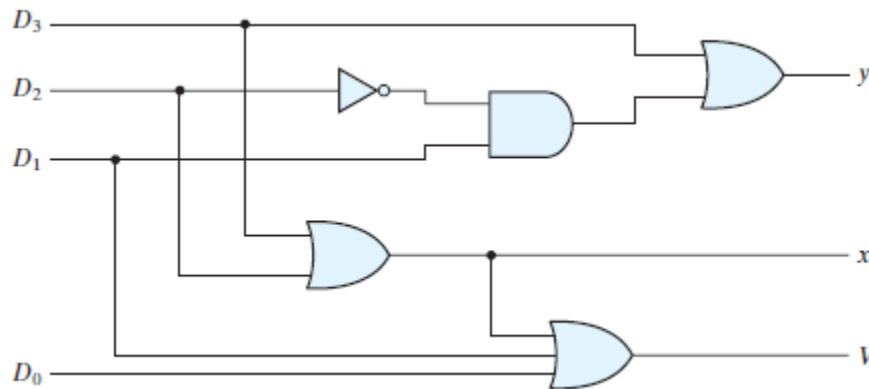
Logic Equations:

$$x = D_2 + D_3$$

$$y = D_3 + D_1D_2$$

$$V = D_0 + D_1 + D_2 + D_3$$

Logic diagram:

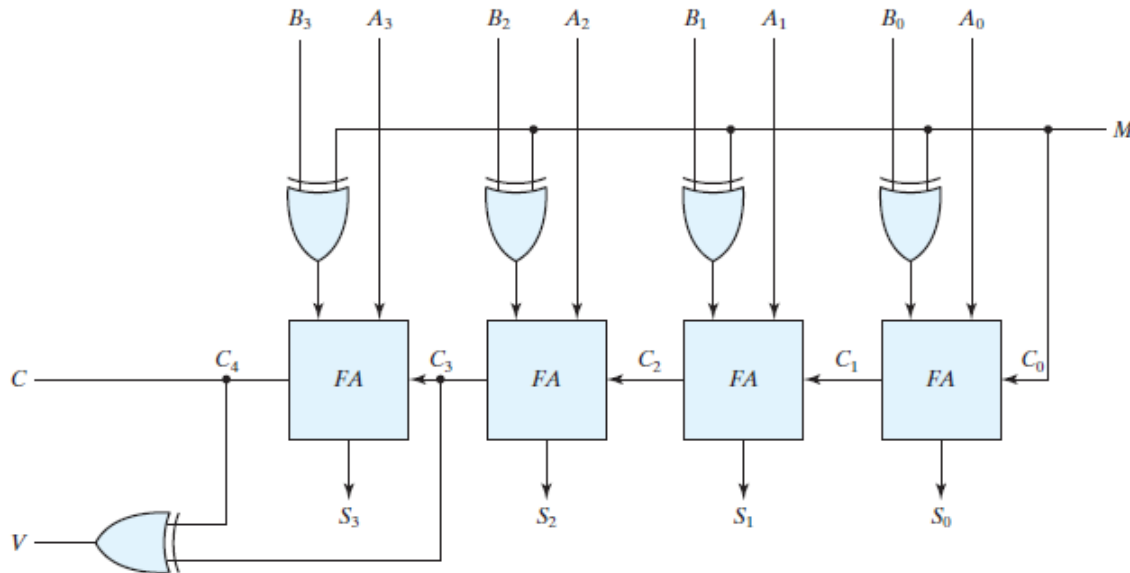


4 bit-Parallel adder/subtractor:

Explain about binary parallel / adder subtractor.

(Dec 2018)

- ❖ The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder.
- ❖ The mode input M controls the operation. When $M = 0$, the circuit is an adder, and when $M = 1$, the circuit becomes a subtractor.



- ❖ It performs the operations of both addition and subtraction.
- ❖ It has two 4bit inputs $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$.
- ❖ The mode input M controls the operation when $M=0$ the circuit is an adder and when $M=1$ the circuits become subtractor.
- ❖ Each exclusive-OR gate receives input M and one of the inputs of B .
- ❖ When $M = 0$, we have $B \text{ xor } 0 = B$. The full adders receive the value of B , the input carry is 0, and the circuit performs $A \text{ plus } B$. This results in sum $S_3S_2S_1S_0$ and carry C_4 .
- ❖ When $M = 1$, we have $B \text{ xor } 1 = B'$ and $C_0 = 1$. The B inputs are all complemented and a 1 is added through the input carry thus producing 2's complement of B .
- ❖ Now the data $A_3A_2A_1A_0$ will be added with 2's complement of $B_3B_2B_1B_0$ to produce the sum i.e., $A-B$ if $A \geq B$ or the 2's complement of $B-A$ if $A < B$.

Parity Checker / Generator:

- A parity bit is an extra bit included with a binary message to make the number of 1's either odd or even. The message, including the parity bit, is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond with the one transmitted.

- The circuit that generates the parity bit in the transmitter is called a *parity generator*. The circuit that checks the parity in the receiver is called a *parity checker*.
- In even parity system, the parity bit is '0' if there are even number of 1s in the data and the parity bit is '1' if there are odd number of 1s in the data.
- In odd parity system, the parity bit is '1' if there are even number of 1s in the data and the parity bit is '0' if there are odd number of 1s in the data.

3-bit Even Parity generator:

2019 (Dec 2018)

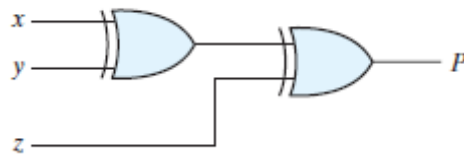
Design an even parity generator , that generates an even parity bit for every input string of 3 bits.

Truth Table:

Three-Bit Message			Parity Bit
<i>x</i>	<i>y</i>	<i>z</i>	<i>P</i>
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$P = x \oplus y \oplus z$$

Logic Diagram:

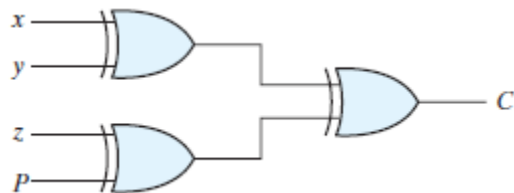


4-bit Even parity checker:
Truth Table:

Four Bits Received				Parity Error Check
<i>x</i>	<i>y</i>	<i>z</i>	<i>P</i>	<i>C</i>
0	0	0	0	0
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	0

$$C = x \oplus y \oplus z \oplus P$$

Logic Diagram:

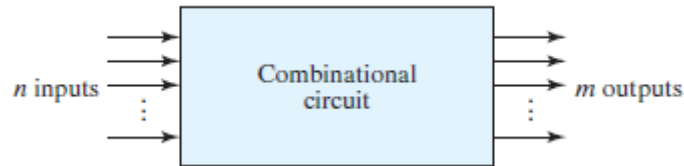


TWO MARK QUESTIONS-ANSWERS

1) Define combinational logic.

(May 2008), (Dec 2014)

A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs. A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.



2) What is sequential circuits?

- ❖ *Sequential circuits contain logic gates as well as memory cells. Their outputs depend on the present inputs and also on the states of memory elements.*
- ❖ Since the outputs of sequential circuits depend not only on the present inputs but also on past inputs.

3) Write the design procedure for combinational circuits?

The procedure involves the following steps:

1. The problem is stated.
2. Identify the input variables and output functions.
3. The input and output variables are assigned letter symbols.
4. The truth table is prepared that completely defines the relationship between the input variables and output functions.
5. The simplified Boolean expression is obtained by any method of minimization algebraic method, Karnaugh map method, or tabulation method.
6. A logic diagram is realized from the simplified expression using logic gates.

4) What is Half adder?

(May 2019)

A half-adder is an arithmetic circuit block that can be used to add two bits and produce two outputs SUM and CARRY.

The Boolean expressions for the SUM and CARRY outputs are given by the equations

$$\text{SUM } S = A.\bar{B} + \bar{A}.B$$

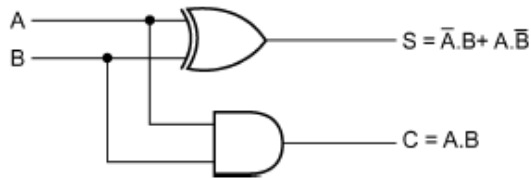
$$\text{CARRY } C = A.B$$

Input variables		Output variables	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

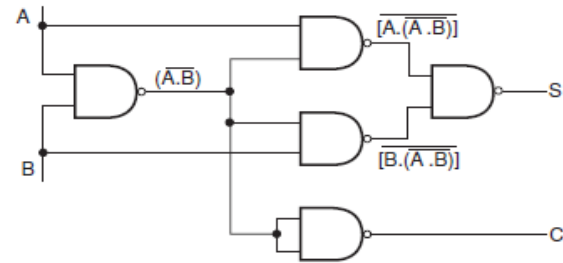
5) Draw the logic diagram of half adder using NAND gate.

(May 2006,13)

Logic Diagram:



Half adder using NAND gate:



6) What is Full adder?

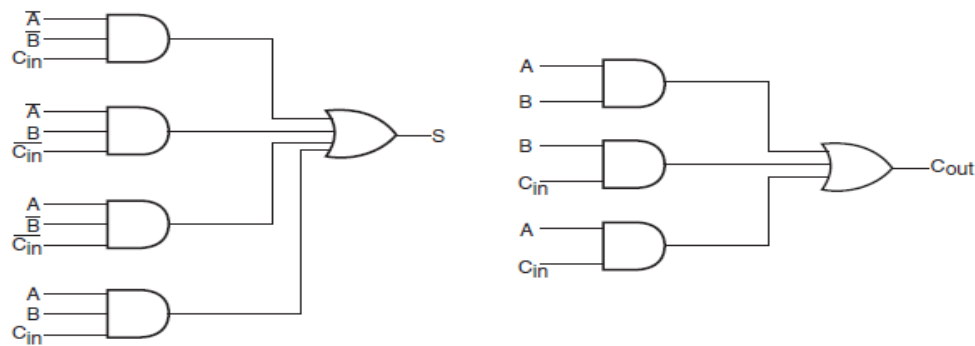
[NOV 2020]

(Dec 2013)

A Full-adder is an arithmetic circuit block that can be used to add three bits and produce two outputs SUM and CARRY.

The Boolean expressions for the SUM and CARRY outputs are given by the equations

7) Draw the Logic diagram of full adder.



8) What is Half subtractor?

(May 2005)

A half-subtractor is a combinational circuit that can be used to subtract one binary digit from another to produce a DIFFERENCE output and a BORROW output.

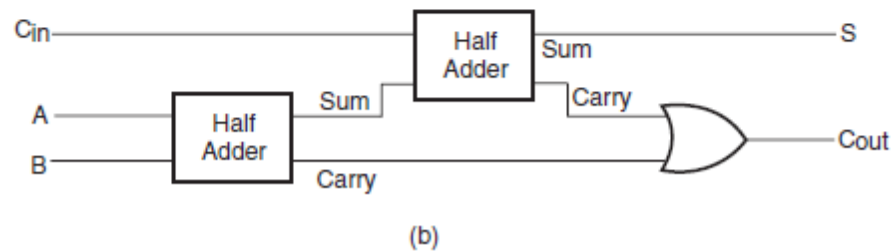
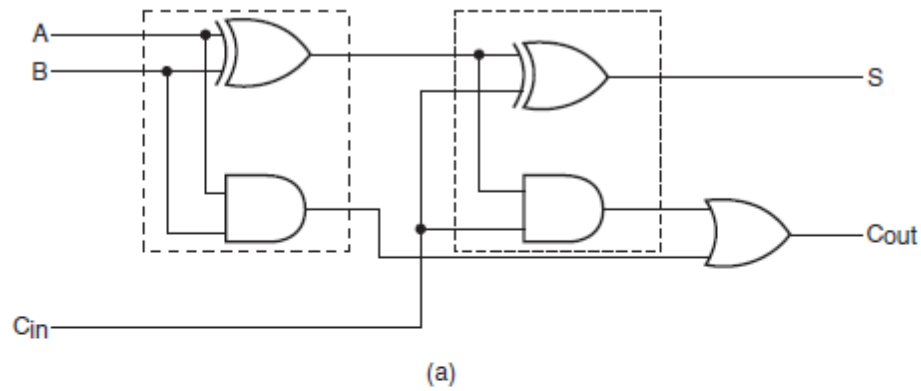
The Boolean expression for difference and barrow is:

$$D = X'Y + XY'$$

$$B = X'Y$$

9) Draw Full adder using Two half adder.

(Dec 2019 (May 2017) (Dec 2018)



10) What is Full subtractor?

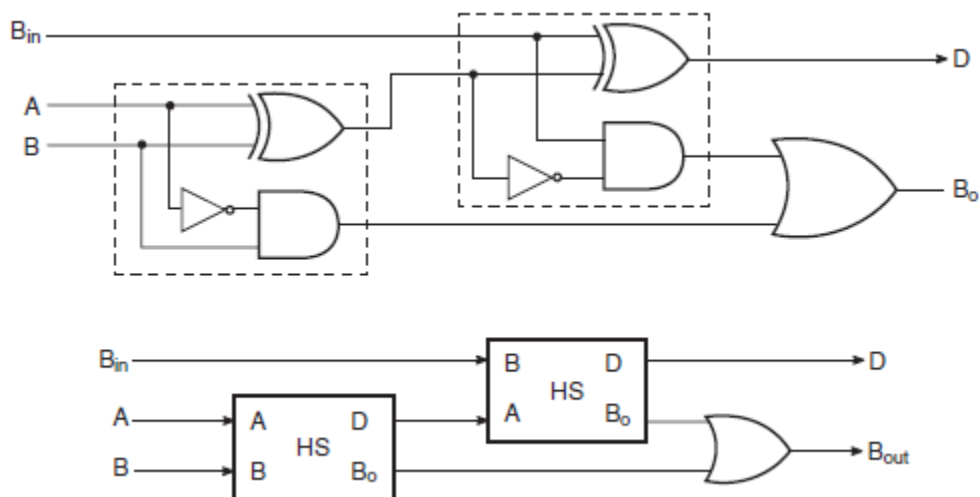
A combinational circuit of full-subtractor performs the operation of subtraction of three bits such as the minuend, subtrahend, and borrow generated from the subtraction operation of previous significant digits and produces the outputs difference and borrow.

The Boolean expression for difference and barrow is:

$$S = X'Y'Z + X'YZ' + XY'Z' + XYZ$$

$$C = X'Z + X'Y + YZ$$

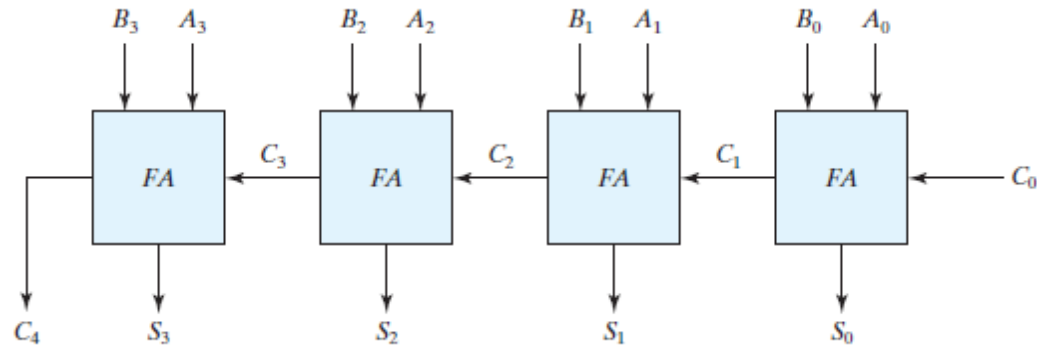
11) Draw Full subtractor using two half subtractor.



12) What is Parallel Binary Adder (Ripple Carry Adder)?

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder in the chain.

13) Draw the logic diagram for four bit binary parallel adder.



14) What is 1's complement of a number?

The 1's complement of a binary number is formed by changing 1 to 0 and 0 to 1.

Example:

1. The 1's complement of 1011000 is 0100111.
2. The 1's complement of 0101101 is 1010010.

15) What is 2's complement of a number?

The 2's complement of a binary number is formed by adding 1 with 1's complement of a binary number.

Example:

- 1) The 2's complement of 1101100 is 0010100
- 2) The 2's complement of 0110111 is 1001001

16) How Subtraction of binary numbers perform using 2's complement addition?

- ✓ The subtraction of unsigned binary number can be done by means of complements.
- ✓ Subtraction of A-B can be done by taking 2's complement of B and adding it to A.
- ✓ Check the resulting number. If carry present, the number is positive and remove the carry.
- ✓ If no carry present, the resulting number is negative, take the 2's complement of result and put negative sign.

(a) $X - Y$ and (b) $Y - X$ by using 2's complements.

(a) $X = 1010100$

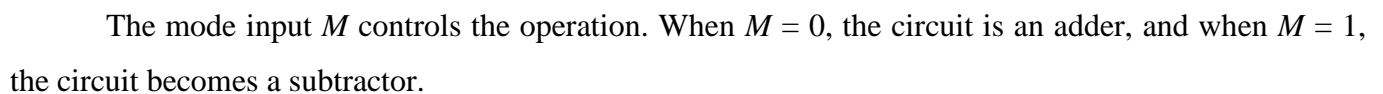
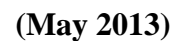
Sum= 10010001

(b) $Y = 1000011$

Sum= 1101111

Therefore, the answer is $Y - X = -(2\text{'s complement of } 1101111) = -0010001$.

(May 2018)



20) What is Magnitude Comparator?**(Dec 2017)**

The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number.

A magnitude comparator is a combinational circuit that compares two numbers A and B and determines their relative magnitudes.

The outcome of the comparison is specified by three binary variables that indicate whether $A > B$, $A = B$, or $A < B$.

21) Design a 1-bit Magnitude Comparator.**Truth table:**

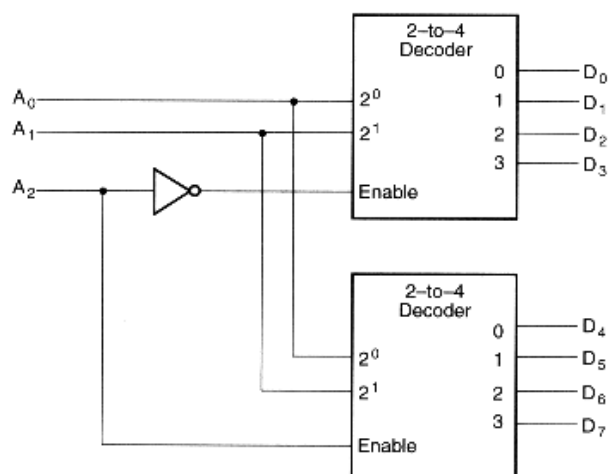
Inputs		Outputs		
B	A	$A > B$	$A = B$	$A < B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

22) What is Decoder? Define binary decoder. [NOV 2020]**(Dec 2014, May 2019)**

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.

23) Give some applications of Decoders.

Decoders have a wide variety of applications in digital systems such as data demultiplexing, digital display, digital to analog converting, memory addressing, etc.

24) Design a 3 to 8 decoder with 2 to 4 decoder.

25) What is Encoder?**(May 2012)**

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or fewer) input lines and n output lines. The output lines, as an aggregate, generate the binary code corresponding to the input value.

26) What is Priority Encoder?

- A priority encoder is an encoder circuit that includes the priority function.
- Special type of encoder that senses when two or more inputs are activated simultaneously and then generates a code corresponding to the highest numbered input.

27) Define Multiplexer (MUX) (or) Data Selector.**(Dec 2006, May 2011)**

A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.

The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

28) What is De-multiplexer?

The de-multiplexer performs the inverse function of a multiplexer, that is it receives information on one line and transmits its onto one of 2^n possible output lines. The selection is by n input select lines.

29) Give the applications of Demultiplexer.

- i) It finds its application in Data transmission system with error detection.
- ii) One simple application is binary to Decimal decoder.

30) Mention the uses of Demultiplexer.**(May 2015)**

Demultiplexer is used in computers when a same message has to be sent to different receivers. Not only in computers, but any time information from one source can be fed to several places.

33) Give other name for Multiplexer and Demultiplexer.

Multiplexer is otherwise called as Data selector.

Demultiplexer is otherwise called as Data distributor.

34) What is the function of the enable input in a Multiplexer?

The function of the enable input in a MUX is to control the operation of the unit.

35) List out the applications of decoder?

(Dec 2006)

- a. Decoders are used in counter system.
- b. They are used in analog to digital converter.
- c. Decoder outputs can be used to drive a display system.

36) What are the Application of MUX?

1. They are used as a data selector to select one output of many data inputs.
2. They can be used to implement combinational logic circuits
3. They are used in time multiplexing systems.
4. They are used in frequency multiplexing systems.
5. They are used in A/D & D/A Converter.
6. They are used in data acquisition system.

37) List out the applications of comparators?

- a. Comparators are used as a part of the address decoding circuitry in computers to select a specific input/output device for the storage of data.
- b. They are used to actuate circuitry to drive the physical variable towards the reference value.
- c. They are used in control applications.

38) What is carry look-ahead addition?

The speed with which an addition is performed limited by the time required for the carries to propagate or ripple through all of the stage of the adder. One method of speeding up the process is by eliminating the ripple carry delay.

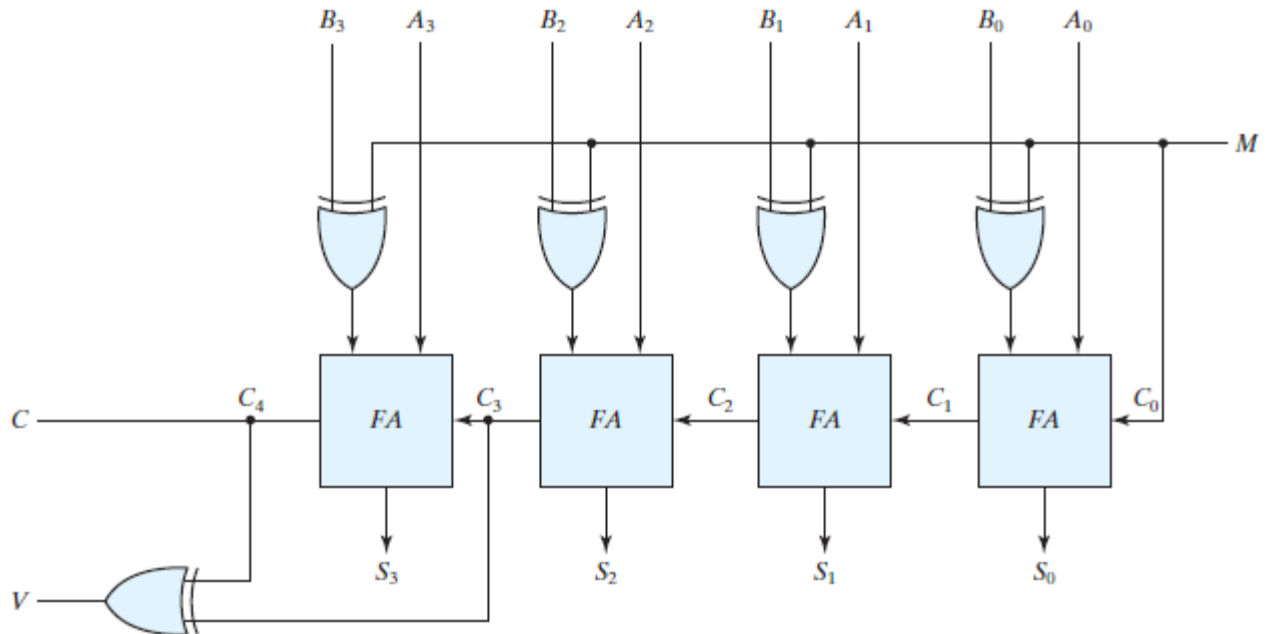
39) What are the Difference between Decoder & Demux.?

S.No	Decoder	Demux
1	Decoder is a many inputs to many outputs	Demux is a single input to many outputs
2	There are no selection lines.	The selection of specific output line is controlled by the value of selection lines.

Parallel adder/subtractor:

Explain the working of 4bit Adder- subtractor circuit.

(May 2019)



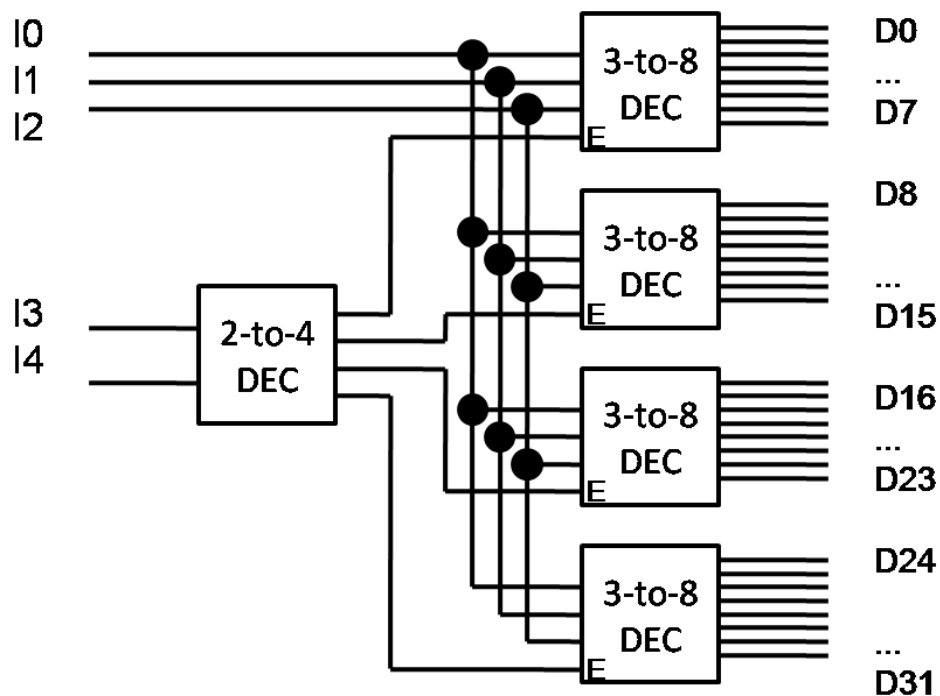
- The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder.
- The mode input M controls the operation. When $M = 0$, the circuit is an adder, and when $M = 1$, the circuit becomes a subtractor.
- Each exclusive-OR gate receives input M and one of the inputs of B .
 - When $M = 0$, we have $B \text{ xor } 0 = B$. The full adders receive the value of B , the input carry is 0, and the circuit performs A plus B .
 - When $M = 1$, we have $B \text{ xor } 1 = B'$ and $C_0 = 1$. The B inputs are all complemented and a 1 is added through the input carry.
- The circuit performs the operation A plus the 2's complement of B . (The exclusive-OR with output V is for detecting an overflow.)
- The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be signed or unsigned.
- The binary adder-subtractor circuit with outputs C and V is shown in Fig.
- If the two binary numbers are considered to be unsigned, then the C bit detects a carry after addition or a borrow after subtraction.
- If the numbers are considered to be signed, then the V bit detects an overflow.
 - If $V = 0$ after an addition or subtraction, then no overflow occurred and the n -bit result is correct.
 - If $V = 1$, then the result of the operation contains $n + 1$ bits, but only the rightmost n bits of the number fit in the space available, so an overflow has occurred.

1. Design a 5x 32 decoder using 3x8 decoder and summarize how many decoders required (

connect d3 and d4 to 2-to-4 line decoder
connect d0, d1, and d2 to all 3-to-8 line decoders.

Now connect output of 2-to-4 line decoder to enable pins of 3-to-8 line decoders such that the first makes first 3-to-8 line decoders enable.

that's it 32 output of 3-to-8 line decoders are your required output



5-to-32 line decoder

2 marks

(May 2018)

1. A binary ripple counter is required to count up to 16,383. How many Flipflops are required? If the frequency is 8.192 MHz, What is the frequency at the output of MSB?

3 Flip flops are required.
Average frequency = $3 / 8 * 8.192 = 3.072\text{M}$

(May 2018)

2. What is the basic principle used to check or generate the proper parity bit in a given code word?
The modulo sum of an even number of 1s is always 0 and the modulo sum of an odd number is always 1.

CODE CONVERSION

Design a binary to gray converter.
2017)

(Nov-2009)(Nov

Binary to Grayconverter

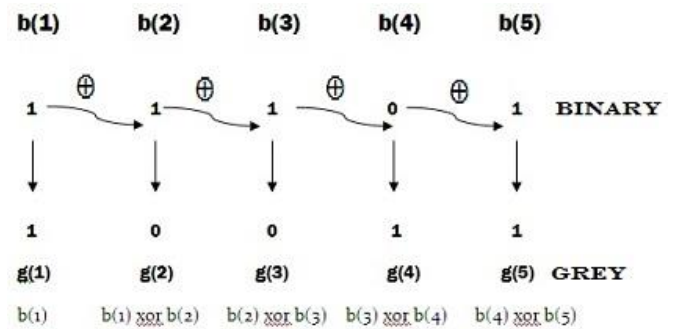
Gray code is unit distance code.

Input code: Binary [B_3 B_2 B_1 B_0]

output code: Gray [G_3 G_2 G_1 G_0]

Truth Table

B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



K-MAP FORG3:

$B_1 B_0$	00	01	11	10
11	0	0	0	0
01	0	0	0	0
10	1	1	1	1
00	1	1	1	1

$$G_3 = B_3$$

K-MAP FORG2:

$B_1 B_0$	00	01	11	10
11	0	0	0	0
01	1	1	1	1
10	0	0	0	0
00	1	1	1	1

$$G_2 = B_3' B_2 + B_3 B_2' = B_3 \oplus B_2$$

K-MAP FORG1:

B1B0 \ B3B2	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

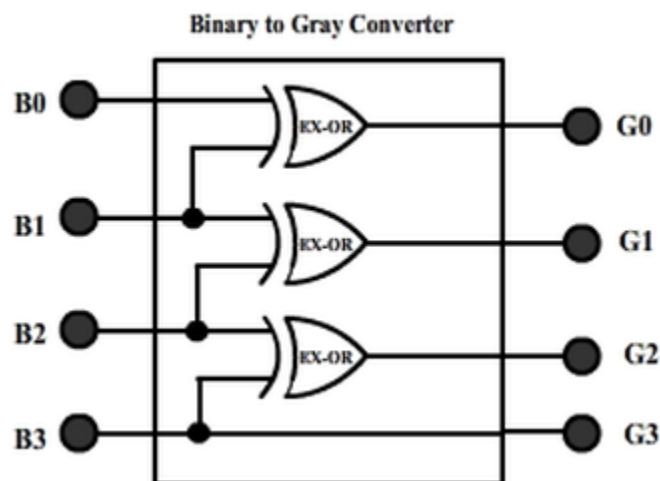
$$G1 = B1'B2 + B1B2' = B1 \oplus B2$$

K-MAP FORG0:

B1B0 \ B3B2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$G0 = B1'B0 + B1B0' = B1 \oplus B0$$

Logic diagram:



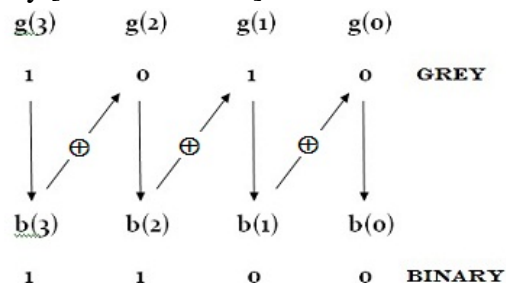
Gray to Binary converter:

Design a gray to binary converter.(OR) Design a combinational circuit that converts a four bit gray code to a four bit binary number using exclusive – OR gates. (Nov-2009) [NOV – 2019]

Gray code is unit distance code.

Input code: Gray [G₃ G₂ G₁ G₀]

output code: Binary [B₃ B₂ B₁ B₀]



i.e

$$b(3) = g(3)$$

$$b(2) = b(3) \oplus g(2)$$

$$b(1) = b(2) \oplus g(1)$$

$$b(0) = b(1) \oplus g(0)$$

Truth Table:

Gray code				Natural-binary code			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

K-Map:

For B₃

		G ₁ G ₀			
		00	01	11	10
G ₃ G ₂	00	0	0	0	0
	01	0	0	0	0
	11	1	1	1	1
	10	1	1	1	1

$$B_3 = G_3$$

For B₂

		G ₁ G ₀			
		00	01	11	10
G ₃ G ₂	00	0	0	0	0
	01	1	1	1	1
	11	0	0	0	0
	10	1	1	1	1

$$B_2 = G_3'G_2 + G_3G_2'$$

$$= G_3 \oplus G_2$$

For B₁

		G ₁ G ₀			
		00	01	11	10
G ₃ G ₂	00	0	0	1	1
	01	1	1	0	0
	11	0	0	1	1
	10	1	1	0	0

For B₀

		G ₁ G ₀			
		00	01	11	10
G ₃ G ₂	00	0	1	0	1
	01	1	0	1	0
	11	0	1	0	1
	10	1	0	1	0

From the above K-map,

$$B_3 = G_3$$

$$B_2 = G_3'G_2 + G_3G_2'$$

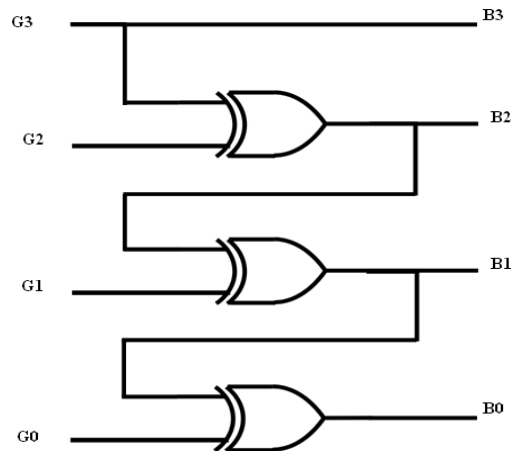
$$B_2 = G_3 \oplus G_2$$

$$\begin{aligned} B_1 &= G_3'G_2'G_1 + G_3'G_2G_1' + G_3G_2G_1 + G_3G_2'G_1' \\ &= G_3' (G_2'G_1 + G_2G_1') + G_3 (G_2G_1 + G_2'G_1') \\ &= G_3' (G_2 \oplus G_1) + G_3 (G_2 \oplus G_1)' \quad [x \oplus y = x'y + xy'], [(x \oplus y)' = xy + x'y'] \end{aligned}$$

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

$$\begin{aligned} B_0 &= G_3'G_2'G_1'G_0 + G_3'G_2'G_1G_0' + G_3'G_2G_1'G_0 + G_3'G_2G_1G_0' + G_3'G_2G_1'G_0' + \\ &\quad G_3'G_2G_1G_0' + G_3'G_2G_1G_0 + G_3'G_2G_1G_0' \\ &= G_3'G_2' (G_1'G_0 + G_1G_0') + G_3'G_2 (G_1'G_0 + G_1G_0') + G_1'G_0' (G_3'G_2 + G_3G_2') + \\ &\quad G_1G_0 (G_3'G_2 + G_3G_2') \\ &= G_3'G_2' (G_0 \oplus G_1) + G_3'G_2 (G_0 \oplus G_1) + G_1'G_0' (G_2 \oplus G_3) + G_1G_0 (G_2 \oplus G_3) \\ &= G_0 \oplus G_1 (G_3'G_2' + G_3G_2) + G_2 \oplus G_3 (G_1'G_0' + G_1G_0) \\ &= (G_0 \oplus G_1) (G_2 \oplus G_3)' + (G_2 \oplus G_3) (G_0 \oplus G_1) \quad [x \oplus y = x'y + xy'] \\ B_0 &= (G_0 \oplus G_1) \oplus (G_2 \oplus G_3). \end{aligned}$$

Logic Diagram:



BCD to Excess -3 converter:

Design a combinational circuits to convert binary coded decimal number into an excess-3 code.

- ❖ Excess-3 code is modified form of BCD code. (Nov-06,09,10, May-08,10)
- ❖ Excess -3 code is derived from BCD code by adding 3to each coded number.

Truth table:

Decimal	BCD code				Excess-3 code			
	B ₃	B ₂	B ₁	B ₀	E ₃	E ₂	E ₁	E ₀
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

K-Map:

For E₃

B ₃ B ₂	B ₁ B ₀			
	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	x	x	x	x
10	1	1	x	x

$$E_3 = B_3 + B_2 (B_0 + B_1)$$

For E₁

B ₃ B ₂	B ₁ B ₀			
	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	x	x	x	x
10	1	0	x	x

$$E_1 = B_1' B_0' + B_1 B_0$$

$$= B_1 \odot B_0$$

For E₂

B ₃ B ₂	B ₁ B ₀			
	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	x	x	x	x
10	0	1	x	x

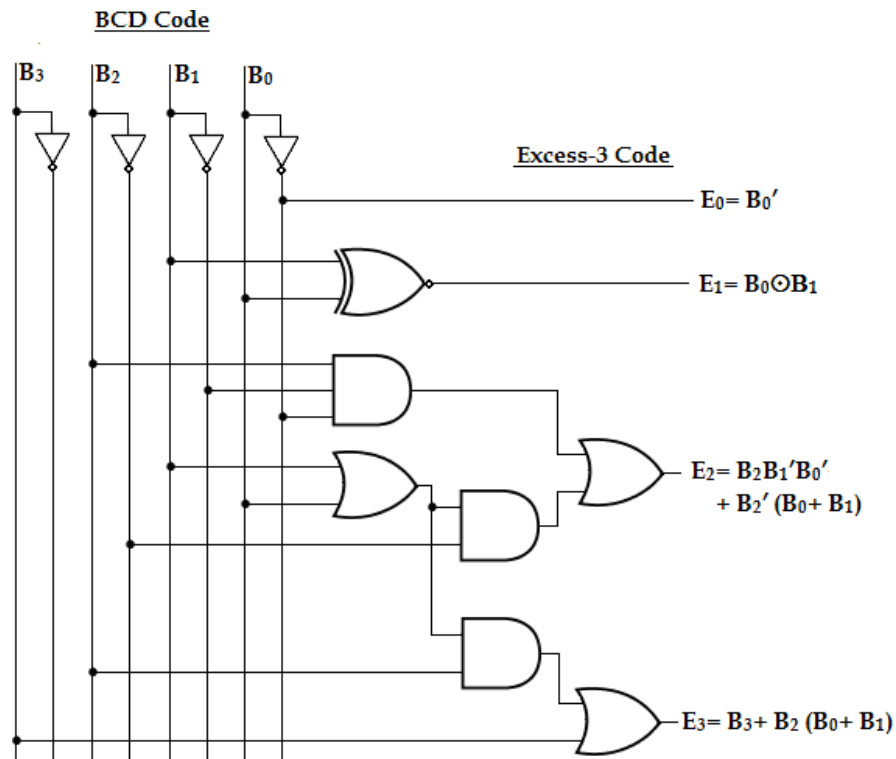
$$E_2 = B_2 B_1' B_0' + B_2' (B_0 + B_1)$$

For E₀

B ₃ B ₂	B ₁ B ₀			
	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	x	x	x	x
10	1	0	x	x

$$E_0 = B_0'$$

Logic Diagram



Excess -3 to BCD converter:

Design a combinational circuit to convert Excess-3 to BCD code.

(May 2007)

Truth table:

Decimal	Excess-3 code				BCD code			
	E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
3	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	0	1	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	0	1	0	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	1	1
11	1	0	1	1	1	0	0	0
12	1	1	0	0	1	0	0	1

K-map simplification

For B_0

$E_3E_2 \backslash E_1E_0$	00	01	11	10
00	X	X	0	X
01	1	0	0	1
11	1	X	X	X
10	1	0	0	1

$B_0 = \overline{E}_0$

For B_1

$E_3E_2 \backslash E_1E_0$	00	01	11	10
00	X	X	0	X
01	0	1	0	1
11	0	X	X	X
10	0	1	0	1

$B_1 = \overline{E}_1E_0 + E_1\overline{E}_0$
 $= E_1 \oplus E_0$

For B_2

$E_3E_2 \backslash E_1E_0$	00	01	11	10
00	X	X	0	X
01	0	0	1	0
11	0	X	X	X
10	1	1	0	1

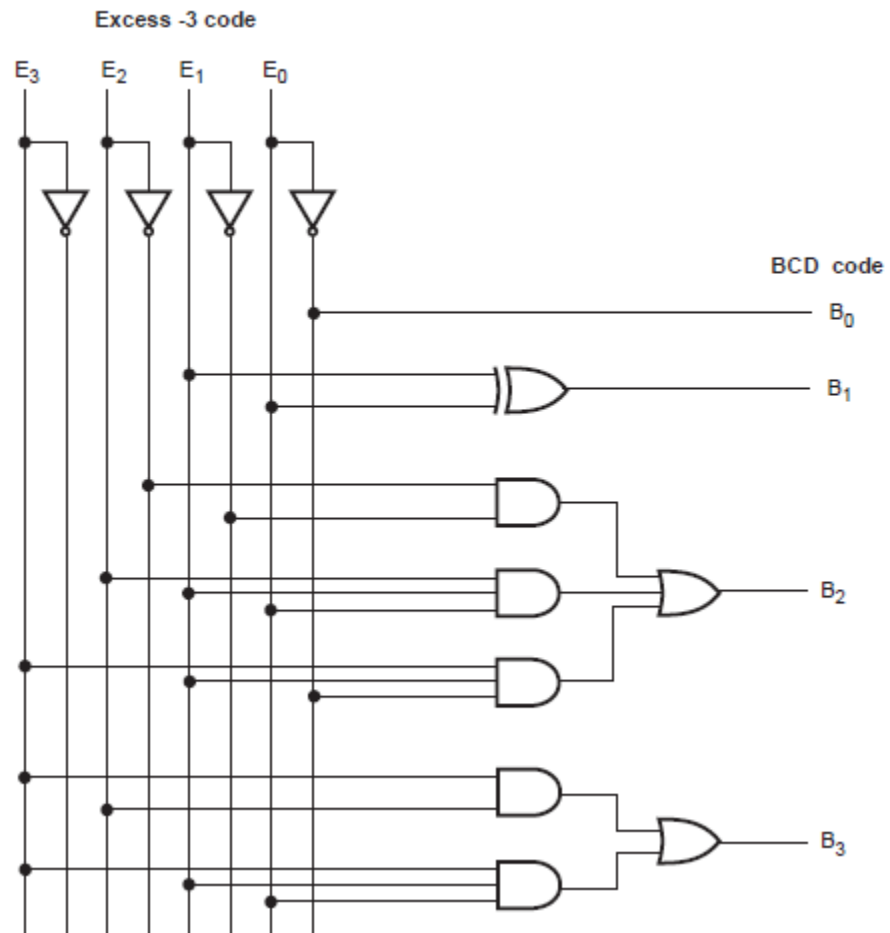
$B_2 = \overline{E}_2\overline{E}_1 + E_2E_1E_0 + E_3E_1\overline{E}_0$

For B_3

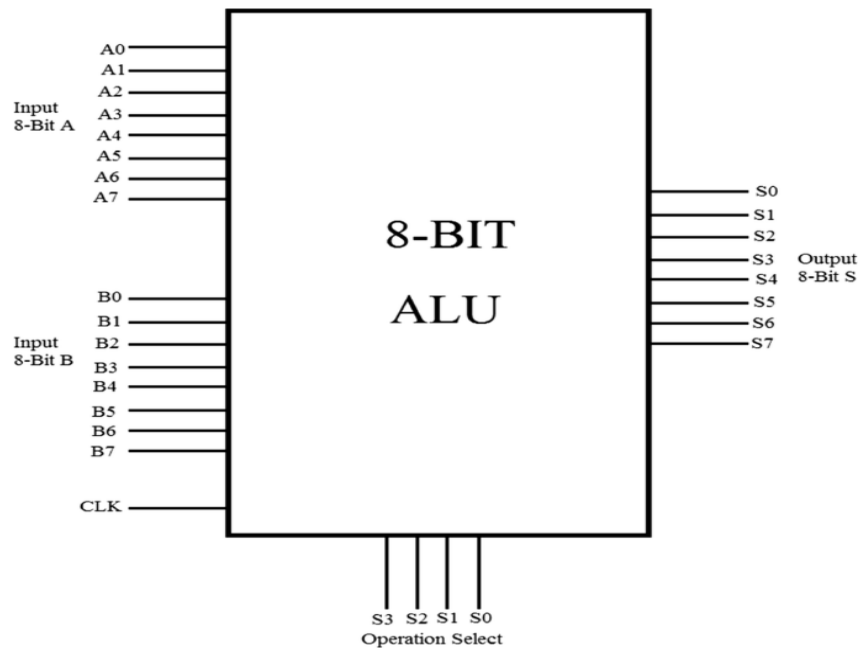
$E_3E_2 \backslash E_1E_0$	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	1	X	X	X
10	0	0	1	0

$B_3 = E_3E_2 + E_3E_1E_0$

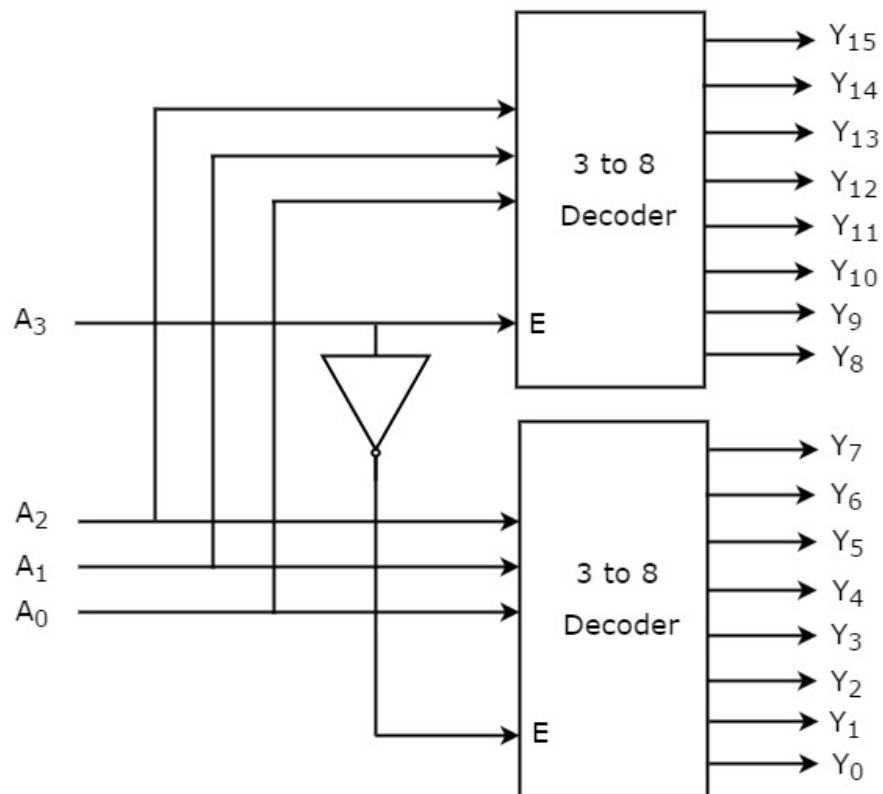
Logic diagram



Digital trans-receiver / 8 bit Arithmetic and logic unit



The block diagram of 4 to 16 decoder using 3 to 8 decoders

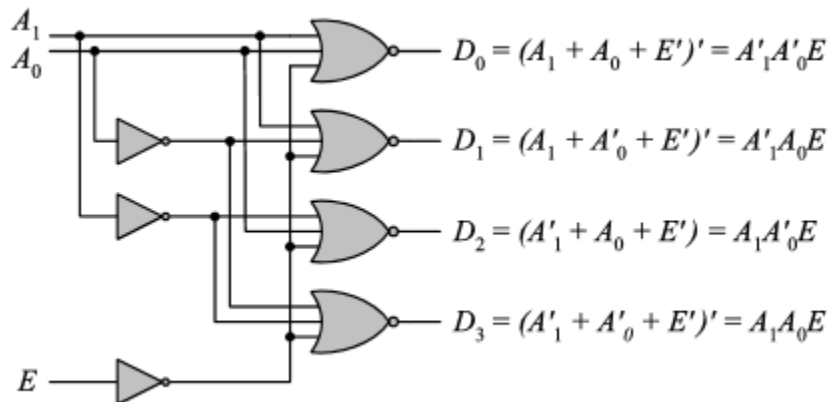


Design an octal-to-binary priority encoder. Provide an output ‘V’ to indicate that at least one of the inputs is present. The input with the highest subscript number has the highest priority. What will be the value of the four outputs if inputs D2 and D6 are 1 at the same time?
[NOV / DEC 2021]

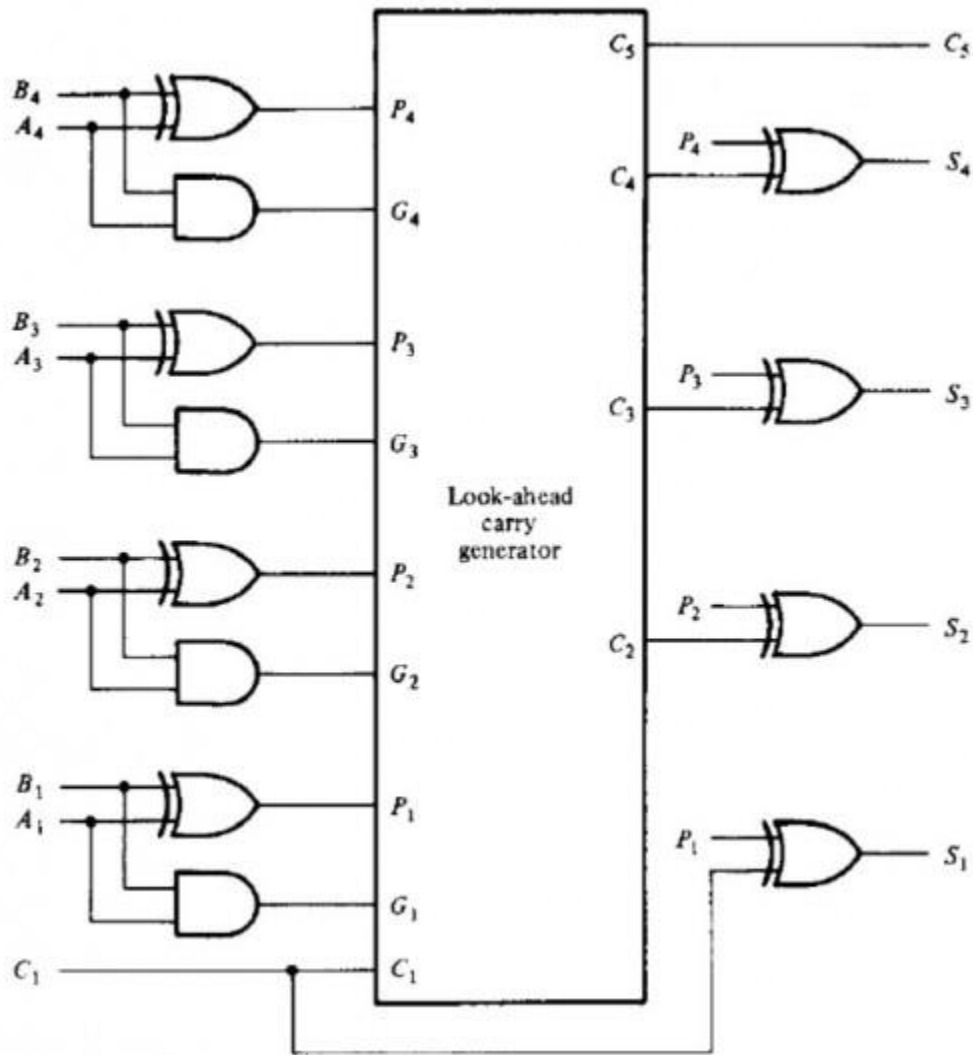
Inputs								Outputs			
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z	V
0	0	0	0	0	0	0	0	x	x	x	0
1	0	0	0	0	0	0	0	0	0	0	1
x	1	0	0	0	0	0	0	0	0	1	1
x	x	1	0	0	0	0	0	0	1	0	1
x	x	x	1	0	0	0	0	0	1	1	1
x	x	x	x	1	0	0	0	1	0	0	1
x	x	x	x	x	1	0	0	1	0	1	1
x	x	x	x	x	x	1	0	1	0	0	1
x	x	x	x	x	x	x	1	1	1	1	1

If $D_2 = 1, D_6 = 1$, all others = 0
Output xyz = 100 and $V = 1$

Draw the logic diagram for 2 to 4 line decoder using NOR gates. [NOV / DEC 2021]



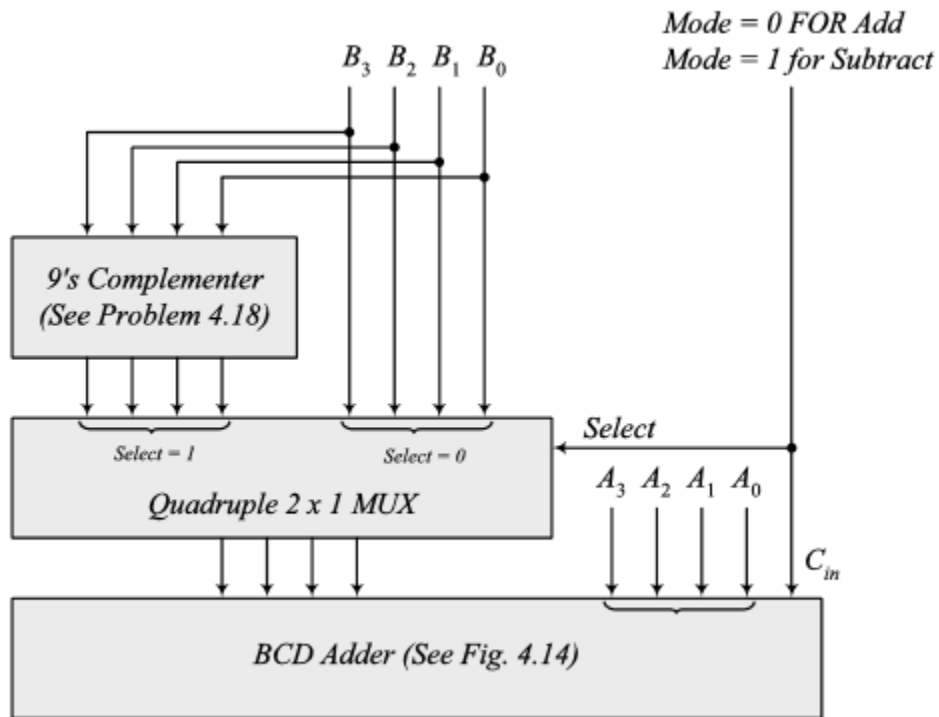
Assume that the Ex-OR gate has a propagation delay of 10 ns and that the AND or OR gates have a propagation delay of 5 ns. What is the total propagation delay time in the four-bit adder?
[NOV / DEC 2021]



xor AND OR XOR

$$10 + 5 + 5 + 10 = 30 \text{ ns}$$

Construct the BCD adder-subtractor circuit. Explain its working by providing relevant inputs to the circuit.
[NOV / DEC 2021]



9'S COMPLEMENTER:

Inputs $ABCD$	Outputs $wxyz$
0000	1001
0001	1000
0010	0111
0011	0110
0100	0101
0101	0100
0110	0011
0111	0010
1000	0001
1001	0000

$d(A, b, c, d) = \Sigma(10, 11, 12, 13, 14, 15)$

AB \ CD		C			
		00	01	11	10
A	00	m_0 1	m_1 1	m_3	m_2
	01	m_4	m_5	m_7	m_6
	11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
	10	m_8	m_9	m_{11} x	m_{10} x

$$w = A'B'C'$$

AB \ CD		C			
		00	01	11	10
A	00	m_0	m_1	m_3 1	m_2 1
	01	m_4 1	m_5 1	m_7	m_6
	11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
	10	m_8	m_9	m_{11} x	m_{10} x

$$x = BC' + B'C = B \oplus C$$

CD		C			
		00	01	11	10
A	00	m_0	m_1	m_3 1	m_2 1
	01	m_4	m_5	m_7 1	m_6 1
	11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
	10	m_8	m_9	m_{11} x	m_{10} x

$$y = C$$

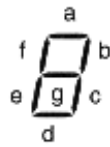
AB \ CD		C			
		00	01	11	10
A	00	m_0 1	m_1	m_3	m_2 1
	01	m_4 1	m_5 1	m_7	m_6 1
	11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
	10	m_8 1	m_9	m_{11} x	m_{10} x

$$z = D'$$

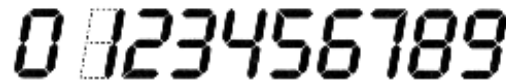
BCD to seven segment decoder

Design a BCD to seven segment code converter.

(May-06,10, Nov- 09)



(a) Segment designation



(b) Numeric designation for display

Truth table:

	BCD code				7-Segment code						
Digit	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

K-Map:

For (a)

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$$a = A + C + BD + B'D'$$

For (b)

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

$$b = B' + C'D' + CD$$

For (c)

AB \ CD	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$$c = B + C' + D$$

For (d)

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	X	X	X	X
10	1	1	X	X

$$d = B'D' + CD' + BC'D + B'C + A$$

For (e)

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

$$e = B'D' + CD'$$

For (f)

AB \ CD	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

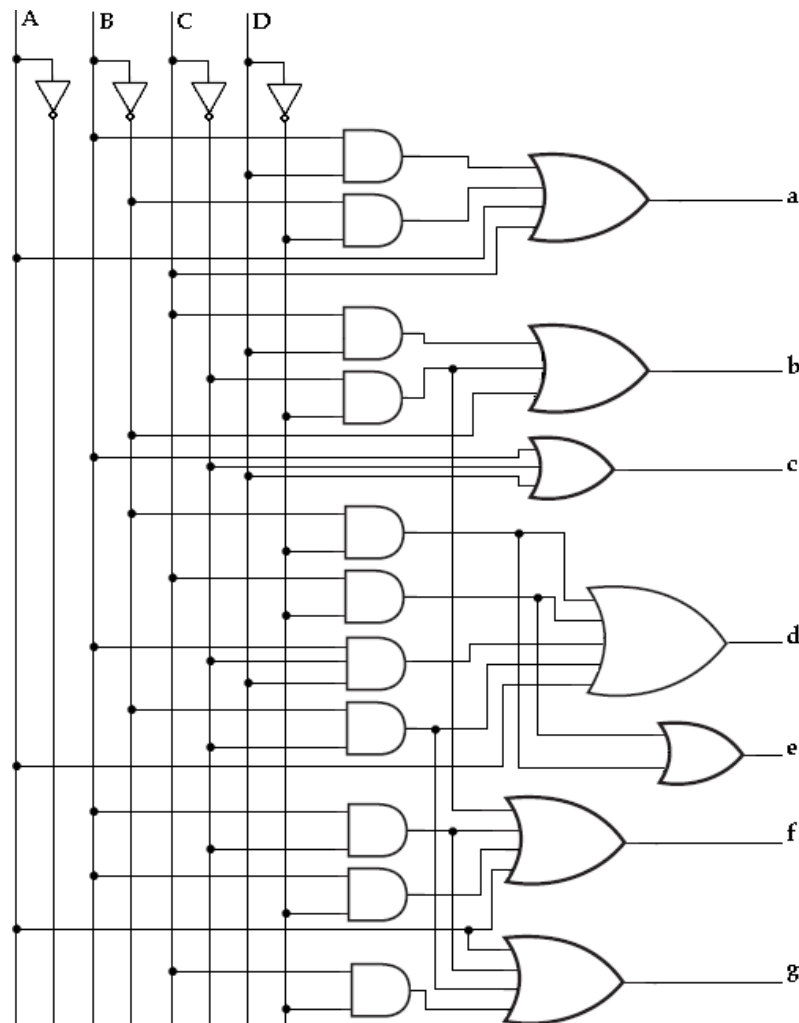
$$f = A + C'D' + BC' + BD'$$

For (g)

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

$$g = A + BC' + B'C + CD'$$

Logic Diagram:



- ❖ The specification above requires that the output be zeroes (none of the segments are lighted up) when the input is not a BCD digit.
- ❖ In practical implementations, this may defer to allow representation of hexadecimal digits using the seven segments.

UNIT III

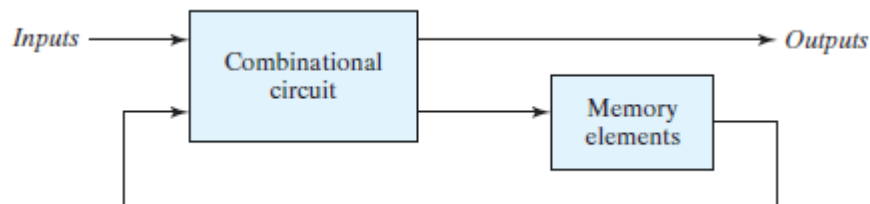
SYNCHRONOUS SEQUENTIAL CIRCUITS

Flip flops – SR, JK, T, D, Master/Slave FF – operation and excitation tables, Triggering of FF, Analysis and design of clocked sequential circuits – Design - Moore/Mealy models, state minimization, state assignment, circuit implementation – Design of Counters- Ripple Counters, Ring Counters, Shift registers, Universal Shift Register.

SEQUENTIAL CIRCUITS

Sequential circuits:

- Sequential circuits employ storage elements in addition to logic gates. Their outputs depend upon the function of the inputs and the state of the storage elements.
- Because the state of the storage elements is a function of previous inputs, the outputs of a sequential circuit depend not only on present values of inputs, but also on past inputs, and the circuit behavior must be specified by a time sequence of inputs and internal states.



Types of sequential circuits:

There are two main types of sequential circuits, and their classification is a function of the timing of their signals.

1. Synchronous sequential circuit:

It is a system whose behavior can be defined from the knowledge of its signals at discrete instants of time.

2. Asynchronous sequential circuits:

The behavior of an asynchronous sequential circuit depends upon the input signals at any instant of time and the order in which the inputs change. The storage elements commonly used in asynchronous sequential circuits are time-delay devices.

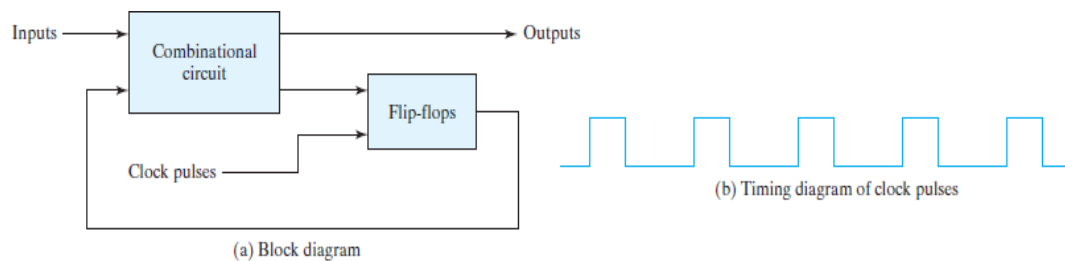
LATCHES AND FLIP FLOPS

Flip-Flop:

- The storage elements (memory) used in clocked sequential circuits are called flipflops. A flip-flop is a binary storage device capable of storing one bit of information.
- In a stable state, the output of a flip-flop is either 0 or 1.
- A sequential circuit may use many flip-flops to store as many bits as necessary. The block diagram of a synchronous clocked sequential circuit is shown in Fig.
- A storage element in a digital circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit), until directed by an input signal to switch states.
- The major differences among various types of storage elements are in the number of inputs they possess and in the manner in which the inputs affect the binary state.

Latch:

- The storage elements that operate with signal levels (rather than signal transitions) are referred to as latches; those controlled by a clock transition are flip-flops. Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices.

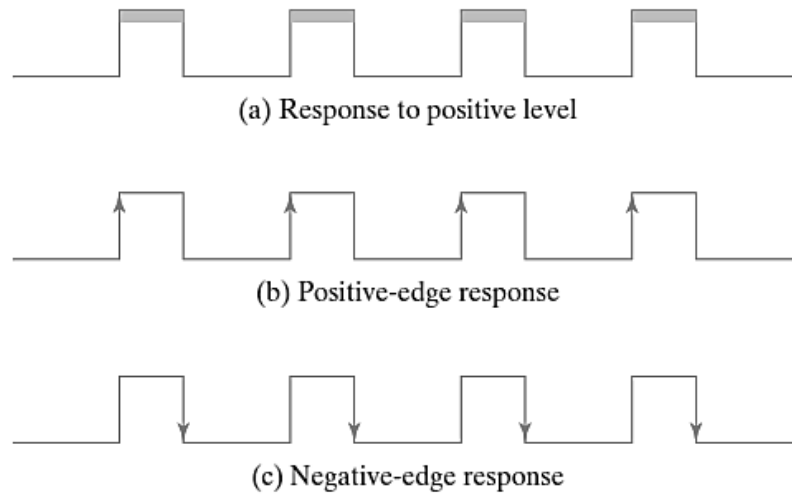


Synchronous clocked sequential circuit

Triggering of Flip Flops

Explain about triggering of flip flops in detail.

- The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a *trigger*, and the transition it causes is said to trigger the flip-flop.



Level Triggering:

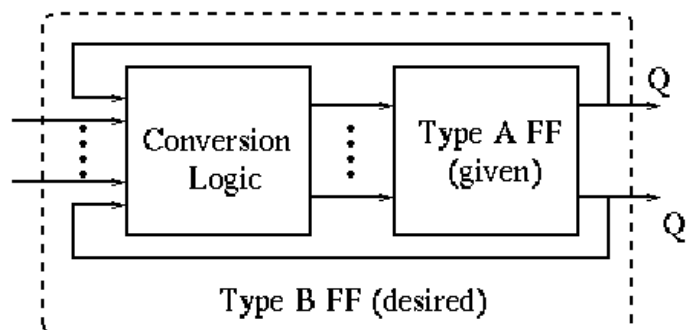
- SR, D, JK and T latches are having enable input.
- Latches are controlled by enable signal, and they are level triggered, either positive level triggered or negative level triggered as shown in figure (a).
- The output is free to change according to the input values, when active level is maintained at the enable input.

Edge Triggering:

- A clock pulse goes through two transitions: from 0 to 1 and the return from 1 to 0.
- As shown in above Fig (b) and (c), the positive transition is defined as the positive edge and the negative transition as the negative edge.

FLIP FLOP CONVERSIONS

- The purpose is to convert a given type A FF to a desired type B FF using some conversion logic.



- To use the excitation table, which shows the necessary triggering signal (S,R, J,K, D and T) for a

desired flip-flop state transition $Q_t \rightarrow Q_{t+1}$:

Excitation table for all flip flops:

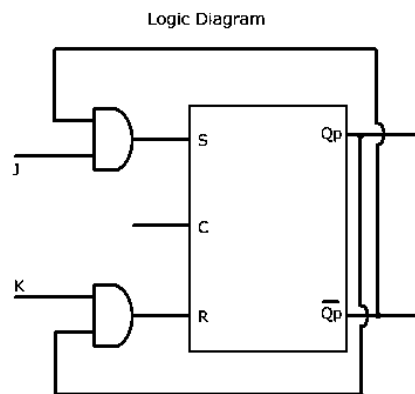
Q_t	Q_{t+1}	S	R	D	J	K	T
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	0	0	1	0	X	1	1
1	1	X	0	1	X	0	0

1. Design of JK Flip-flop using SR Flip-Flop.

S-R Flip Flop to J-K Flip Flop

Conversion Table

J-K Inputs		Outputs		S-R Inputs	
J	K	Q_p	Q_{p+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1



J \ KQ_p				
	00	01	11	10
0	0 ⁰	X ¹	0 ³	0 ²
1	1 ⁴	X ⁵	0 ⁷	1 ⁶

$$S = \bar{J}Q_p$$

J \ KQ_p				
	00	01	11	10
0	X ⁰	0 ¹	1 ³	X ²
1	0 ⁴	0 ⁵	1 ⁷	0 ⁶

$$R = KQ_p$$

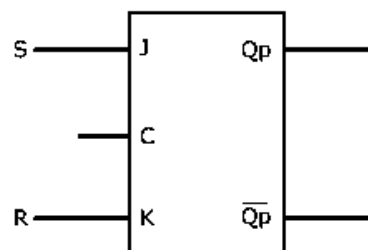
2. Design of SR Flip-flop using JK Flip-Flop.

J-K Flip Flop to S-R Flip Flop

Conversion Table

S-R Inputs		Outputs		J-K Inputs	
S	R	Q_p	Q_{p+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invalid		Dont care	
1	1	Invalid		Dont care	

Logic Diagram



S \ RQ_p				
	00	01	11	10
0	0 ⁰	X ¹	X ³	0 ²
1	1 ⁴	X ⁵	X ⁷	X ⁶

$$J=S$$

K-maps

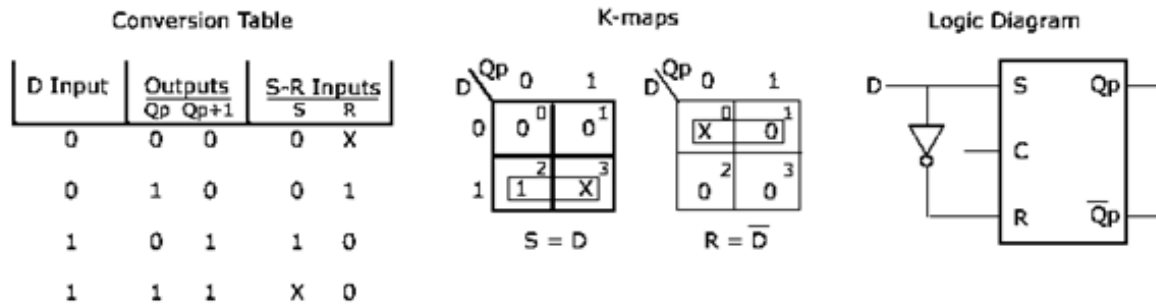
S \ RQ_p				
	00	01	11	10
0	X ⁰	0 ¹	1 ³	X ²
1	X ⁴	0 ⁵	X ⁷	X ⁶

$$K=R$$

3. Design of D Flip-flop using SR Flip-Flop.

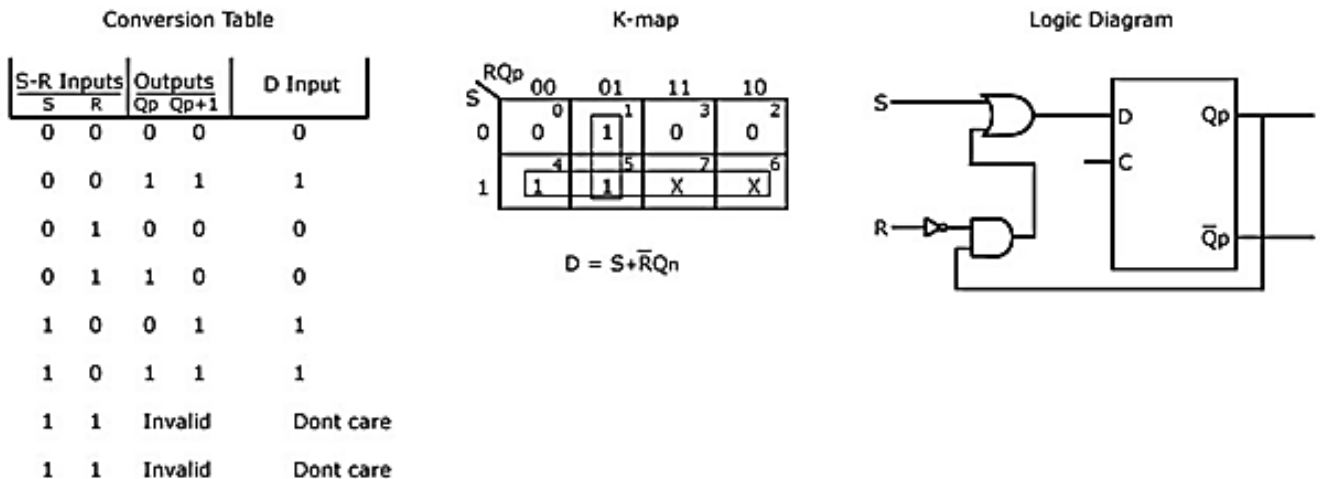
Show how S-R flip flop is converted into D-flip flop. [NOV 2020]

S-R Flip Flop to D Flip Flop



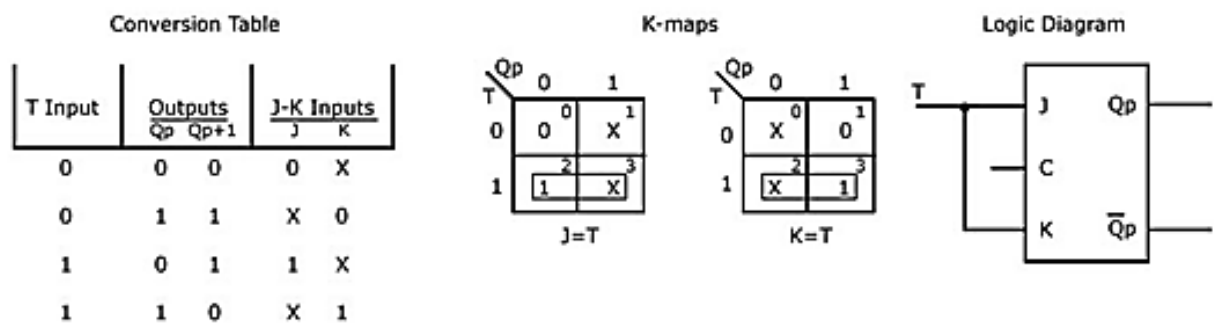
4. Design of SR Flip-flop using D Flip-Flop.

D Flip Flop to S-R Flip Flop



5. Design of T Flip-flop using JK Flip-Flop.

J-K Flip Flop to T Flip Flop

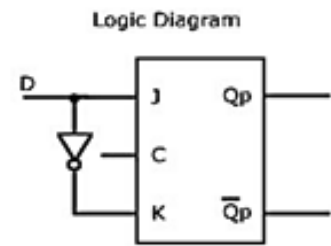
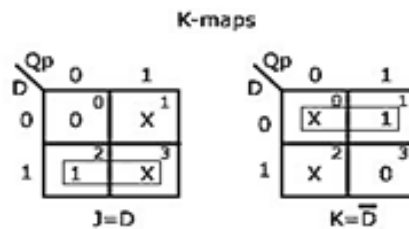


6. Design of D Flip-flop using JK Flip-Flop.

J-K Flip Flop to D Flip Flop

Conversion Table

D Input	Outputs Q_p Q_{p+1}		J-K Inputs J K	
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0



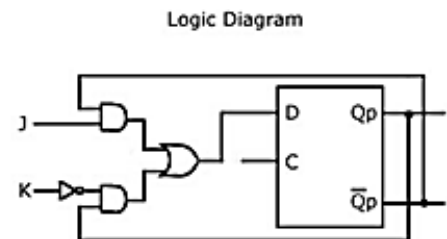
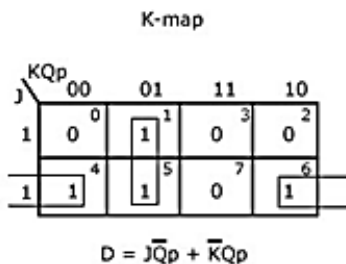
7. Design of JK Flip-flop using D Flip-Flop.

Dec 2009,11

D Flip Flop to J-K Flip Flop

Conversion Table

J-K Input J K	Outputs Q_p Q_{p+1}		D Input
0 0	0	0	0
0 0	1	1	1
0 1	0	0	0
0 1	1	0	0
1 0	0	1	1
1 0	1	1	1
1 1	0	1	1
1 1	1	0	0



MEALY AND MOORE MODELS

Write short notes on Mealy and Moore models in sequential circuits.

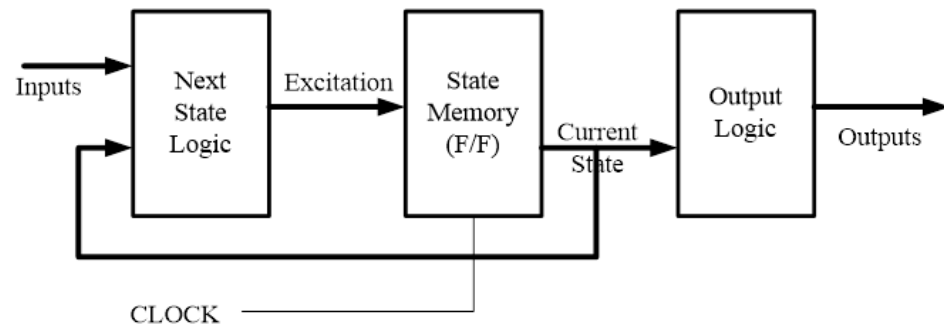
- In synchronous sequential circuit the outputs depend upon the order in which its input variables change and can be affected at discrete instances of time.

General Models:

- There are two models in sequential circuits. They are:
 1. Mealy model
 2. Moore model

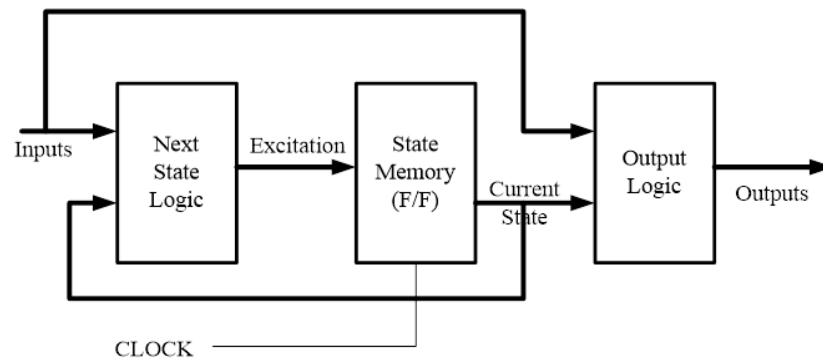
Moore machine:

- In the Moore model, the outputs are a function of present state only.



Mealy machine:

- In the Mealy model, the outputs are a function of present state and external inputs.



COUNTERS

Counter:

- A counter is a register (group of Flip-Flop) capable of counting the number of clock pulse arriving at its clock input.
- A counter that follows the binary number sequence is called a binary counter.
- Counter are classified into two types,
 1. Asynchronous (Ripple) counters.
 2. Synchronous counters.
- In ripple counter, a flip- flop output transition serves as clock to next flip-flop.
 - With an asynchronous circuit, all the bits in the count do not all change at the same time.
- In a synchronous counter, all flip-flops receive common clock.
 - With a synchronous circuit, all the bits in the count change synchronously with the assertion of the clock
- A counter may count up or count down or count up and down depending on the input control.

Uses of Counters:

The most typical uses of counters are

- ✓ To count the number of times that a certain event takes place; the occurrence of event to be counted is represented by the input signal to the counter
- ✓ To control a fixed sequence of actions in a digital system
- ✓ To generate timing signals
- ✓ To generate clocks of different frequencies

Modulo 16 ripple /Asynchronous Up Counter/Serial up counter

Explain the operation of a 4-bit binary ripple counter.

(May 2014, Dec 2015)

- The output of up-counter is incremented by one for each clock transition.
- A 4-bit asynchronous up-counter consists of 4JK Flip-Flops.
- The external clock signal is connected to the clock input of the first FlipFlop.
- The clock inputs of the remaining Flip-Flops are triggered by the Q output of the previous stage.
- We know that in JK Flip-Flop, if $J=1$, $K=1$ and clock is triggered the past output will be complemented.

➤ Initially, the register is cleared, $Q_D Q_C Q_B Q_A = 0000$.

➤ During the **first clock pulse**, Flip-Flop A triggers, therefore $Q_A=1$, $Q_B=Q_C=Q_D=0$.

$$Q_D Q_C Q_B Q_A = 0001$$

➤ At the **second clock pulse** FlipFlop A triggers, therefore Q_A changes from 1 to 0, which triggers FlipFlop B, therefore $Q_B=1, Q_A=Q_C=Q_D=0$

$$Q_D Q_C Q_B Q_A = 0010$$

➤ At the **third clock pulse** FlipFlop A triggers, therefore Q_A changes from 0 to 1.

- This never triggers FlipFlop B because 0 to 1 transition gives a positive edge triggering,
- but here the FlipFlops are triggered only at negative edge(1 to 0 transition) ,therefore $Q_A=Q_B=1$, $Q_C=Q_D=0$.

$$Q_D Q_C Q_B Q_A = 0011$$

➤ At the **fourth clock pulse** Flip-Flop A triggers, therefore Q_A changes from 1 to 0.

- This triggers FlipFlop B therefore Q_B changes from 1 to 0.
- The change in Q_B from 1 to 0 triggers C Flip-Flop,

➤ Therefore Q_C changes from 0 to 1. Therefore $Q_A=Q_B=Q_D=0$, $Q_C=1$.

$$Q_D Q_C Q_B Q_A = 0100$$

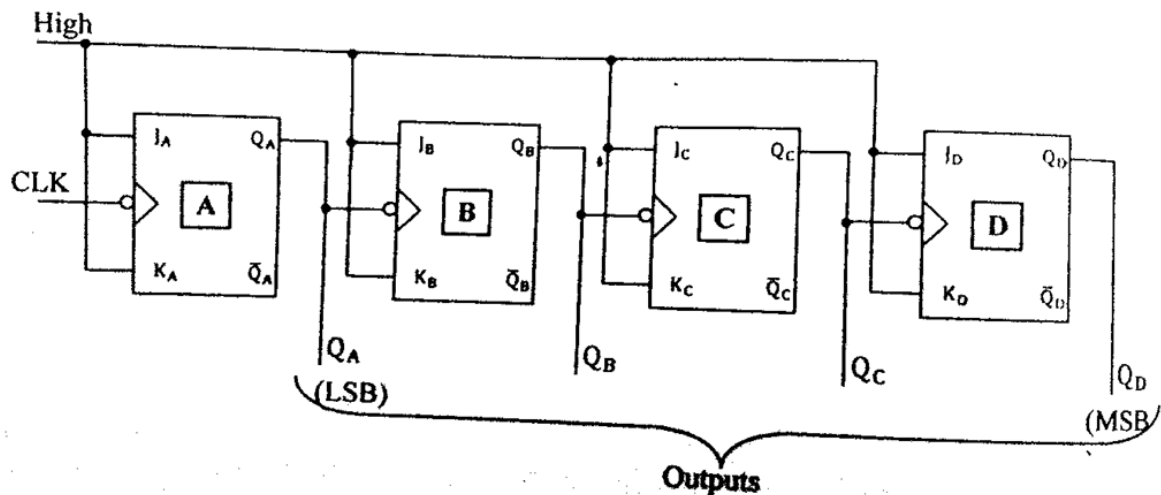


Figure 4-bit Asynchronous Up-counter

Truth table:

CLK	Outputs			
	Q _D	Q _C	Q _B	Q _A
-	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Truth table for 4-bit asynchronous up-counter

Timing diagram:

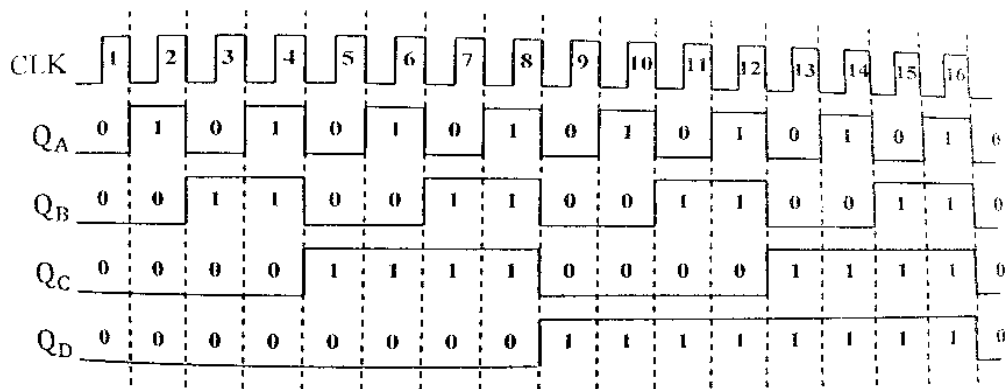


Figure 4.37 Timing diagram of 4-bit asynchronous up-counter.

Modulo 16 /4 bit Ripple Down counter/ Asynchronous Down counter/Serial down counter

Explain about Modulo 16 /4 bit Ripple Down counter.

- The output of down-counter is decremented by one for each clock transition.
- A 4-bit asynchronous down-counter consists of 4JK Flip-Flops.
- The external clock signal is connected to the clock input of the first Flip-Flop.
- The clock inputs of the remaining Flip-Flops are triggered by the \overline{Q} output of the previous stage.
- We know that in JK Flip-Flop, if $J=1$, $K=1$ and clock is triggered the past output will be complemented.

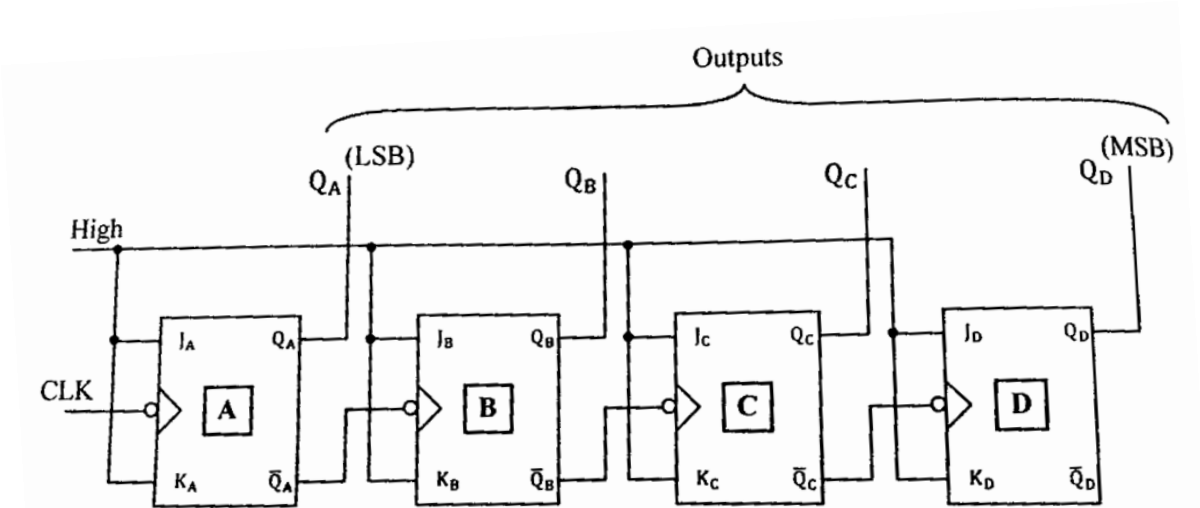


Figure Logic diagram of 4-bit asynchronous down-counter

- Initially, the register is cleared, $Q_D Q_C Q_B Q_A = 0000$.
- During the **first clock pulse**, Flip-Flop A triggers, therefore Q_A changes from 0 to 1 also \overline{Q}_A changes from 1 to 0. This triggers Flip-Flop B, therefore Q_B changes from 0 to 1, also \overline{Q}_B changes from 1 to 0 which triggers Flip-Flop C.
- Hence Q_C changes from 0 to 1 and \overline{Q}_C changes from 1 to 0, which further triggers, Flip-Flop D.

$$\begin{aligned} Q_D Q_C Q_B Q_A &= 1111 \\ \overline{Q_D} \overline{Q_C} \overline{Q_B} \overline{Q_A} &= 0000 \end{aligned}$$

- During the **second clock pulse** Flip-Flop A triggers, therefore Q_A changes from 1 to 0 also \overline{Q}_A changes from 0 to 1 which never triggers B Flip-Flop.
- Therefore C and D Flip-Flop are not triggered.

$$Q_D Q_C Q_B Q_A = 1110$$

- The same procedure repeats until the counter decrements upto 0000.

CLK	Outputs			
	Q _D	Q _C	Q _B	Q _A
-	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0

Table 4.1 Truth table for 4-bit asynchronous down-counter

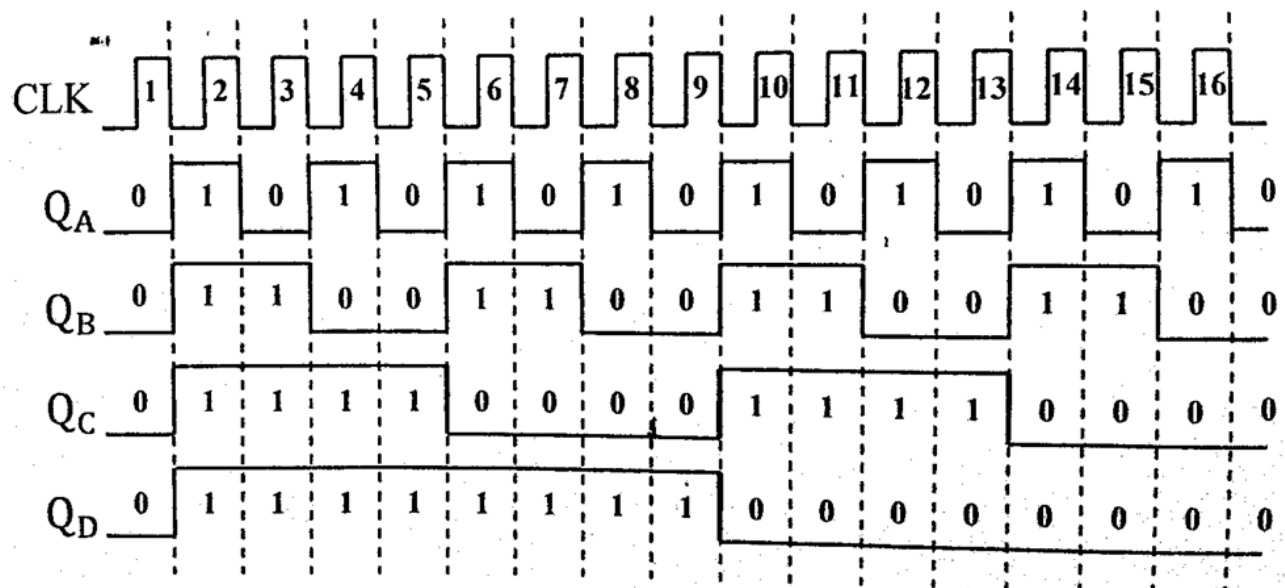


Figure 4.1 Timing diagram of 4-bit asynchronous down-counter.

Asynchronous Up/Down Counter:

Explain about Asynchronous Up/Down counter.

- The up-down counter has the capability of counting upwards as well as downwards. It is also called multimode counter.
- In asynchronous up-counter, each flip-flop is triggered by the normal output Q of the preceding flip-flop.
- In asynchronous down counter, each flip-flop is triggered by the complement output \bar{Q} of the preceding flip-flop.
- In both the counters, the first flip-flop is triggered by the clock output.

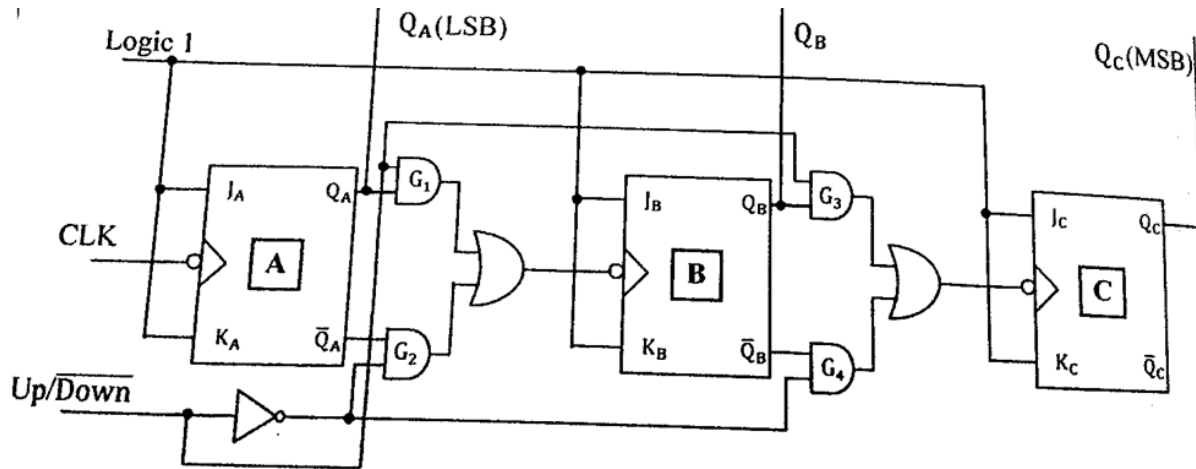


Figure 3-bit asynchronous up/down-counter

- If $\overline{Up/Down} = 1$, the 3-bit asynchronous up/down counter will perform up-counting. It will count from 000 to 111.
 - If $\overline{Up/Down} = 1$ gates G_2 and G_4 are disabled and gates G_1 and G_3 are enabled. So that the circuit behaves as an up-counter circuit.
- If $\overline{Up/Down} = 0$, the 3-bit asynchronous up/down counter will perform down-counting. It will count from 111 to 000.
 - If $\overline{Up/Down} = 0$ gates G_2 and G_4 are enabled and gates G_1 and G_3 are disabled. So that the circuit behaves as an down-counter circuit.

$\overline{Up/Down}$	Q_C	Q_B	Q_A
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	0
0	0	1	1
0	0	0	1
0	0	0	0

Table : Truth table for 3-Bit asynchronous Up/Down-counter

4- bit Synchronous up-counter:

Explain about 4-bit Synchronous up-counter.

Explain the operation of synchronous up-counter.

2018

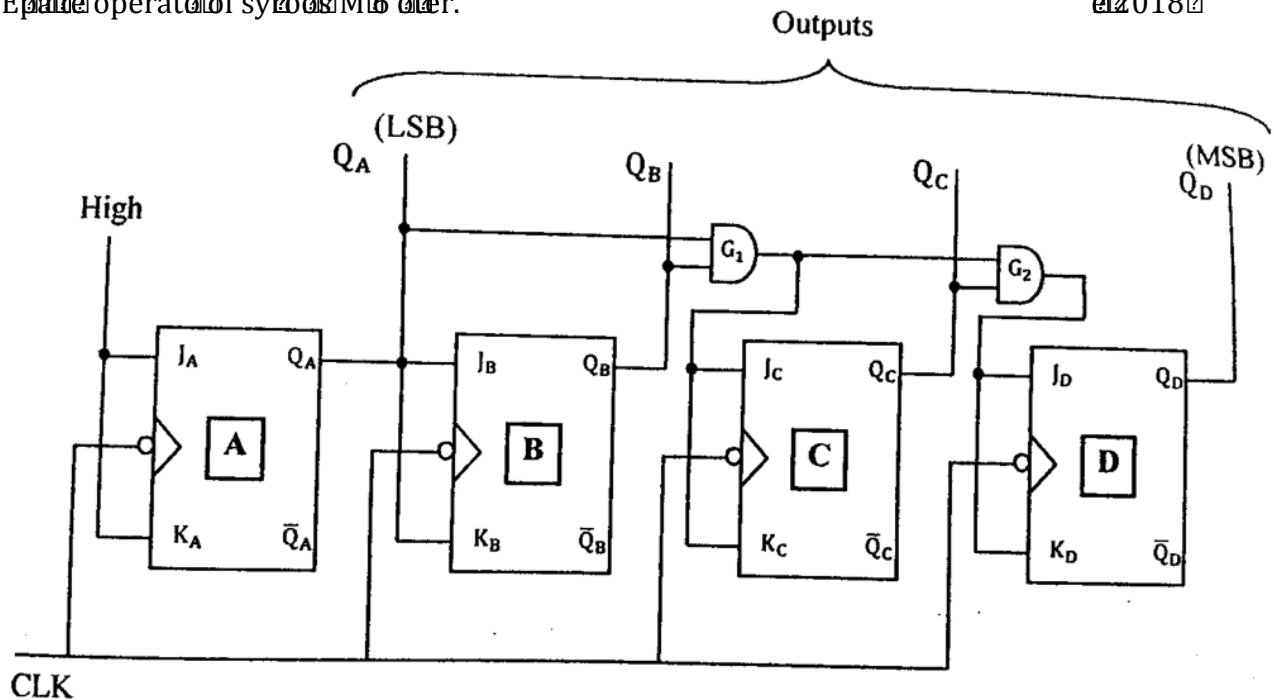


Figure Logic diagram of 4-bit Synchronous up-counter

- In JK Flip-Flop, If $J=0$, $K=0$ and clock is triggered, the output never changes. If $J=1$ and $K=1$ and the clock is triggered, the past output will be complemented.
- Initially the register is cleared $Q_D Q_C Q_B Q_A = 0000$.
- During the first clock pulse, $J_A = K_A = 1$, Q_A becomes 1, Q_B, Q_C, Q_D remains 0.
 $Q_D Q_C Q_B Q_A = 0001$.
- During second clock pulse, $J_A = K_A = 1$, $Q_A = 0$.
 $J_B = K_B = 1$, $Q_B = 1$, Q_C, Q_D remains 0.
 $Q_D Q_C Q_B Q_A = 0010$.
- During third clock pulse, $J_A = K_A = 1$, $Q_A = 1$.
 $J_B = K_B = 0$, $Q_B = 1$, Q_C, Q_D remains 0.
 $Q_D Q_C Q_B Q_A = 0011$.
- During fourth clock pulse, $J_A = K_A = 1$, $Q_A = 0$.
 $J_B = K_B = 1$, $Q_B = 0$
 $J_C = K_C = 1$, $Q_C = 1$, Q_D remains 0
 $Q_D Q_C Q_B Q_A = 0100$.
- The same procedure repeats until the counter counts up to 1111.

CLK	Outputs			
	Q _D	Q _C	Q _B	Q _A
-	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Table Truth table for 4-bit synchronous up-counter

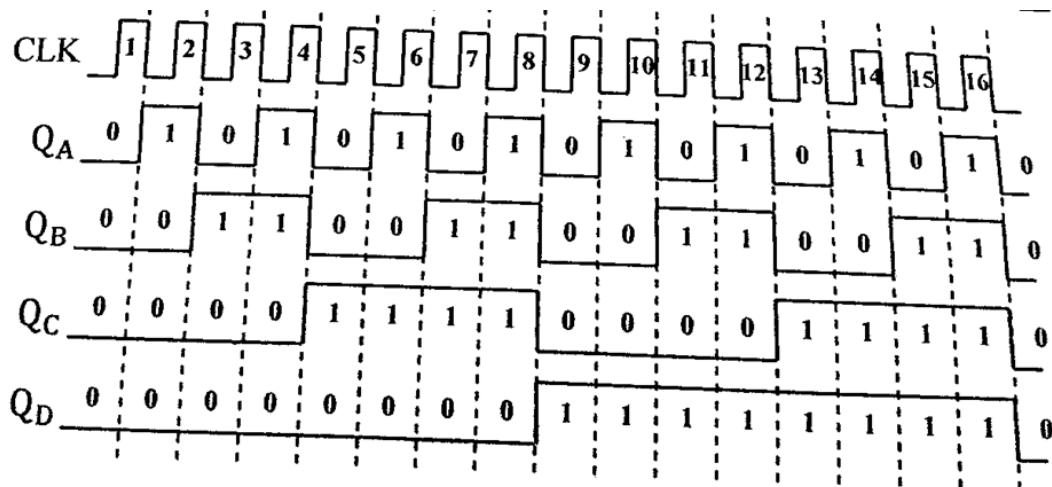


Figure Timing diagram of 4-bit synchronous up-counter

4- bit Synchronous down-counter:

Explain about 4-Bit Synchronous down counter.

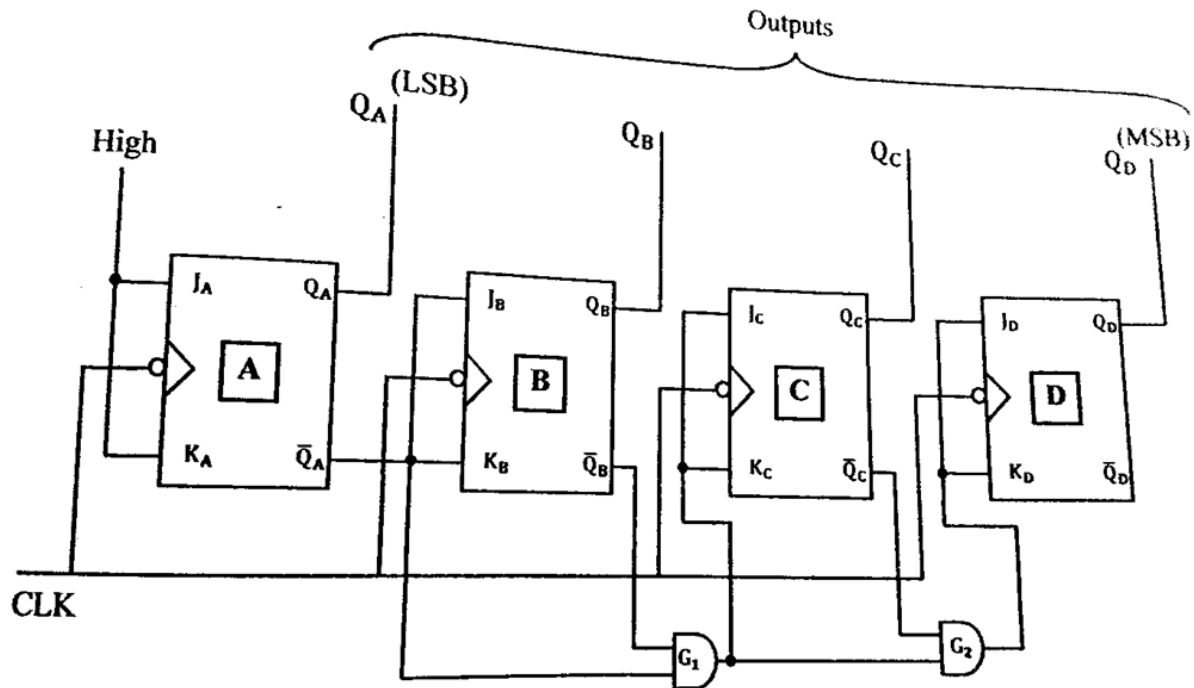


Figure 3 Logic diagram of 4-bit synchronous down-counter

In JK Flip-Flop, If $J=0$, $K=0$ and clock is triggered, the output never changes. If $J=1$ and $K=1$ and the clock is triggered, the past output will be complemented.

Initially the register is cleared $Q_D Q_C Q_B Q_A = 0000$

$$\overline{Q_D} \overline{Q_C} \overline{Q_B} \overline{Q_A} = 1111$$

During the *first clock pulse*, $J_A = K_A = 1$, $Q_A = 1$

$$J_B = K_B = 1, Q_B = 1$$

$$J_C = K_C = 1, Q_C = 1$$

$$J_D = K_D = 1, Q_D = 1$$

$$Q_D Q_C Q_B Q_A = 1111$$

$$\overline{Q_D} \overline{Q_C} \overline{Q_B} \overline{Q_A} = 0000$$

During the *second clock pulse*, $J_A = K_A = 1$, $Q_A = 0$

$$J_B = K_B = 0, Q_B = 1$$

$$J_C = K_C = 0, Q_C = 1$$

$$J_D = K_D = 0, Q_D = 1$$

$$Q_D Q_C Q_B Q_A = 1110$$

$$\overline{Q_D} \overline{Q_C} \overline{Q_B} \overline{Q_A} = 0001$$

During the *third clock pulse*, $J_A = K_A = 1$, $Q_A = 1$

$J_B = K_B = 1$, $Q_B = 0$

$J_C = K_C = 0$, $Q_C = 1$

$J_D = K_D = 0$, $Q_D = 1$

$Q_D Q_C Q_B Q_A = 1101$

The process repeats until the counter down-counts up to 0000.

CLK	Outputs			
	Q_D	Q_C	Q_B	Q_A
-	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0

Table Truth table of 4-bit synchronous down-counter

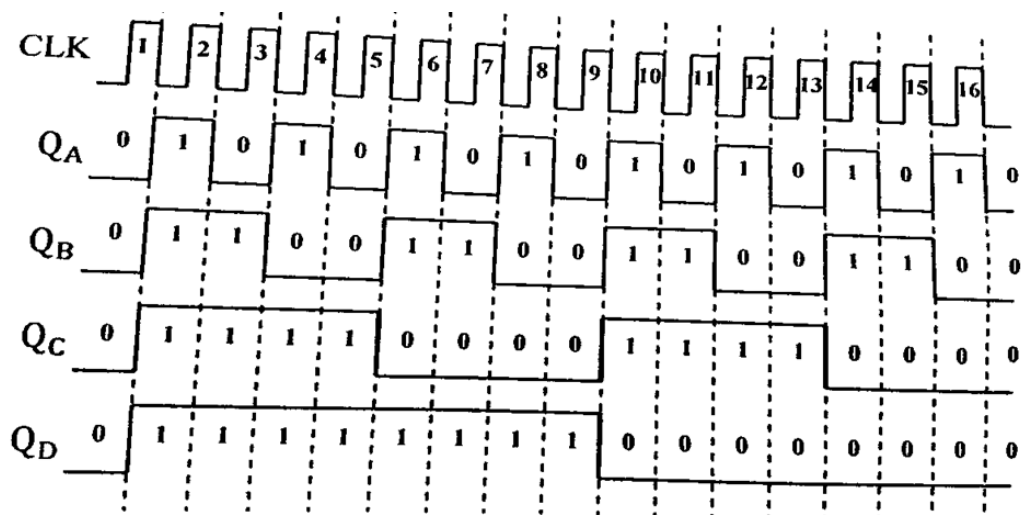


Figure 4.11 Timing diagram of 4-bit synchronous down-counter

Modulo 8 Synchronous Up/Down Counter:

Explain about Modulo 8 Synchronous Up/Down Counter.

Explain the operation of synchronous three bit counter. [NOV 2020]

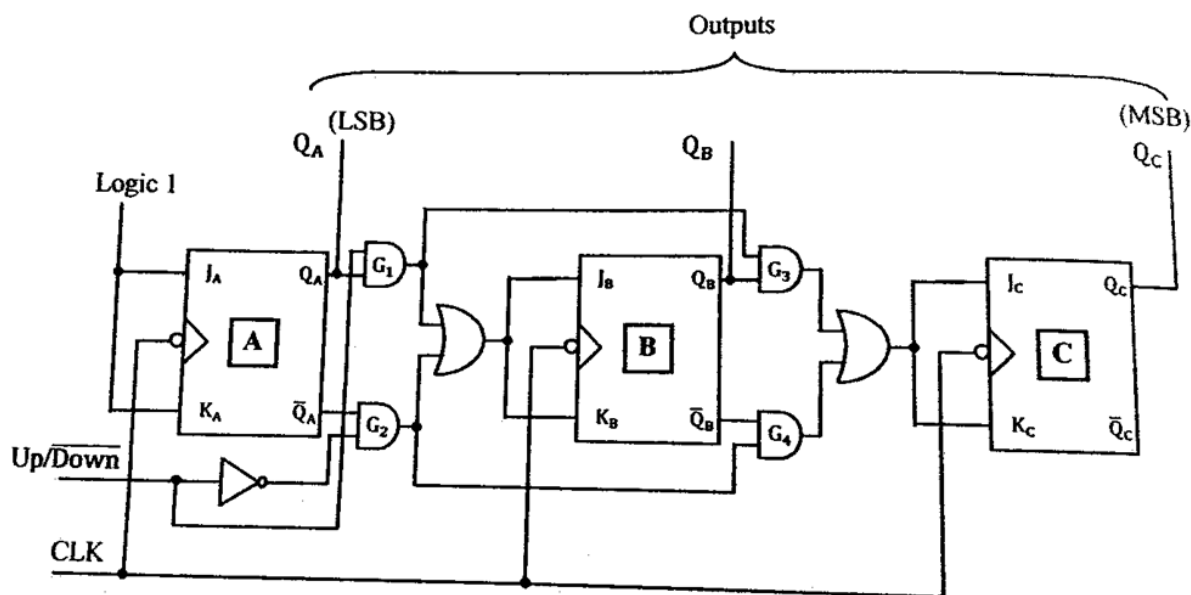


Figure : 3-bit synchronous up/down-counter

In synchronous up-counter the Q_A output is given to J_B , K_B and Q_A . Q_B is given to J_C , K_C . But in synchronous down-counter $\overline{Q_A}$ output is given to J_B , K_B and $\overline{Q_A}$. $\overline{Q_B}$ is given to J_C , K_C .

A control input $\overline{Up/Down}$ is used to select the mode of operation.

If $\overline{Up/Down} = 1$, the 3-bit asynchronous up/down counter will perform up-counting. It will count from 000 to 111. If $\overline{Up/Down} = 1$ gates G_2 and G_4 are disabled and gates G_1 and G_3 are enabled. So that the circuit behaves as an up-counter circuit.

If $\overline{Up/Down} = 0$, the 3-bit asynchronous up/down counter will perform down-counting. It will count from 111 to 000. If $\overline{Up/Down} = 0$ gates G_2 and G_4 are enabled and gates G_1 and G_3 are disabled. So that the circuit behaves as a down-counter circuit.

Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Up/Down = 1 (downward arrow) Up/Down = 0 (upward arrow)

Table : Truth table for 3-Bit asynchronous Up/Down-counter

DESIGN OF SYNCHRONOUS COUNTERS

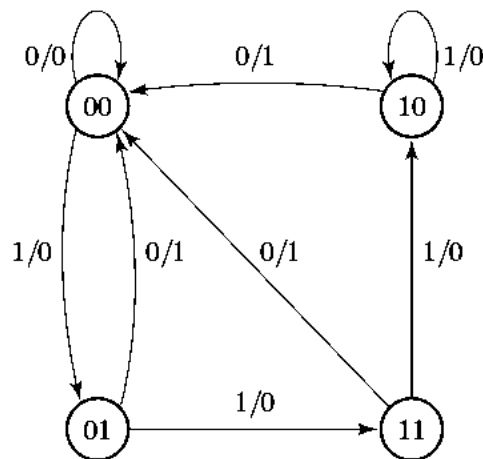
Design and analyze of clocked sequential circuit with an example.

The procedure for designing synchronous sequential circuit is given below,

1. From the given specification, Draw the state diagram.
2. Plot the state table.
3. Reduce the number of states if possible.
4. Assign binary values to the states and plot the transition table by choosing the type of Flip-Flop.
5. Derive the Flip flop input equations and output equations by using K-map.
6. Draw the logic diagram.

State Diagram:

- State diagram is the *graphical representation of the information available in a state table.*
- In state diagram, a state is represented by a circle and the transitions between states are indicated by directed lines connecting the circles.
- The state diagram for the logic circuit in below figure.



State Table:

- A state table gives the time sequence of inputs, outputs and flip flops states. The table consists of four sections labeled present state, next state, input and output.
- The present state section shows the states of flip flops A and B at any given time 'n'. The input section gives a value of x for each possible present state.
- The next state section shows the states of flip flops one clock cycle later, at time n+1.
- The state table for the circuit is shown. This is derived using state equations.

Present State		Input	Next State		Output
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

The above state table can also be expressed in different forms as follows.

Present State		Next State				Output	
		x = 0		x = 1		x = 0	x = 1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

State Minimization or state reduction:

- State reduction is concerned with the procedures for reducing the number of states in the state table.
- When two states in the state table are equivalent by producing the same next state and same output then one of the states in the state table can be removed without altering the input output relationship.

State Assignment:

- In order to design a sequential circuit, it is necessary to assign binary values to the state.
- For a circuit with 'm' states, the codes must contain 'n' bits, where $2^n \geq m$.
- For an example, with three bits we can assign codes to maximum of 8 states.

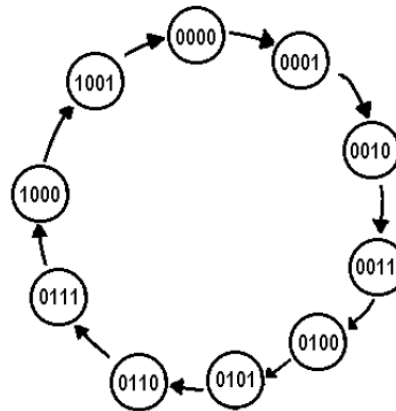
Flip-Flop Input Equations:

The part of the circuit that generates the inputs to flip flops is described algebraically by a set of Boolean functions called flip flop input equations.

A synchronous decade counter will count from zero to nine and repeat the sequence.

State diagram:

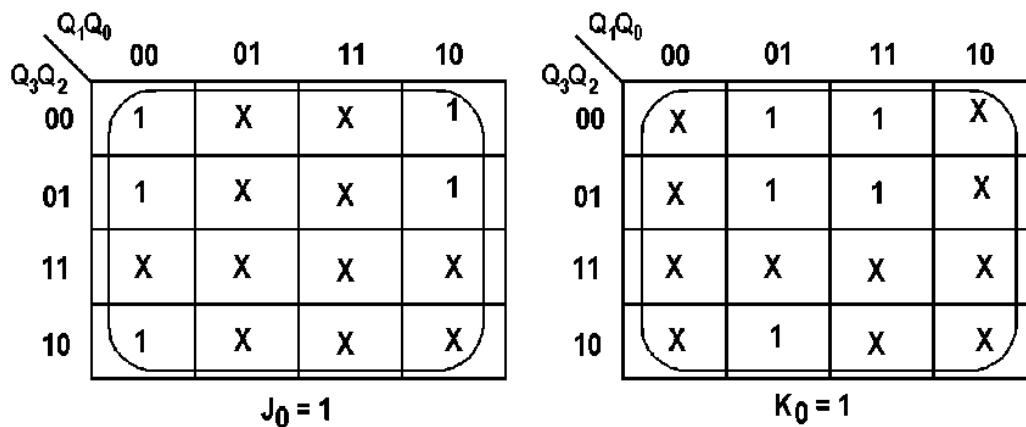
The state diagram of this counter is shown in Fig.



Excitation table:

Present State				Next State				Output							
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1

K-Map:



Q_3Q_2	Q_1Q_0 00	01	11	10
00		1	X	X
01		1	X	X
11	X	X	X	X
10			X	X

$$J_1 = \bar{Q}_3 Q_0$$

Q_3Q_2	Q_1Q_0 00	01	11	10
00	X	X	1	
01	X	X	1	
11	X	X	X	X
10	X	X	X	X

$$K_1 = \bar{Q}_3 Q_0$$

Q_3Q_2	Q_1Q_0 00	01	11	10
00			1	
01	X	X	X	X
11	X	X	X	X
10			X	X

$$J_2 = Q_1 Q_0$$

Q_3Q_2	Q_1Q_0 00	01	11	10
00	X	X	X	X
01			1	
11	X	X	X	X
10			X	X

$$K_2 = Q_1 Q_0$$

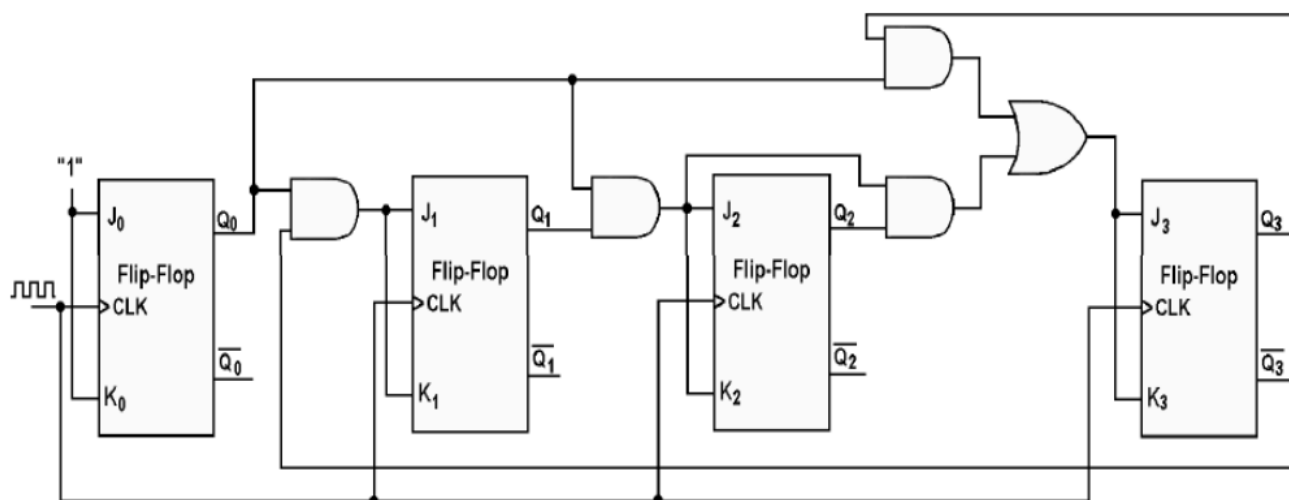
Q_3Q_2	Q_1Q_0 00	01	11	10
00				
01			1	
11	X	X	X	X
10	X	X	X	X

$$J_3 = Q_3 Q_0 + Q_2 Q_1 Q_0$$

Q_3Q_2	Q_1Q_0 00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10		1	X	X

$$K_3 = Q_3 Q_0 + Q_2 Q_1 Q_0$$

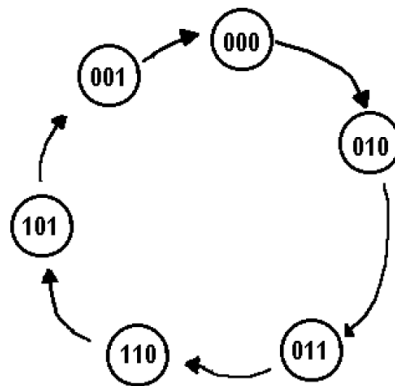
Logic Diagram:



The modulus six counters will count 0, 2, 3, 6, 5, and 1 and repeat the sequence. This modulus six counter requires three SR flip-flops for the design.

May 2019

State diagram:



Truth table:

[Present State			Next State			Output					
Q ₂	Q ₁	Q ₀	Q ₂	Q ₁	Q ₀	R ₂	S ₂	R ₁	S ₁	R ₀	S ₀
0	0	0	0	1	0	0	X	1	0	0	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	1	1	0	1	0	X	0	0	1
1	1	0	1	0	1	X	0	0	1	1	0
1	0	1	0	0	1	0	1	0	X	X	0
0	0	1	0	0	0	0	X	0	X	0	1

K-Map:

Q ₂	Q ₁ Q ₀			
	00	01	11	10
0	0	0	0	1
1	X	X	X	1

$$R_0 = Q_1 \cdot \overline{Q_0}$$

Q ₂	Q ₁ Q ₀			
	00	01	11	10
0	X	1	1	0
1	X	0	X	0

$$S_0 = \overline{Q_2} \cdot Q_0$$

Q ₂	Q ₁ Q ₀			
	00	01	11	10
0	1	0	X	X
1	X	0	X	0

$$R_1 = \overline{Q_1} \cdot \overline{Q_0}$$

Q ₂	Q ₁ Q ₀			
	00	01	11	10
0	0	X	0	0
1	X	X	X	1

$$S_1 = Q_2$$

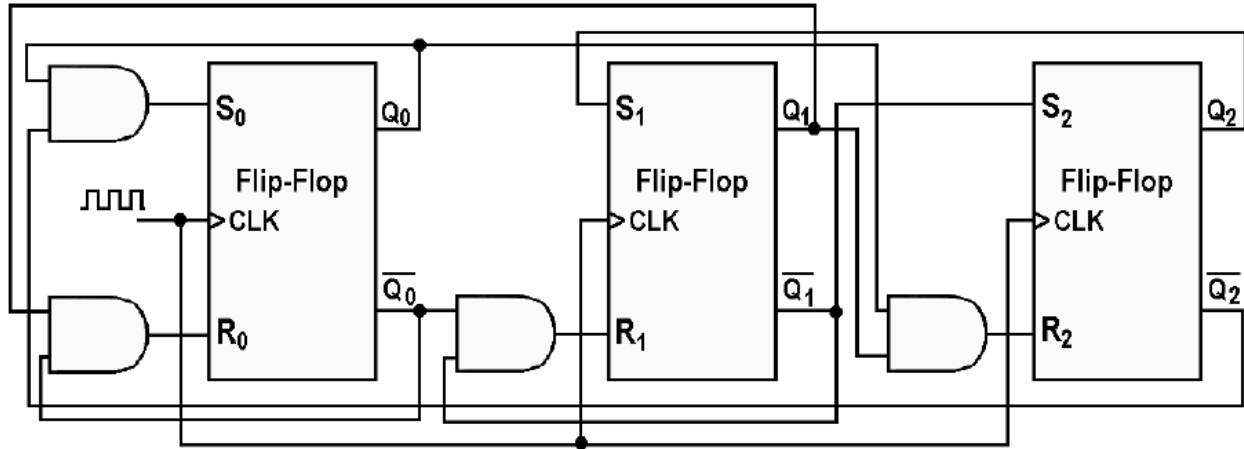
$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	0	0	1	0
1	X	0	X	X

$$R_2 = Q_1 \cdot Q_0$$

$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	X	X	0	X
1	X	1	X	0

$$S_2 = \overline{Q_1}$$

Logic Diagram:



SHIFT REGISTERS

Explain various types of shift registers.

- A register capable of shifting the binary information held in each cell to its neighboring cell in a selected direction is called a shift register.
- There are four types of shift registers namely:
 1. Serial In Serial Out Shift Register,
 2. Serial In Parallel Out Shift Register
 3. Parallel In Serial Out Shift Register
 4. Parallel In Parallel Out Shift Register

With neat logic diagram and timing diagram, explain the working of PISO and SIPO shift registers. [NOV/DEC 2021]

1. Serial In Serial Out Shift Register

- The block diagram of a serial out shift register is as below.

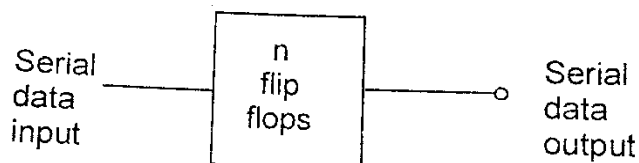
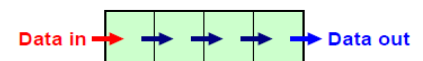


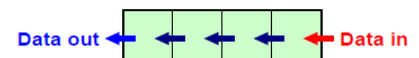
Fig. Block Diagram of SISO

1. Serial-in Serial-out

1(a) Left to Right



1(b) Right to Left



- It accepts data serially i.e., one bit at a time on a single input line. It produces the stored information on its single output also in serial form.
- Data may be shifted left using shift left register or shifted right using shift right register.

Shift Right Register

The circuit diagram using D flip-flops is shown in figure

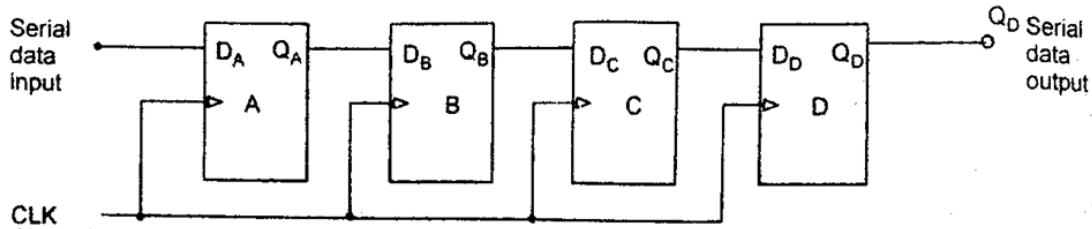


Fig. Serial in serial out right shift register

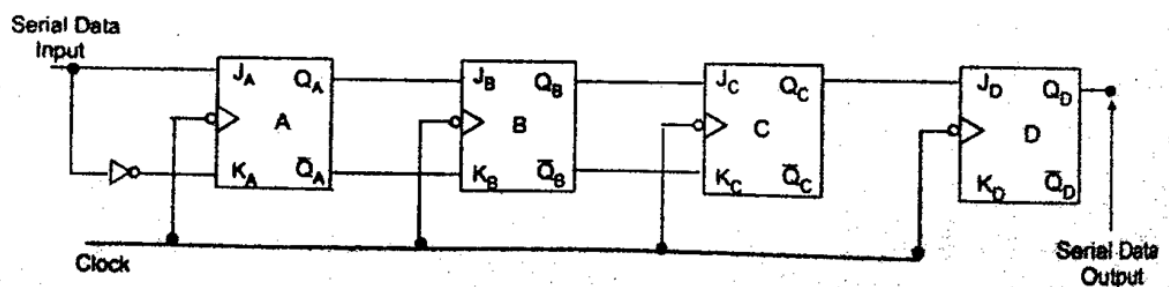


Fig. : SISO Shift Register using JK Flip-flop

- As shown in above figure, the clock pulse is applied to all the flip-flops simultaneously.
- The output of each flip-flop is connected to D input of the flip-flop at its right.
- Each clock pulse shifts the contents of the register one bit position to the right.
- New data is entered into stage A whereas the data presented in stage D are shifted out.
- For example, consider that all stages are reset and a steady logical 1 is applied to the serial input line.
- When the *first clock pulse* is applied, flip-flop A is set and all other flip-flops are reset.
- When the *second clock pulse* is applied, the '1' on the data input is shifted into flip-flop A and '1' that was in flip flop A is shifted to flip-flop B.
- This continues till all flip-flop sets.
- The data in each stage after each clock pulse is shown in table below

Shift Pulse	Serial Data Input	Q_A	Q_B	Q_C	Serial Output Q_D
0	1	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	1	0
4	1	1	1	1	1

Shift Left Register

The figure below shows the shift left register using D flip-flops.

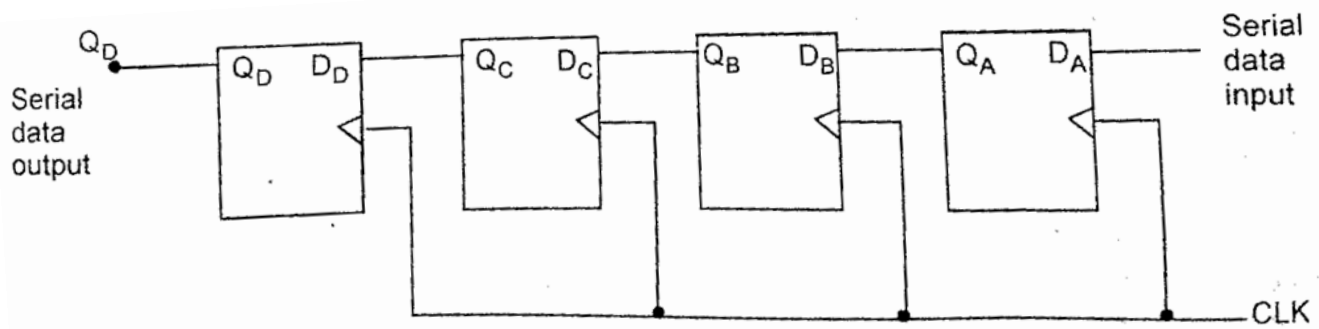


Fig. : Serial in serial out shift left register

- The clock is applied to all the flip-flops simultaneously. The output of each flip-flop is connected to D input of the flip-flop at its left.
- Each clock pulse shifts the contents of the register one bit position to the left.
- Let us illustrate the entry of the 4-bit binary number 1111 into the register beginning with the right most bit.
- When the *first clock pulse* is applied, flip flop A is set and all other flip-flops are reset.
- When *second clock pulse* is applied, '1' on the data input is shifted into flip-flop A and '1' that was in flip flop A is shifted to flip-flop B. This continues till all flip-flop are set.
- The data in each stage after each clock pulse is shown in table below.

Q _D	Q _C	Q _B	Q _A	Serial Input Data	Clock Pulse
0	0	0	0	1	0
0	0	0	1	1	1
0	0	1	1	1	2
0	1	1	1	1	3
1	1	1	1	1	4

2. Serial in Parallel out shift register:

A 4 bit serial in parallel out shift register is shown in figure.

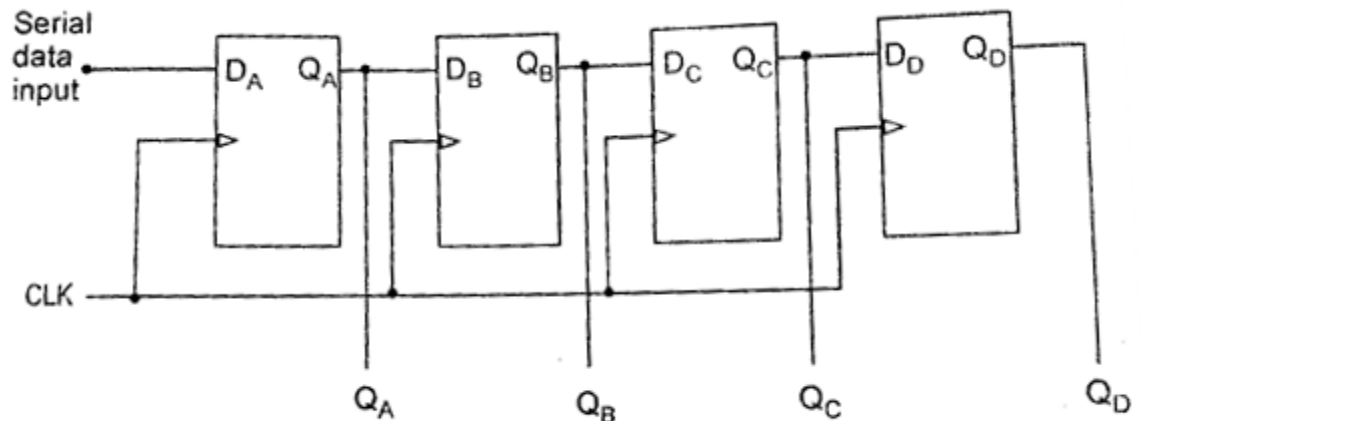


Fig. 3.42: Serial in parallel out shift register

- It consists of one serial input and outputs are taken from all the flip-flops simultaneously.
- The output of each flip-flop is connected to D input of the flip-flop at its right. Each clock pulse shifts the contents of the register one bit position to the right.
- For example, consider that all stages are reset and a steady logical '1' is applied to the serial input line.
- When the *first clock pulse* is applied flip flop A is set and all other flip-flops are reset.
- When the *second pulse* is applied the '1' on the data input is shifted into flip flop A and '1' that was in flip flop A is shifted into flip-flop B.
- This continues till all flip-flops are set. The data in each stage after each clock pulse is shown in table below.

Shift Pulse	Serial Data Input	Parallel Outputs			
		Q _A	Q _B	Q _C	Q _D
0	1	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	1	0
4	1	1	1	1	1

3. Parallel In Serial Out Shift register:

- For register with parallel data inputs, register the bits are entered simultaneously into their respective stages on parallel lines.
- A four bit parallel in serial out shift register is shown in figure. Let A,B,C and D be the four parallel data input lines and $\overline{\text{SHIFT/LOAD}}$ is a control input that allows the four bits of data to be entered in parallel or shift the serially.

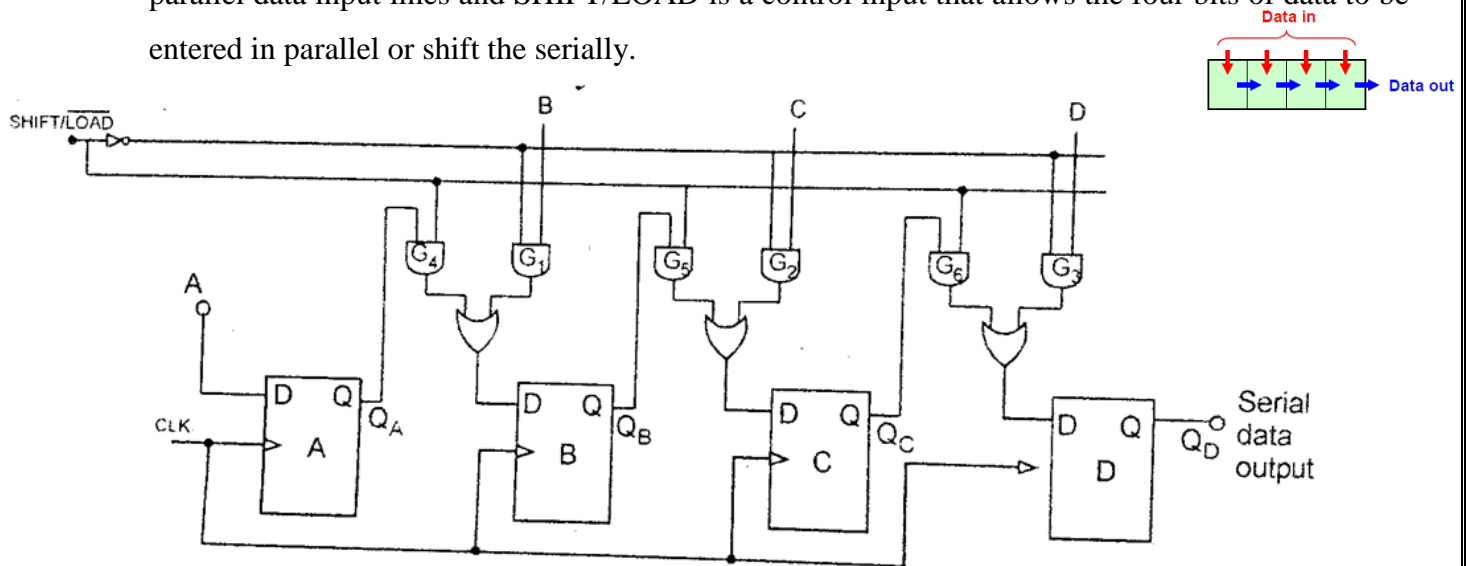


Fig. : Parallel in serial and shift register

- When $\overline{\text{SHIFT/LOAD}}$ is low, gates G1 through G3 are enabled, allowing the data at parallel inputs to the D input of its respective flip-flop.
 - When the clock pulse is applied the flip-flops with D=1 will set and those with D=0 will reset, thereby storing all four bits simultaneously.
- When $\overline{\text{SHIFT/LOAD}}$ is high. AND gates G1 through G3 are disabled and gates G4 through G6 are enabled, allowing the data bits to shift right from one stage to next.
 - The OR gates allow either the normal shifting operation or the parallel data entry operation, depending on which AND gates are enabled by the level on the $\overline{\text{SHIFT/LOAD}}$ input.

Parallel In Parallel Out Shift Register:

- In parallel in parallel out shift register, data inputs can be shifted either in or out of the register in parallel.
- A four bit parallel in parallel out shift register is shown in figure. Let A, B, C, D be the four parallel data input lines and Q_A , Q_B , Q_C and Q_D be four parallel data output lines. The $\overline{\text{SHIFT/LOAD}}$ is the control input that allows the four bits data to enter in parallel or shift the serially.

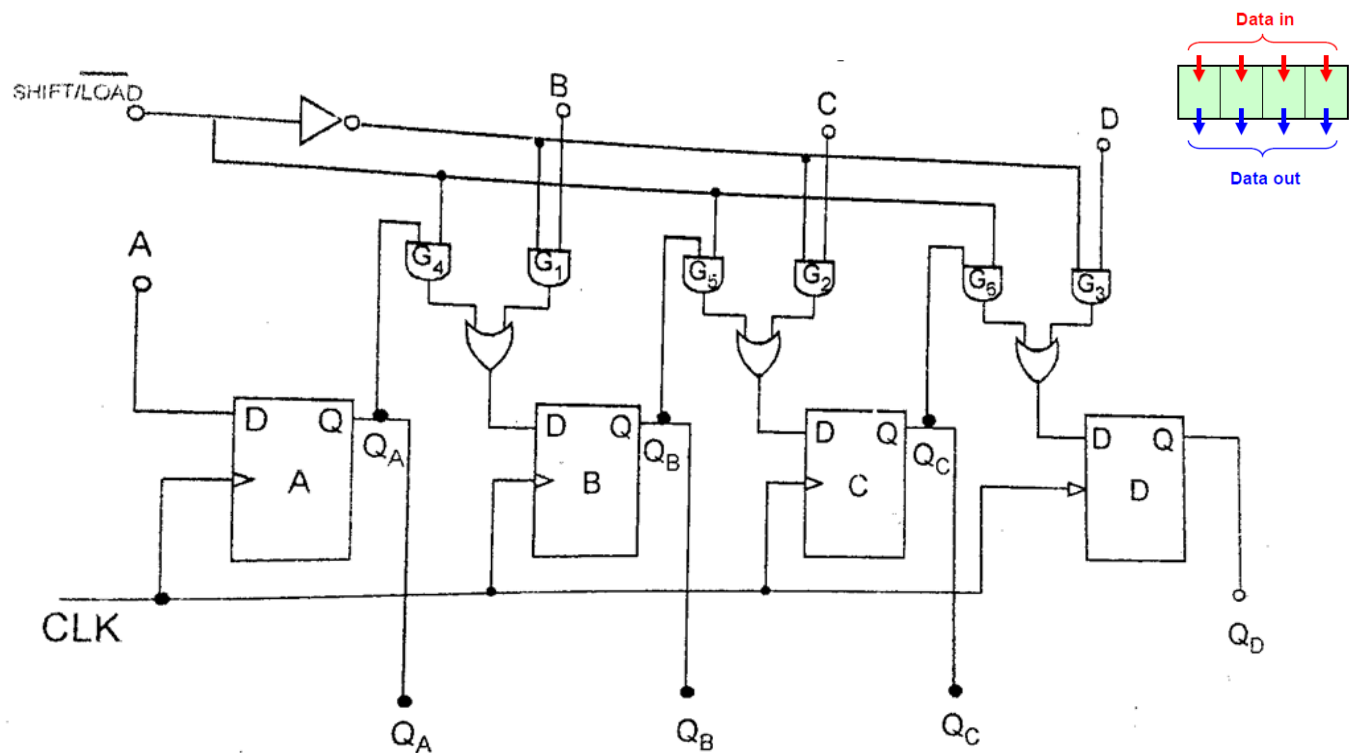
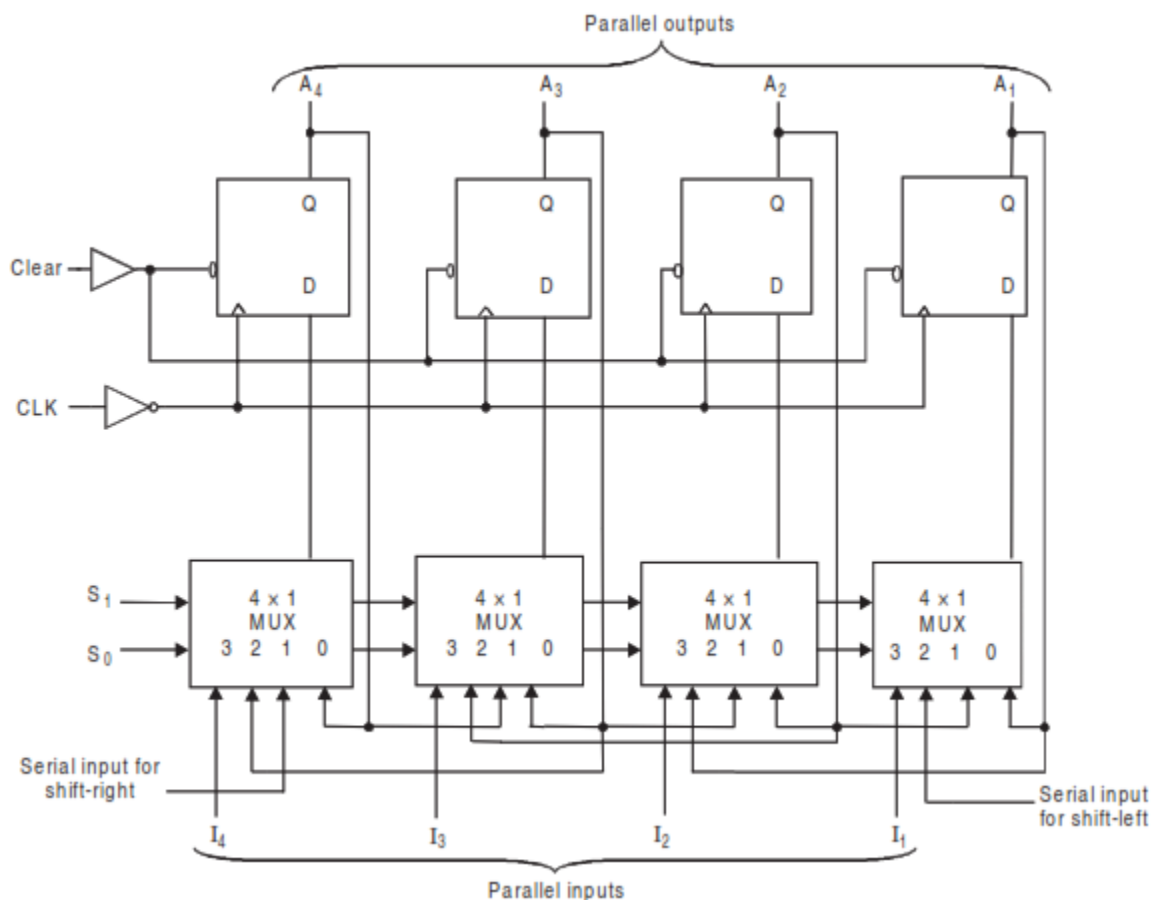


Fig. : Parallel in parallel out shift register

- When $\overline{SHIFT/LOAD}$ is low, gates G1 through G3 are enabled, allowing the data at parallel inputs to the D input of its respective flip-flop.
 - When the clock pulse is applied, the flip-flops with D = 1 will set those with D = 0 will reset thereby storing all four bits simultaneously.
 - These are immediately available at the outputs Q_A , Q_B , Q_C and Q_D .
- When $\overline{SHIFT/LOAD}$ is high, gates G1, through G3 are disabled and gates G4 through G6 are enabled allowing the data bits to shift right from one stage to another.
 - The OR gates allow either the normal shifting operation or the parallel data entry operation, depending on which AND gates are enabled by the level on the $\overline{SHIFT/LOAD}$ input.

Explain about universal shift register.

- A register that can shift data to right and left and also has parallel load capabilities is called universal shift register.
- It has the following capabilities.
 1. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift-right.
 2. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.
 3. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
 4. n parallel output lines.
 5. A clear control to clear the register to 0.
 6. A CLK input for clock pulses to synchronize all operations.
 7. A control state that leaves the information in the register unchanged even though clock pulses are continuously applied.



Mode of operation:

- It consists of four D flip-flop and four multiplexers.
- When $S_1 S_0 = 00$, the present value of the register is applied to the D inputs of the flip-flops.
 - Hence this condition forms a path from the output of each flip-flop into the input of the same flip-flop.
 - The next clock pulse transition transfers into each flip-flop the binary value held previously, and no change of state occurs.
- When $S_1 S_0 = 01$, terminals 1 of each of the multiplexer inputs have a path to the D inputs of each of the flip-flops.
 - This causes a shift-right operation, with the serial input transferred into flip-flop A_4 .
- Similarly, with $S_1 S_0 = 10$, a shift-left operation results, with the other serial input going into flip-flop A_1 .
- Finally, when $S_1 S_0 = 11$, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock pulse.

Mode Control		
s_1	s_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

SHIFT REGISTER COUNTERS:

Explain about Johnson and Ring counter.

[May 2018] (Dec 2019)

Most common shift register counters are Johnson counter and ring counter.

Johnson counter:

- A 4 bit Johnson counter using D flip-flop is shown in figure. It is also called shift counter or twisted counter.

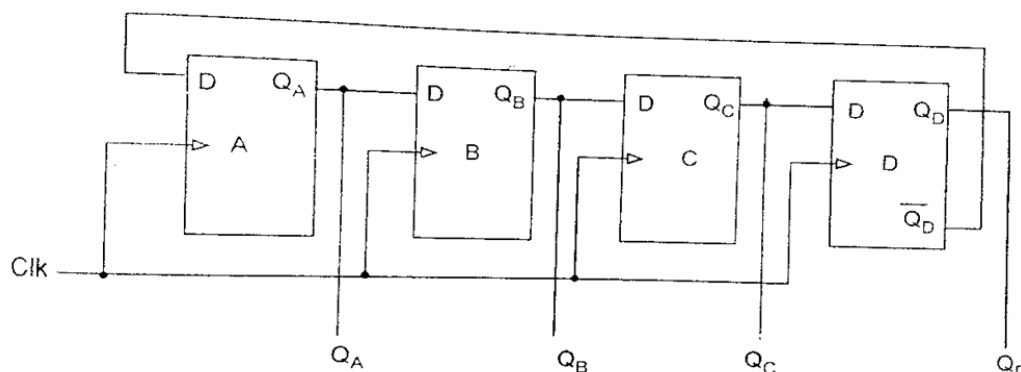
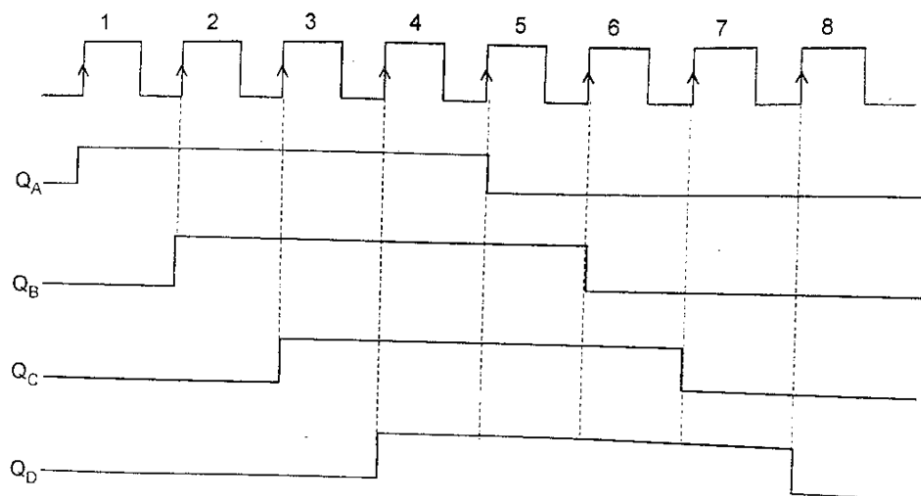


Fig. : Johnson Counter

- The output of each flip-flop is connected to D input of the next stage. The inverted output of last flip-flop $\overline{Q_D}$ is connected to the D input of the first flip-flop A.
- Initially, assume that the counter is reset to 0. i.e., $Q_A Q_B Q_C Q_D = 0000$. The value at $D_B = D_C = D_D = 0$, whereas $D_A = 1$ since $\overline{Q_D}$.
- When the *first clock pulse* is applied, the first flip-flop A is set and the other flip-flops are reset. i.e., $Q_A Q_B Q_C Q_D = 1000$.
- When the *second clock pulse* is applied, the counter is $Q_A Q_B Q_C Q_D = 1100$. This continues and the counter will fill up with 1's from left to right and then it will fill up with 0's again.
- The sequence of states is shown in the table. As observed from the table, a 4-bit shift counter has 8 states. In general, an n -flip-flop Johnson counter will result in $2n$ states.

Clock Pulse	Q_A	Q_B	Q_C	Q_D	$\overline{Q_D}$
0	0	0	0	0	1
1	1	0	0	0	1
2	1	1	0	0	1
3	1	1	1	0	1
4	1	1	1	1	0
5	0	1	1	1	0
6	0	0	1	1	0
7	0	0	0	1	0
0	0	0	0	0	1

The
diagram
counter is



timing
of Johnson
as follows:

Fig. : Timing Diagram of Johnson Counter

Ring Counter:

A 4-bit ring counter using D Flip-Flop is shown in figure.

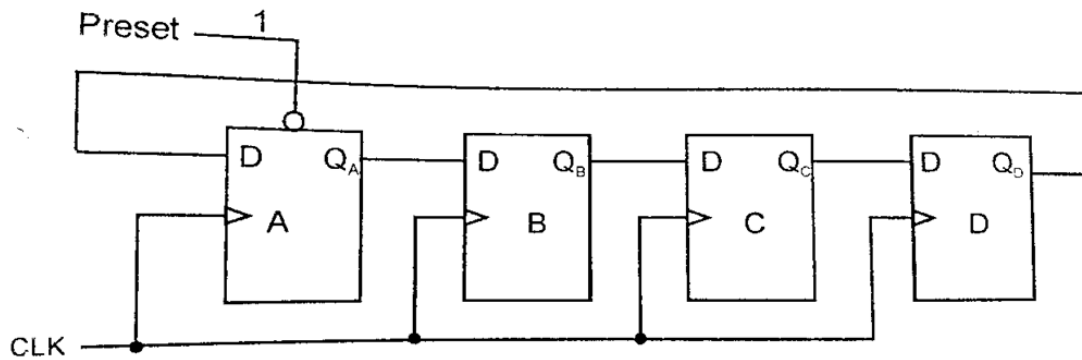
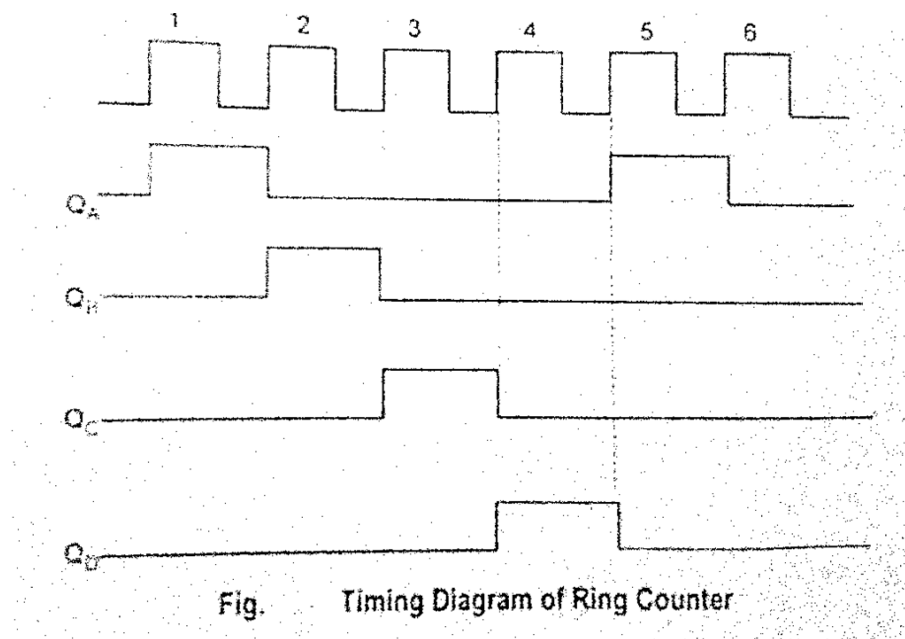


Fig. : Ring Counter

- As shown in figure, the true output of flip-flop D. i.e., Q_D is connected back to serial input of flip-flop A.
- Initially, 1 preset into the first flip-flop and the rest of the flip-flops are cleared i.e., $Q_A Q_B Q_C Q_D = 1000$.
- When the *first clock pulse is applied*, the second flip-flop is set to 1 while the other three flip-flops are reset to 0.
- When the second clock pulse is applied, the '1' in the second flip-flop is shifted to the third flip-flop and so on.
- The truth table which describes the operation of the ring counter is shown below.

Clock Pulse	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0

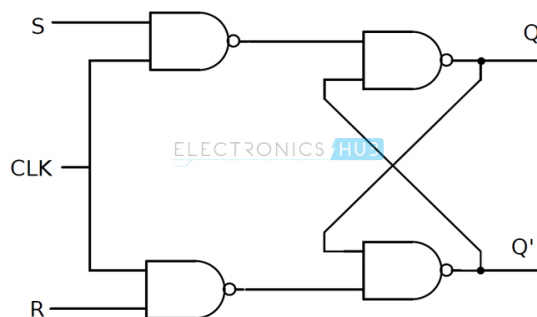
- As seen a 4-bit ring counter has 4 states. In general, an n -bit ring counter has n states. Since a single '1' in the register is made to circulate around the register, it is *called a ring counter*. The timing diagram of the ring counter is shown in figure.



TIMEMARK

May 2019

1. Draw logic diagram and truth table of a Ratchet SR flip-flop.



Inputs			Outputs		Comments
E	S	R	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	Q_n	\overline{Q}_n	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	x	x	Indeterminate

1. Difference between Combinational & Sequential Circuits.

S.no	Combinational Circuits	Sequential Circuits
1	The output at all times depends only on the present combination of input variables.	The output not only depends on the present input but also depends on the past history input variables.
2	Memory unit is not Required	Memory unit is required to store the past history of input variable
3	Clock input is not needed.	Clock input is needed.
4	Faster in Speed	Speed is Slower
5	Easy to design. Eg:Mux, Demux, Encoder, Decoder, Adders, Subtractors.	Difficult to design. Eg: Shift Register, Counters.

2. What are the classifications of sequential circuits?

The sequential circuits are classified on the basis of timing of their signals in to two types. They are

1) Synchronous sequential circuit.2) Asynchronous sequential circuit.

3. Define Latch.

The basic unit for storage is Latch. A Latch maintain its output state either at 1or 0 until directed by an input signal to change its state.

4. Define a flip flop.

A flip-flop is a storage device capable of storing one bit of information. It has two states either 0 or 1. It is also called bistable multivibrator.

5. What are the different types of flip-flop?

The various types of flip flops are 1). SR flip-flop 2). D flip-flop 3). JK flip-flop 4). T flip-flop

6. What is the main difference between a latch and flip flop?

- ✓ The output of latch changes immediately when its input changes.
- ✓ The output of a flip-flop changes only when its clock pulse is active and its input changes.
Input changes do not affect output if its clock is not activated.

7. State few application of Flip-Flop.

- ✓ Used as a memory element.
- ✓ Used as delay elements.
- ✓ Data transfer
- ✓ Used as a building block in sequential circuits such as counters and registers.

8. What is the operation of D flip-flop?

In D flip-flop during the occurrence of clock pulse if $D=1$, the output Q is set and if $D=0$, the output is reset. Set – 1, Reset – 0.

9. What is the operation of JK flip-flop?

When K input is low and J input is high the Q output of flip-flop is set.

When K input is high and J input is low the Q output of flip-flop is reset.

When both the inputs K and J are low the output does not change

When both the inputs K and J are high it is possible to set or reset the flip-flop (ie) the output toggle on the next positive clock edge.

10. What is the operation of T flip-flop?

T flip-flop is also known as Toggle flip-flop. 1). When $T=0$ there is no change in the output. 2). When $T=1$ the output switch to the complement state (ie) the output toggles.

11. Define race around condition.

In JK flip-flop output is fed back to the input. Therefore change in the output results change in the input. Due to this in the positive half of the clock pulse if both J and K are high then output toggles continuously. This condition is called 'race around condition'.

12. What is triggering? What is the need for trigger in flip-flop?

A flip-flop is made to change its state by application of a clock pulse after giving inputs. This is called triggering. The clock (triggering input) is given to synchronize the change in the output with it.

13. What is meant by level and edge-triggering?

- ✓ If flip-flop changes its state when the clock is positive (high) or negative (low) then, that flip-flop is said to be *level triggering flip-flop*.
- ✓ If the flip-flop changes its state at the positive edge (rising edge) or negative edge (falling edge) of the clock is sensitive to its inputs only at this transition of the clock then flip-flop is said to be *edge triggered flip-flop*.

14. How do you eliminate race around condition in JK flip flop. ?

Using master-slave flip-flop which consists of two flip-flops where one circuit serves as a master and the other as a slave race around condition in JK flip flop is eliminated .

15. Define rise time.

The time required to change the voltage level from 10% to 90% is known as rise time (t_r).

16. Define fall time.

The time required to change the voltage level from 90% to 10% is known as fall time (t_f).

17. Define skew and clock skew.

The phase shift between the rectangular clock waveforms is referred to as skew and the time delay between the two clock pulses is called clock skew.

18. Define setup time.

The setup time is the minimum time required to maintain a constant voltage levels at the excitation inputs of the flip-flop device prior to the triggering edge of the clock pulse in order for the levels to be reliably clocked into the flip flop.

19. Define hold time.

The hold time is the minimum time for which the voltage levels at the excitation inputs must remain constant after the triggering edge of the clock pulse in order for the levels to be reliably clocked into the flip flop.

20. Define propagation delay.

A propagation delay is the time required to change the output after the application of the input

21. Explain the flip-flop excitation tables for RS FF.

In RS flip-flop there are four possible transitions from the present state to the Next state. They are

- 1). $0 \rightarrow 0$ transition: This can happen either when $R=S=0$ or when $R=1$ and $S=0$.
- 2). $0 \rightarrow 1$ transition: This can happen only when $S=1$ and $R=0$.
- 3). $1 \rightarrow 0$ transition: This can happen only when $S=0$ and $R=1$.
- 4). $1 \rightarrow 1$ transition: This can happen either when $S=1$ and $R=0$ or $S=0$ and $R=0$.

22. Give some applications of clocked RS Flip-flop.

Clocked RS flip flops are used in Calculators & Computers.

It is widely used in modern electronic products.

23. What is the drawback of SR Flipflop? How is this minimized?

In SR flipflop when both S and R inputs are one it will generate a Undetermined state. This is Minimized by providing feedback path or by using JK flip flop.

24. How many flip flops are required to build a Binary counter that counts from 0 to 1023?

$2^{10} = 1024$ hence 10 flipflops are required.

25. What is mealy and Moore circuit? Or what are the models used to represent clocked sequential circuits?

(Dec 2019)

- ✓ *Mealy circuit* is a network where the output is a function of both present state and input.
- ✓ *Moore circuit* is a network where the output is function of only present state

26. What is counter?

A counter is a register (group of Flip-Flop) capable of counting the number of clock pulse arriving at its clock input.

27. What is binary counter?

A counter that follows the binary number sequence is called a binary counter.

28. State the applications of counters.

1. Used as a memory Element.
2. Used as a Delay Element.
3. Used as a basic building block in sequential circuits such as counters and registers.
4. Used for Data Transfer, Frequency Division & Counting.

29. List the types of counters.

Counter are classified into two types,

- ✓ Asynchronous (Ripple) counters.
- ✓ Synchronous counters.

(Dec 2017) (May 2018)

30. Give the comparison between synchronous & Asynchronous counters. (Nov/Dec 2009)

S.No	Asynchronous counters	Synchronous counters
1.	In this type of counter flip-flops are connected in such a way that output of 1 st flip-flop drives the clock for the next flip - flop.	In this type there is no connection between output of first flip-flop and clock input of the next flip – flop
2	All the flip-flops are not clocked simultaneously	All the flip-flops are clocked simultaneously
3	Logic circuit is very simple even for more number of states	Design involves complex logic circuit as number of states increases
4	Counters speed is low.	Counters speed is high.

31. State the Steps or Design procedure for Synchronous Counter.

- Preparation of
- 1). State Diagram
 - 2). State Table
 - 3). State Assignment
 - 4). Excitation Table (Consider which Memory Unit Using)
 - 5). K-Map
 - 6). Circuit Diagram

32. What is modulo-N counter?

A modulo- n counter will count n states. For example a mod-6 counter will count the sequence 000,001,010,011,100,101 and then recycles to 000. Mod -6 counter skips 110 and 111 states and it goes through only six different states.

33. Define state diagram.

State diagram is the graphical representation of the information available in a state table. In state diagram, a state is represented by a circle and the transitions between states are indicated by directed lines connecting the circles.

34. What is the use of state diagram?

- i) Behavior of a state machine can be analyzed rapidly.
- ii) It can be used to design a machine from a set of specification.

35. What is state table?

A state table is a table that represents relationship between inputs, outputs and flip-flop states, is called state table. Generally it consists of four section present state, next state, input and output.

36. What is a state equation?

A state equation also called, as an application equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation denotes the next state of the flip-flop and the right side, a Boolean function specifies the present state.

37. Define sequential circuit.

Sequential circuits are circuits in which the output variables dependent not only on the present input variables but they also depend up on the past output of these input variables.

38. What do you mean by present state?

The information stored in the memory elements at any given time defines the present state of the sequential circuit.

39. What do you mean by next state?

The present state and the external inputs determine the outputs and the next state of the sequential circuit.

40. Define synchronous sequential circuit.

Synchronous Sequential circuits are circuits in which the signals can affect the memory elements only at discrete instant of time.

41. What are the steps for the design of asynchronous sequential circuit?

- iii) Construction of primitive flow table
- iv) Reduction of flow table
- v) State assignment is made
- vi) Realization of primitive flow table

42. Define registers.

A register is a group of flip-flops. A n-bit register has a group of n flip-flops and is capable of storing any binary information/number containing n-bits.

43. Define shift registers. What is shift register ? [NOV 2020] (Dec 2018

A register capable of shifting its binary information in one or both directions is called as a shift register. It consists of a chain of flip flops in cascade, with the output of one flip flop connected to the input of the next flip-flop

44. What are the different types of shift registers?[Nov/Dec 2010, April/May 2007]

- ✓ Serial In Serial Out Shift Register
- ✓ Serial In Parallel Out Shift Register
- ✓ Parallel In Serial Out Shift Register
- ✓ Parallel In Parallel Out Shift Register
- ✓ Bidirectional Shift Register

45. State the applications of shift register.

Shift registers are widely used in

- ✓ Time delay circuits
- ✓ As Serial to parallel converter
- ✓ As Parallel to serial converters
- ✓ As Counters

46. Define Shift Register Counter.

A shift register can also be used as a counter. A shift register with the serial output connection back to the serial input is called Shift register counter

47. What is bi-directional shift register and unidirectional shift register?

A register capable of shifting both right and left is called bi-directional shift register. A register capable of shifting only one direction is called unidirectional shift register.

48. What are the two types of shift register counters?[April/May 2007, Nov/Dec 2006, 2011, 2012]

There are 2 types of shift Register counters are:

Ring counter:

A ring counter is a circular shift register with only one flip flop being set, at any particular time, all others are cleared.

Johnson counters:

The Johnson counter is K-bit switch-tail rings counter $2k$ decoding gates to provide outputs for $2k$ timing signals.

49. How can a SIPO shift register is converted in to SISO shift register? (Apr/May 2010)

By taking output only on the Q output of last flip flop SIPO shift register is converted in to SISO shift register.

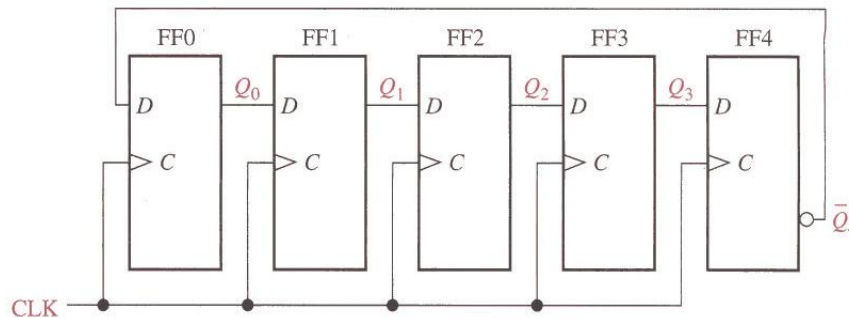
50. What is bi-directional shift register and unidirectional shift register?

A register capable of shifting both right and left is called bi-directional shift register. A register capable of shifting only one direction is called unidirectional shift register.

51. What is sequence generator?

The sequential circuit used to repeat a particular sequence repeatedly is called Sequence generator.

52. Draw 5-bit Johnson counter.



53. Draw the circuit diagram of Ring counter. [NOV/DEC 2021]

2018

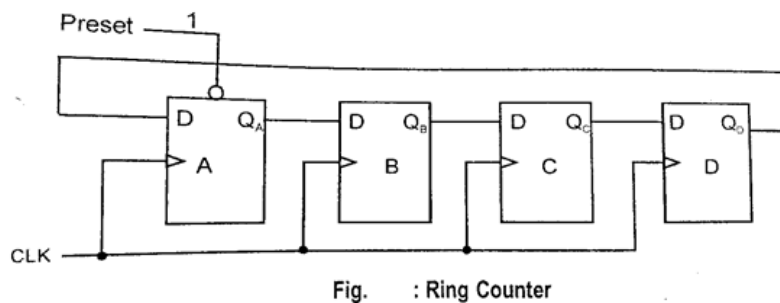


Fig. : Ring Counter

54. Give few applications of shift register.

- ✓ Serial to parallel converter
- ✓ Parallel to serial converter
- ✓ As a counter
- ✓ To introduce delay in a digital circuit.

55. How many flip-flops are needed to implement a 10-bit ripple counter to reach the state 1001100111 after the reset of 1001100111?

May 2019

Add 1 to the binary number 1001100111 to generate the next state. The 4th bit from the right is toggled to generate the next state. The 4th bit is 1, so it is toggled to 0, and the 3rd bit is 1, so it is toggled to 0. The 2nd bit is 0, so it is toggled to 1. The 1st bit is 1, so it is toggled to 0. The next state is 1001101000.

1001100111 is 1001101000. 4 bits are toggled, 4 bits are needed.

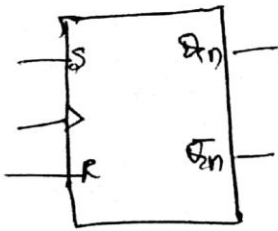
FLIP-FLOPS

* SR FLIP-FLOP:-

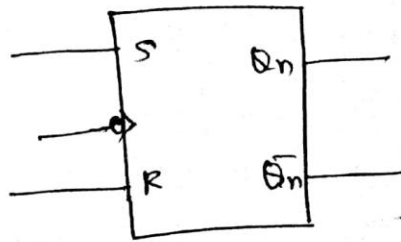
[May 2018]

Draw SR FF circuit and explain the operation with truth table and suggest how to eliminate the undetermined stage? write some applications. [DEC-2017]

* Symbol

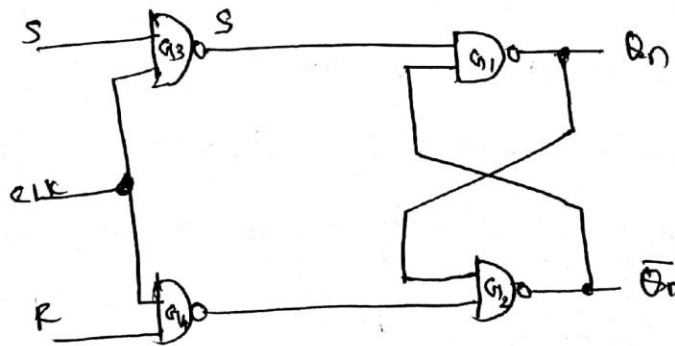


a) +ve edge triggered



b) -ve edge triggered

* Logic Diagram



* Truth Table

S	R	Q_{n+1}	state
0	0	Q_n	NC
0	1	0	Reset
1	0	1	Set
1	1	X	Indeter- minate

CLK	S	R	Q_n	Q_{n+1}	state
0	x	x	0	0	No change
0	x	x	1	1	
↑	0	0	0	0	No change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	x	Indeter- -minate
↑	1	1	1	x	

OPERATION:-

case i):- when $S=0$ & $R=0$

- * Positive edge triggered, S_1 becomes '1', R_1 becomes '1'.
- * Assume $Q_n=1$ & $\overline{Q}_n=0$.
- * o/p of G_2 remain logic '0'. o/p of G_1 remains logic '1'.
- * o/p doesnot change when S & R are low.

case ii):- when $S=0$ & $R=1$

- * $S_1 \rightarrow$ logic '1' ; $R_1 \rightarrow$ logic '0' ; Assume $Q_n=1, \overline{Q}_n=0$
- * o/p of G_2 becomes logic '1'.
- * o/p of G_1 becomes logic '0'.
- * Hence o/p is in RESET condition

case iii:- when $S=1$ & $R=0$

* $S_1 \rightarrow \text{logic '0'}$; $R_1 \rightarrow \text{logic '1'}$

* Assume $Q_n=1$ & $\bar{Q}_n=0$

* o/p of G_2 becomes logic '0'.

* o/p of G_1 becomes logic '1'

* Hence the o/p is in SET condition.

case iv:- when $S=1$ & $R=1$

* $S_1 \rightarrow \text{logic '0'}$; $R_1 \rightarrow \text{logic '0'}$.

* Assume $Q_n=1$, $\bar{Q}_n=0$

* o/p of G_2 & G_1 becomes logic '1'.

* Hence the flip combination is avoided.

This is called indeterminate condition.

K-map for Q_{n+1} :-

S	$R Q_n$			
	00	01	11	10
0		1		
1	1	1	x	x

\therefore the characteristic eq. of

SR-FF is

$$Q_{n+1} = S + \bar{R} Q_n$$

Applications:

* Memory

* Counters

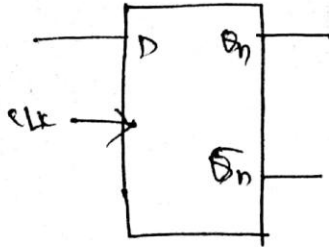
* Registers

With neat sketches and function table, briefly explain the working of a positive edge triggered D flip-flop and also mention its significance. [NOV/DEC 2021]

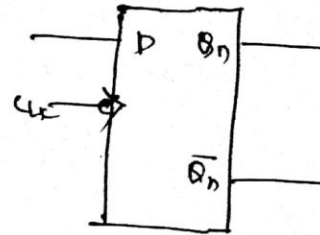
D-FLIP FLOP

Explain the operation of DFF using truth table.

Symbol:

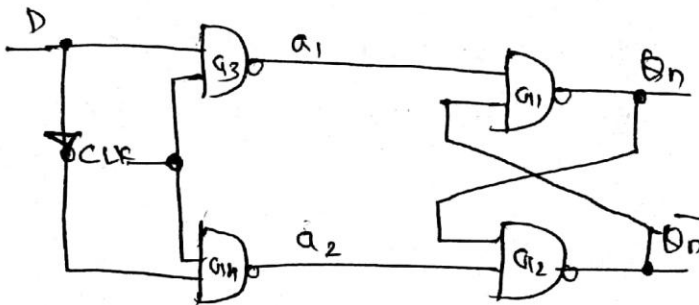


a) +ve edge Triggered



b) -ve edge Triggered.

Logic diagram:



Truth Table:

clk	D	Q_n	Q_{n+1}	state
0	x	0	0	No change
0	x	1	1	
↑	0	0	0	RESET
↑	0	1	0	
↑	1	0	1	SET
↑	1	1	1	

D	Q_{n+1}	state
0	0	RESET
0	1	SET

OPERATION:-

Case i, \therefore when $D=0$

- * Positive triggered in all cases
- * a_1 becomes logic '1'; a_2 becomes logic '0'.
- * Assume $Q_n=1$ & $\bar{Q}_n=0$
- * o/p of G_1 is '0'.
- * o/p of G_2 is '1'.
- * Hence the o/p Q_n is in RESET condition.

Case ii, \therefore when $D=1$

- * a_1 becomes logic '0'; a_2 becomes logic '1'.
- * o/p of G_1 is '1'.
- * o/p of G_2 is '0'.
- * Hence the o/p Q_n is in SET condition.

K-map of Q_{n+1} :-

$D \backslash Q_n$	0	1
0	0	0
1	1	1

\therefore the characteristic eq for D Flip flop is

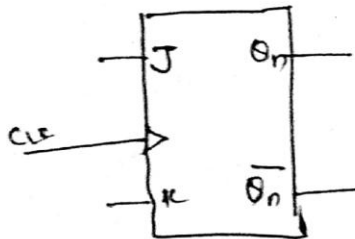
$$\boxed{Q_{n+1} = D}$$

* J-K Flip-Flop:

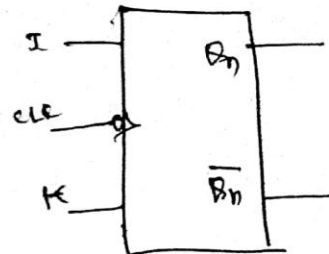
Explain JK Flip flop with its truth table and find the characteristic equation.

(Dec 2018)

* Symbol:

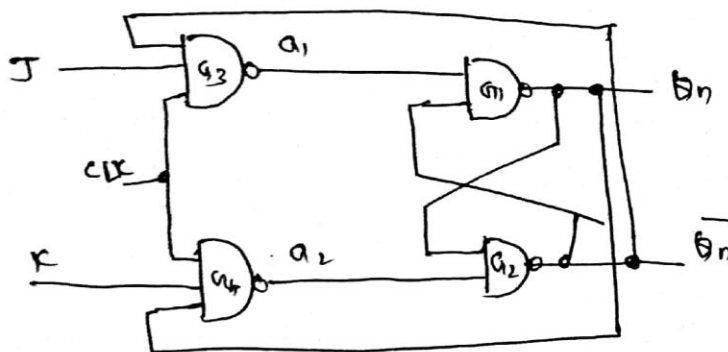


a) Positive Triggered



b) Negative Triggered

* Logic Diagram:



* Truth Table:

clk	J	K	Q_n	Q_{n+1}	state
0	X	X	0	0	No change
0	X	X	1	1	No change
↑	0	0	0	0	No change
↑	0	0	1	1	No change
↑	0	1	0	0	RESET
↑	0	1	1	0	RESET
↑	1	0	0	1	SET
↑	1	0	1	1	SET
↑	1	1	0	1	Toggles
↑	1	1	1	0	Toggles

J	K	Q_{n+1}	state
0	0	Q_n	No change
0	1	0	RESET
1	0	1	SET
1	1	\bar{Q}_n	Toggles

OPERATION:-

case i, when $J=0$ & $K=0$

- * Assume $Q_n=1$; $\bar{Q}_n=0$ in all cases.
- * Positive edge triggered in all cases.
- * Q_1 becomes '1'; Q_2 becomes '1'.
- * o/p of G_1 is $Q_n=1$; o/p of G_2 is $\bar{Q}_n=0$.
- * Hence the o/p does not change if J, K are low.

case ii, when $J=0$ & $K=1$

- * o/p $Q_1=1$; $Q_2=0$
- * o/p of G_1 is $Q_n=0$; the o/p of G_2 is $\bar{Q}_n=1$.
- * Hence the o/p is in RESET condition.

case iii, when $J=1$ & $K=0$

- * o/p $Q_1=1$; $Q_2=1$
- * o/p of G_1 is $Q_n=0$; the o/p of G_2 is $Q_n=1$.
- * Hence the o/p is in SET condition.

case iv, when $J=1$ & $K=1$

- * o/p $Q_1=1$; $Q_2=0$
- * the o/p of G_1 is $Q_n=0$;
- * the o/p of G_2 is $\bar{Q}_n=1$
- * Hence the o/p is in complement form.

RACE AROUND CONDITION:

- * In JK-FF, o/p is fed back to the input.
- * Therefore change in the o/p results change in the input.
- * During positive edge triggering, if both J & K are high then output toggles continuously.
- * This condition is called race around condition.
- * Avoided by using Master slave flip flop.

K-map of Q_{n+1} :-

J \ K Q_n				
	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

∴ the characteristic eq. of JK FF is

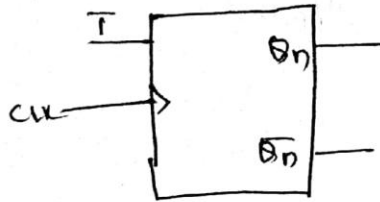
$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

← x ←

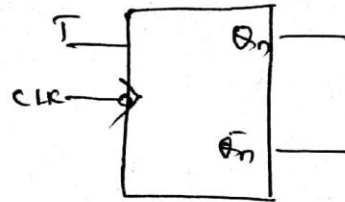
* T- FLIP FLOP:

Explain the operation of T-FF with its Truth Table.

* Symbol:

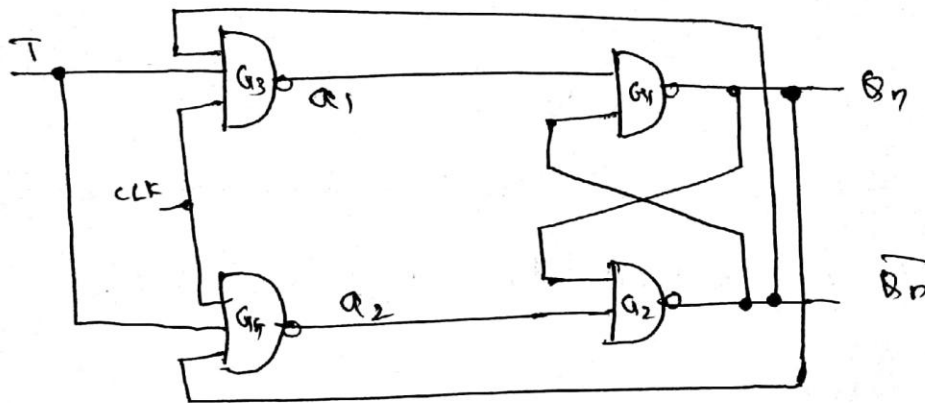


a) Positive Triggered



b) Negative edge Triggered.

* Logic Diagram:



* Truth Table:

clk	T	Q_n	Q_{n+1}	State
0	x	0	0	No change
0	x	1	1	
↑	0	0	0	No change
↑	0	1	1	
↑	1	0	1	Toggles
↑	1	1	0	

T	Q_{n+1}	State
0	Q_n	No change
1	\bar{Q}_n	Toggles

OPERATION:-

case i, when $T=0$

- * Assume $Q_n=1$ & $\bar{Q}_n=0$
- * positive edge triggered is applied.
- * o/p $a_1=1$; $a_2=1$
- * the o/p of G_1 is $Q_n=1$;
- * the o/p of G_2 is $\bar{Q}_n=0$.
- * Hence the o/p does not change if $T=0$.

case ii, when $T=1$

- * the o/p $a_1=1$; $a_2=0$.
- * the o/p of G_1 is $Q_n=0$; the o/p of G_2 is $\bar{Q}_n=1$.
- * Hence the output Q_{n+1} toggles if $T=1$.

K-map for Q_{n+1}

T	Q_n	
	0	1
0	0	1
1	1	0

$$\therefore Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$$

\therefore the characteristic equation of T-FF is

$$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$$

Explain the operation of Master Slave flip flop and show how the race around condition is eliminated.

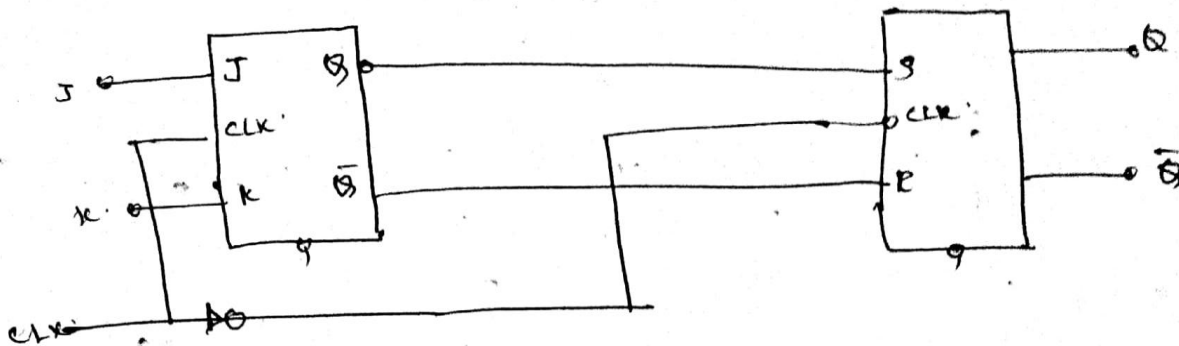
[NOV 2020]

MASTER SLAVE J-K - FLIP FLOP

(Dec 2018)

Explain the operation of Master Slave J-K Flip Flop.

* The Block diagram of a Master Slave JK FF is shown in fig.



* It consists of clocked JK Flip Flop as a master and clocked JK Flip Flop as a slave.

* The clock signal is connected directly to the master flip flop.

* But it is connected through inverter to the slave flip flop.

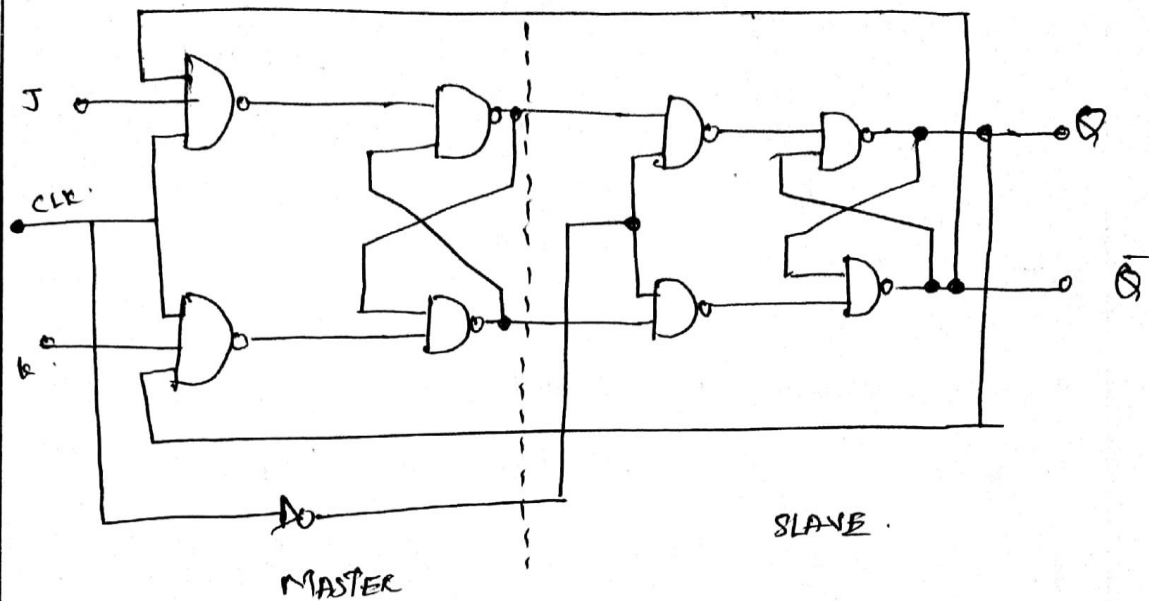
* When clock goes high the information present at the JK inputs is transmitted to the output of Master flip flop and it is held there, since slave is inactive.

* When clock goes low the slave flip flop transfers the output of master flip flop.

* Since master is inactive at this instant, it does not respond to the feedback from Q & Q-bar.

* Thus the master slave flip flop, it does not have race around problem.

* A master slave JK flip flop using NAND gate is shown in Fig.



OPERATION:

* When $J=1$ and $K=0$ and clock is high, the master will be set making 'S' high and 'R' low.

* When clock becomes low, the slave will be set making Q high and Q-bar low.

* When $J=0$ and $K=1$, the master resets on the positive clock. This makes 'S' low and 'R' high.

- * When clock goes low, the slave will also be RESET making Q low and \bar{Q} high.
- * When $J=1$ and $K=1$ and the previous state of the flip flop is SET (Q of slave $= 1$) the master will be RESET on positive clock.
- * When clock goes low, the slave will also be RESET.
- * When the clock goes high, again the master will be set and this will set the slave. when clock goes low. the FF is said to be TOGGLE.
- * No change in output if both J & K inputs are low.

_____ x _____

- * write short notes on state minimization and state assignment.

State Minimization:-

- * State reduction algorithm is concerned with the procedure for reducing the number of states in the state table.
- * When two states in the state table are equivalent by producing the ~~same~~ same next state and same output then one of states in the state table can be removed without altering the input output relationship.

State Assignment:-

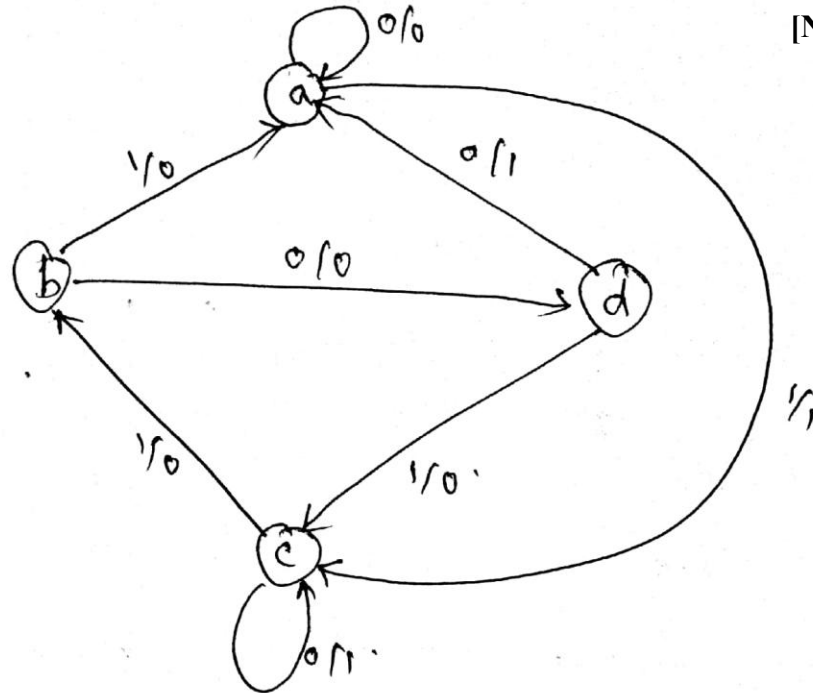
- * In order to design a sequential circuit, it is necessary to assign binary values to the state.
- * For a circuit with 'm' states, the codes must contain 'n' bits, where $2^n \geq m$.

————— x —————

Problem:

* A sequential circuit has one input and one output. The state diagram is shown in Fig. Design the sequential circuit with SR flip flop.

[NOV/DEC 2021]



Sol:

Step 1: state Table.

Present State	Next state		output	
	x=0	x=1	Z=0	Z=1
a	a	c	0	1
b	d	a	0	0
c	c	b	1	0
d	a	0	1	0

Step 2: Reduce the no. of states if possible.

Step 3: Apply state assignment

$$a = 00$$

$$b = 01$$

$$c = 10$$

$$d = 11$$

* Excitation table for SR Flip flop.

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Step 4: Transition Table.

present state		input	Next state		Flip Flop i/p's				output
A	B	X	A^+	B^+	S_A	R_A	S_B	R_B	Z
0	0	0	1	0	0	X	0	X	0
0	0	1	1	0	1	0	0	X	1
0	1	0	1	1	1	0	X	0	0
0	1	1	0	0	0	X	0	1	0
1	0	0	0	X	X	0	0	X	1
1	0	1	1	0	0	1	1	0	0
1	1	0	0	0	0	1	0	1	1
1	1	1	0	X	X	0	0	1	0

step 5: Derive the flip flop input equations and output equations using K-map.

A \ Bx	00	01	11	10
0		1		1
1	x		x	

$$S_A = \bar{A}\bar{B}x + \bar{A}B\bar{x}$$

A \ Bx	00	01	11	10
0	x		x	
1		1		1

$$R_A = A\bar{B}x + AB\bar{x}$$

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

$$Z = A\bar{x} + \bar{A}Bx$$

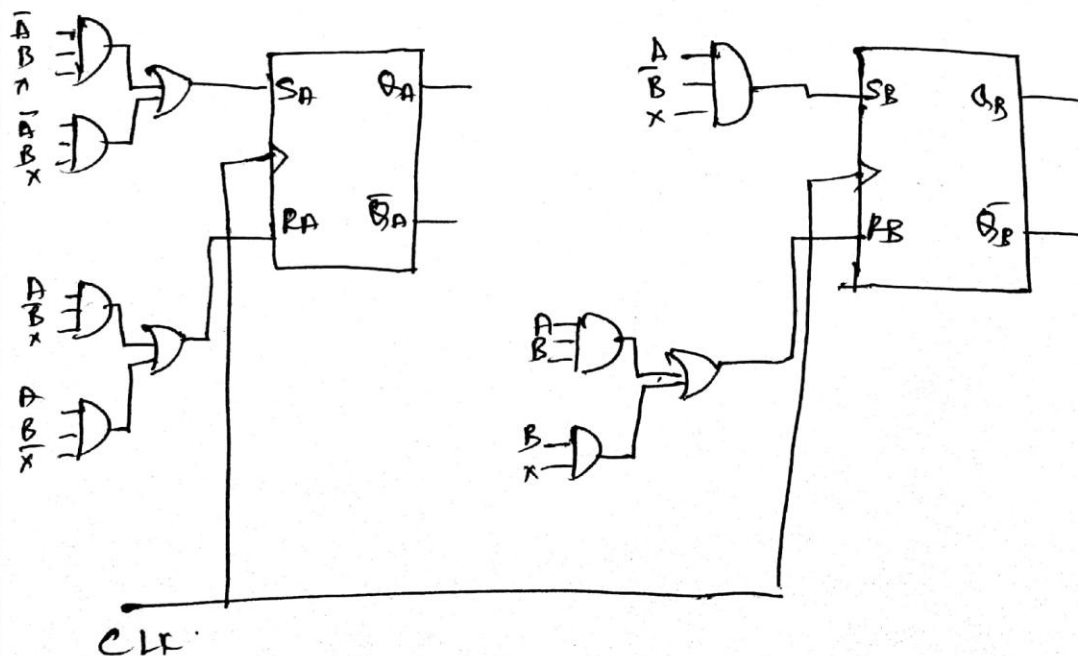
A \ Bx	00	01	11	10
0				x
1		1		

$$S_B = A\bar{B}x$$

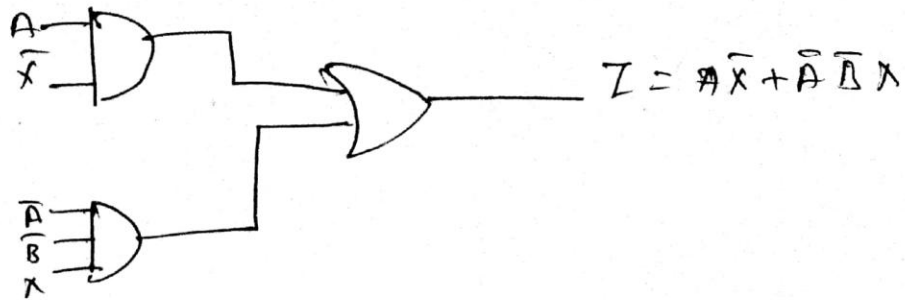
A \ Bx	00	01	11	10
0	x	x	1	
1	x		1	1

$$R_B = AB + Bx$$

Step 6: Logic Diagram:



output:



_____ X _____

* Write down the characteristic equation and excitation table for T flip-flop. [NOV/DEC 2021]
write the excitation table for flip flop.

Q_t	Q_{t+1}	S	R	D	J	K	T
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	0	0	1	0	X	1	1
1	1	X	0	1	X	0	0

K-map for Q_{n+1}

	Q_n	0	1
T	0	0	1
	1	1	0

$$\therefore Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$$

(Dec 2019)

Example 3.3 Design a sequential circuit with 3D Flip-Flops one input x and one output y for the state diagram given below.

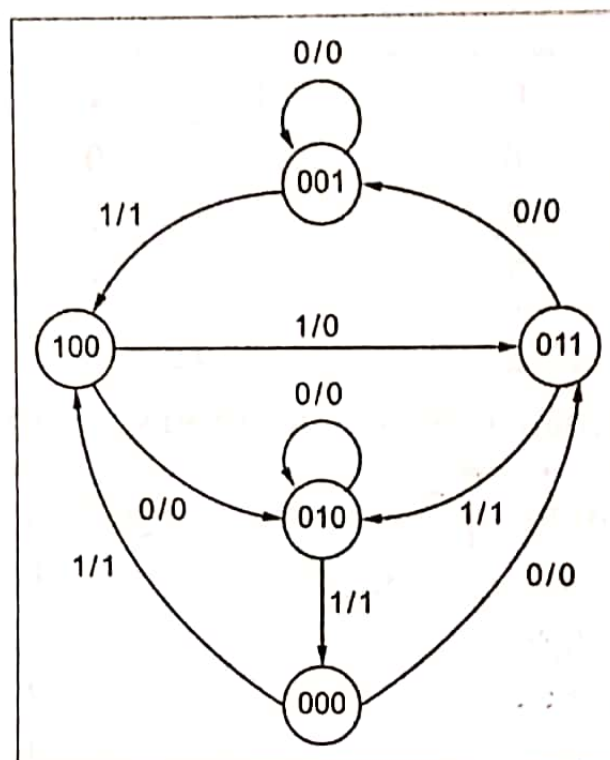


Fig. 3.55. State diagram

😊 Solution:

Present state			Input x	Next state			Output y
A	B	C		A	B	C	
0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	0	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	1	0

Transition Table

Present State			Next State						Output	
			$x = 0$			$x = 1$			$x = 0$	$x = 1$
A	B	C	A	B	C	A	B	C	Y	Y
0	0	0	0	1	1	1	0	0	0	1
0	0	1	0	0	1	1	0	0	0	1
0	1	0	0	1	0	0	0	0	0	1
0	1	1	0	0	1	0	1	0	0	1
1	0	0	0	1	0	0	1	1	0	0

2. Excitation Table for D Flip-Flop

Excitation Table for D Flip-Flop

Q	Q + 1	D
0	0	0
0	1	1
1	0	0
1	1	1

In the given state diagram five states are there. So three D-Flip-Flops required.

Excitation Table for given State Diagram

Present State			Input	Next State			Flip-Flop inputs			Output
A	B	C	x	A	B	C	D _A	D _B	D _C	y
0	0	0	0	0	1	1	0	1	1	0
0	0	0	1	1	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	1	0	0	1	0	0	1
0	1	0	0	0	1	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	1
0	1	1	0	0	0	1	0	0	1	0
0	1	1	1	0	1	0	0	1	0	1
1	0	0	0	0	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1	1	0

3. K-Map Simplification

Determine the expressions for D_A, D_B, D_C and output y.

For D_A:

AB \ CX					
		00	01	11	10
00	00	0	1	1	0
	01	0	0	0	0
11	11	X	X	X	X
	10	0	0	X	X

$$D_A = \bar{A} \bar{B} X$$

For D_B:

AB \ CX					
		00	01	11	10
00	00	1	0	0	0
	01	1	0	1	0
11	11	X	X	X	X
	10	1	1	X	X

$$D_B = A + \bar{C} \bar{X} + BCX$$

For D_C :

AB \ CX	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	0	1	X	X

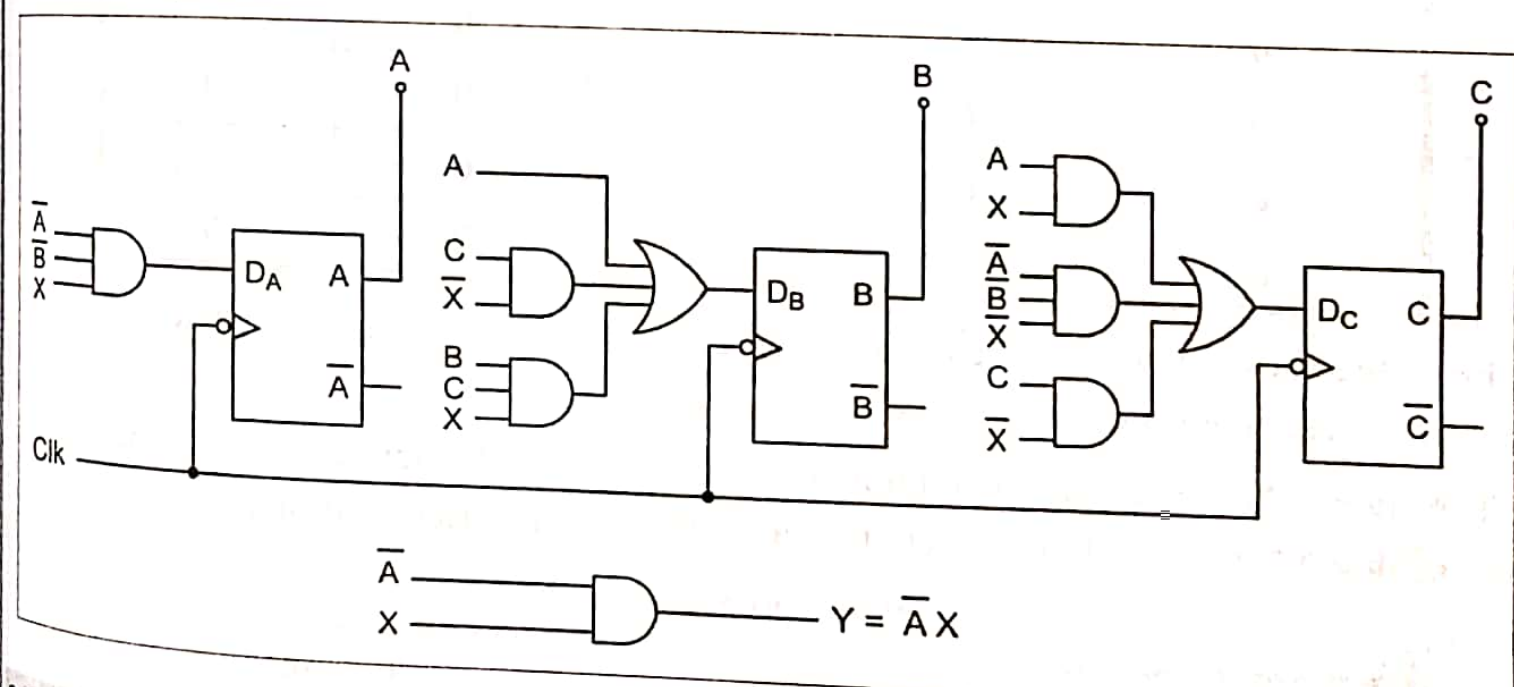
$$D_C = AX + C\bar{X} + \bar{A}\bar{B}\bar{X}$$

For Y:

AB \ CX	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	X	X	X	X
10	0	0	X	X

$$Y = \bar{A}X$$

4. Logic Diagram



UNIT IV

ASYNCHRONOUS SEQUENTIAL CIRCUITS

Stable and Unstable states, output specifications, cycles and races, state reduction, race free assignments, Hazards, Essential Hazards, Pulse mode sequential circuits, Design of Hazard free circuits.

ASYNCHRONOUS SEQUENTIAL CIRCUITS

Write short notes on types of Asynchronous sequential circuits.

- Sequential circuits without clock pulses are called Asynchronous Sequential Circuits. They are classified into 2 types:
 1. Fundamental mode circuits
 2. Pulse mode circuits

Fundamental Mode Circuits:

It assumes that:

- ✓ The input variables should change only when the circuit is *stable*.
- ✓ Only one input variable can change at a given time.
- ✓ Inputs and outputs are represented by *levels* (0 or 1).

Pulse Mode Circuits:

It assumes that:

- ✓ Inputs and outputs are represented by *pulses*.
- ✓ The input pulses must be long enough to intimate a state changes.
- ✓ Pulses are not so wide that the input is still true after a new state is reached.

Stable state:

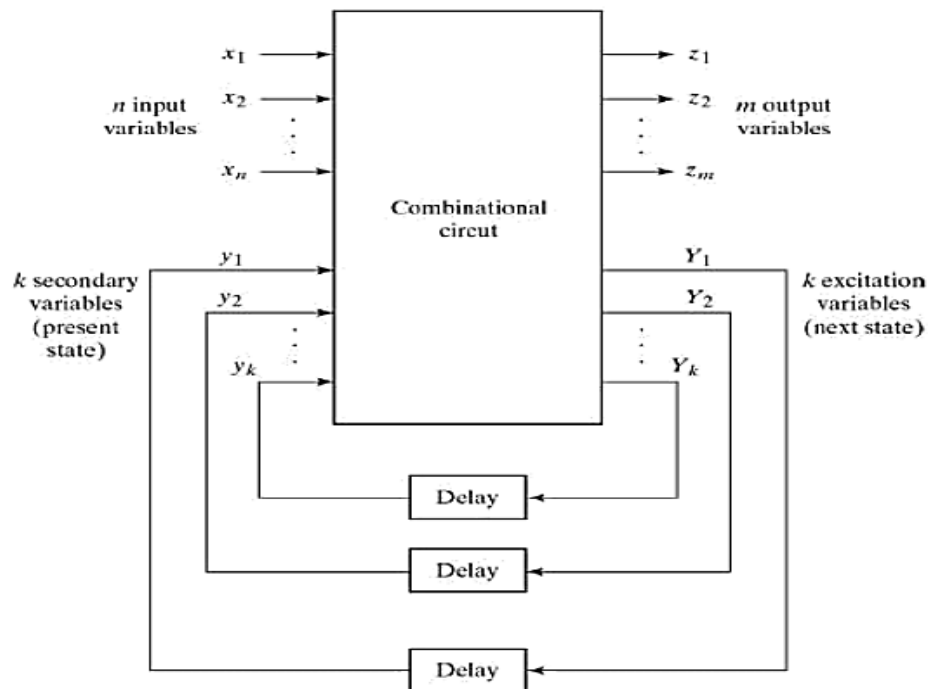
- If the circuit reaches a steady state condition with ***present state*** $y_i = \text{next state } Y_i$ for $i=1, 2, 3 \dots K$ then the circuit is said to be stable state.
- A transition from one stable to another occurs only in response to a change in an input variable.

Unstable state:

- In a circuit, if ***present state*** $y_i \neq \text{next state } Y_i$ for $i=1, 2, 3 \dots K$ then the circuit is said to be unstable state.
- The circuit will be in continuous transition till it reached a stable state.

Draw a block diagram of asynchronous sequential circuits.[NOV 2020]

Block diagram of Asynchronous Sequential circuits

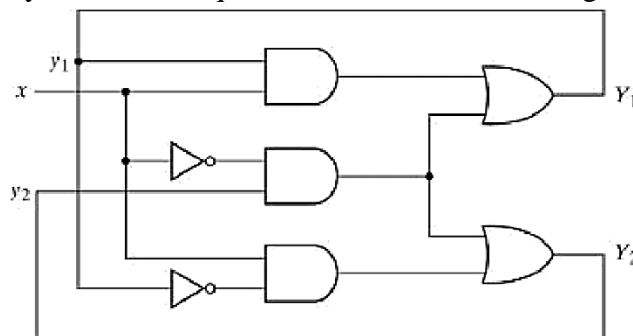


- The communication of two units, with each unit having its own independent clock, must be done with asynchronous circuits.

ANALYSIS PROCEDURE OF FUNDAMENTAL MODE SEQUENTIAL CIRCUITS 2018

Explain about analysis procedure of fundamental mode sequential circuits. (Dec2011, May 2019) 2019

- The analysis of asynchronous sequential circuits consists of obtaining a table or a diagram that described the sequence of internal states and outputs as a function of changes in the input variables.
- Let us consider the asynchronous sequential circuit is shown in figure.



- The analysis of the circuit starts by considering the excitation variables (Y_1 and Y_2) as outputs and the secondary variables (y_1 and y_2) as inputs.

Step1:

- The Boolean expressions are,\

$$Y_1 = xy_1 + x'y_2$$

$$Y_2 = xy_1' + x'y_2$$

Step 2:

- The next step is to plot the Y_1 and Y_2 functions in a map

		x	
		0	1
$y_1 y_2$			
00		0	0
01		1	0
11		1	1
10		0	1

Map for
 $Y_1 = xy_1 + x'y_2$

		x	
		0	1
$y_1 y_2$			
00		0	1
01		1	1
11		1	0
10		0	0

Map for
 $Y_2 = xy_1' + x'y_2$

- Combining the binary values in corresponding squares, the following transition table is obtained.
- The transition table shows the value of $Y = Y_1 Y_2$ inside each square. Those entries where $Y = y$ are circled to indicate a *stable condition*.
- The circuit has four stable total states, $y_1 y_2 x = 000, 011, 110$, and 101 and four unstable total states- $001, 010, 111$ and 100 .
- The state table of the circuit is shown below:

Present State		Next State			
		$x = 0$		$x = 1$	
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	1	1	1	0

- This table provides the same information as the transition table.

Step 3:

Transition table

- The transition table is obtained by combining the maps for Y_1 and Y_2 .

		x	
		0	1
$y_1 y_2$	00	(00)	01
	01	11	(01)
	11	(11)	10
	10	00	(10)

- The transition table is a table which gives the relation between present state, input and next state. If the secondary variables $y_1 y_2$ is same as excitation variables $Y_1 Y_2$, the state is said to be stable.
- The stable states are indicated by circles. An uncircled entry represents an unstable state.
- In a transition table, usually there will be at least one stable state in each row. Otherwise, all the states in that row will be unstable.

Step 4:

Primitive Flow table

- In a flow table the states are named by letter symbols. Examples of flow tables are as follows:

		x	
		0	1
y	a	(a)	b
	b	c	(b)
	c	(c)	d
	d	a	(d)

(a) Four states with one input

- In order to obtain the circuit described by a flow table, it is necessary to assign to each state a distinct value.

Critically examine cycles and races in asynchronous sequential circuits. [NOV 2020]

Explain the problems in asynchronous circuits with examples. (Dec 2010, Dec 2012, Dec 2013)

Cycles

- A cycle occurs when an asynchronous circuit makes a transition through a *series of unstable state*.
- When a state assignment is made so that it introduces cycles, care must be taken that it terminates with a stable state.
- Otherwise, the circuit will go from one unstable state to another, until the inputs are changed.
- Examples of cycles are:

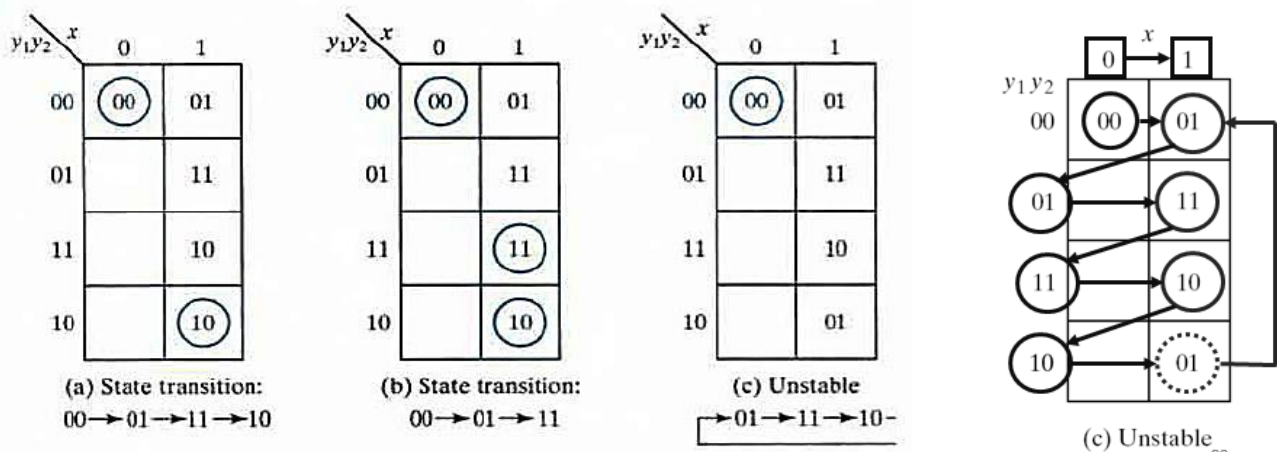


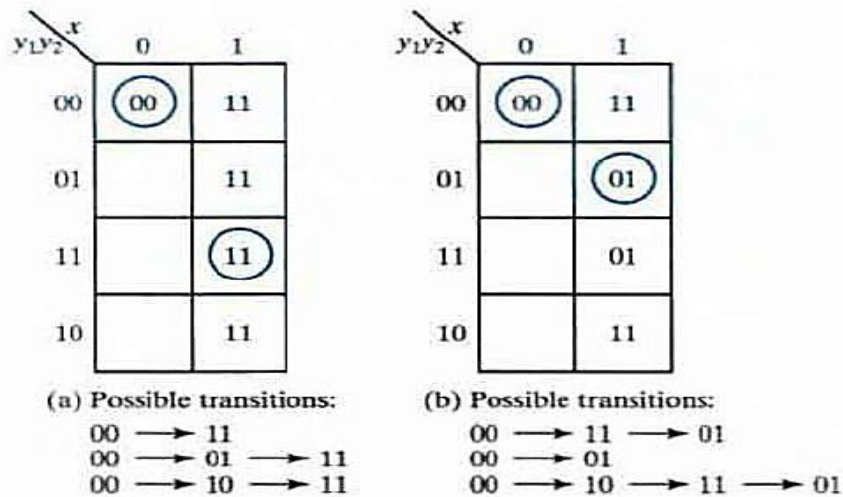
Fig: Examples of cycles

Race Conditions

Discuss about Race

(Dec 2019)

- A race condition exists in an asynchronous circuit when two or more binary state variables change value in response to a change in an input variable.
- When unequal delays are encountered, a race condition may cause the state variable to change in an unpredictable manner.
- If the final stable state that the circuit reaches **does not depend on the order** in which the state variables change, the race is called a noncritical race.
- If the final stable state that the circuit reaches **depends on the order** in which the state variables change, the race is called a critical race.
- **Examples of noncritical races** are illustrated in the transition tables below:



- Initial stable state is $y_1y_2x = 000$ and then input changes from 0 to 1.
- The state variables y_1y_2 must change from 00 to 11, (race condition).

Possible transitions are

$00 \rightarrow 11$
 $00 \rightarrow 01(y_2 \text{ faster}) \rightarrow 11$
 $00 \rightarrow 10(y_1 \text{ faster}) \rightarrow 11$

- In all cases final stable state is same, which results in a non-critical race condition.
- **Examples of critical races** are illustrated in the transition tables below:

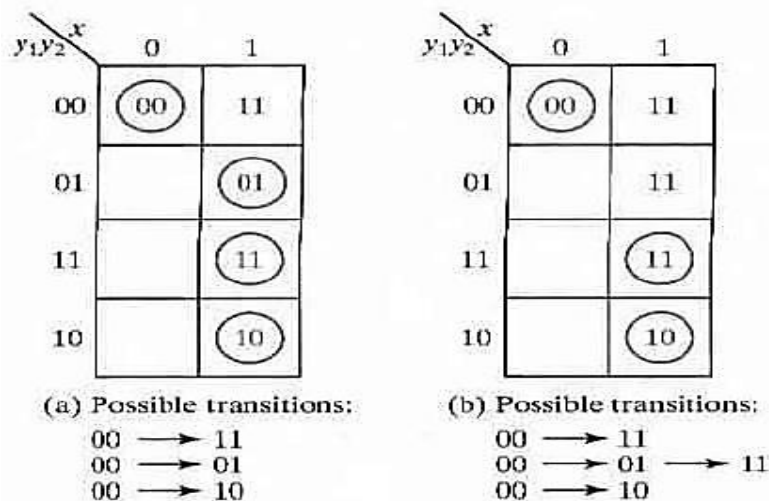


Fig: Examples of critical races

- The initial stable state is $y_1y_2x=000$ and let us consider that the input changes from 0 to 1. Then , the state variables must change from 00 to 11.
- If they change simultaneously, the final total state is 111.
- Due to unequal propagation delay, if y_2 changes to 1 before y_1 does, then the circuit goes to total stable state $y_1y_2x=011$ and remains there.

- If y_1 changes first, then the circuit will be in total stable state is $y_1y_2x=101$.
- Hence the race is critical because the circuit goes to different stable states depending on the order in which the state variables change.

DESIGN PROCEDURE OF ASYNCHRONOUS SEQUENTIAL CIRCUITS (Dec 2016)

Explain in detail about design procedure of asynchronous sequential circuits. (May 2011, Dec 2017)

- There are a number of steps that must be carried out in order to minimize the circuit complexity and to produce a stable circuit without critical races.

The design steps are as follows:

- ✓ Obtain a primitive flow table from the given specification.
- ✓ Reduce the flow table by merging rows in the primitive flow table.
- ✓ Assign binary states variables to each row of the reduced flow table to obtain the transition table.
- ✓ Assign output values to the dashes associated with the unstable states to obtain the output maps.
- ✓ Simplify the Boolean functions of the excitation and output variables and draw the logic diagram.
- ✓ The design process will be demonstrated by going through a specific example:

Example:

Design a gated latch circuit with two inputs, G (gate) and D (data), and one output Q . The gated latch is a memory element that accepts the value of D when $G = 1$ and retains this value after G goes to 0. Once $G = 0$, a change in D does not change the value of the output Q . (May 2016, Dec 2016)

(Or)

Design an asynchronous sequential circuit with two inputs D and G with one output Z . Whenever G is 1, input D is transferred to Z . When G is 0, the output does not change for any change in D . Use SR latch for implementation of the circuit.

Primitive Flow Table

- A primitive flow table is a flow table with only one stable total state in each row. The total state consists of the internal state combined with the input.
- To derive the primitive flow table, first a table with all possible total states in the system is needed:

State	Inputs		Output	Comments
	D	G	Q	
a	0	1	0	$D = Q$ because $G = 1$
b	1	1	1	$D = Q$ because $G = 1$
c	0	0	0	After state a or d
d	1	0	0	After state c
e	1	0	1	After state b or f
f	0	0	1	After state e

- Each row in the above table specifies a total state; the resulting primitive table for the gated latch is shown below:

	Inputs DG			
	00	01	11	10
a	c, -	(a), 0	b, -	-, -
b	-, -	a, -	(b), 1	e, -
c	(c), 0	a, -	-, -	d, -
d	c, -	-, -	b, -	(d), 0
e	f, -	-, -	b, -	(e), 1
f	(f), 1	a, -	-, -	e, -

- First, fill in one square in each row belonging to the stable state in that row.
- Next recalling that both inputs are not allowed to change at the same time.
- Then enter dash marks in each row that differs in two or more variables from the input variables associated with the stable state.

Reduction of primitive flow table:

- Two or more rows in the primitive flow table can be merged into one row if there are non-conflicting states and outputs on each of the columns.
- This can be done by implication table and merger diagram.
- The implication table has all states except the first vertically and all states except the last across bottom horizontally.
- The tick (✓) mark denotes that the pair (rows) is compatible.
- Two states are compatible, if the states are identical with non-conflicting outputs.
- The cross (x) mark implies non-compatible.

b	✓				
c	✓	d, e X			
d	✓	d, e X	✓		
e	c, f X	✓	d, e X c, f X	X	
f	c, f X	✓	X	c, f X d, e X	✓
	a	b	c	d	e

Fig. : Implication table

- The compatible pairs are

$(a,b), (a,c), (a,d), (b,e), (b,f), (c,d), (e,f)$

Merger Diagram:

- The *maximum compatible sets* can be obtained from merger diagram as shown in figure.
- The merger diagram is a graph in which each state is represented by a dot placed along the circumference of a circle.
- Lines are drawn between any two corresponding dot that form a compatible pair.
- Based on the geometrical patterns formed by the lines, all the possible compatibilities can be obtained.

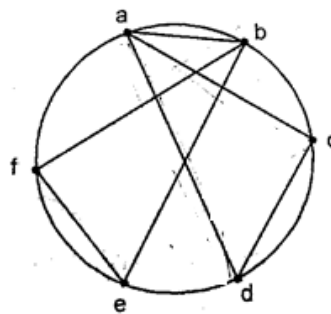


Fig. : Merger Diagram

- An isolated dot represents a state that is not compatible with any other state.
- A line represents a compatible pair.
- A triangle constitutes a compatible with three states.
- An n-state compatible is represented in the merger diagram by an n-sided polygon with all its diagonal connected.
- So, the maximal compatibilities are

$(a,b), (a,c,d), (b,e,f)$

Closed covering condition:

- In the above, if only (a,c,d) and (b,e,f) are selected, all the six states are included.
- This set satisfies the covering condition.
- Thus, the rows a,c,d can be merged as one row and b,e,f states can be merged as another row.

DG		00	01	11	10
States	a, c, d	Ⓒ, 0	Ⓐ, 0	b, -	Ⓓ, 0
	b, e, f	Ⓕ, 1	a, -	Ⓑ, 1	Ⓔ, 1

Fig. : Reduced flow table

- Consider $a,c,d = a$ and $b,e,f = b$

DG		00	01	11	10
States	a	Ⓐ, 0	Ⓐ, 0	b, -	Ⓐ, 0
	b	b, 1	a, -	Ⓑ, 1	Ⓑ, 1

Fig. : Reduced flow table with common symbol

- A race free binary assignment is made and transition table and output map is obtained.

a -> 0, b -> 1

DG		00	01	11	10
Q	0	0	0	1	0
	1	1	0	1	1

Fig. : Transition table

DG		00	01	11	10
Q	0	0	0	1	0
	1	1	0	1	1

Fig. : Output map

Logic Diagram using SR Latch:

Excitation table of SR flip-flop is used to find expressions for S and R.

Excitation Table of SR Flip-Flop

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Q \ DG	00	01	11	10
0	0	0	1	0
1	X	0	X	X

$$S = DG$$

Q \ DG	00	01	11	10
0	X	X	0	X
1	0	1	0	0

$$R = \overline{D}G$$

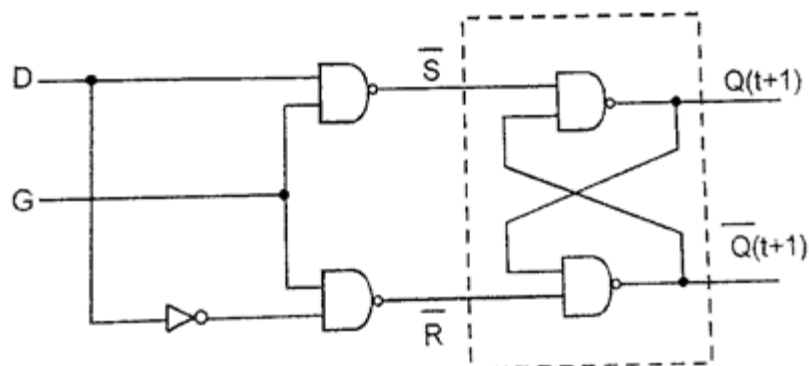


Fig. : Logic diagram using SR latch

Explain in detail about race -free state assignment.

May 2012, Dec. 2014, Dec 2015 2018

- To avoid critical races, it is necessary that present state and next state should be given adjacent assignments.
- If the present state and next state are said to be adjacent, if the binary value differ in only one bit.
- For an example 010 and 011 are adjacent because they differ only in the third bit.
- The binary values 010 and 111 are not adjacent because the first and third bit differs.
- Consider the reduced state table is shown below.

Present State	Next State $A^+ B^+$ Output Z			
	xy=00	xy=01	xy=11	xy=10
S_0	$(S_0), 0$	$(S_0), 0$	$S_2, -$	$S_1, -$
S_1	$S_0, -$	$S_2, -$	$(S_1), 0$	$(S_1), 0$
S_2	$S_0, -$	$(S_2), 1$	$(S_2), 1$	$S_1, -$

- Now if we assign $S_0 = 00$, $S_1 = 01$ and $S_2 = 10$

Present State	Next State $A^+ B^+$ Output Z			
	xy=00	xy=01	xy=11	xy=10
$S_0(00)$	00, 0	00, 0	10, -	01, -
$S_1(01)$	00, -	10, -	01, 0	01, 0
$S_2(10)$	00, -	10, 1	10, 1	01, -

Table: State Assignment without race free

- Here if the present state is $AB = 01$ and input is $xy = 01$, the next state is $A^+ B^+ = 10$.
- Also if the present state is AB and input is 10, the next state $A^+ B^+ = 01$.
- In both cases present state and next state are not adjacent.
- A race free assignment can be obtained if we add an extra row to the flow table.
- The first three rows represent the same conditions as the original three –row table.
- The fourth row is assigned with 11.
- Now the transition of 01 to 10 for input $xy = 01$ must go through 11.
- Also the transition of 10 and 01 for input $xy = 10$ must go through 11.

Present State	Next State A ⁺ B ⁺		Output Z	
	xy=00	xy=01	xy=11	xy=10
S ₀ (00)	00, 0	00, 0	10, -	01, -
S ₁ (01)	00, -	11 , -	01, 0	01, 0
S ₂ (10)	00, -	10, 1	10, 1	11 , -
S ₃ (11)	-, -	10 , -	-, -	01 , -

Table: Race free assignment

- Since the present states and next states differ by single bit, the circuit for this flow table will be free from races.

Incompletely Specified State Machines

- Sequential circuits in which some of the states are left unspecified are called Incompletely Specified State Machines.
- In sequential circuits, not all combinations of states and inputs are possible.
- For example, consider the state table is shown below.

Present State	Next State		Output Z
	X=0	X=1	
a	d, 1	b, 0	
b	-, -	c, 0	
c	a, 1	b, -	
d	a, 0	d, 1	

Table: State Table

- Here the state 'b' will never receive a '0' input and hence the next state and outputs are left unspecified by a dash (-).
- In some situation, the state transitions are completely defined but for some combinations of states and inputs, the output values may be left specified.
- In the state table, the next state of 'c' is specified as 'b' for the input '1' but the output is unspecified as dash.
- When the state transition is unspecified, the future behavior of the sequential machine may become unpredictable.

HAZARDS

Explain in detail about hazards. (May 2012, May 2013, May 2011, Dec 2011, Dec 2012, Dec. 2013, May 2010, May 2009) (Dec 2018)

- **Hazards** are unwanted switching transients that may appear at the output of a circuit because different paths exhibit different propagation delays.
- Hazards occur in combinational circuits, where they may cause a temporary false output value.
- But in asynchronous sequential circuits hazards may result in a transition to a wrong stable state.

Types of Hazards

- ✓ **Static Hazard**
- ✓ **Dynamic Hazard**
- ✓ **Essential Hazard**

Static Hazard

- Static Hazard is a condition which results in *a single momentary incorrect output* due to change in a single input variable when the output is expected to remain in the same state.
- The static hazard may be either static-0 or Static -1.

Hazards in Combinational Circuits

- A hazard is a condition in which a change in a single variable produces a momentary change in output when no change in output should occur.

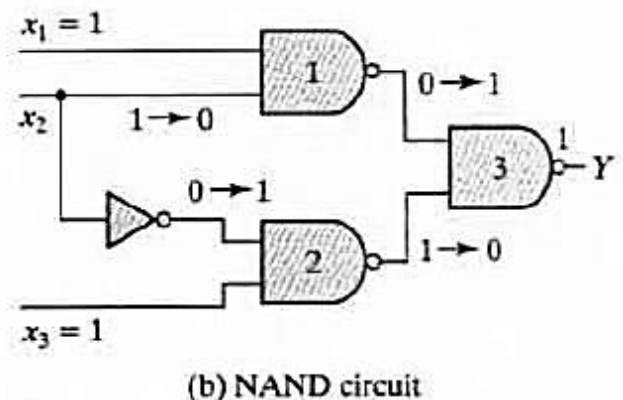
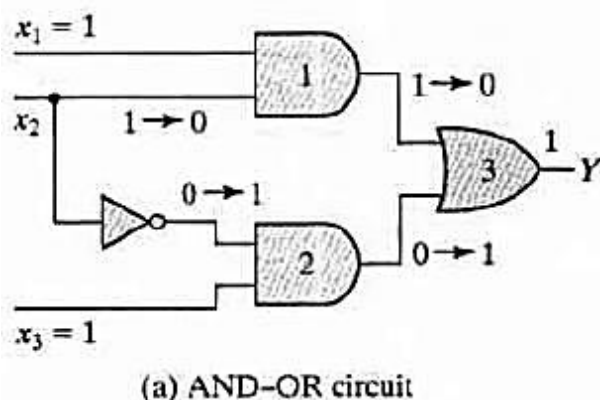


Fig: Circuits with Hazards

- Assume that all three inputs are initially equal to 1.
- This causes the output of gate 1 to be 1, that of gate 2 to be 0 and that of the circuit to be 1.

- Now consider a change in x_2 from 1 to 0.
- Then the output of gate 1 changes to 0 and that of gate 2 changes to 1, leaving the output at 1.
- However, the output may momentarily go to 0 if the propagation delay through the inverter is taken into consideration.
- The delay in the inverter may cause the output of gate 1 to change to 0 before the output of gate 2 changes to 1.
- The two circuits shown in Fig implement the Boolean function in sum-of-products form:

$$Y = x_1x_2 + \bar{x}_2x_3$$

- This type of implementation may cause the output to go to 0 when it should remain a 1.
- If however, the Circuit is implemented instead in product-of-sums form namely,

$$Y = (x_1 + \bar{x}_2)(x_2 + x_3)$$

then the output may momentarily go to 1 when it should remain 0.

- The first case is referred to a static 1-hazard and the second case as static 0-hazard.
- A third type of hazard, known as **dynamic hazard**, causes the output to change three or more times when it should change from 1 to 0 or from 0 to 1.

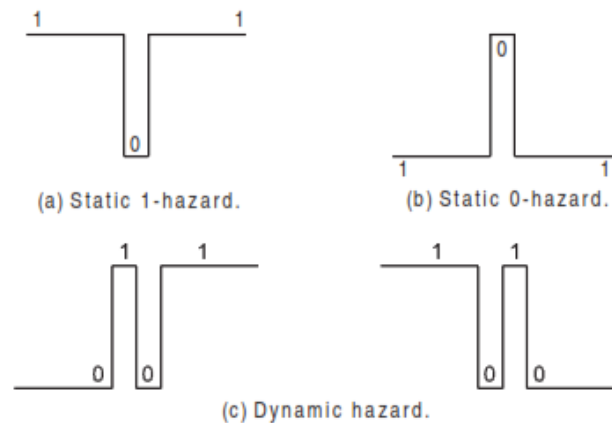


Fig: Types of hazards

- The change in x_2 from 1 to 0 moves the circuit from minterm 111 to minterm 101.
- The hazard exists because the change in input results in a different product term covering the two minterm.

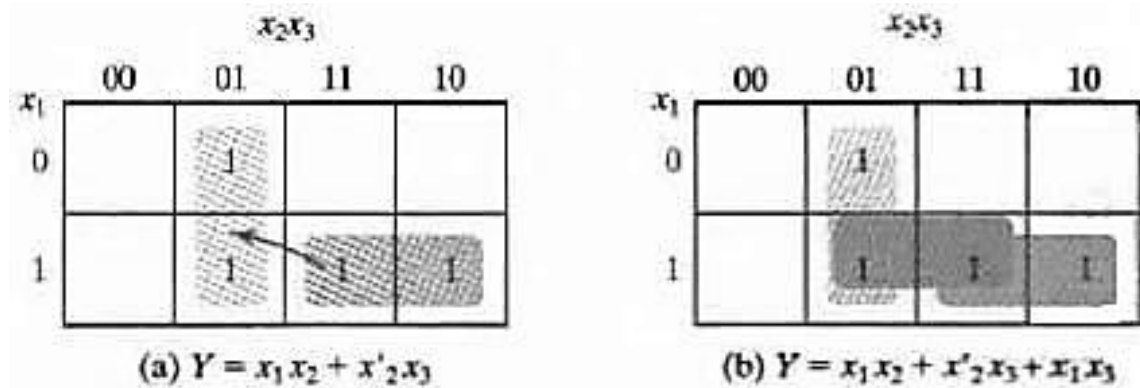
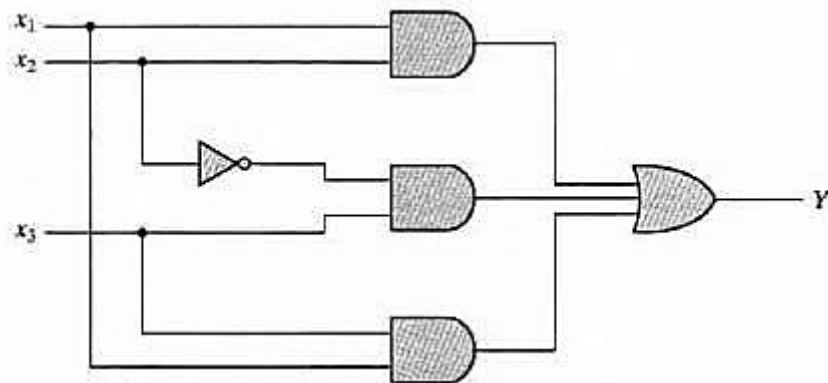
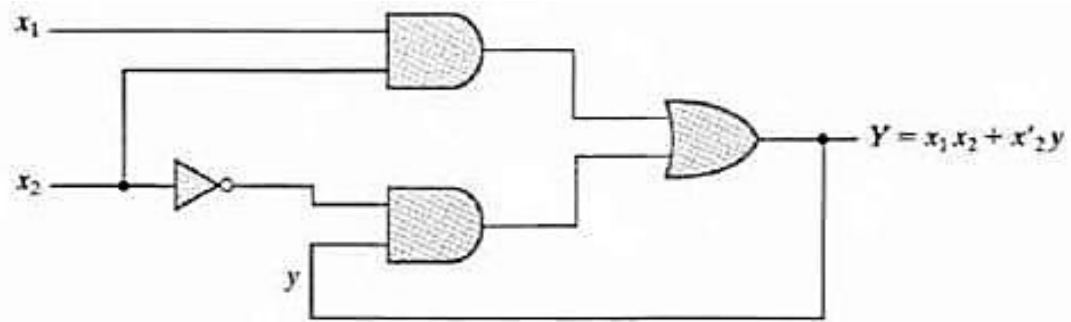


Fig: Illustrates hazard and its removal

- Minterm 111 is covered by the product term implemented in gate 1 and minterm 101 is covered by the product term implemented in gate 2.
- The remedy for eliminating a hazard is to enclose the two minterms with another product term that overlaps both groupings.
- The hazard-free circuit obtained by such a configuration is shown in figure below.
- The extra gate in the circuit generates the product term x_1x_3 .
- In general, hazards in combinational circuits can be removed by covering any two minterms that may produce a hazard with a product term common to both.
- The removal of hazards requires the addition of redundant gates to the circuit.



Hazards in Sequential Circuits:



(a) Logic diagram

	x_1x_2			
	00	01	11	10
y				
0	0	0	1	0
1	1	0	1	1

(b) Transition table

	x_1x_2			
	00	01	11	10
y				
0			1	
1	1		1	1

(c) Map for Y

Fig: Hazard in an Asynchronous sequential circuit

- In normal combinational-circuit design associated with synchronous sequential circuits, hazards are of no concern, since momentary erroneous signals are not generally troublesome.
- However, if a *momentary incorrect signal is fed back in an asynchronous sequential circuit*, it may cause the circuit to go to the wrong stable state.
- If the circuit is in total stable state $yx_1x_2 = 111$ and input x_2 changes from 1 to 0, the next total stable state should be 110.
- However, because of the hazard, output Y may go to 0 momentarily.
- If this false signal feeds back into gate 2 before the output of the inverter goes to 1, the output of gate 2 will remain at 0 and the circuit will switch to the incorrect total stable state 010.
- This malfunction can be eliminated by adding an extra gate.

Essential Hazards

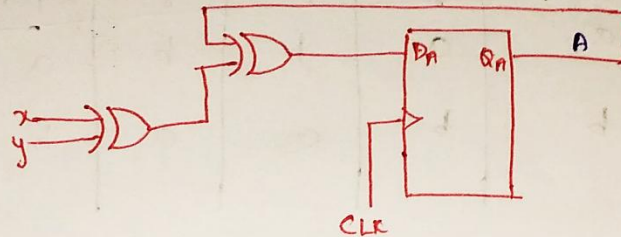
- **Essential** hazard is caused by *unequal delays along two or more paths* that originate from the same input.
- An excessive delay through an inverter circuit in comparison to the delay associated with the feedback path may cause such a hazard.

- Essential hazards cannot be corrected by adding redundant gates as in static hazards.
- The problem that they impose can be corrected by adjusting the amount of delay in the affected path.
- To avoid essential hazards, each feedback loop must be handled with individual care to ensure that the delay in the feedback path is long enough compared with delays of other signals that originate from the input terminals.

ANALYSIS OF SYNCHRONOUS SEQUENTIAL CIRCUITS

- 1) Construct the transition table, state table and state diagram for the sequential circuit.

(Dec 2019 May 2016)



Solution:-

Step 1 :- Determine the Flip-Flop input equation & output equations from the circuit.

$$D_A = (x \oplus y) \oplus A$$

Step 2 :- Plot the transition table

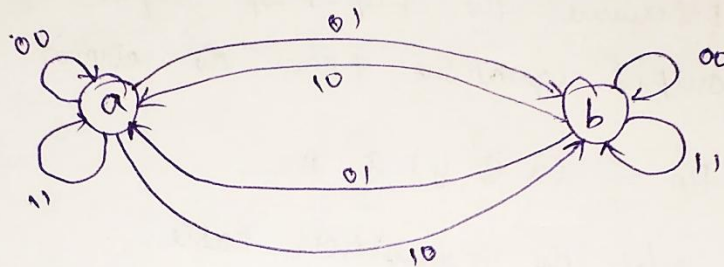
IPS		Present state 'A'	Next State 'A ⁺ '	Flip Flop IPS 'D _A '
x	y			
0	0	0	0	0 ✓
0	0	1	1	1 ✓
0	1	0	1	1 ✓
0	1	1	0	0 ✓
1	0	0	1	1 ✓
1	0	1	0	0 ✓
1	1	0	0	0 ✓
1	1	1	1	1 ✓

(or) arrange PS in A⁺ columns as '0' or '1'

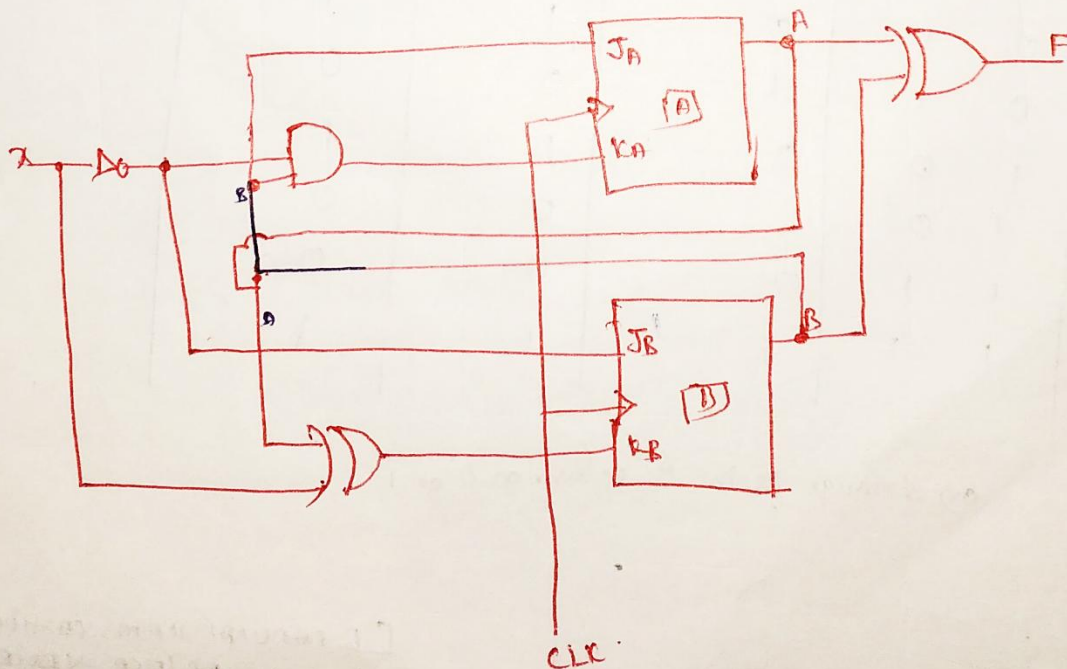
step 2:- Plot the state table, let $a=0$, $b=1$

Present state A	Next state 'A'			
	$xy=00$	$xy=01$	$xy=10$	$xy=11$
a	a	b	b	a
b	b	a	a	b

step 4:- Draw the state diagram.



2) Construct the transition table, state table and state diagram for the moore model sequential circuit.
(May 2019)



Step 1:- Determine the flip flop i/p eq and o/p equation.

$$\begin{aligned} \text{i/p eq :- } J_A &= B & ; & \quad K_A = B\bar{x} \\ J_B &= \bar{x} & ; & \quad K_B = x \oplus A \end{aligned}$$

$$\text{o/p eq :- } F = A \oplus B$$

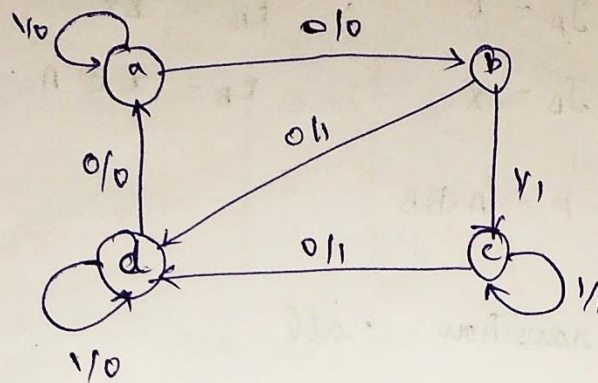
Step 2:- Plot the Transition Table.

Present state		input	Next state		Flip flop i/p's				o/p
A	B	x	A ⁺	B ⁺	J _A	K _A	J _B	K _B	F
0	0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0
0	1	0	1	1	1	1	1	0	1
0	1	1	1	0	1	0	0	1	1
1	0	0	1	1	0	0	1	1	1
1	0	1	1	0	0	0	0	0	1
1	1	0	0	0	1	1	1	1	0
1	1	1	1	1	1	0	0	0	0

Step 3:- Plot the state table:-

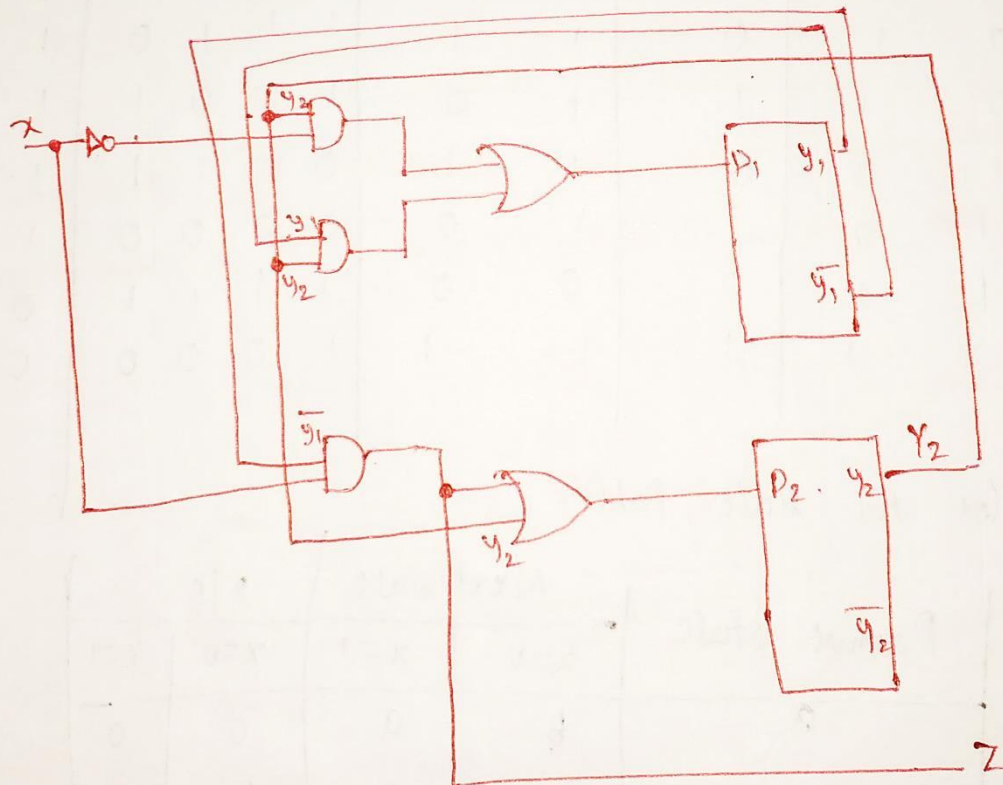
Present state	Next state		o/p	
	x=0	x=1	x=0	x=1
a	b	a	0	0
b	d	c	1	1
c	d	c	1	1
d	a	d	0	0

step 4:- draw the state Diagram.



ANALYSIS OF PULSE MODE SEQUENTIAL CIRCUITS:

1) Explain the design procedure of pulse mode sequential circuits with an example. (Dec 2018)



* The analysis of pulse mode sequential circuit is same as that of fundamental mode sequential circuit.

Step 1:- determine the next state equations and o/p equations

next state eqs:- $Y_1 = D_1 \quad \therefore D_1 = y_2 \bar{x} + y_1 y_2$

$Y_2 = D_2 \quad \therefore D_2 = x \bar{y}_1 + y_2$

o/p equation:-

$Z = x \bar{y}_1$

Step 2:- plot the Truth Table.

i/p x	present states		next states		o/p Z	states
	y ₂	y ₁	Y ₂	Y ₁		
0	0	0	0	0	0	stable
0	0	1	0	0	0	unstable
0	1	0	1	1	0	unstable
0	1	1	1	1	0	stable
1	0	0	1	0	1	unstable
1	0	1	0	0	0	unstable
1	1	0	1	0	1	stable
1	1	1	1	1	0	stable

Step 3:- plot the Transition Table.

i/p x	present state				
	y ₂ y ₁ 00	01	11	10	
0	00	00	11	11	→ unstable states
1	10	00	11	10	

→ stable states

Step 4:- Assign states for the binary values and plot the state table. Let $a=00$, $b=01$, $c=10$, $d=11$.

present state

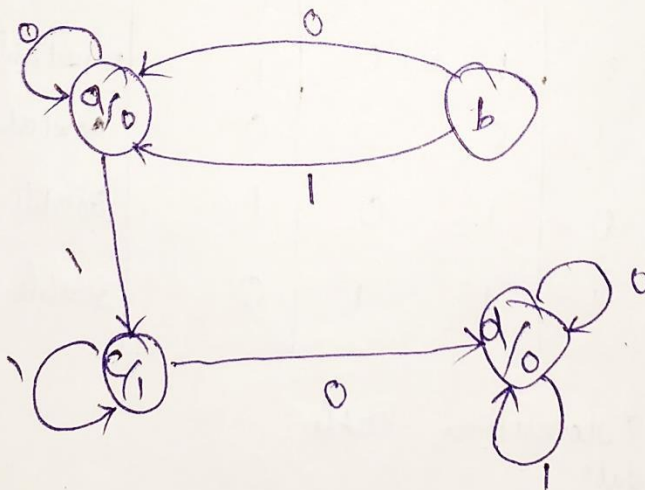
ip x		a	b	d	c
0	→	a	a	d	d
1	→	c	a	d	c

Fig. State table.

Step 5:- Plot the output map.

x		a	b	d	c
0	→	0	-	0	-
1	→	-	-	0	1

Step 6:- Plot the state diagram.



← x →

❖ Design an asynchronous sequential circuit that has two inputs X_2 and X_1 and one output Z . the output is to remain a 0 as long as X_1 is 0. The first change in X_2 that occurs while X_1 is a 1 will cause output Z to be 1. The output Z will remain 1 until X returns to 0. (Dec 2013)(May 2014)

Solution:

[NOV 2020]

Step 1:

States	Inputs		Outputs	Comments
	X_2	X_1	Z	
a	0	0	0	after b or c or f
b	1	0	0	after a or d or e
c	0	1	0	after a
d	1	1	0	after b
e	1	1	1	after c or f
f	0	1	1	after d or e

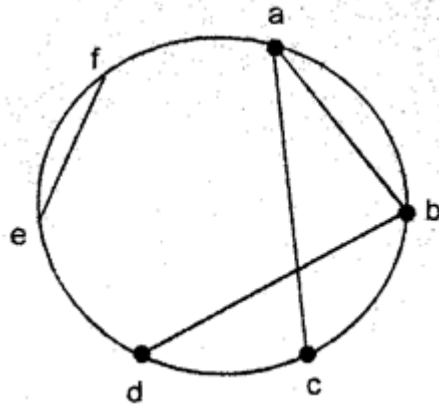
Step 2: Primitive flow table.

		Input $X_2 X_1$			
		00	01	11	10
Present State	a	(a), 0	c, 0	-, -	b, 0
	b	a, 0	-, -	d, 0	(b), 0
	c	a, 0	(c), 0	e, -	-, -
	d	-, -	f, -	(d), 0	b, 0
	e	-, -	f, 1	(e), 1	b, -
	f	a, -	(f), 1	e, 1	-, -

Step 3: A reduced flow table is obtained using implication table and merger diagram.

b	✓				
c	✓	d, e X			
d	c, f X	✓	c, f X d, e X		
e	X	X	X	X	
f	X	X	X	X	✓
	a	b	c	d	e

- The compatible pairs are (a,b) (a,c) (b,d) (e,f)
- The merger diagram is used to find more compatible pairs.



- 4 separate lines are obtained.
- The compatible pairs are again (a,b) (a,c) (b,d) (e,f) .
- If (a,b) compatible pair is removed, then the remaining pairs (a,c) (b,d) (e,f) covers all the 6 states.
- Therefore the reduced flow table is as follows:

States \backslash x_2x_1		00	01	11	10
a, c		$\textcircled{a}, 0$	$\textcircled{c}, 0$	e, -	b, 0
b, d		a, 0	f, -	$\textcircled{d}, 0$	$\textcircled{b}, 0$
e, f		a, -	$\textcircled{f}, 1$	$\textcircled{e}, 1$	b, -

(or)

States \backslash x_2x_1		00	01	11	10
S_0		$\textcircled{S_0}, 0$	$\textcircled{S_0}, 0$	$S_2, -$	$S_1, 0$
S_1		$S_0, 0$	$S_2, -$	$\textcircled{S_1}, 0$	$\textcircled{S_1}, 0$
S_2		$S_0, -$	$\textcircled{S_2}, 1$	$\textcircled{S_2}, 1$	$S_1, -$

Step 4: In order to avoid critical race, one more stable state is added and values are assigned for states.

Present state.			Next State and output for X_2X_1 inputs			
	y_2	y_1	00	01	11	10
S_0	0	0	$\textcircled{S_0}, 0$	$\textcircled{S_0}, 0$	$S_3, -$	$S_1, 0$
S_1	0	1	$S_0, 0$	$S_2, -$	$\textcircled{S_1}, 0$	$\textcircled{S_1}, 0$
S_2	1	1	$S_3, -$	$\textcircled{S_2}, 1$	$\textcircled{S_2}, 1$	$S_1, -$
S_3	1	0	$S_0, -$	$-, -$	$S_2, -$	$-, -$

Step 5: The transition table and output maps are as follows:

Transition Table

x_2x_1 y_2y_1	00	01	11	10
00	00	00	10	01
01	00	11	01	01
11	10	11	11	01
10	00	-	11	-

Output Table Z

x_2x_1 y_2y_1	00	01	11	10
00	0	0	X	0
01	0	X	0	0
11	X	1	1	X
10	X	X	X	X

$$Z = y_2$$

Map for Y_2

x_2x_1 y_2y_1	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	1	1	1	0
10	0	X	1	X

$$Y_2 = \bar{y}_1x_2x_1 + y_1\bar{x}_2x_1 + y_2y_1x_2 + y_2x_1$$

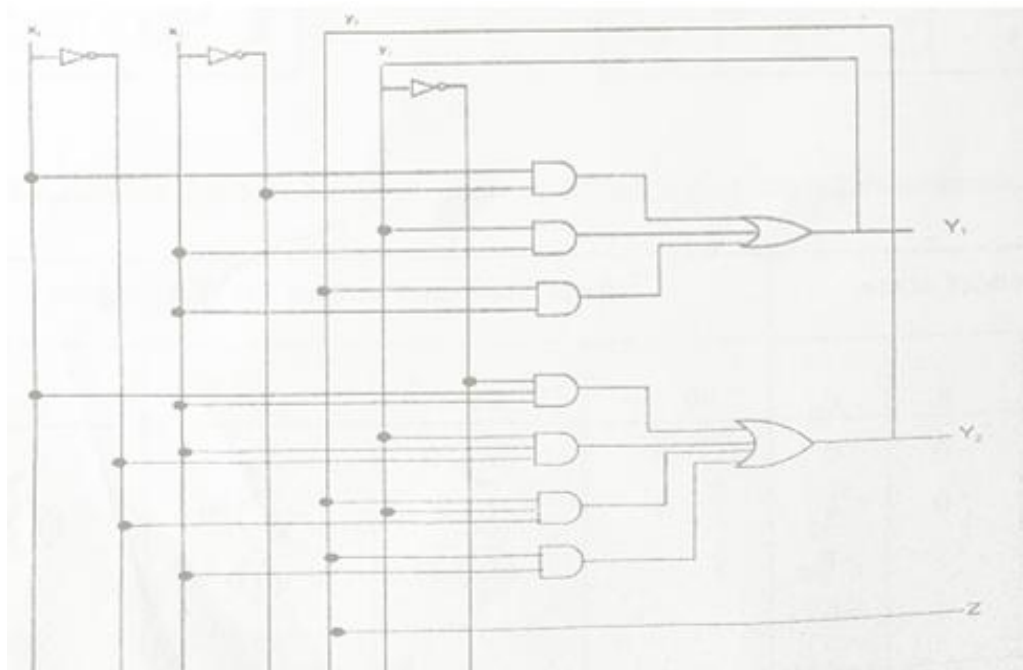
Map for Y_1

x_2x_1 y_2y_1	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	0	1	1	1
10	0	X	1	X

$$Y_1 = x_2\bar{x}_1 + y_1x_1 + y_2x_1$$

Step 6: Logic Diagram

x_1 x_2 y_1 y_2



Y_1

Y_2

Two Marks

UNIT-4

ASYNCHRONOUS SEQUENTIAL CIRCUITS

1. What are classifications of sequential circuits?

The sequential circuits are classified on the basis of timing of their signals in to two types. They are

- Synchronous sequential
- Asynchronous sequential circuits

2. What is synchronous sequential circuit?

(Dec 2013)

Synchronous Sequential circuits are circuits in which the signals can affect the memory elements only at discrete instant of time

3. Define Asynchronous sequential circuit.

In asynchronous sequential circuits change in input signals can affect memory element at any instant of time.

4. Give the comparison between synchronous & Asynchronous sequential circuits. (Dec 2009)

Synchronous sequential circuits	Asynchronous sequential circuits.
<ul style="list-style-type: none">• Memory elements are clocked flip-flops• Easier to design.	<ul style="list-style-type: none">• Memory elements are either unlocked flip - flops or time delay elements.• More difficult to design.

5. What is Moore and Mealy circuit?

(May 2009,Dec 2010,Dec 2007 ,May 2010,Dec 2011, May2011,May2012, Dec 2017)

Moore circuit is a network where the output depends only on the present state of the flip flops.

Mealy circuit is a network where the output depends on both the present state of the flip flops and on the inputs.

6. What are the steps for the designing of asynchronous sequential circuit? (May 2017)

- ✓ Construction of a primitive flow table from the problem statement.
- ✓ Primitive flow table is reduced by eliminating redundant states using the state reduction
- ✓ State assignment is made
- ✓ The primitive flow table is realized using appropriate logic elements.

7. What is merger graph?

It contains the same number of vertices as the state table contains states. A line drawn between the two state vertices indicates each compatible state pair. Two states are incompatible if no connecting line is drawn.

8. What is closed covering?

A Set of compatibles is said to be closed if, for every compatible contained in the set, all its implied compatibles are also contained in the set. A closed set of compatibles, which contains all the states of M, is called a closed covering.

9. What is meant by state table?

(May 2012)

For the design of sequential counters which deals with present states and next states. The table, which represents the relationship between present states and next states, is called state table.

10. Define total state.

The combination of level signals that appear at the inputs and the outputs of the delays is called the total state of the circuit

11. Define primitive flow table.

(May 2019)

It is defined as a flow table which has exactly one stable state for each row in the table. The design process begins with the construction of primitive flow table.

12. What are the types of asynchronous sequential circuits ?

(May 2011)

1. Fundamental mode circuits
2. Pulse mode circuits

13. Give the comparison between state Assignment Synchronous circuit and state assignment asynchronous circuit.

In synchronous circuit, the state assignments are made with the objective of circuit reduction. In asynchronous circuits, the objective of state assignment is to avoid critical races.

14. What is the difference between stable and unstable state in an asynchronous sequential circuit?

If the present state and next state are same then it is said to be stable state.

If the present and next states are different then it is said to be unstable state.

15. What are the problems involved in asynchronous circuits?

The asynchronous sequential circuits have three problems namely

- a. Cycle
- b. Races
- c. Hazards.

16. What are races?

When two or more binary state variables change their value in response to a change in an input variable, race condition occurs in an asynchronous sequential circuit. In case of unequal delays, a race condition may cause the state variables to change in an unpredictable manner.

17. Define non-critical race.**(Dec 2010, Dec 2016)**

If the final stable state that the circuit reaches does not depend on the order in which the state variable changes, the race condition is not harmful and it is called a non-critical race.

18. Define critical race.**(Dec 2010, Dec 2014, Dec 2016, May 2016)**

If the final stable state depends on the order in which the state variable changes, the race condition is harmful and it is called a critical race.

19. What is a cycle?

A cycle occurs when an asynchronous circuit makes transition through a series of unstable states. If a cycle does not contain a stable state, the circuit will go from one unstable to stable to another, until the inputs are changed.

20. What is fundamental mode circuit?

A transition from one stable state to another occurs only in response to a change in the input state. After a change in one input has occurred, no other change in any input occurs until the circuit enters a stable state. Such a mode of operation is referred to as a fundamental mode.

21. Write a short note on pulse mode circuit. [Nov/Dec 2021]

Pulse mode circuit assumes that the input variables are pulses instead of level. The width of the pulses is long enough for the circuit to respond to the input and the pulse width must not be so long that it is still present after the new state is reached.

22. Compare Fundamental mode and Pulse mode circuit.**(Dec 2012)**

Fundamental mode	Pulse mode circuit
Inputs are levels and not pulses.	Input variables are pulses instead of levels
Input variables changes only when the circuit is in stable	The width of the pulses is long enough for the circuit to respond to the input.
Only one input variable change at a given time.	The pulse width must not be long that it is still present after the new state is reached.

23. Define secondary variables

The delay elements provide a short term memory for the sequential circuit. The present state and next state variables in asynchronous sequential circuits are called secondary variables.

24. Define flow table in asynchronous sequential circuit.

In asynchronous sequential circuit state table is known as flow table because of the behavior of the asynchronous sequential circuit. The stage changes occur in independent of a clock, based on the logic propagation delay, and cause the states to flow from one to another.

25. What are the different techniques used in state assignment?

(May 2016)

- Shared row state assignment
- One hot state assignment

26. What is the significance of state assignment?

(May 2017)

In synchronous circuits-state assignments are made with the objective of circuit reduction. For Asynchronous circuits- its objective is to avoid critical races.

27. Define Race condition. How can it be avoided?

(Dec 2011)

If an input changes in two or more feedback variables then it is known as race condition. Races can be avoided by using a race free state assignment while designing the circuit.

28. Write short note on shared row state assignment.

Only one state variable can change at any one state variable can change at any one time when a state transition occurs. To accomplish this, it is necessary that states between which transitions occur be given adjacent assignments.

29. Write short note on one hot state assignment.

Only one variable is active or hot for each row in the original flow table, i.e., it requires one state variable for each row of the flow table. Additional row are introduced to provide single variable changes between internal state transitions.

[NOV 2020] (May 2016)

30. What is hazard? (Dec 2009,May2010,Dec 2013, May 2013, Dec 2015, Dec 2016&2017)

Unwanted switching transients that may occur in an output of combinational circuit is Called hazards

31. What happens when a hazard in a logic circuit?

In Combinational circuits, hazards may cause temporary false output. But in sequential circuits hazards may cause a transition to a wrong state.

32. What is static 1 hazard?

[NOV 2020]

(May 2009,Dec 2015, Dec 2013)

The output goes momentarily 0 when it should remain at 1 is called static 1 hazard

33. What is static 0 hazard?

(Dec 2013)(May 2009)

The output goes momentarily 1 when it should remain at 0 is called static 0 hazard.

34. What is dynamic hazard?

(May 2018) (Dec 2019)

The output changes 3 or more times when it changes from 1 to 0 or 0 to 1 is called dynamic hazard.

35. What is the cause for essential hazards?

Unequal delays along 2 or more path from same input cause essential hazards.

36. What are the basic elements of ASM chart?

(Dec 2014, May 2011)

An ASM chart mainly consists of three elements:

State box - one box per system state

Decision box - Basic condition, i.e. logic flow control. Only the decision boxes depend on inputs.

Conditional box- An action/operation to be undertaken conditioned on some earlier decision box.

37. What is Algorithmic State Machine (ASM)?

(Dec 2014) (May 2013) (Dec 2015)

- ✓ An ASM is a flow chart representation that describes the behavior of a sequential circuit.
- ✓ An ASM chart differs from a conventional flow chart, a conventional flow chart describes the procedure steps and decision path for an algorithm without concern of their time relationship.
- ✓ An ASM chart also describes the timing relationship between the states and outputs.

38. What is state diagram?

(May 2014)

It is the graphical representation of state table. In state diagram, a state is represented by a circle, and the transitions between the states are indicated by directed lines connecting circles.

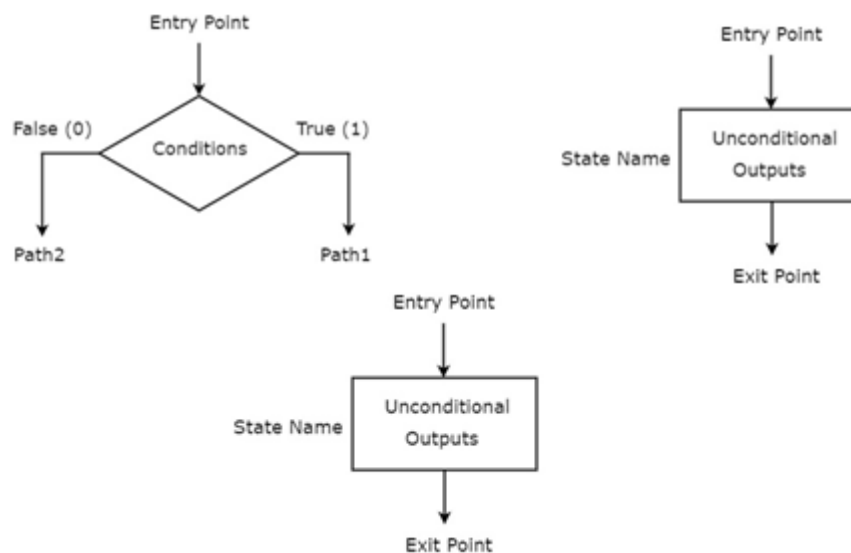
39. What are hazard free digital circuits?

[April/May-2010]

A circuit which has no hazard like static-0-hazard and static-1-hazard is called hazard free digital circuit.

40. Draw the general model of ASM.

((Dec 2018)



Design a negative edge triggered T flip-flop. The circuit has 2 inputs, T (toggle) and C (clock) and outputs Q and \overline{Q} . The output state is complemented if T = 1 and the clock changes from 1 to 0. Otherwise under any other input condition, the output Q remains unchanged.

Solution :

Step 1: A table with all possible total states in the system is formed. A primitive flow table is obtained from this table.

States	Inputs		Outputs Q	Comments
	T	C		
a	1	1	0	initial output is 0
b	1	0	1	after state a
c	1	1	1	after output is 1
d	1	0	0	after state c
e	0	0	0	after state d or f
f	0	1	0	after state e or a
g	0	0	1	after state b or h
h	0	1	1	after state g or c

Primitive Flow Table

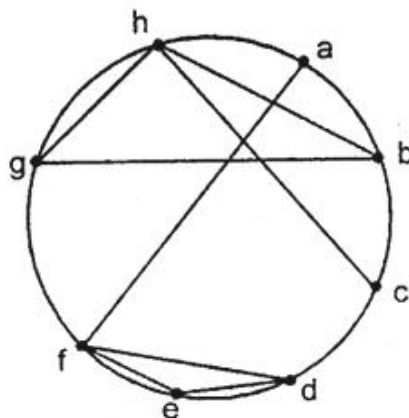
		TC			
		00	01	11	10
Present State	a	-, -	f, -	Ⓐ, 0	b, -
	b	g, -	-, -	c, -	Ⓑ, 1
	c	-, -	h, -	Ⓒ, 1	d, -
	d	e, -	-, -	a, -	Ⓓ, 0
	e	Ⓔ, 0	f, -	-, -	d, -
	f	e, -	Ⓘ, 0	a, -	-, -
	g	Ⓕ, 1	h, -	-, -	b, -
	h	g, -	Ⓖ, 1	c, -	-, -

Step 2: Reduced Flow Table is obtained using implication table and merger diagram.

b	a,c X						
c	X	b,d X					
d	b,d X	X	a,c X				
e	b,d X	e,g X b,d X	f,h X	✓			
f	✓	e,g X a,c X	f,h X a,c X	✓	✓		
g	f,h X	✓	b,d X	e,g X b,d X	X	e,g X f,h X	
h	f,h X a,c X	✓	✓	e,g X a,c X	e,g X f,h X	X	✓
	a	b	c	d	e	f	g

The compatible pairs are (a, f) (b, g) (b, h) (c, h) (e, f) (g, h) (d, e) (d, f).

Merger Diagram



The maximal compatibles are (a, f) (c, h) (b, g, h) (d, e, f).

The reduced flow table is given below.

Present State	TC	00	01	11	10
	a,f	e, -	(f), 0	(a), 0	b -
	c,h	g, -	(h), 1	(c), 1	d -
	b,g,h	(g), 1	(h), 1	c, -	(b), 1
	d,e,f	(e), 0	(f), 0	a, -	(d), 0

Present State	TC	00	01	11	10
	a	d, -	(a), 0	(a), 0	c -
	b	c, -	(b), 1	(b), 1	d -
	c	(c), 1	(c), 1	b, -	(c), 1
	d	(d), 0	(d), 0	a, -	(d), 0

Step 3: A race free binary state assignment is made. Transition table and output map is obtained.

Transition Table

$y_1 y_2$	TC			
	00	01	11	10
00	10	(00)	(00)	01
01	(01)	(01)	11	(01)
11	01	(11)	(11)	10
10	(10)	(10)	00	(10)

Output Map

$y_1 y_2$	TC			
	00	01	11	10
00	X	0	0	X
01	1	1	X	1
11	X	1	1	X
10	0	0	X	0

$$Q = y_2$$

Step 4: Logic diagram.

Map for Y_1

y_1	TC			
	00	01	11	10
0	1	0	0	0
0	0	0	1	0
1	0	1	1	1
1	1	1	0	1

Map for Y_2

y_2	TC			
	00	01	11	10
0	0	0	0	1
1	1	1	1	1
1	1	1	1	0
0	0	0	0	0

Use SR latches.

The excitation table for SR latch is

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

y ₁ y ₂	TC			
	00	01	11	10
00	1	0	0	0
01	0	0	1	0
11	0	X	X	X
10	X	X	0	X

$$S_1 = \bar{y}_2 \bar{T} \bar{C} + y_2 TC$$

y ₁ y ₂	TC			
	00	01	11	10
00	0	X	X	X
01	X	X	0	X
11	1	0	0	0
10	0	0	1	0

$$R_1 = y_2 \bar{T} \bar{C} + \bar{y}_2 TC$$

y ₁ y ₂	TC			
	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	0
10	0	0	0	0

$$S_2 = \bar{y}_1 T \bar{C}$$

y ₁ y ₂	TC			
	00	01	11	10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

$$R_2 = y_1 T \bar{C}$$

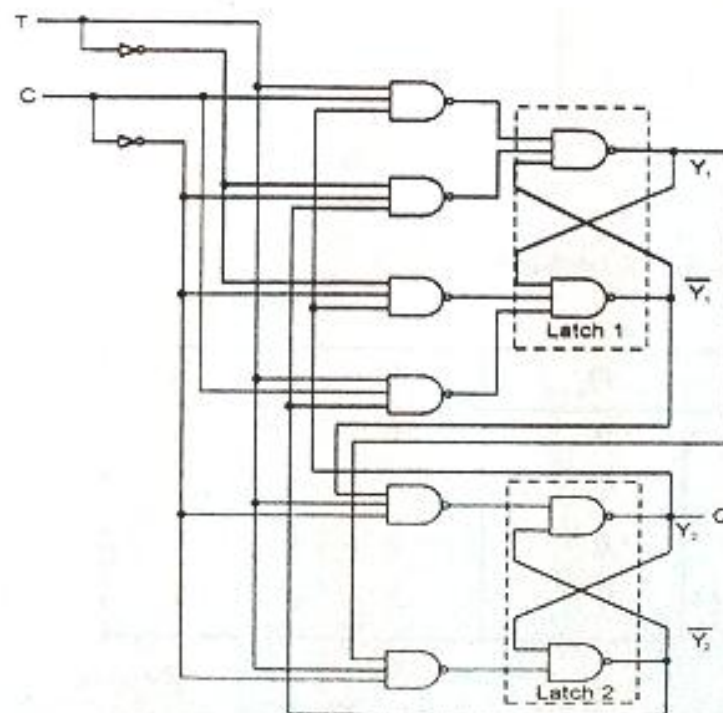


Fig. 4.17: Logic diagram of Example 4.3

Example:

Design an asynchronous sequential circuit with inputs x_1 and x_2 and one output z . Initially and at any time if both the inputs are 0, output is equal to 0. When x_1 or x_2 becomes 1, z becomes 1. When second input also becomes 1, $z = 0$; The output stays at 0 until circuit goes back to initial state.

Ans: Step 1:

States	Inputs		Outputs Z	Comments
	X_2	X_1		
a	0	0	0	after b or c
b	0	1	1	after a
c	1	0	1	after a
d	1	1	0	after b or c
e	1	0	0	after d
f	0	1	0	after d

Step 2: Primitive flow table.

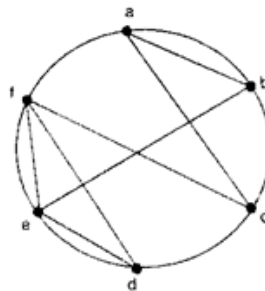
	00	01	11	10
a	$\textcircled{a}0$	b--	--	c--
b	a--	$\textcircled{b}1$	d--	--
c	a--	--	d--	$\textcircled{c}1$
d	--	f--	$\textcircled{d}0$	e--
e	a--	--	d--	$\textcircled{e}0$
f	a--	$\textcircled{f}0$	d--	--

Step 3: A reduced flow table is obtained using Implication table and merger diagram.

	a	b	c	d	e
b	✓				
c	✓	✓			
d	b, fX c, eX	b, f X	c, e X		
e	c, e X	✓	X	✓	
f	b, f X	X	✓	✓	✓

The compatible pairs are (a, b) (a, c) (b, c) (b, e) (c, f) (d, e) (d, f) (e, f).

Merger Diagram:



The Maximal Compatibles are (a, b, c) (d, e, f) (c, f) (b, e) (c, f) and (b, e) can be removed. Since the remaining terms themselves cover all six states.

	X_2X_1			
States	00	01	11	10
a, b, c	$\textcircled{a}0$	$\textcircled{b}1$	d,--	$\textcircled{c}1$
d, e, f	a,--	$\textcircled{f}0$	$\textcircled{d}0$	$\textcircled{e}0$

Step 4: Stable assignment.

$$a = b = c = 0 \quad d = e = f = 1.$$

x_2x_1	00	01	11	10
y				
0	0	0	1	0
1	0	1	1	1

Fig. : Transition Table

x_2x_1	00	01	11	10
y				
0	0	1	X	1
1	X	0	0	0

Fig. : Output Map

$$Y = X_2X_1 + yX_1 + yX_2 \quad Z = \bar{y}X_1 + \bar{y}X_2.$$

Step 5: Logic diagram.

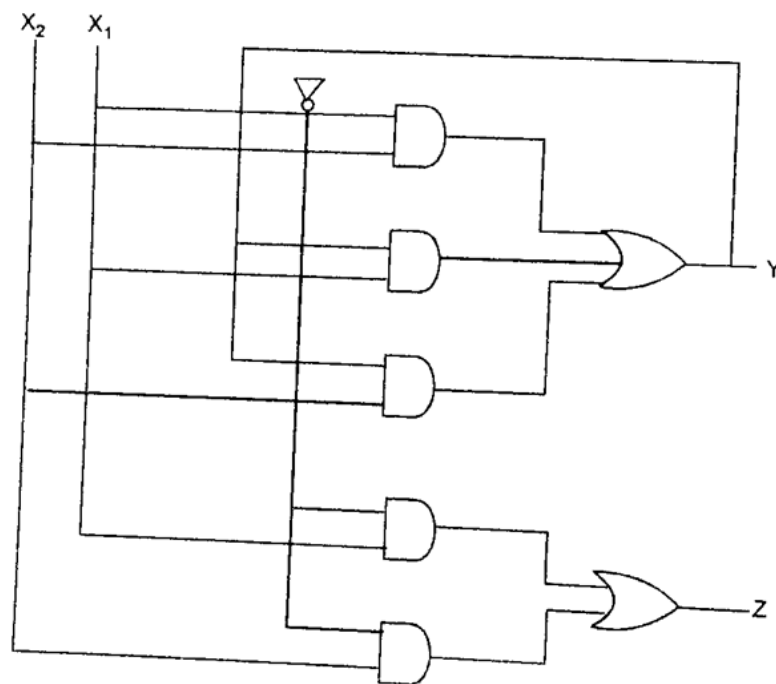


Fig.

Problems on hazards:

Example

Give hazard free realization for the following Boolean functions.

$$f(A, B, C, D) = \sum m(1, 3, 6, 7, 13, 15).$$

✍ Solution :

The simplified expression for the given function can be obtained using k-map.

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	0	1	1
11	0	1	1	0
10	0	0	0	0

The simplified expression is,

$$f(A, B, C, D) = \overline{A}\overline{B}D + \overline{A}BC + ABD$$

But to remove hazards, we have to include another two terms, which is shown in dotted lines in map.

$$f(A, B, C, D) = \overline{A}\overline{B}D + \overline{A}BC + ABD + \overline{A}CD + BCD$$

The logic diagram is as shown in Figure 4.38.

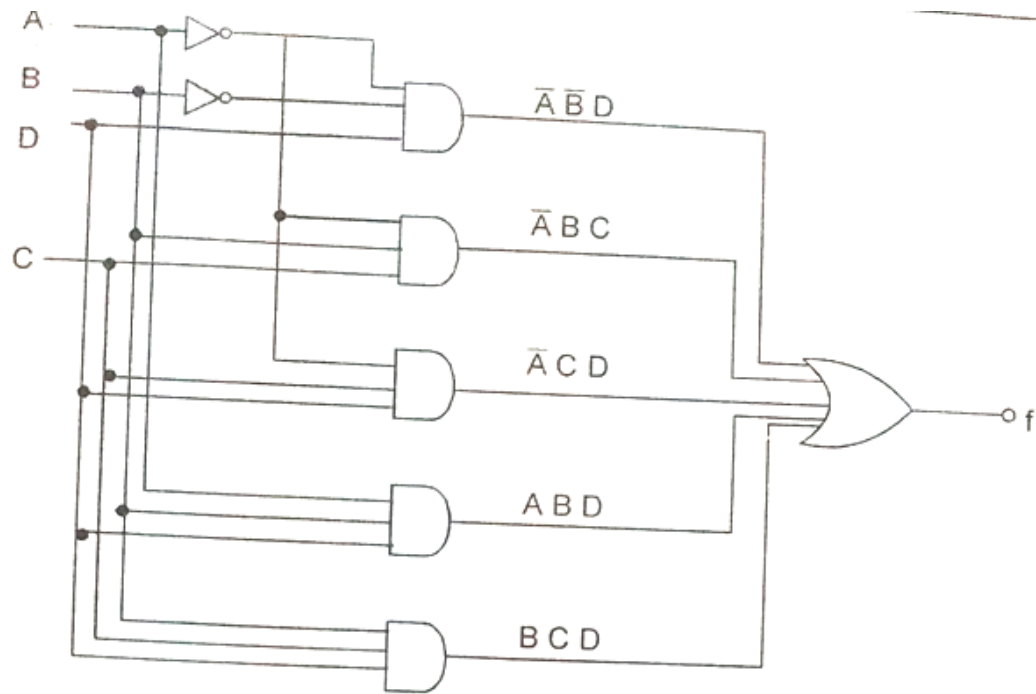


Fig. 4.38 : Logic diagram of Example 4.14

Example

Give hazard free realization for the following Boolean functions.

May 2017

$$F(A, B, C, D) = \sum m(0, 1, 5, 6, 7, 9, 11).$$

Solution :

The simplified expression for the given function can be obtained using K-map.

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	1	1	0	0
$\bar{A}B$ 01	0	1	1	1
AB 11	0	0	0	0
$A\bar{B}$ 10	0	1	1	0

The simplified expression is,

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}BD + \bar{A}BC + A\bar{B}D$$

But to remove hazards, we have to include two more terms, that are shown as dotted lines in K-map.

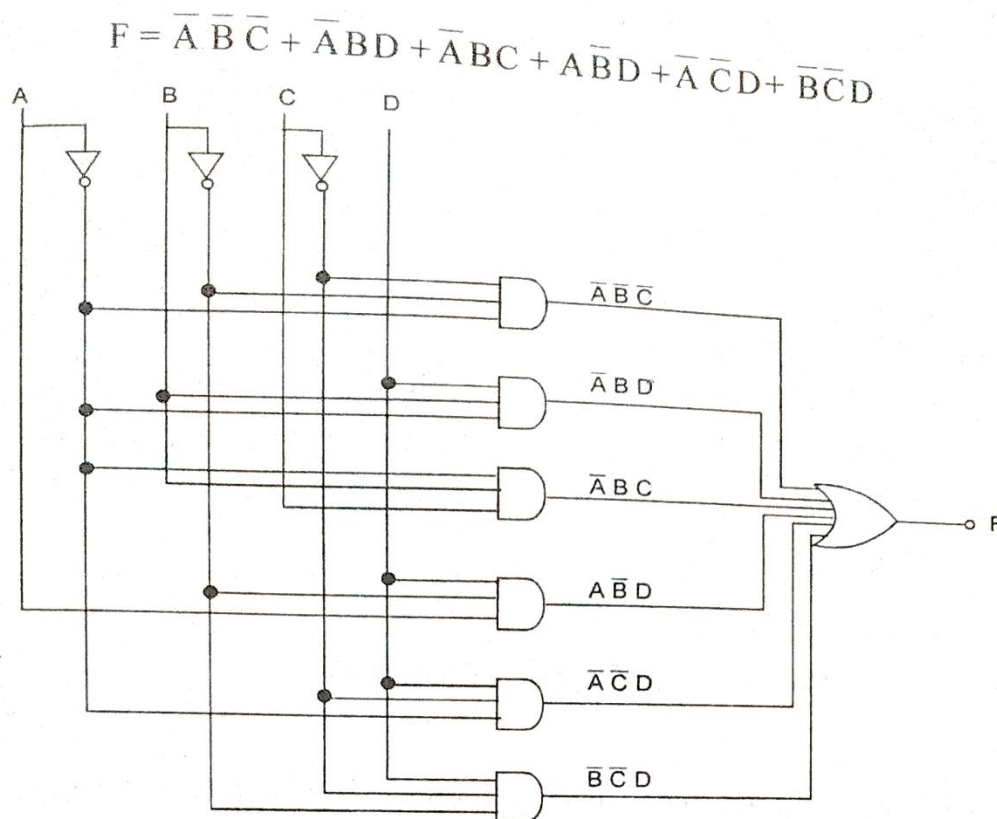


Fig. 4.39 : Logic diagram of Example 4.15

Example .

Give hazard free realization for the following Boolean function

$$F(I, J, K, L) = \sum m(1, 3, 4, 5, 6, 7, 9, 11, 15).$$

✎ Solution :

The simplified expression for the given function can be obtained using K-map.

IJ \ KL	KL			
	$\bar{K}\bar{L}$ 00	$\bar{K}L$ 01	KL 11	$K\bar{L}$ 10
$\bar{I}\bar{J}$ 00	0	1	1	0
$\bar{I}J$ 01	1	1	1	1
$I\bar{J}$ 11	0	0	1	0
IJ 10	0	1	1	0

The simplified expression is,

$$F = \bar{I}\bar{J} + KL + \bar{J}L$$

But to remove hazards, one more redundant quad have to be included, which is shown as dotted lines in K-map.

$$F = \bar{I}\bar{J} + KL + \bar{J}L + \bar{I}L$$

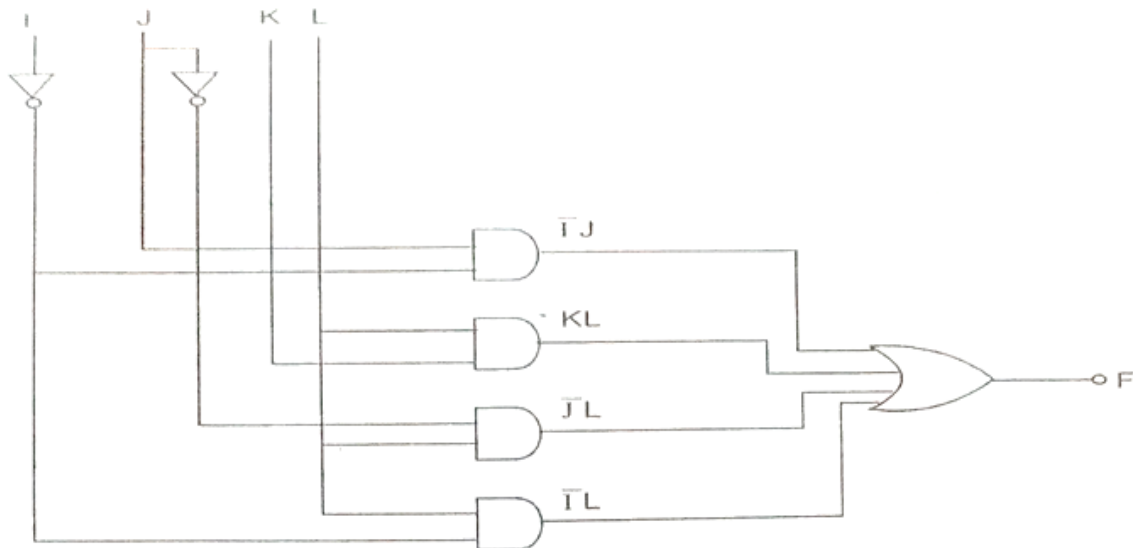


Fig. 4.40 : Logic Diagram of Example 4.16

Example 4.17

Find a static and dynamic hazard free realization for the following function using

- (i) NAND gates. (ii) NOR gates.

$$f(a, b, c, d) = \sum m(1, 5, 7, 14, 15).$$

Solution :

The simplified expression for the given function can be obtained using K-map.

		cd			
		00	01	11	10
ab	00	0	1	0	0
	01	0	1	1	0
	11	0	0	1	1
	10	0	0	0	0

The simplified expression is,

$$f = \bar{a}\bar{c}d + \bar{a}bd + abc$$

But to remove static and dynamic hazards one more term has to be included, which is shown as dotted lines in K-map.

$$f = \bar{a}\bar{c}d + \bar{a}bd + abc + bcd$$

This can be implemented using NAND gates as follows.

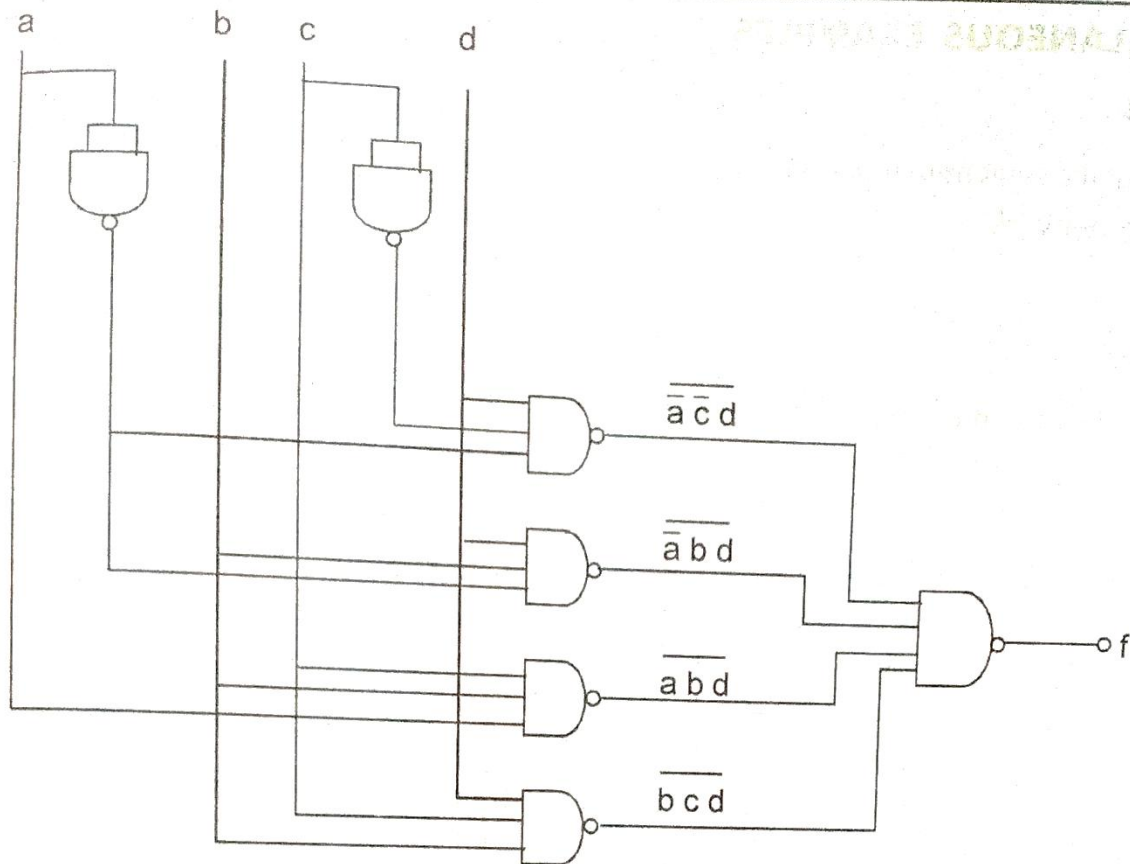


Fig. 4.41: Hazard free circuit using NAND gates

The above expression can also be implemented using NOR gates, as shown in Figure 4.40.

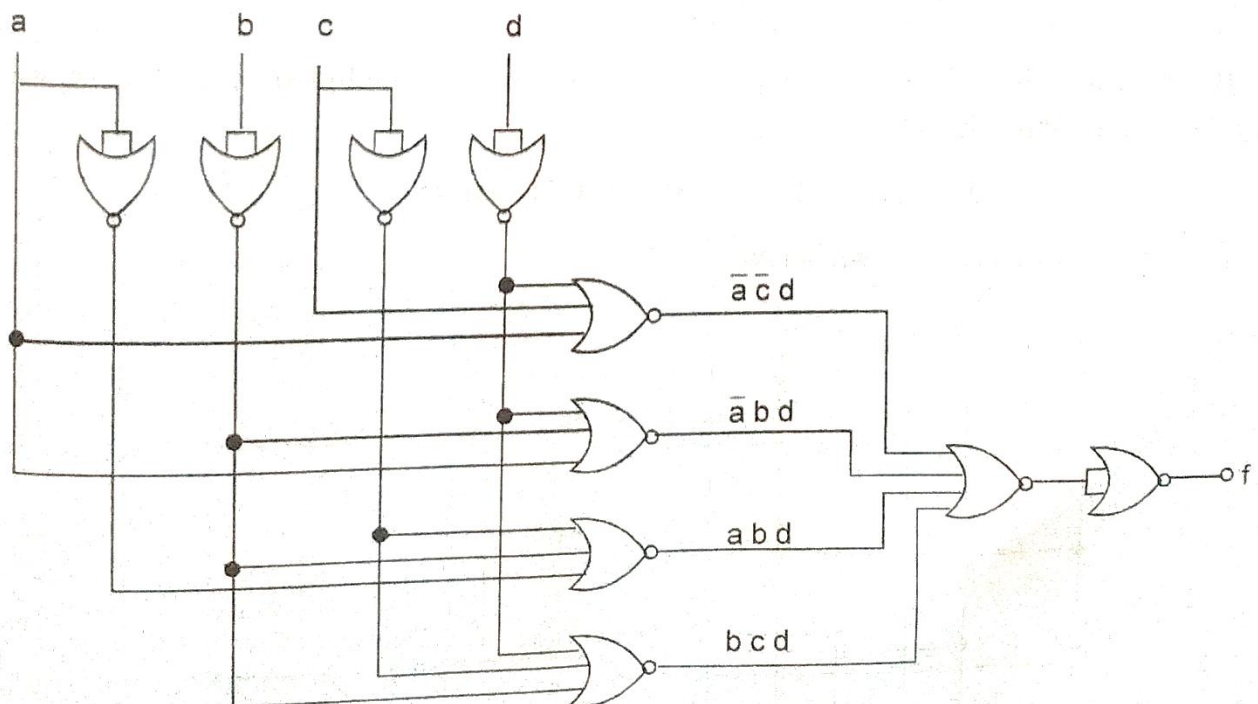


Fig. 4.42: Hazard free circuit using NOR gates

Problem:

A staircase light is controlled by two switches one at the top of the stairs and another at the bottom of stairs a. Make a truth table for this system. (May 2018)

b. Write the logic equation in SOP form.

c. Realize the circuit using AND-OR gates.

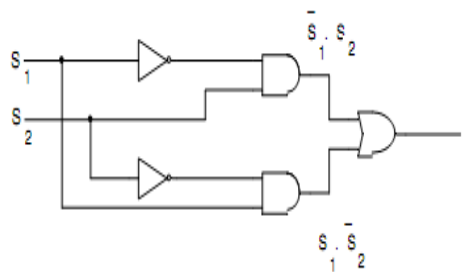
Solution:

a. The truth table for the system is illustrated in given truth table

A	B	Light
0	0	0
0	1	1
1	0	1
1	1	0

b. The logic equation for given system is specified by $L = A' B + A B'$

c. Realization of given case, the circuit using AND-OR gates is demonstrated in fig



* Differentiate dynamic hazard and static hazard. [NOV/DEC 2021]

° **Static 1-hazard**

- Input change causes output to go from 1 to 0 to 1



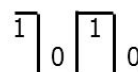
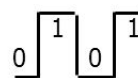
° **Static 0-hazard**

- Input change causes output to go from 0 to 1 to 0



° **Dynamic hazards**

- Input change causes a double change from 0 to 1 to 0 to 1 OR from 1 to 0 to 1 to 0



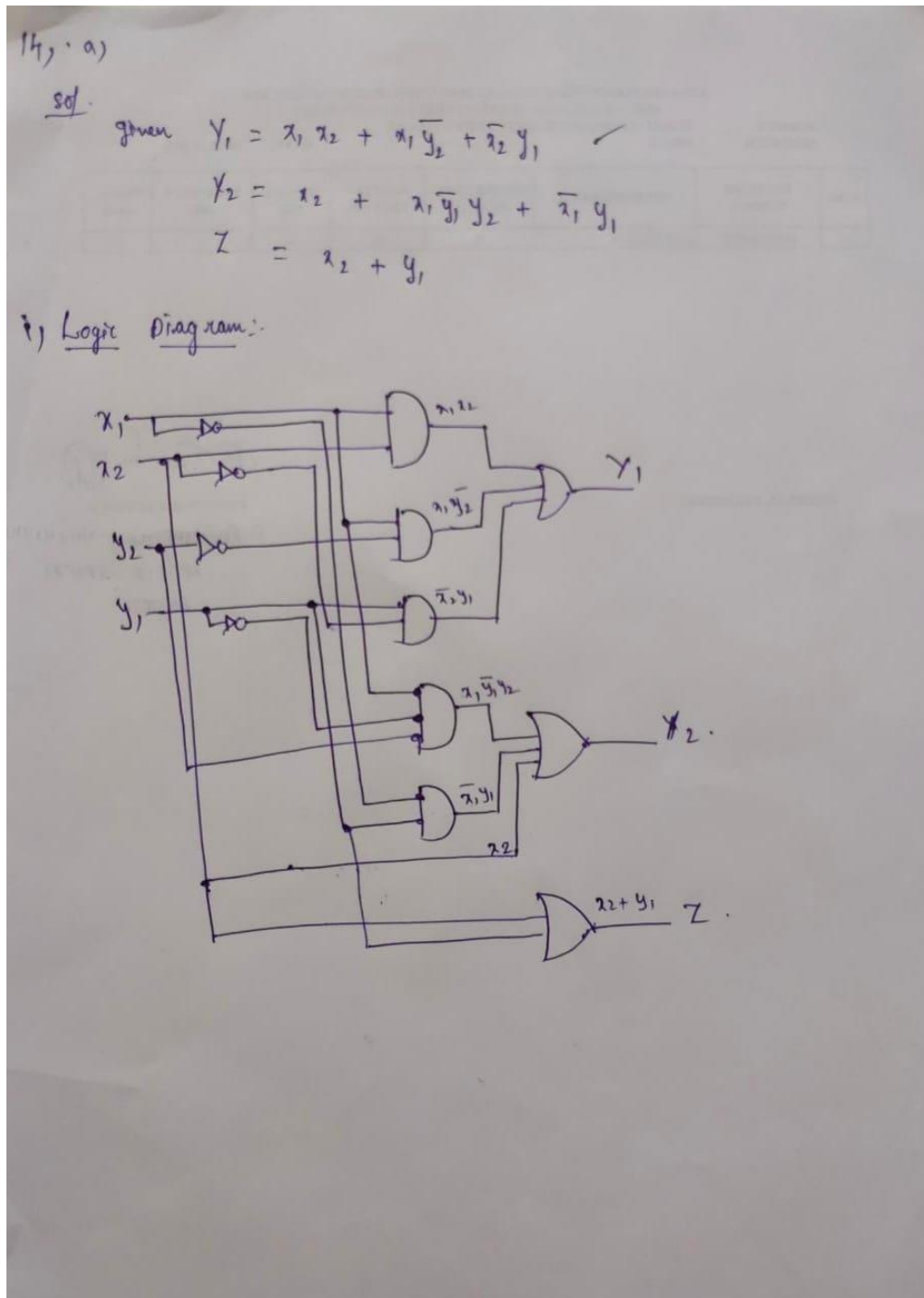
An asynchronous sequential circuit has two internal states and one output. The two excitation functions and one output function describing the circuit are, respectively.

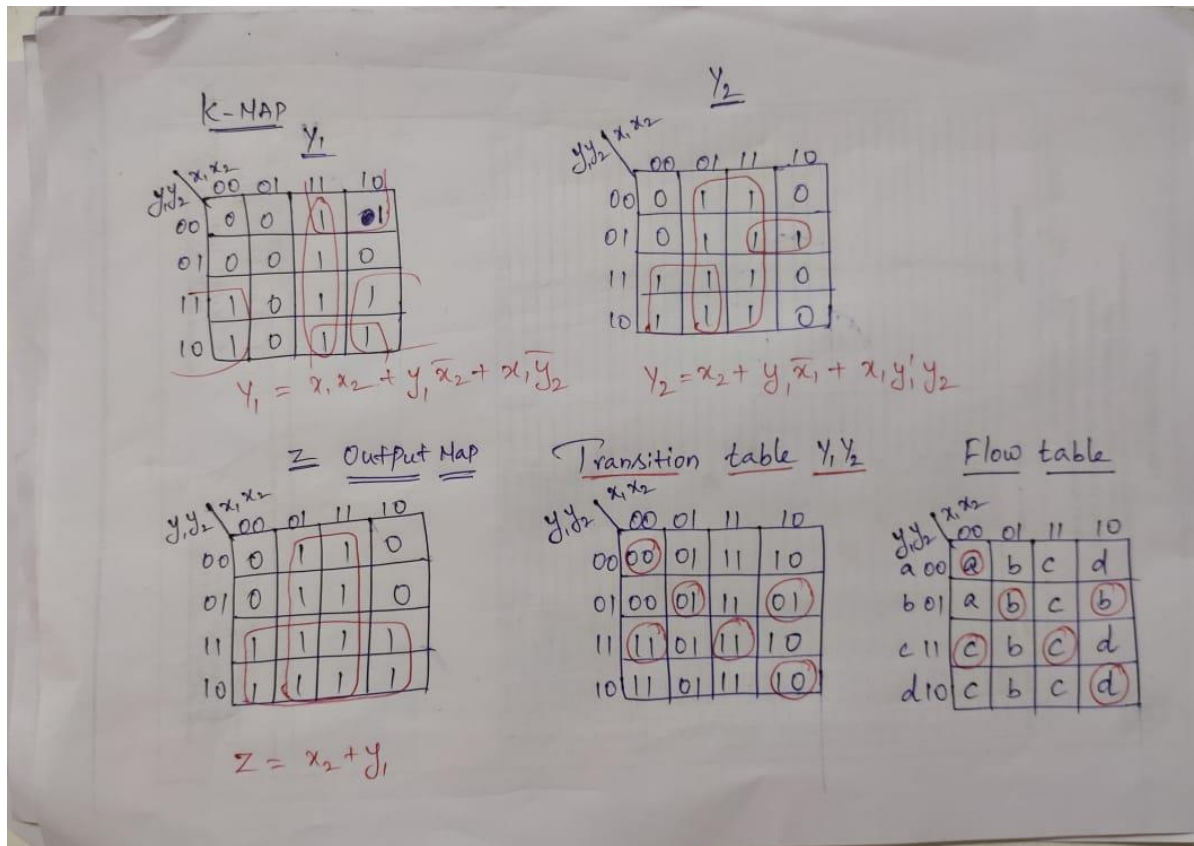
$$Y_1 = x_1x_2 + x_1y_2' + x_2'y_1$$

$$Y_2 = x_2 + x_1y_1'y_2 + x_1'y_1$$

$$Z = x_2 + y_1$$

Draw the logic diagram of the circuit. Obtain the transition table, flow table and output map for the circuit.
[NOV / DEC 2021]





Obtain the binary state assignment for the reduced flow table shown in Fig. 1. Avoid critical race conditions. Also draw the logic diagram of the circuit using NAND latches and gates.
[NOV / DEC 2021]

	x_1, x_2			
	00	01	11	10
a	a, 0	a, 1	b, -	d, -
b	a, -	b, 0	b, 0	c, -
c	a, -	-, -	d, -	c, 0
d	a, -	a, -	a, 1	a, 1

Binary state assignment

$$a = 00$$

$$b = 01$$

$$c = 11$$

$$d = 10$$

Transition table

y_1, y_2	x_1, x_2			
	00	01	11	10
00	00	00	01	10
01	00	01	01	11
11	00	-	10	11
10	00	00	10	10

y_1, y_2	x_1, x_2			
	00	01	11	10
00	0	1	X	X
01	X	0	0	X
11	X	X	X	0
10	X	X	1	1

Reduced flow table

y_1	x_1, x_2			
y_1, y_2	00	01	11	10
00	0	0	0	1
01	0	0	0	2
11	0	X	1	1
10	0	0	1	1

$$Y_1 = y_1 x_1 + x_1 \bar{x}_2$$

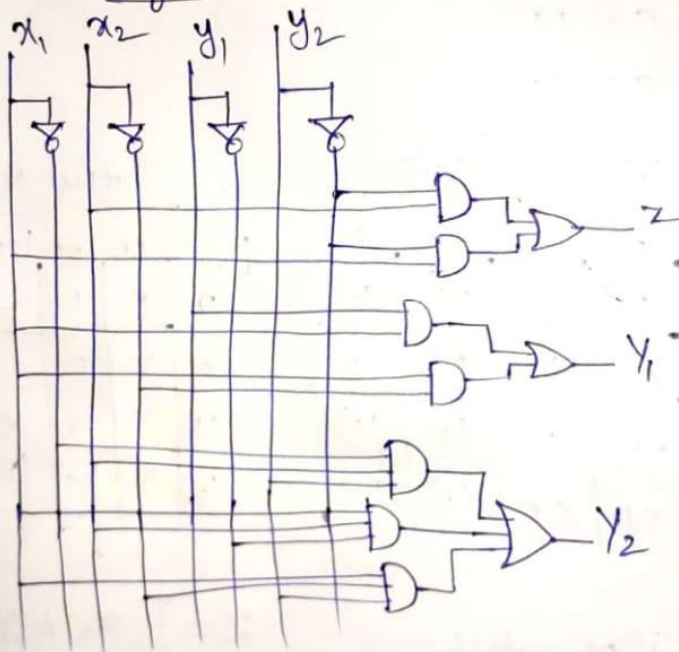
$$Z = \bar{y}_2 x_2 + \bar{y}_2 x_1$$

y_2

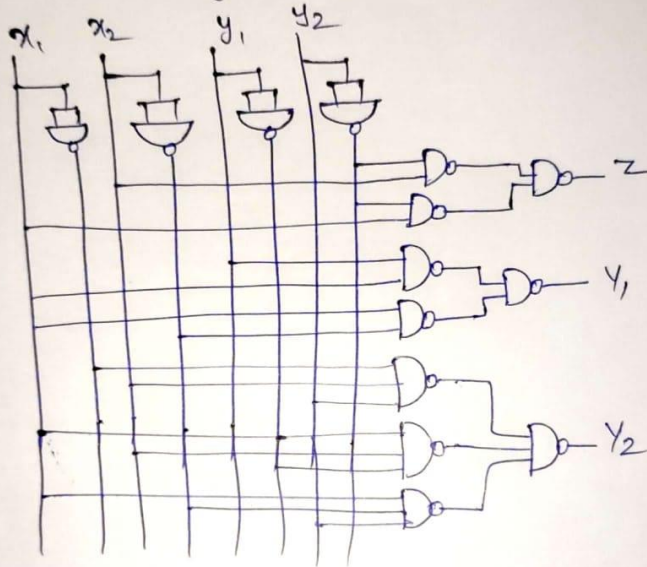
$y_2 \backslash x_1 x_2$	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	0	X	0	1
10	0	0	0	0

$$y_2 = \bar{x}_1 x_2 y_2 + x_1 x_2 \bar{y}_1 + x_1 \bar{x}_2 y_2$$

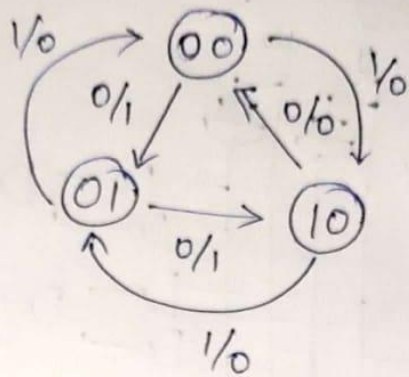
Logic diagram



Replace all gates by NAND



Design an asynchronous circuit using positive edge triggered JK flip-flops with minimal combinational gating to generate the following sequence 0-1-2-0; if input $X = 0$ and 0-2-1-0; if input $X = 1$, provide an output which goes high to indicate the non-zero state in the 0-1-2-0 sequence. Is this a mealy machine? [NOV / DEC 2021]



Excitation table

Q_A	Q_A^+	J	k
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

State table & Excitation table

Present State $Q_A Q_B$	Input x	Next State $Q_A^+ Q_B^+$	J_A	K_A	J_B	K_B	Output z
00	0	01	0	x	1	x	1
00	1	10	1	x	0	x	0
01	0	10	1	x	x	1	1
01	1	00	0	x	x	1	0
10	0	00	x	1	0	x	0
10	1	01	x	1	1	x	0

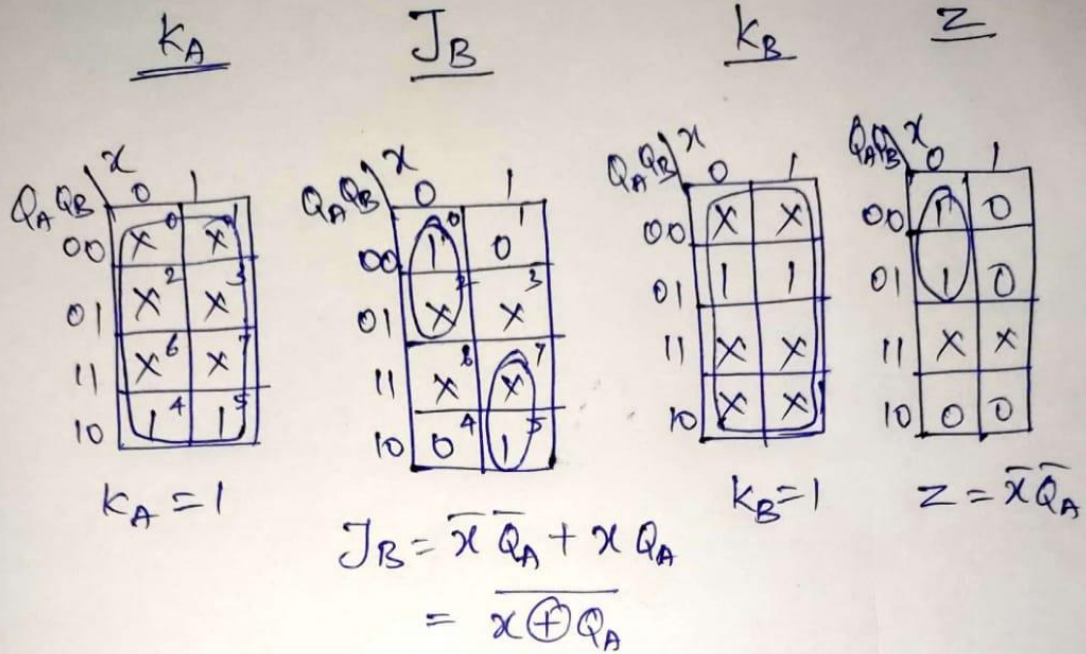
K-Map

J_A

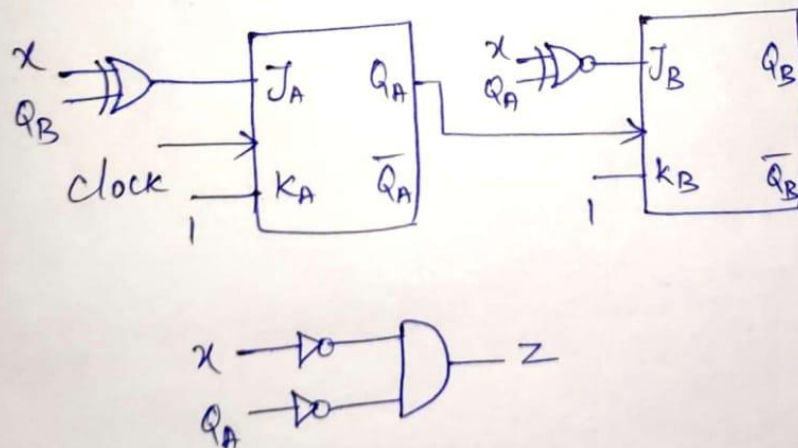
$x \backslash Q_A Q_B$	00	01	11	10
0	0	1	x	x
1	1	0	x	x

$$J_A = \bar{x} Q_B + x \bar{Q}_B$$

$$= x \oplus Q_B$$



Logic diagram



MEMORY DEVICES AND DIGITAL INTEGRATED CIRCUITS

Basic memory structure – ROM -PROM – EPROM – EEPROM –EAPROM, RAM – Static and dynamic RAM - Programmable Logic Devices – Programmable Logic Array (PLA) - Programmable Array Logic (PAL) – Field Programmable Gate Arrays (FPGA) – Implementation of combinational logic circuits using PLA, PAL.

Digital integrated circuits: Logic levels, propagation delay, power dissipation, fan-out and fanin, noise margin. logic families and their characteristics-RTL. TTL. ECL. CMOS

CLASSIFICATION OF MEMORIES

Discuss the classification of ROM and RAM memories.(Dec 2017),(May 2006,12,13,15), Dec 2013)

- A **memory unit** is a device to which binary information is transferred for storage and from which information is retrieved when needed for processing.
- When data processing takes place, information from memory is transferred to selected registers in the processing unit.
- A memory unit is a collection of cells capable of storing a large quantity of binary information.

TWO TYPES OF MEMORIES:

- There are two types of memories that are used in digital systems:
 - *Random-access memory (RAM) and Read-only memory (ROM)*
- (i) **Random-access memory (RAM)**
 - ✓ RAM stores new information for later use.
 - ✓ The process of storing new information into memory is referred to as a memory “write” operation.
 - ✓ The process of transferring the stored information out of memory is referred to as a memory “read” operation.
 - ✓ RAM can perform both write and read operations.
- (ii) **Read-only memory (ROM)**
 - ✓ ROM can perform only the read operation.
 - ✓ This means that suitable binary information is already stored inside memory and can be retrieved or read at any time.
 - ✓ However, that information cannot be altered by writing.
- ROM is a *programmable logic device (PLD)*.
- The binary information that is stored within such a device in some fashion and then embedded within the hardware in a process is referred to as *programming* the device.

TYPES OF ROM

Briefly explain EPROM and EEPROM technology.

(May 2009, May 2011)

The required paths in a ROM may be programmed in four different ways.

➤ **Mask programming**

- ✓ It is done by the semiconductor company during the *last fabrication process* of the unit.
- ✓ This procedure is costly because the *vendor charges the customer a special fee* for custom masking the particular ROM.

➤ **Programmable read-only memory- PROM.**

- ✓ Economical for small quantity.
- ✓ The fuses in the PROM are blown by the application of a high-voltage pulse to the device through a special pin.
- ✓ A *blown fuse* defines a *binary 0* state and an *intact fuse* gives a *binary 1* state.
- ✓ The procedure is irreversible and once programmed; the fixed pattern is permanent and cannot be altered.

➤ **Erasable PROM or EPROM**

- ✓ This can be restructured to the initial state even though it has been programmed previously.
- ✓ It is *erased* by placing under a special *ultraviolet light* for a given length of time.

➤ **Electrically erasable PROM (EEPROM) or Electrically Alterable PROM (EAPROM)**

- ✓ *Electrical signals* are used to erase the previously programmed connections *instead of ultraviolet light*.
- ✓ The advantage is that the device can be erased without removing it from its socket.

➤ **EAPROM stands for Electronically Alterable Programmable Read-Only Memory.**

- ✓ It is a type of PROM whose contents can be changed.
- ✓ It acts as a non-volatile storage device, and its individual bits can be re-programmed during the course of system operation.
- ✓ There are some timing constraints that cause the part to need more time for erasure or programming then is needed to read data from the part.

Draw the basic circuit of a Rom cell and describe its working principle with its architecture. [NOV 2020] **READ-ONLY MEMORY(ROM)**

Explain in detail about read-only memory.

(May 2013, May 2011, Dec 2009, Dec 2011)

- ROM is a non-volatile memory. It can hold data even if power is turned off.
- A ROM is essentially a memory device in which permanent binary information is stored.
- It is embedded in the unit and cannot be altered.
- It consists of 'k' inputs and 'n' outputs.

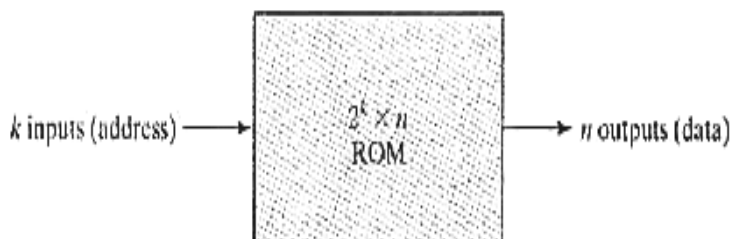


Fig: ROM block diagram

ROM Organization:

- The inputs provide the address for memory, and the outputs give the data bits of the stored word that is selected by the address.
- Number of words is get from number of address inputs, here it is 'k', hence 2^k words of n bits each is present in the memory.

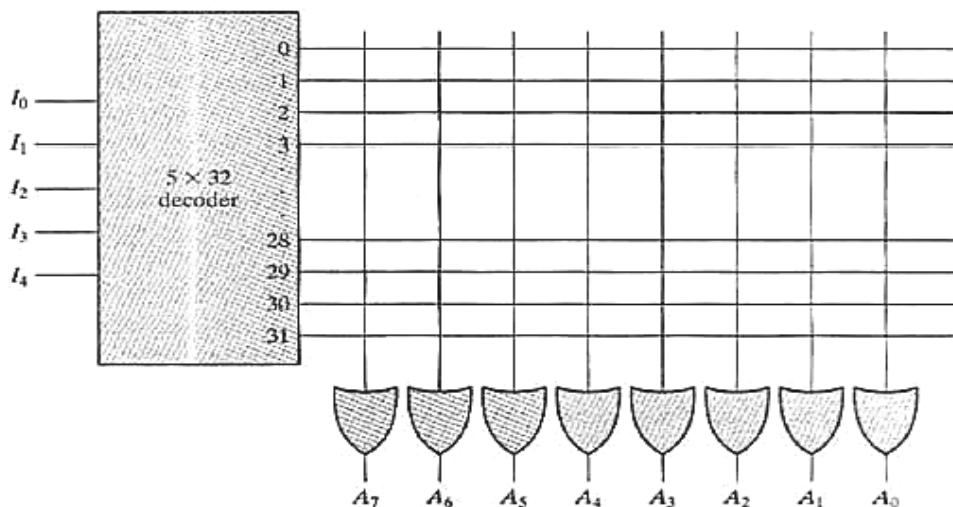


Fig: Internal logic of a 32x8 ROM

Example:

- The five inputs are decoded into 32 distinct outputs by means of a 5x32 decoder. Each output of the decoder represents a memory address.
- The 32 outputs of the decoder are connected to each of the eight OR gates. Each OR gate must be considered as having 32 inputs.

- Each output of the decoder is connected to one of the inputs of each OR gate.
- Since each OR gate has 32 input connections and there are 8 OR gates, the ROM contains $32 \times 8 = 256$ internal connections.
- A programmable connection between *two lines* is logically equivalent to a switch that can be altered to be either *closed* (two lines are connected) or *open* (two lines are disconnected).
- The programmable intersection between two lines is sometimes called a *cross point*.

ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		\vdots						\vdots				
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

- For an example, programming the ROM according to the truth table given by table.
- Every 0 listed in the truth table specifies the *absence* of a connection and every 1 listed specifies a path that is *obtained* by a connection.

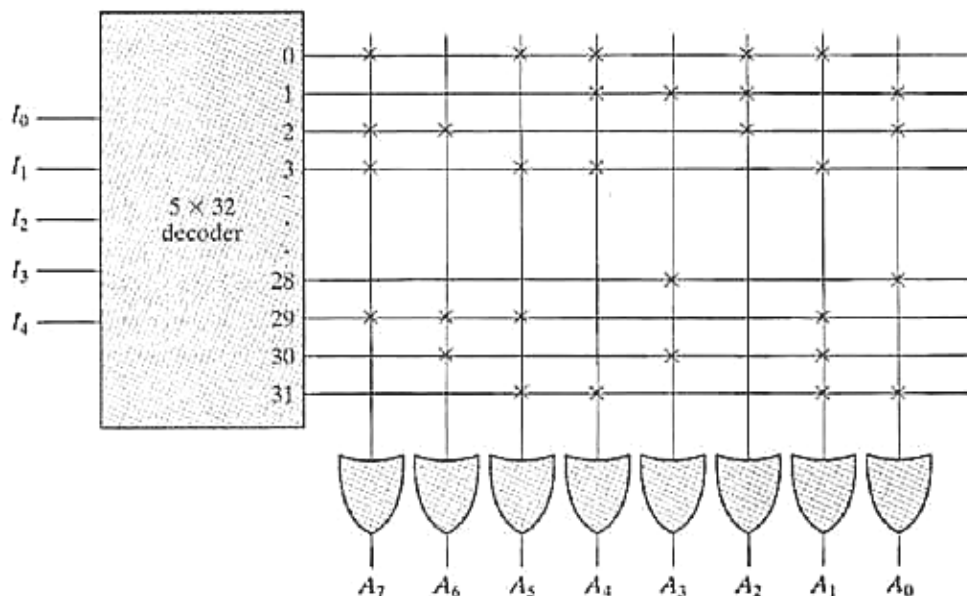


Fig: Programming the ROM according to ROM truth table

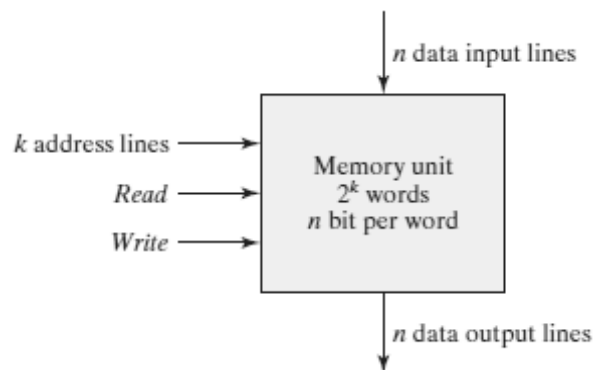
RANDOM ACCESS MEMORY(RAM)

Explain in detail about Random Access Memory.

- RAM stores new information for later use.
- The process of storing new information into memory is referred to as a memory *write* operation.
- The process of transferring the stored information out of memory is referred to as a memory *read* operation.
- RAM can perform both write and read operations.

RAM Organization:

A block diagram of a memory unit is shown in Fig.



- The n data input lines provide the information to be stored in memory, and the n data output lines supply the information coming out of memory.
- The k address lines specify the particular word chosen among the many available.
- The two control inputs specify the direction of transfer desired: *Write* input and *Read* input.
- A memory unit stores binary information in groups of bits called words.
- Each word in memory is assigned an identification number called an address starting from 0 up to $2^k - 1$, where k is the number of address lines.
- Consider for example, a memory unit with a capacity of 1K words of 16 bits each. Since
- Here 1024 x 16 RAM consists of 10 x 1024 decoder ($1K = 1024 \text{ bytes} = 2^{10}$), where the decoder inputs are the 10 address lines.
- The decoder accepts the address lines and provides the path needed to select the word specified.

Memory address		Memory content
Binary	Decimal	
000000000	0	1011010101011101
000000001	1	1010101110001001
000000010	2	0000110101000110
	⋮	⋮
111111101	1021	1001110100010100
111111110	1022	0000110100011110
111111111	1023	1101111000100101

Fig: Contents of a 1024×16 memory

Read and write operations:

- The two operations that RAM can perform are the write and read operations.

Steps to Write operation as follows:

- ✓ Apply the binary address of the desired word to the address lines.
- ✓ Apply the data bits that must be stored in memory to the data input lines.
- ✓ Activate the *write* input.

Steps to Read operation as follows:

- ✓ Apply the binary address of the desired word to the address lines.
- ✓ Activate the *read* input.
- The *memory enable or chip select* is used to enable the particular memory chip in a multichip implementation of a large memory.

Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

- When the memory enable is inactive, the *memory chip is not selected and no operation* is performed. When the *memory enable input is active*, the *read/write operation* to be performed.

Memory Cycle and Timing Waveforms:

With neat timing diagram, explain the write and read operations.

(May 2009)

- The operation of the memory unit is controlled by an external device such as a central processing unit (CPU).
- The CPU is usually synchronized by its own clock.
- The *access time* of memory is the time required to *select a word and read it*.
- The *cycle time* of memory is the time required to complete a *write operation*.

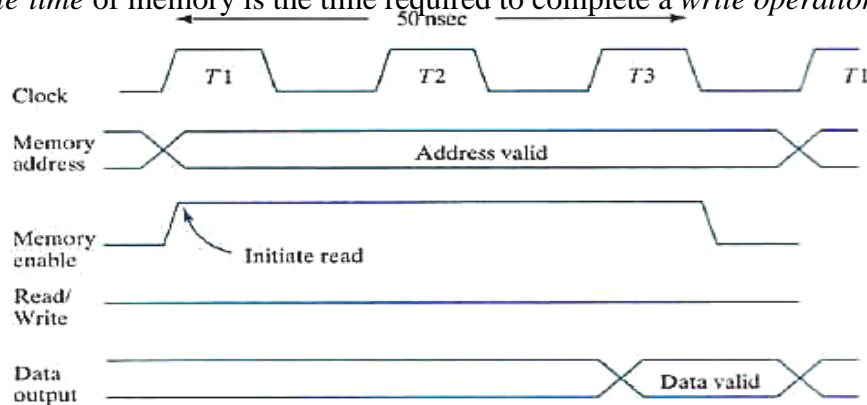


Fig: Memory read cycle timing waveform

- The memory- enable and read/write signals must be in their high level for a read operation.

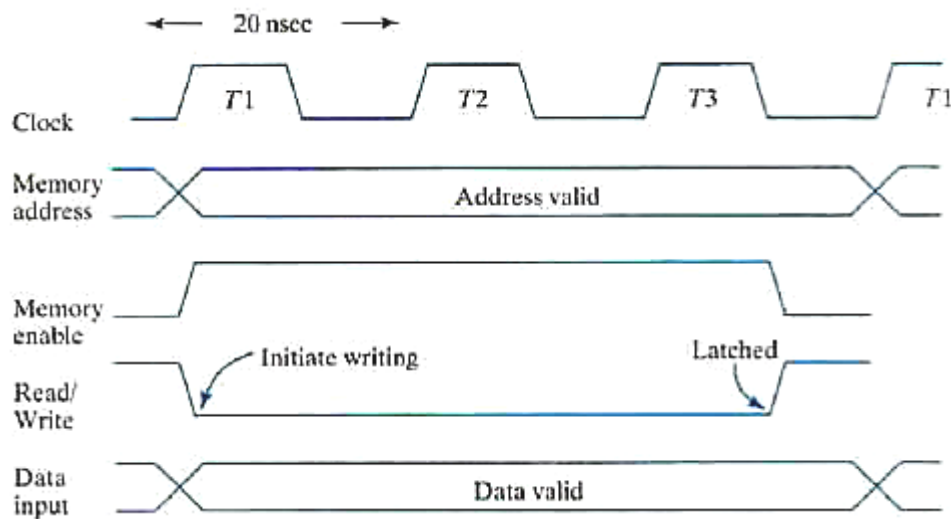


Fig: Memory write cycle timing waveform

- The memory enable signal switches to the high level and the read/write signal switches to the low level to indicate a write operation.

TYPES OF RAM

Explain the types of RAM with neat diagram.

(Dec 2011) (May-2010)(May 2018) (Dec 2018)

RAM is classified into two types.

1. *Static RAM*
2. *Dynamic RAM*

STATIC RAM (SRAM):

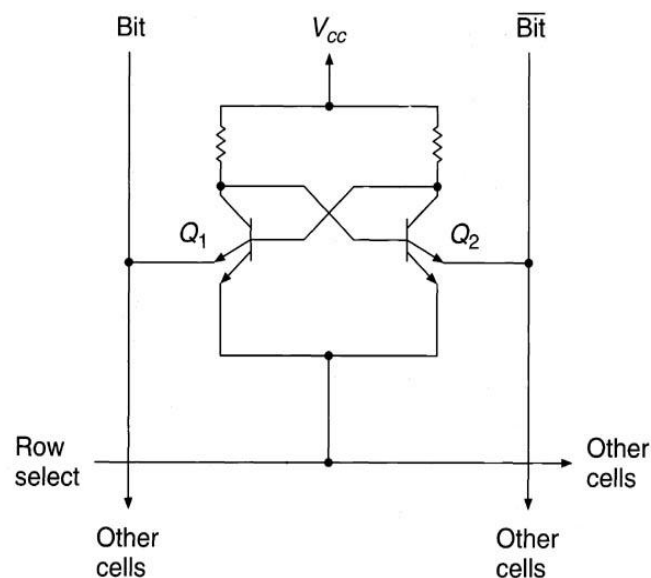
- Memories that consist of circuits capable of *retaining their state as long as power is applied* are known as static memories.
- Two basic SRAM cell technologies are
 - Bipolar and MOS.
- All those types use cross-coupled transistors to make up the basic flip-flop storage cell.

Bipolar Static RAM cell:

- It is implemented using TTL (Transistor-Transistor Logic) multiple emitter technology.
- It can store either 0 or 1 as long as power is applied.

Operation:

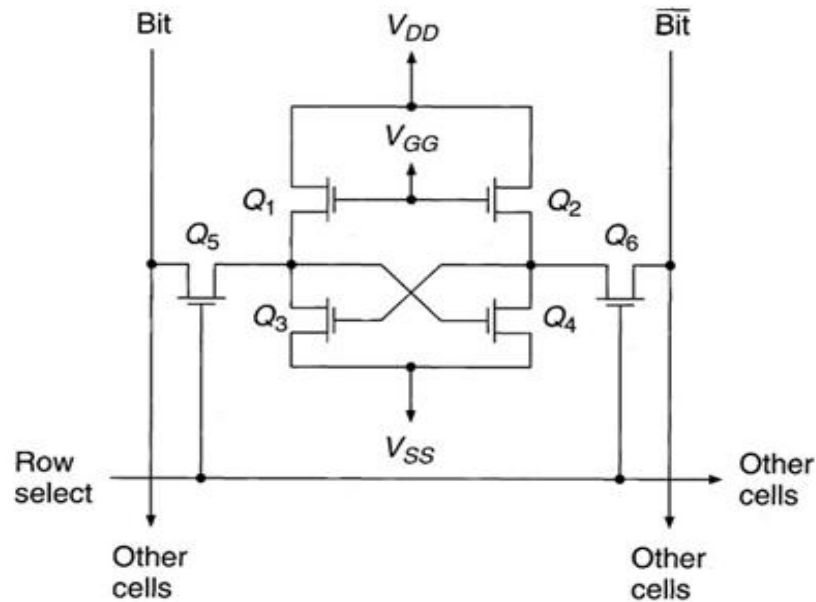
- Row and column select lines select a cell.
- The Q1 and Q2 are cross coupled inverters.
- A “1” is stored in the cell if Q1 is ON and Q2 is OFF.
- A “0” is stored in the cell if Q2 is ON and Q1 is OFF.
- When pulsing HIGH on Q1 emitter (SET), State is changed to ‘0’.
- When pulsing HIGH on Q2 emitter (RESET), State is changed to ‘1’



a. Bipolar SRAM cell

Explain static RAM using MOSFET.

(DEc 2019



b. MOS SRAM cell

Operation:

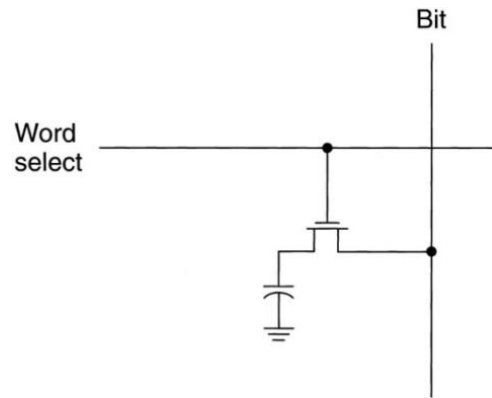
- In the basic NMOS cell, Q_1 and Q_2 are always biased to act as a Load Resistor for Q_3 and Q_4 .
- The Data in a cell can be read by setting $ROW_SELECT = 1$ to turn on Pass Transistors Q_5, Q_6 .
- The Data from cell is then “passed” to the BIT Line and $(BIT)'$ Line.
- To store a ‘0’, place a 0 on the bit line and set $ROW_SELECT = 1$. This turns on the Pass Transistors (Q_5, Q_6) to place a 0 to Q_4 (it is off). Q_3 is then ON to store the 0.
- A ‘1’ can be stored in a similar fashion.

Dynamic RAM cell:

Write note on dynamic RAM cell.

(Dec 2005, May 2014)

- Dynamic RAM (DRAM) stores data as *a charge on capacitors*.
- The stored charge on the capacitors tends to discharge with time, and the capacitors must be *periodically* recharged by refreshing the dynamic memory.
- Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge.
- DRAM offers *reduced power consumption and larger storage capacity* in a single memory chip.
- Memory units that lose stored information when power is turned off are said to be *volatile*.



Comparison of Static and Dynamic RAM.

(May 2009, May 2017)

- Integrated circuit RAM units are available in two operating modes: *static* and *dynamic*.

S.No	Static RAM	Dynamic RAM
1	Static RAM contains less memory cells per unit area	Dynamic RAM contains more memory cells as compared to static RAM per unit area.
2	It has less access time hence faster memories.	Its access time is greater than static RAMs.
3	Static RAM consists of number of flip flops. Each flip flop stores one bit.	Dynamic RAMs store the data as a charge on the capacitor. It consists of MOSFET and the capacitor for each cell.
4.	Refreshing circuitry is not required.	Refreshing circuitry is required to maintain the charge on the capacitors after every few milliseconds
5	Cost is more	Cost is less

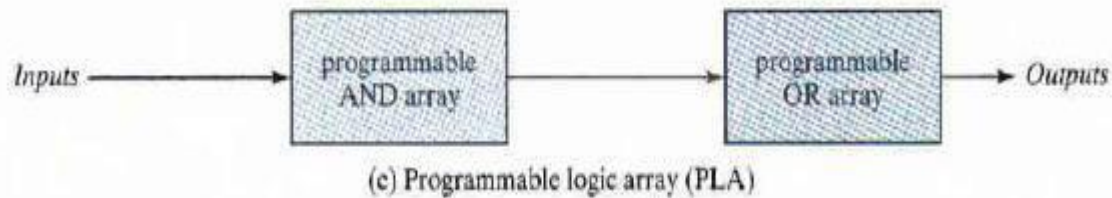
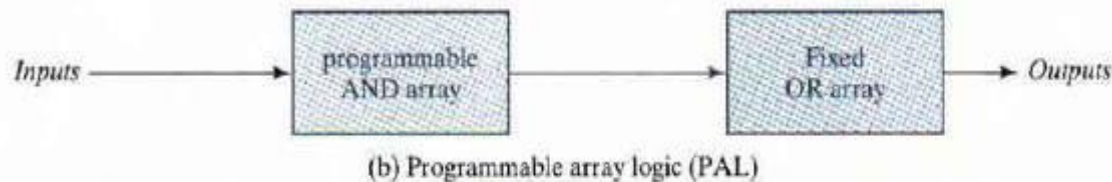
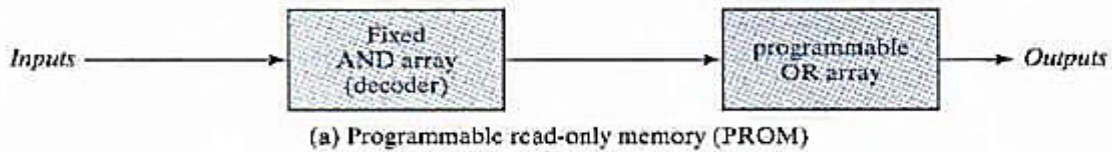
PROGRAMMABLE LOGIC DEVICES (PLDs)

Write brief notes on combinational programmable logic device PLD.

Combinational PLDs

- The PROM is a combinational programmable logic device (PLD)-an integrated circuit with programmable gates divided into an *AND* array and an *OR* array to provide an AND-OR *sum of-product* implementation.
- There are three major *types of combinational PLDs*, differing in the placement of the programmable connections in the AND-OR array.

1. *PROM- Fixed AND array and a programmable OR array.*
2. *PAL - Programmable AND array and a fixed OR array.*
3. *PLA - Programmable AND array and a programmable OR array.*



Present the basic concepts of PLA and its applications. [NOV 2020]

PROGRAMMABLE LOGIC ARRAY (PLA)

(May 2011, May 2012)

Write short notes on PLA.

(Dec 2019)

- Programmable logic arrays (PLAs) is a type of fixed architecture logic devices with *programmable AND gates followed by programmable OR array.*
- PLA is used to implement a complex combinational circuit.
- The AND and OR gates inside the PLA are initially fabricated with fuses among them.
- The specific Boolean functions are implemented in sum of products (SOP) form by blowing appropriate fuses and leaving the desired connections.
- For an example, the Boolean expressions are,

$$F_1 = A\bar{B} + AC + \bar{A}B\bar{C}$$

$$F_2 = \overline{(AC + BC)}$$

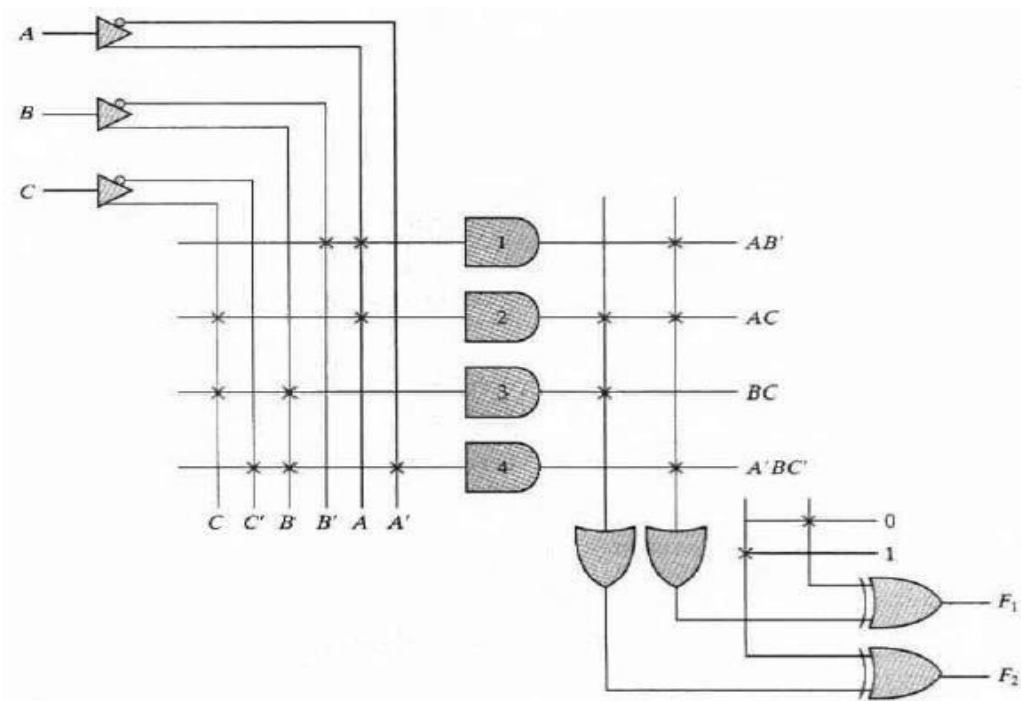


Fig: PLA with three inputs, four product terms and two outputs

- The fuse map of a PLA can be specified in a tabular form. The first section lists the product terms numerically.
- The second section specifies the required paths between inputs and AND gates.
- The third section specifies the paths between the AND and OR gates. For each output variable, we may have a T (for true) or C (for complement) for programming the XOR gate.
- For each product term, the inputs are marked with 1, 0, or - (dash). If a variable in the product term *appears* in the form in which it is true, the corresponding input variable is *marked with a 1*.
- If it *appears complemented*, the corresponding input variable is *marked with a 0*. If the variable is absent from the product term, it is marked with a dash.

PLA Programming Table

		Inputs			Outputs	
		A	B	C	(T) F_1	(C) F_2
AB'	1	1	0	—	1	—
AC	2	1	—	1	1	1
BC	3	—	1	1	—	1
$A'BC'$	4	0	1	0	1	—

* A combinational circuit is defined by the function

$$F_1(A, B, C) = \Sigma(3, 5, 6, 7)$$

$$F_2(A, B, C) = \Sigma(0, 2, 4, 7)$$

Implement the circuit with PLA having 3 i/p's, 4 Product terms and 2 o/p's. with True & complement.

sol.

Step 1: K-map F_1

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

Complement, $F_1 = (\overline{B}\overline{C} + \overline{A}\overline{C} + \overline{A}\overline{B})'$

K-map F_2

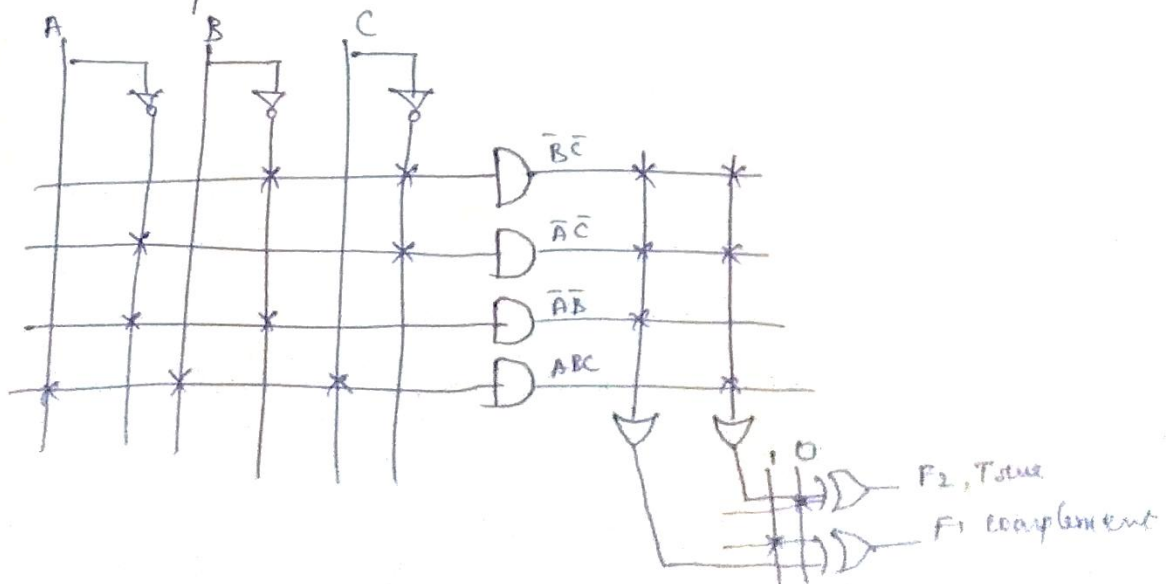
A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	1	0

True, $F_2 = \overline{B}\overline{C} + \overline{A}\overline{C} + ABC$

Step 2: PLA Table

	Product Term	i/p's			o/p's	
		A	B	C	F ₁	F ₂
$\overline{B}\overline{C}$	1	—	0	0	1	1
$\overline{A}\overline{C}$	2	0	—	0	1	1
$\overline{A}\overline{B}$	3	0	0	—	1	—
ABC	4	1	1	1	—	1
					C	T
					T/C	

Steps: PLA Diagram



❖ Implement the following two Boolean functions with a PLA: (May 2012, May 2014, Dec 2013)

$$F_1(A, B, C) = \sum(0, 1, 2, 4)$$

(May 2019)

$$F_2(A, B, C) = \sum(0, 5, 6, 7)$$

Solution:

Kmap

PLA programming table						
	Product term	Inputs			Outputs	
					(C)	(T)
		A	B	C	F ₁	F ₂
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
A'B'C'	4	0	0	0	–	1

A \ BC	B			
	00	01	11	10
0	m ₀ 1	m ₁ 1	m ₃ 0	m ₂ 1
1	m ₄ 1	m ₅ 0	m ₇ 0	m ₆ 0

A \ BC	B			
	00	01	11	10
0	m ₀ 1	m ₁ 0	m ₃ 0	m ₂ 0
1	m ₄ 0	m ₅ 1	m ₇ 1	m ₆ 1

Boolean Expressions:

$$F_1 = (AB + AC + BC)'$$

$$F_2 = AB + AC + A'B'C'$$

PLA Diagram:

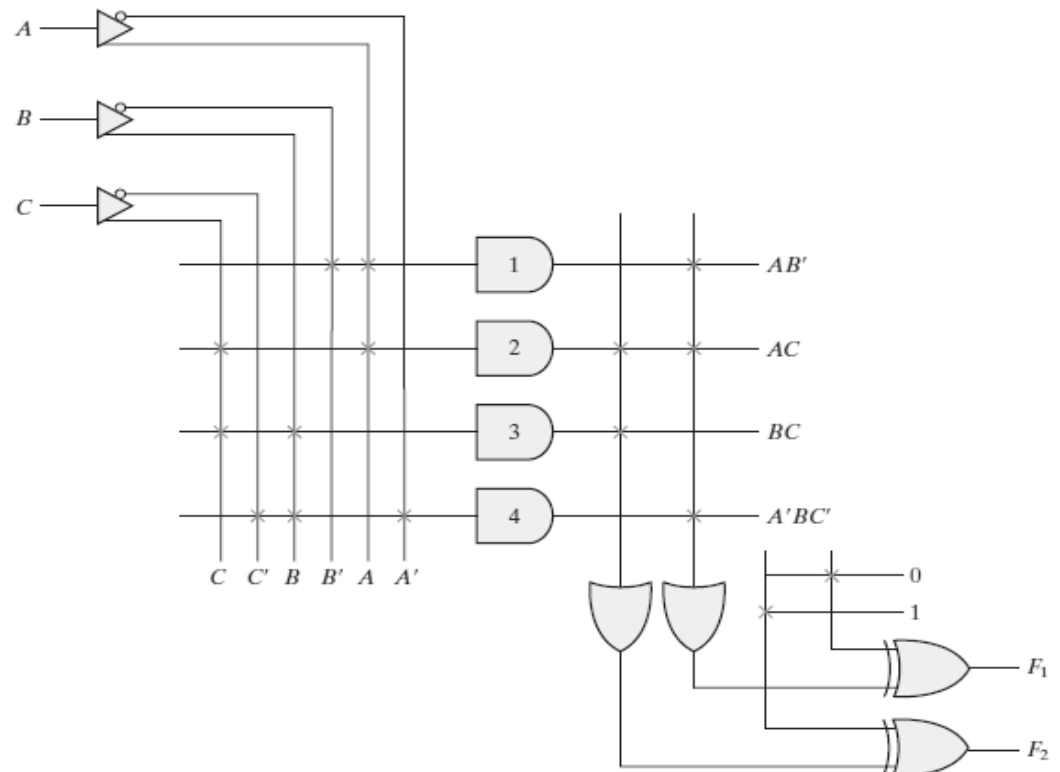


FIGURE
PLA with three inputs, four product terms, and two outputs

Problems using PROM:

Design a large circuit for the following Boolean expressions using PROM.

$$F_1(x, y, z) = \sum m(1, 2, 4, 7).$$

$$F_2(x, y, z) = \sum m(3, 5, 6, 7).$$

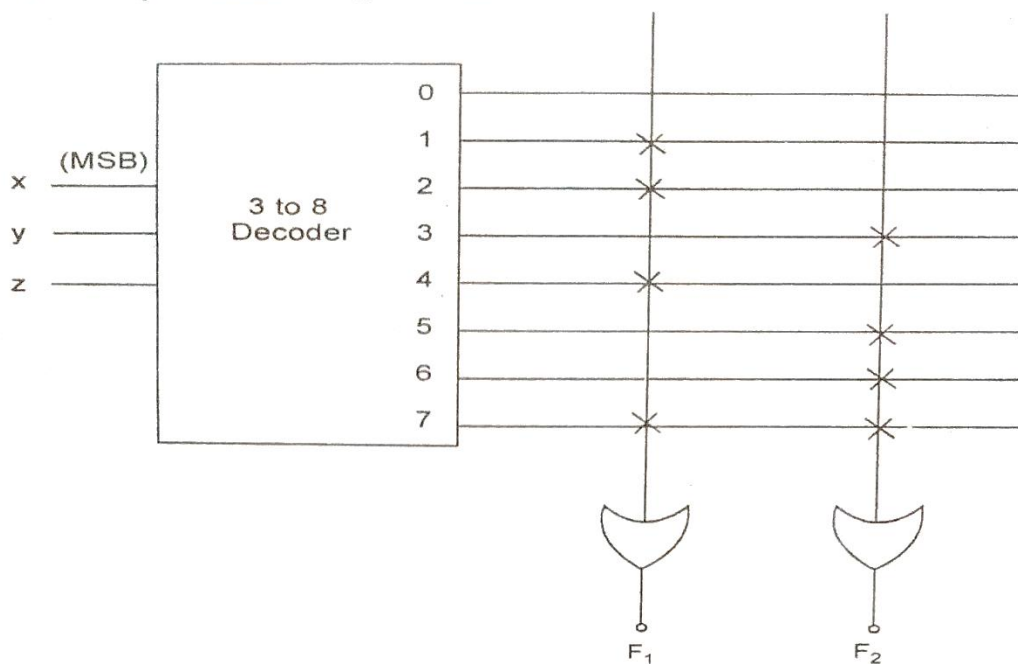
✍ Solution :

The Truth Table for the functions F_1 and F_2 are given below.

∧

Inputs			Outputs	
x	y	z	F_1	F_2
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

The PROM implementation is given below.



PROGRAMMABLE ARRAY LOGIC (PAL):

Write short notes on PAL.

(Dec2011,Nov-2016)

- The PAL is a programmable logic device with a *fixed OR array and a programmable AND* array.
- Because only the AND gates are programmable, the PAL is easier to program than but is not as flexible as the PLA.
- The PAL is a programmable logic device with a fixed OR array and a programmable AND array.
- Below figure shows the logic configuration of a typical PAL with four inputs and four outputs.
- Each input has a buffer–inverter gate, and each output is generated by a fixed OR gate.
- There are four sections in the unit, each composed of an AND–OR array that is *three wide*.
- Each AND gate has 10 programmable input connections, shown in the diagram by 10 vertical lines intersecting each horizontal line.
- The horizontal line symbolizes the multiple-input configuration of the AND gate.
- One of the outputs is connected to a buffer–inverter gate and then fed back into two inputs of the AND gates.

Example:

Implement the following Boolean functions, using PAL.

(May 2013)

$$w(A, B, C, D) = \sum(2, 12, 13)$$

$$x(A, B, C, D) = \sum(7, 8, 9, 10, 11, 12, 13, 14, 15)$$

$$y(A, B, C, D) = \sum(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$$

$$z(A, B, C, D) = \sum(1, 2, 8, 12, 13)$$

Sol:

Simplify the functions using K Map:

Simplify the functions using K-map.

W

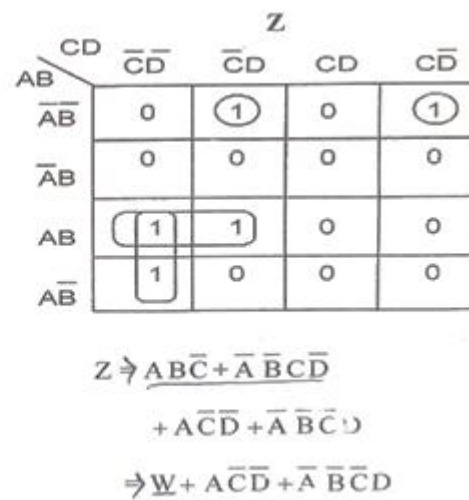
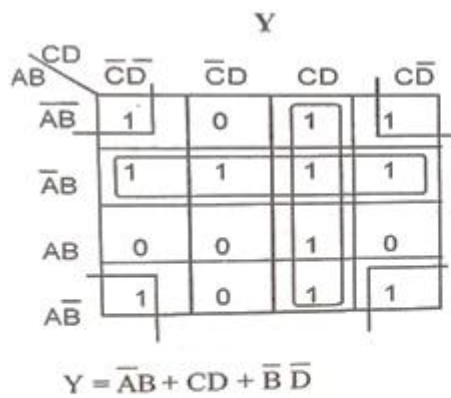
	CD	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
AB	$\overline{A}\overline{B}$	0	0	0	1
	$\overline{A}B$	0	0	0	0
	AB	1	1	0	0
	$A\overline{B}$	0	0	0	0

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

X

	CD	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
AB	$\overline{A}\overline{B}$	0	0	0	0
	$\overline{A}B$	0	0	1	0
	AB	1	1	1	1
	$A\overline{B}$	1	1	1	1

$$X = A + BCD$$

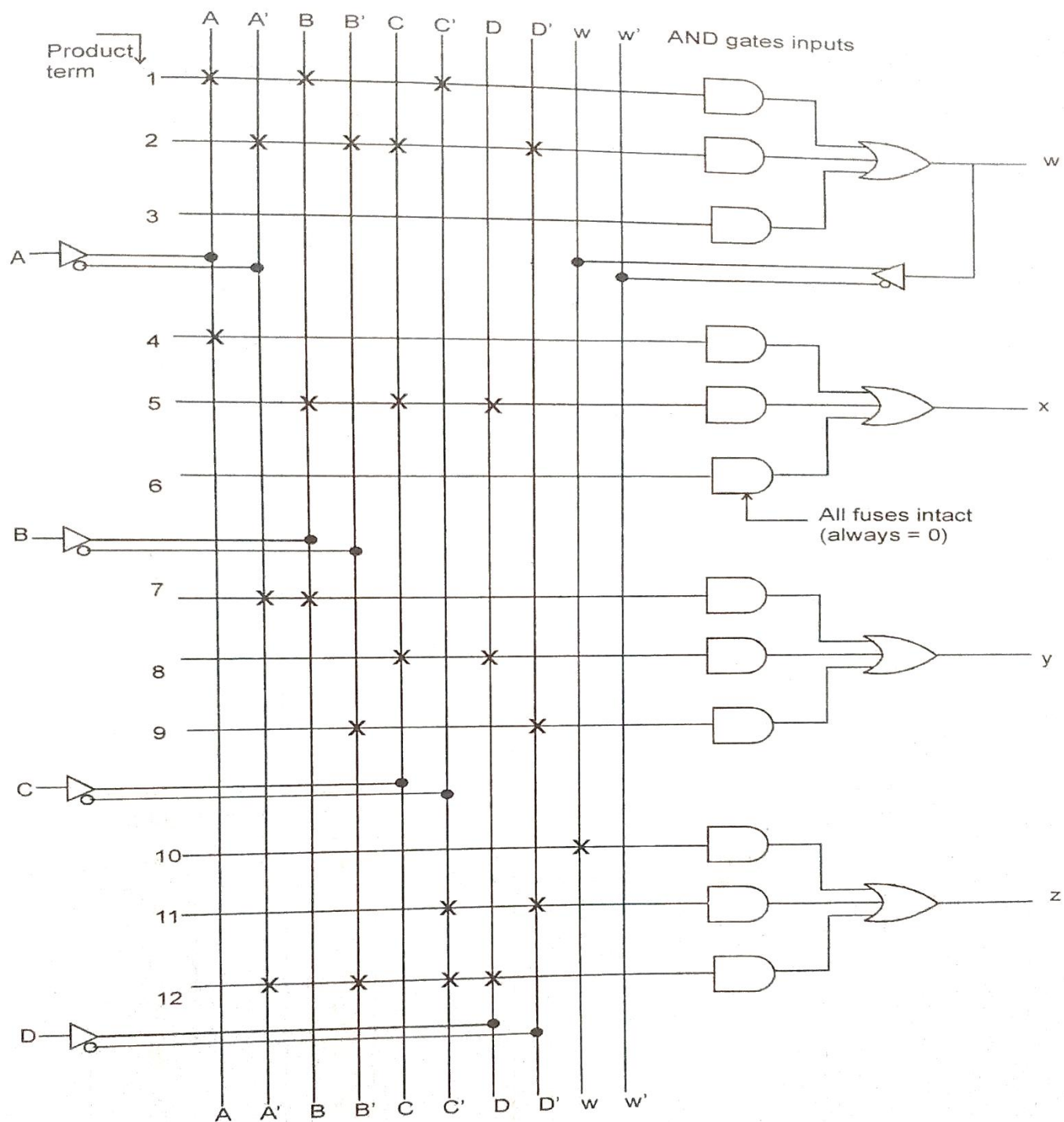


PAL Programming Table:

PAL Programming Table

Product Term	AND Inputs					Outputs
	A	B	C	D	W	
1	1	1	0	-	-	$w = ABC'$ $+ A'B'CD'$
2	0	0	1	0	-	
3	-	-	-	-	-	
4	1	-	-	-	-	$x = A$ $+ BCD$
5	-	1	1	1	-	
6	-	-	-	-	-	
7	0	1	-	-	-	$y = A'B$ $+ CD$ $+ B'D'$
8	-	-	1	1	-	
9	-	0	-	0	-	
10	-	-	-	-	1	$z = w$ $+ AC'D'$ $+ A'B'C'D$
11	1	-	0	0	-	
12	0	0	0	1	-	

PAL Logic Diagram:



Problem 1:

(Dec 2009, Dec 2011)

Implement the following function using PLA

[NOV/DEC 2021]

$$A(x, y, z) = \sum m(1, 2, 4, 6)$$

$$B(x, y, z) = \sum m(0, 1, 6, 7)$$

$$C(x, y, z) = \sum m(2, 6).$$

Ans:

	yz	00	01	11	10
x					
0		0	1	0	1
1		1	0	0	1

$$A = \bar{x}\bar{y}z + x\bar{z} + y\bar{z}$$

	yz	00	01	11	10
x					
0		1	1	0	0
1		0	0	1	1

$$B = \bar{x}\bar{y} + xy$$

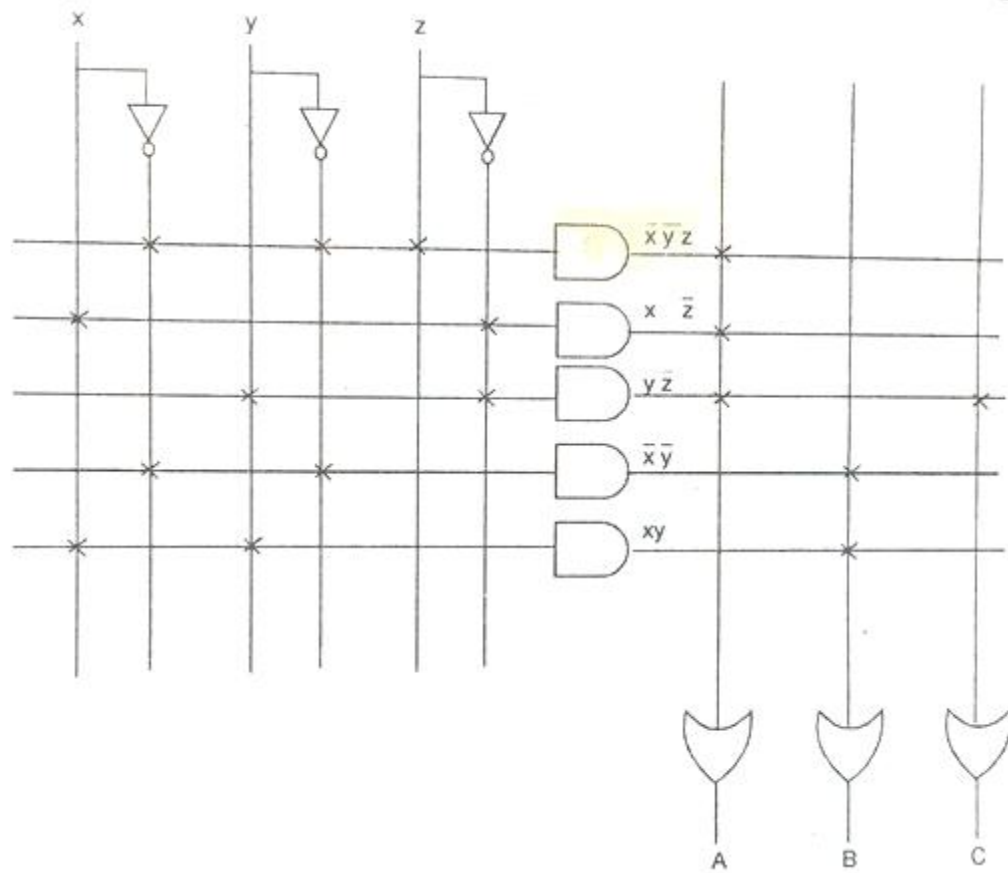
	yz	00	01	11	10
x					
0		0	0	0	1
1		0	0	0	1

$$C = y\bar{z}$$

PLA Programming Table:

Product Term	Inputs			Outputs		
	x	y	z	A	B	C
$\bar{x}\bar{y}z$	0	0	1	1	1	0
$x\bar{z}$	1	-	0	1	-	-
$y\bar{z}$	-	1	0	1	-	1
$\bar{x}y$	0	0	-	-	1	-
xy	1	1	-	-	1	-

PLA Logic Diagram:



Implement the switching function:

$$Z_1 = \overline{a}\overline{b}\overline{d}e + \overline{a}\overline{b}\overline{c}\overline{d}\overline{e} + bc + de$$

$$Z_2 = \overline{a}\overline{c}e$$

$$Z_3 = bc + de + \overline{c}\overline{d}\overline{e} + bd$$

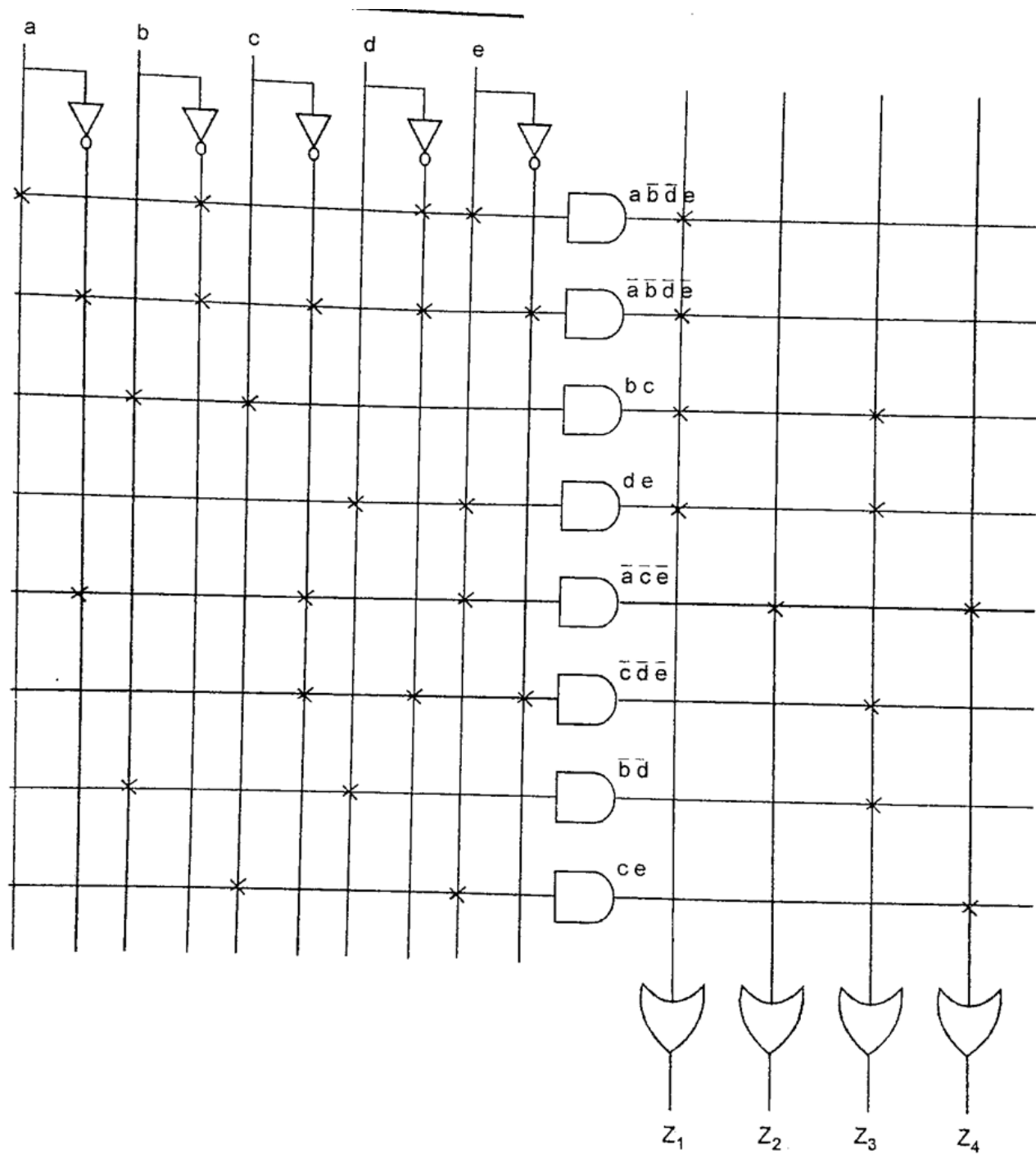
$$Z_4 = \overline{a}\overline{c}e + ce \text{ using } 5 \times 8 \times 4 \text{ PLA.}$$

✍ Solution :

PLA Programming Table

S.No.	Product Term	Inputs	Outputs
		a b c d e	Z ₁ Z ₂ Z ₃ Z ₄
1	$\overline{a}\overline{b}\overline{d}e$	1 0 - 0 1	1 - - -
2	$\overline{a}\overline{b}\overline{c}\overline{d}\overline{e}$	0 0 0 0 0	1 - - -
3	bc	- 1 1 - -	1 - 1 -
4	de	- - - 1 1	1 - 1 -
5	$\overline{a}\overline{c}e$	0 - 0 - 1	- 1 - 1
6	$\overline{c}\overline{d}\overline{e}$	- - 0 0 0	- - 1 -
7	bd	- 1 - 1 -	- - 1 -
8	ce	- - 1 - 1	- - - 1

PLA logic Diagram:



Field-Programmable Gate Array (FPGA)

Write short notes on FPGA.(May2010,May2012 ,May 2013, May 2015,Nov-2016, Dec 2013, Dec 2017)) (Dec 2019)

- A Field Programmable Gate Array (FPGA) is a VLSI circuit that can be programmed at the user's location.
- A typical FPGA consists of an array of millions of logic blocks, surrounded by programmable input and output blocks and connected together via programmable interconnections.
- There is a wide variety of internal configurations within this group of devices.
- The performance of each type of device depends on the circuit contained in its logic blocks and the efficiency of its programmed interconnections.

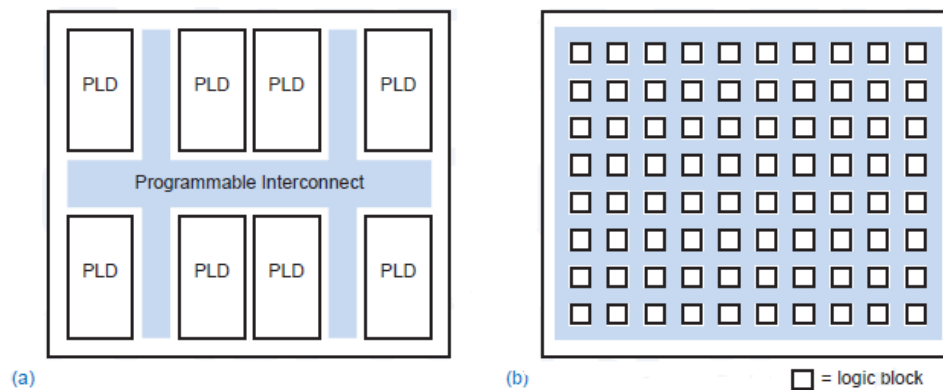


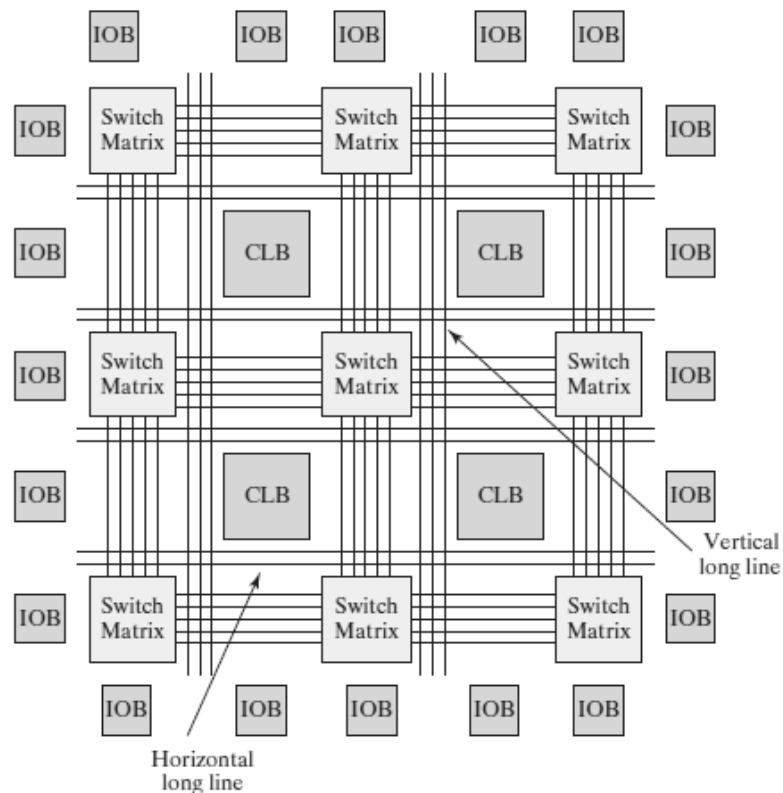
Fig: Large programmable-logic-device scaling approaches: (a) CPLD; (b) FPGA.

- A typical FPGA logic block consists of lookup tables, multiplexers, gates, and flip-flops.
- A lookup table is a truth table stored in an SRAM and provides the combinational circuit functions for the logic block.
- The combinational logic section, along with a number of programmable multiplexers, is used to configure the input equations for the flipflop and the output of the logic block.
- An FPGA contains a number of relatively independent configurable logic modules, configurable I/Os and programmable interconnection paths or routing channels.
- All the resources of this device are uncommitted and these must be selected, configured, and interconnected by the user to form a logic system for his application.
- FPGAs are specified by their size, configuration of their logic modules, and interconnection requirements.
- FPGA with larger logic modules may not be sufficiently utilized to perform simple logic functions and thereby wasting the logic modules.

- Use of smaller logic modules leads to a larger number of interconnections with the device causing significant propagation delay as well as consuming a large percentage of FPGA area.
- The designer must optimize the logic module size and interconnection requirements according to the application of logic system design.
- For a given FPGA device, there are many possible ways to configure to meet the design requirements.
- The *advantage* of using RAM instead of ROM to store the truth table is that the table can be programmed by writing into memory.
- The *disadvantage* is that the memory is volatile and presents the need for the lookup table's content to be reloaded in the event that power is disrupted.
- The program can be downloaded either from a host computer or from an onboard PROM.
- The program remains in SRAM until the FPGA is reprogrammed or the power is turned off. The device must be reprogrammed every time power is turned on.

Basic Xilinx Architecture:

- The basic architecture of Spartan and earlier device families consists of an array of configurable logic blocks (CLBs), a variety of local and global routing resources, and input–output (I/O) blocks (IOBs), programmable I/O buffers, and an SRAM based configuration memory, as shown in Fig.



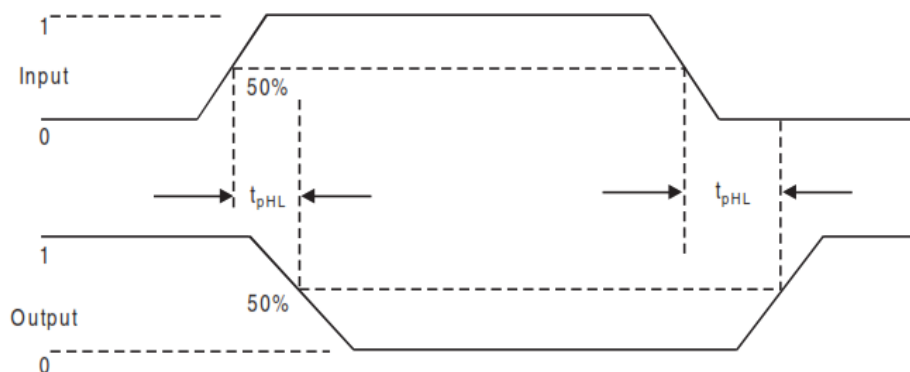
CHARACTERISTICS OF DIGITAL IC

Explain the various parameters used in logic families.

- The various digital logic families are usually evaluated by comparing the characteristics of the basic gates of each family.
- The most important governing parameters of various logic families are listed below.
 1. Propagation delay (speed of operation).
 2. Power dissipation.
 3. Fan in.
 4. Fan out.
 5. Noise immunity.
 6. Operating temperature.
 7. Power supply requirement.
 8. Current and voltage parameters.

Propagation Delay:

- Propagation delay is defined as the time taken for the output of a logic gate to change after the inputs have changed.
- It is the transition time for the signal to propagate from input to output.
- This factor governs the speed of operation of a logic circuit.



- Two types of propagation delay times, which are defined as
 - (a) t_{pLH} : It is the propagation delay time for a signal to change from logic LOW (0 state) to HIGH (1 state).
 - (b) t_{pHL} : It is the propagation delay time for a signal to change from logic HIGH (1 state) to LOW (0 state).
- The delay times are measured by time lapsed between the 50% voltage levels of the
- input and the output waveforms while making the transition.

Power dissipation:

- Power dissipation is the measure of the power consumed by logic gates when fully driven by all inputs.
- The average power or the DC power dissipation is the product of DC supply voltage and the mean current consumed from that supply.

Fan In:

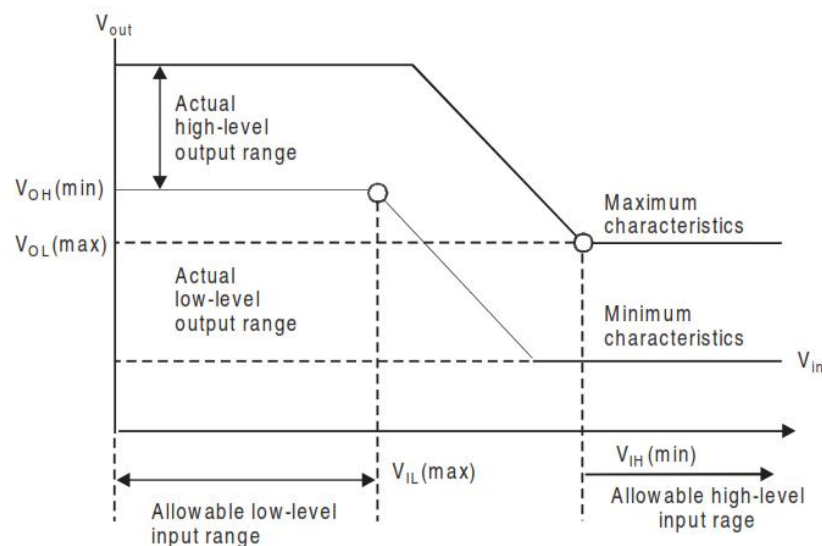
- The maximum number of inputs that can be connected to a logic gate without any impairment of its normal operation is referred to as fan in.

Fan Out:

- Fan out refers to the maximum number of standard loads that the output of the gate can drive without any impairment or degradation of its normal operation.

Noise Margin:

- Noise immunity or the noise margin is the limit of noise voltage that may appear at the input of the logic gate without any impairment of its proper logic operation.
- The difference between the operating input logic voltage level and the threshold voltage is the noise margin of the circuit.
- Noise margin is related with the input-output transfer characteristics of a logic gate.
- It depends on loading factors, power supply, operating temperature, fabrication process by the manufacturers, etc.



- From the ***maximum characteristics***, any input voltage level *less than* $V_{IL}(\max)$ is referred as low-voltage level or logic 0.
 - On the other hand, any input voltage level *greater than* $V_{IH}(\min)$ is referred as as high level or logic 1.

- From the **minimum characteristics**, the manufacture specifies that low level or logic 0 output voltage does not exceed $V_{OL}(\text{max})$ and the high level or logic 1 output is always greater than $V_{OH}(\text{min})$.
- The worst-case low-level noise margin is $V_{IL}(\text{max}) - V_{OL}(\text{max})$ and the worst-case high-level noise margin is $V_{OH}(\text{min}) - V_{IH}(\text{min})$.

Explain the working principle of (i) TTL NAND gate (ii) ECL OR/NOR gate with circuit diagram. [NOV 2020]

Transistor Transistor Logic (TTL)

Explain about TTL Logic with neat diagram.

(Dec 2019 (Dec 2018

- It is a logic family implemented with bipolar process technology that combines or integrates NPN transistors, PN junction diodes and diffused resistors in a single monolithic structure to get the desired logic function.
- The NAND gate is the basic building block of this logic family.
- Different subfamilies in this logic family such as standard TTL, low-power TTL, high-power TTL, low-power Schottky TTL, Schottky TTL, advanced low-power Schottky TTL, advanced Schottky TTL and fast TTL.

Standard TTL NAND gate:

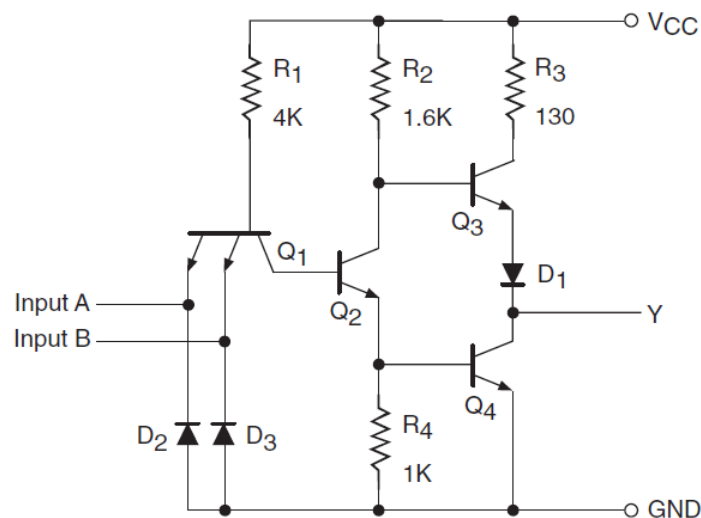


Figure Standard TTL NAND gate.

- The circuit operates as follows; Transistor Q1 is a two-emitter NPN transistor, which is equivalent to two NPN transistors with their base and emitter terminals tied together.
- The two emitters are the two inputs of the NAND gate.
- Diodes D2 and D3 are used to limit negative input voltages.

- When both the inputs are in the logic HIGH state, the current flows through the base-collector PN junction diode of transistor Q_1 into the base of transistor Q_2 .
- Transistor Q_2 is turned ON to saturation, with the result that transistor Q_3 is switched OFF and transistor Q_4 is switched ON. This produces a logic LOW at the output.

Totem-Pole Output Stage:

- It is the same circuit as the open-collector gate, except for the output transistor Q_4 , a diode D_1 , and resistor 130Ω at the collector of Q_4 .
- It is called the totem pole output configuration, because the transistor Q_4 sits upon Q_3 .
- The base of the transistor Q_4 is driven from the collector of Q_2 .
- **Advantages:**
 - It offers low-output impedance in both the HIGH and LOW output states.
 - Because capacitance at the output can be charged or discharged very rapidly, thus allowing quick transitions at the output from one state to the other.
 - When the output is in the logic LOW state, transistor Q_4 would need to conduct a fairly large current if its collector were tied to VCC through R_3 only.
 - A non-conducting Q_3 overcomes this problem.
- **Disadvantage**
 - Switch-off action of Q_4 being slower than the switch-on action of Q_3 .
 - Therefore, a small fraction of time, both the transistors are conducting, thus drawing heavy current from the supply.

NOT Gate (or Inverter):

- It is just the same as that of the NAND gate except that the input transistor is a normal single emitter NPN transistor instead of a multi-emitter one.

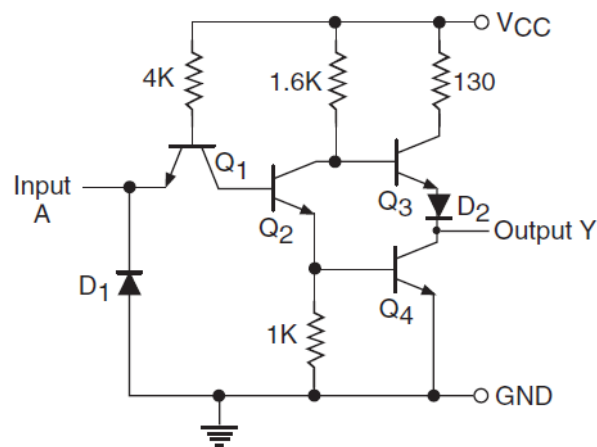


Figure Inverter in the standard TTL.

TTL NOR Gate:

- On the input side there are two separate transistors instead of the multi-emitter transistor of the NAND gate.
- The inputs are fed to the emitters of the two transistors, the collectors of which again feed the bases of the two transistors with their collector and emitter terminals tied together.
- The resistance values used is the same as those used in the case of the NAND gate.
- The output stage is also the same totem-pole output stage.

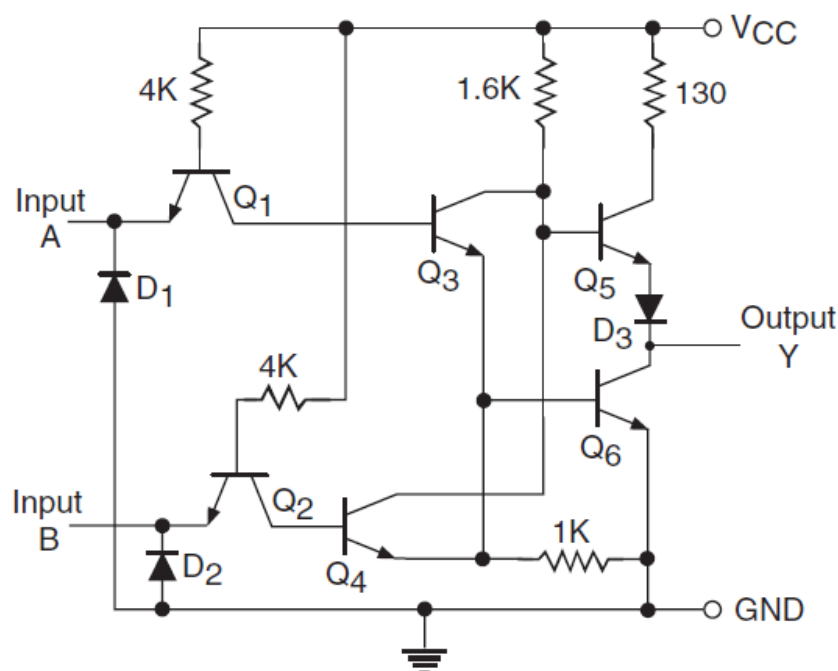


Figure NOR gate in the standard TTL.

[NOV/DEC 2021]

Tristate Gate: Explain about tristate TTL output configuration.

(May 2019)

- A tristate gate has three output states, namely the logic LOW state, the logic HIGH state and the high-impedance state.
- An external enable input decides whether the logic gate works according to its truth table or is in the high-impedance state.
- Figure shows the typical internal schematic of a tristate inverter with an active HIGH enable input.
- The circuit functions as follows.
 - ✓ When the enable input is HIGH, it reverse-biases diode D1 and also applies a logic HIGH on one of the emitters of the input transistor Q1.
 - ✓ The circuit behaves like an inverter. When the enable input is LOW, diode D1 becomes

forward biased.

- ✓ A LOW enable input forces Q2 and Q4 to cut-off. Also, a forward-biased D1 forces Q3 to cut-off.
- ✓ With both output transistors in cut-off, the output essentially is an open circuit and thus presents high output impedance.

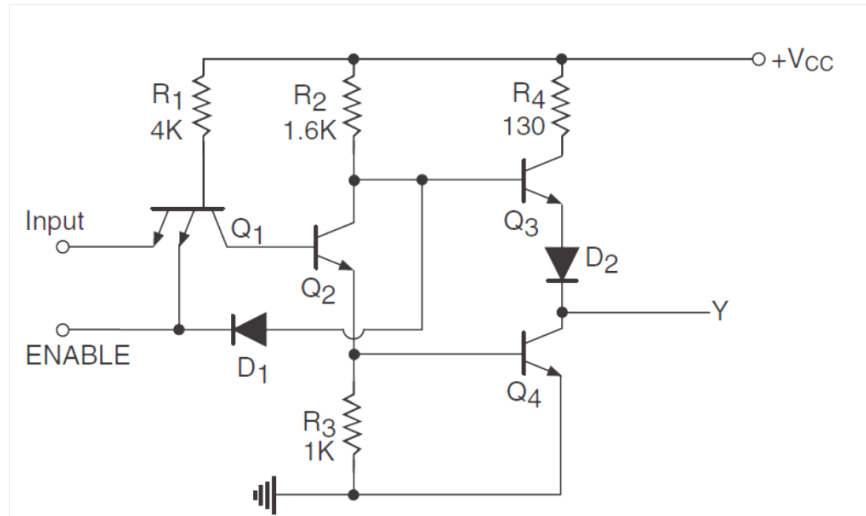
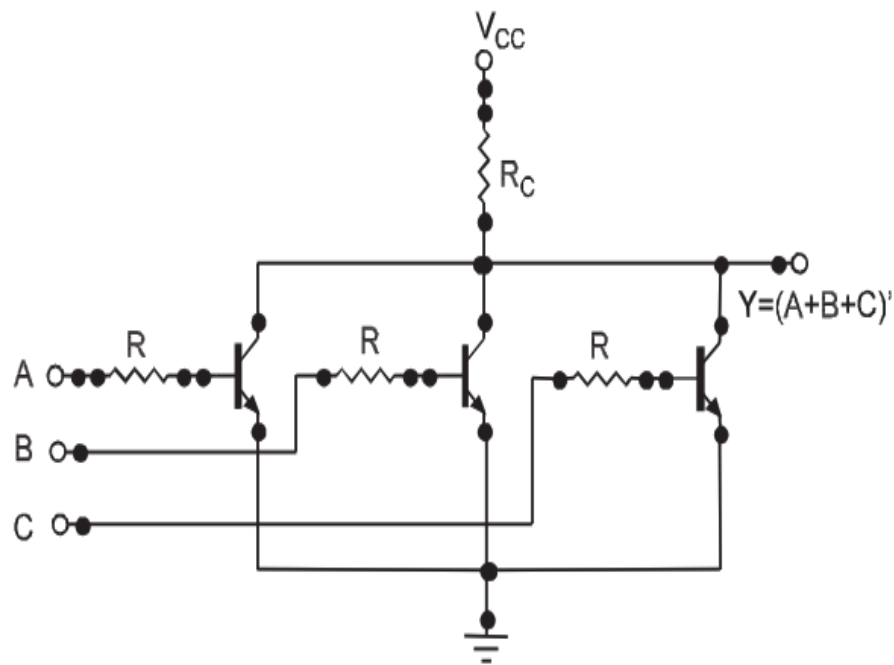


Fig: Tristate inverter in the TTL

RESISTOR-TRANSISTOR LOGIC (RTL)

Explain about RTL logic family.

- Each input is associated with one resistor and a transistor. The collectors of the transistors are tied together with a common resistor to the V_{CC} supply.
- The output is taken from the collectors joint.
- The voltage levels of the circuit are 0.2 V for low level and 1 to 3.6 V for high level.
- If any of the inputs is at high level, the corresponding transistor is at saturation.
- This causes the output at low irrespective of the conditions of other transistors, as all the transistors are connected in parallel.
- If all the inputs are at low level at 0.2 V, all the transistors are at cut-off condition.
- Because base-to-emitter voltage of all the transistors $V_{BE} < 0.6$ V, causing the output of the circuit at high level approaching the value of the supply voltage V_{CC} .
- Thus confirms the conditions of a NOR logic.
- Note that the noise margin for low signal input is $0.6 - 0.2 = 0.4$ V.



Drawbacks:

- Reduce the switching speed of the circuit.
- It degrades the rise and fall times of any input pulse.
- Reduction in base resistors reduces the input resistance, increases power consumption, and decreases the fan in.

The following are the characteristics of the RTL family.

1. Speed of operation is low, i.e., the propagation delay is high up to the order of 500 ns. It cannot operate at more than 4 MHz.
2. Fan out is 4 or 5 with a switching delay of 50 ns and fan in is 4.
3. Poor noise immunity.
4. High average power dissipation. Elimination of base resistors in RTL will reduce the power dissipation, which results in Direct-coupled Transistor Logic (DCTL).
5. Sensitive to temperature.

1. Implement BCD to Gray Code using PLA and PAL.

(Apr 2018)

PALs and PLAs

Design Example: BCD to Gray Code Converter

Truth Table

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	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

Minimized Functions:

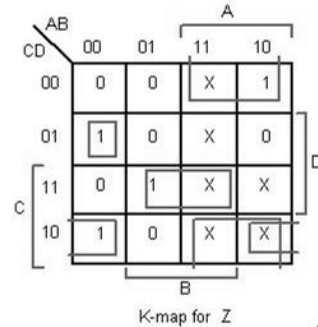
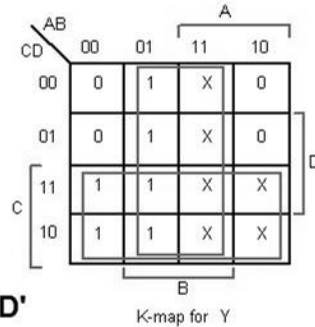
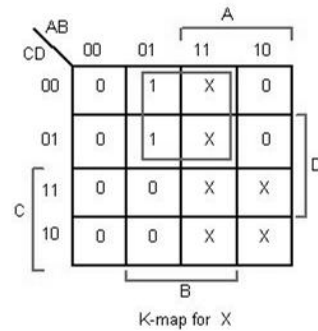
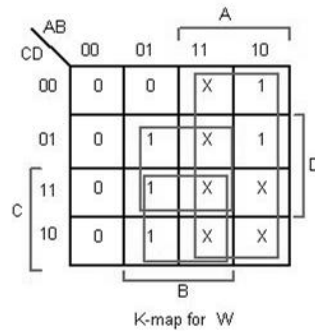
$$W = A + B D + B C$$

$$X = B C'$$

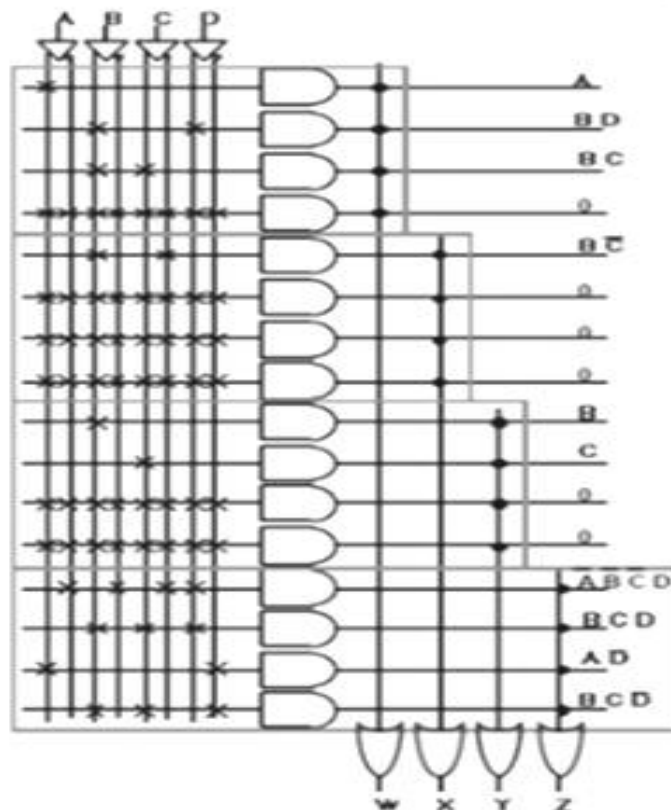
$$Y = B + C$$

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

K-maps



Programmed PLA;



Problem:

Design combinational circuits to convert binary coded decimal number into an excess-3 code using PLA.
(May 2018)

Step 1: Truth Table

BCD Code				Excess-3 Code			
B ₃	B ₂	B ₁	B ₀	E ₃	E ₂	E ₁	E ₀
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

Step 2: K-Map for Excess-3

		B ₁ B ₀			
		00	01	11	10
B ₃ B ₂	00	1	0	0	1
	01	1	0	0	1
	11	X	X	X	X
	10	1	0	X	X

$$E_0 = \overline{B_0}$$

K-Map for D_1

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$$D_1 = \overline{B_1} \overline{B_0} + B_1 B_0$$

$$= B_1 \oplus B_0$$

K-Map for D_2

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

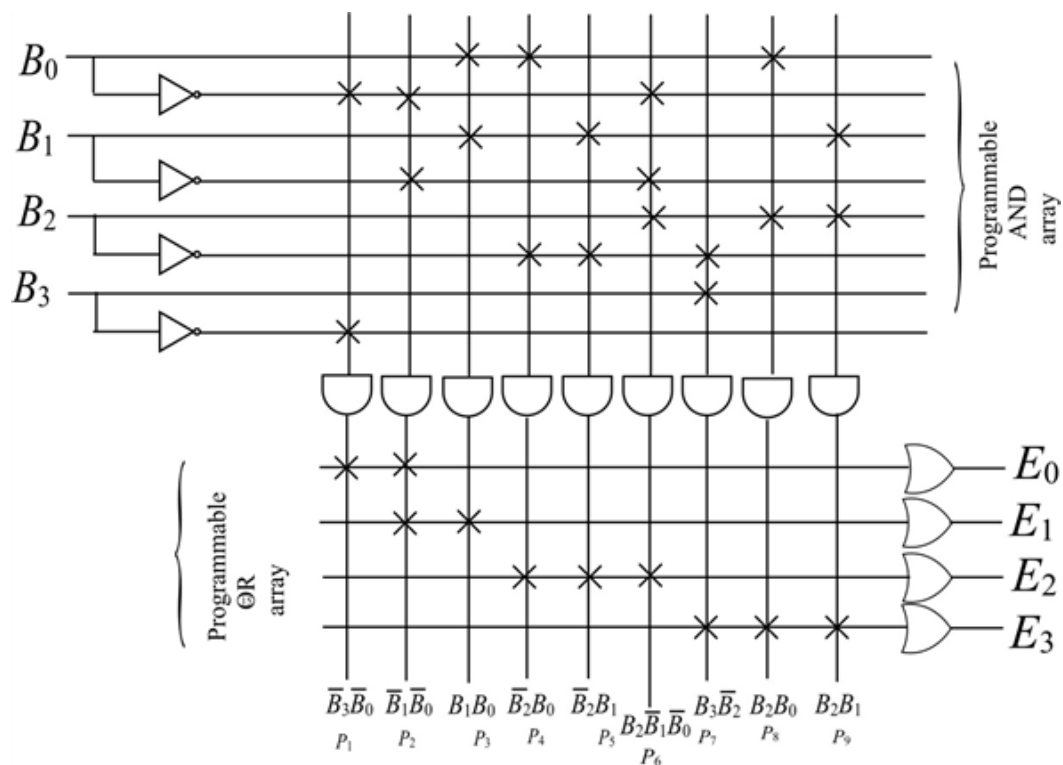
$$D_2 = B_2 \overline{B_1} \overline{B_0} + \overline{B_2} B_0 + \overline{B_2} B_1$$

K-Map for D_3

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$$D_3 = B_3 + B_2 B_0 + B_2 B_1$$

PLA Logic Diagram:



1. Define a memory cell. Give an example. (May-2007, Dec-2004, Dec-2006, Dec 2011)

Memories are made up of registers. Each register consists of storage elements, each of which stores one bit of data. Such a storage element is called memory cell. Example: RAM, ROM

2. Define a memory location. (May-2006)

Memories are made up of registers. Each register in the memory is one storage location also called memory location. Each memory location is identified by an address.

3. List basic types of programmable logic devices.

- Programmable Read only memory
- Programmable logic Array
- Programmable Array Logic

4. What is volatile memory? Give an example. (Dec 2007, Dec-2013)

The memory which cannot hold data when power is turned off is known as volatile memory.
Example: static RAM

5. What is non-volatile memory? (Dec 2013)

The memory which can hold data even when power is turned off is known as volatile memory. Example: ROM

6. What is ROM?

- A read only memory (ROM) is a device that includes both the decoder and the OR gates within a single IC package.
- The number of distinct addresses possible with n input variables is 2^n .

7. Define address and word.

In a ROM, each bit combination of the input variable is called an address.
Each bit combination that comes out of the output lines is called a word.

8. Name the types of ROM. (May 2011, Dec 2011)

There are four types of ROM.

- Masked ROM (Masked Read only memory)
- PROM (Programmable ROM)
- EPROM (Erasable Programmable ROM)
- EEPROM or E2 PROM (Electrically Erasable PROM)

9. What is meant by static and dynamic memories?**(May 2006, May 2007, Dec 2006)**

- Volatile memories which can hold data as long as power is ON are called static RAM (SRAM).
- Dynamic RAM (DRAM) stores data as a charge on the capacitor and they need refreshing of charge on a capacitor after every millisecond to hold data even if power is ON.

10. What is programmable logic array? How it differs from ROM?

A PLA is similar to a ROM in concept, however it does not provide full decoding of the variables and does not generate all the minterms as in the ROM.

11. What is PROM?**(Nov-2016)**

PROM (Programmable Read Only Memory).

It allows user to store data or program. PROMs use the fuses with material like nichrome and polycrystalline. The user can blow these fuses by passing around 20 to 50 mA of current for the period 5 to 20 μ s. The blowing of fuses is called programming of ROM. The PROMs are one time programmable. Once programmed, the information is stored permanent.

12. What is EPROM?**(Nov-2016)**

EPROM (Erasable Programmable Read Only Memory)

- ✓ This can be restructured to the initial state even though it has been programmed previously.
- ✓ It is *erased* by placing under a special *ultraviolet light* for a given length of time.

13. What is EEPROM?

EEPROM (Electrically Erasable Programmable Read Only Memory)

- *Electrical signals* are used to erase the previously programmed connections *instead of ultraviolet light*.
- The advantage is that the device can be erased without removing it from its socket.

14. How individual location in an EEPROM programmed or erased?**(May 2006)**

It is electrically erasable memory, by activating particular row and column it is possible that individual can be programmed or erased.

15. What are the advantages of EPROM over PROM?

PROM is one time programmable but EPROM can be programmed multiple times. The data stored in it can be erased by exposing ultraviolet light by 15 to 20 minutes. Once erased, it can be reprogrammed.

16. Compare and contrast EEPROM and Flash memory.**(Dec 2014)**

S.No	EEPROM	Flash memory
1	Lower packaging density.	Higher packaging density.
2	Higher cost per bit.	Lower cost per bit.
3	Consumes higher power.	Consumes less power.
4	Erasing is done individual byte wise.	Erasing is done in one bulk operation.
5	Erasing and re-programming of EEPROM is slower.	Erasing and re-programming of Flash memory is faster.

17. What is RAM?**(Dec 2006)**

It stands for Random Access Memory (RAM). Read and write operations can be carried out.

It is volatile memory. It can hold data as power is ON.

18. Give the advantages of RAM.**(Dec-2013)**

Advantages of RAM are:

1. RAM is memory where we can read as well as write the data.
2. This memory can be accessed randomly.
3. Higher speed.

19. What is static memory?**(Dec 2006, /May 2007, Dec2011, May 2010)**

Memories that consist of circuits of retaining their state as long as power is applied are known as static memories.

20. Define dynamic RAM.**(May 2010)**

Dynamic RAMs use capacitors as storage elements and cannot retain data very long without capacitors being recharged by a process called refreshing.

21. What is the technique adopted by DRAMs?

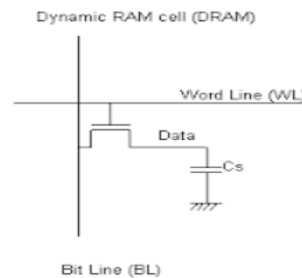
DRAMs use a technique called address multiplexing to reduce the number of address lines.

22. Compare and contrast static and Dynamic RAM.**(May 2012, Dec 2009)**

S.No	Static RAM	Dynamic RAM
1	Static RAM contains Less memory cells per unit area.	Dynamic RAM contains more memory cells per unit area.
2	Faster memory.	Slower memory compared with static memory.
3	Stores data in flip flops as single bit.	Stores data as charge in a capacitor.
4	Refreshing not required.	Periodical refreshing needed.
5	Cost is more.	Cost is less.

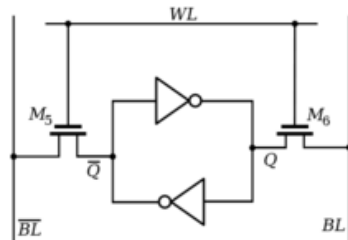
23. Draw the basic dynamic memory cell.

(Dec 2015)



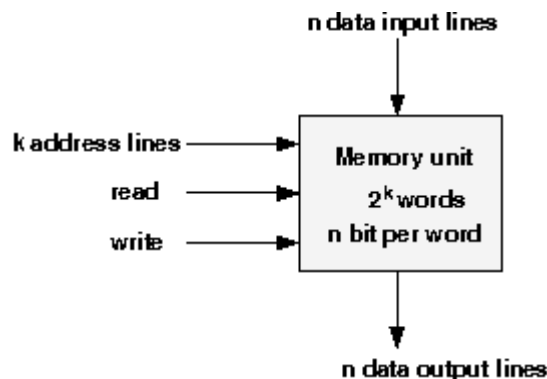
24. Draw the logic diagram of Memory cell.

(Dec 2007)



25. Draw the block diagram of dynamic Memory cell or RAM cell.

(Dec 2011, Dec 2008)



26. What is mask - programmable?

The user must submit a PLA program table to the manufacturer.

27. List the major differences between PLA and PAL

PLA:

Both AND and OR arrays are programmable and Complex Costlier than PAL

PAL

AND arrays are programmable OR arrays are fixed Cheaper and Simpler

28. Define PLD.

(May 2014)

Programmable Logic Devices consist of a large array of AND gates and OR gates that can be programmed to achieve specific logic functions.

29. Give the classification of PLDs.

(Dec 2017)(May 2013)

PLDs are classified as PROM(Programmable Read Only Memory), Programmable Logic Array(PLA), Programmable Array Logic (PAL), and Generic Array Logic(GAL).

30. Compare PLDs.

Type	AND array connections	OR array connections
PROM	Fixed at factory	Programmable by Customer
PLA	Programmable by Customer	Programmable by Customer
PAL	Programmable by Customer	Fixed at factory

31. Define PROM.

(Dec 2019)

PROM is Programmable Read Only Memory. It consists of a set of fixed AND gates connected to a decoder and a programmable OR array.

32. Define PLA.

(Dec 2019 (May 2008))

PLA is Programmable Logic Array (PLA). The PLA is a PLD that consists of a programmable AND array and a programmable OR array.

33. Define PAL.

(Dec 2009)

PAL is Programmable Array Logic. PAL consists of a programmable AND array and a fixed OR array with output logic.

34. Why was PAL developed?

(Dec 2009)

It is a PLD that was developed to overcome certain disadvantages of PLA, such as longer delays due to additional fusible links that result from using two programmable arrays and more circuit complexity.

35. Why the input variables to a PAL are buffered?

The input variables to a PAL are buffered to prevent loading by the large number of AND gate inputs to which available or its complement can be connected.

36. Compare PLA and PAL.

(Dec -10)(May 2011,May 2013,May 2017,Dec 2011)

S.No.	PLA	PAL
1	Both AND and OR arrays are programmable	OR array is fixed and AND array is programmable
2	Costliest and complex than PAL	Cheaper and simpler
3	AND array can be programmed to get desired minterms	AND array can be programmed to get desired minterms
4	Any Boolean functions in SOP form can be implemented using PLA	Any Boolean functions in SOP form can be implemented using PLA

37. List the configurable elements in the FPGA architecture.

The FPGA architecture consists of three types of configurable elements.

- Input / Output Blocks(IOBs)
- Configurable Logic Blocks(CLBs)

38. Differentiate ROM and PLD

S.No	ROM (Read only Memory)	PLD (Programmable Logic Array)
1.	It is a device that includes both the decoder and the OR gates with in a single IC package.	It is a device that includes both AND and OR gates within a single IC package.
2.	ROM does not full decoding of the variables and does generate all the minterms.	PLD does not provide full decoding of the variable and does not generate all the minterms.

Interpret about programmable logic array and infer how it differs from ROM. [NOV 2020]

39. Compare the features of PROM,PAL and PLA.

2018 (May 2009,May 2012)

S.No.	PROM	PLA	PAL
1	AND array is fixed and OR array is programmable	Both AND and OR arrays are programmable	OR array is fixed and AND array is programmable
2	Cheaper and simple to use	Costliest and complex than PAL	Cheaper and simpler
3	All minterms are decoded	AND array can be programmed to get desired minterms	AND array can be programmed to get desired minterms
4	Only Boolean functions in standard SOP form can be implemented using PROM	Any Boolean functions in SOP form can be implemented using PLA	Any Boolean functions in SOP form can be implemented using PLA

40. Give the comparison between PROM and PLA.

S.No	PROM	PLA
1	AND array is fixed and OR array is programmable.	Both AND and OR arrays are Programmable.
2	Cheaper and simple to use.	Costliest and complex than PROM.

41. What is FPGA?

(Dec 2016)

FPGA stands for Field Programmable Gate Array, which is the next generation in the programmable PLDs. The word 'field' refers to the ability of the gate arrays to be programmed for a specific function by the end user. The word 'array' indicates a *series of columns and rows of gates that can be programmed* by the end user.

42. What is volatile and Non-Volatile memory?

(Dec 2013)

In volatile memory the contents present in the memory *will be lost* when the power is removed.

In Non-volatile memory the contents present in the memory *is not lost* when the power is removed.

43. What is access time and cycle time?

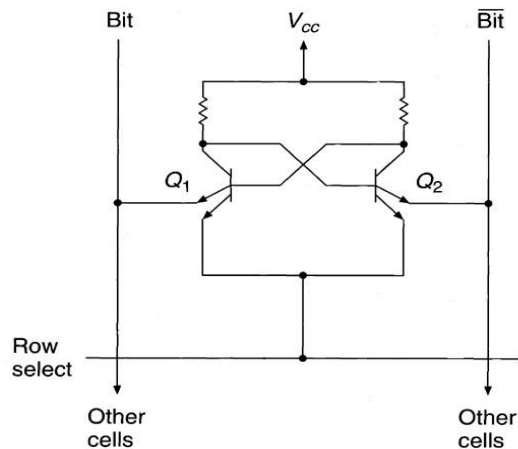
(Dec 2010)

The *access time* of memory is the time required to *select a word and read it*.

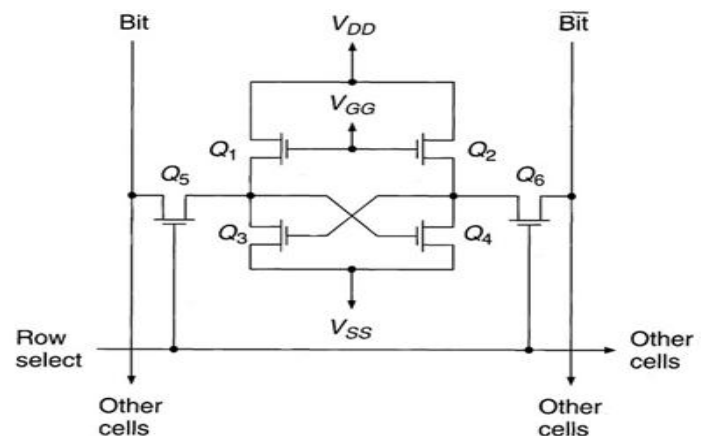
The *cycle time* of memory is the time required to complete a *write operation*.

44. Draw the logic diagram of static and Bipolar RAM cell.

(May 2014) (Dec 2012)



a. Bipolar SRAM cell



b. MOS SRAM cell

45. What are the advantages of static RAM and Dynamic Ram? [April/May-2010,Nov/Dec-2009]

Static RAM:

- ✓ Access time is less.
- ✓ Fast operation.

Dynamic Ram

- ✓ It consumes less power.
- ✓ Cost is low.

46. Mention few applications of PLA and PAL.

[April/May-2012]

- ✓ Implement combinational circuits
- ✓ Implement sequential circuits
- ✓ Code converters
- ✓ Microprocessor based systems

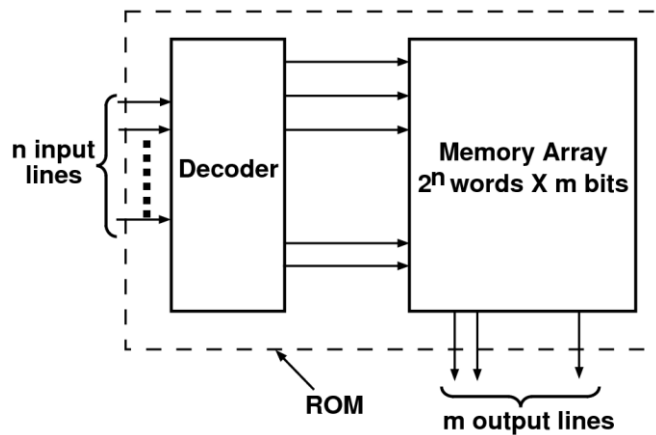
47. List the advantages of PLDs.

[April/May-2014, Nov/Dec-2010]

- ✓ Low and fixed (two gate) propagation delays (typically down to 5 ns),
- ✓ Simple,
- ✓ Low-cost (free),
- ✓ Design tools.

48. Implement a 2-bit multiplier using ROM.

[Nov/Dec-2010]



49. What is the memory capacity of RAM if it has 10bit address lines? (Dec 2017)

$$\text{Memory capacity} = 2^n = 2^{10}$$

N= address lines

Hence, 1024 bytes is the memory capacity of 10 bit address lines.

50. How does ROM retain information?

(May 2017)

A ROM chip is a non-volatile storage medium, which means it does not require a constant source of power to retain the information stored on it.

51. Define the term Fan out.

[NOV/DEC 2021]

[Nov/Dec-2011]

[Dec 2019]

It is the maximum number of inputs which have same family that the gate can drive maintaining its output within the specified limits.

52. What is the significance of high impedance state in tri-state gates?

[Nov/Dec-2010]

- High impedance state of a three-state gate provides a special feature not available in other gates.
- Because of this features a larger number of three state gate output can be connected with wires to form a common line without endangering loading effects.

53. Write a note on tri-state gates.

[April/May-2015]

It is a digital circuit that exhibits three states. Two of the states are signals equivalent to logic 1 and logic 0. The third state is high impedance state. High impedance state behaves like an open circuit.

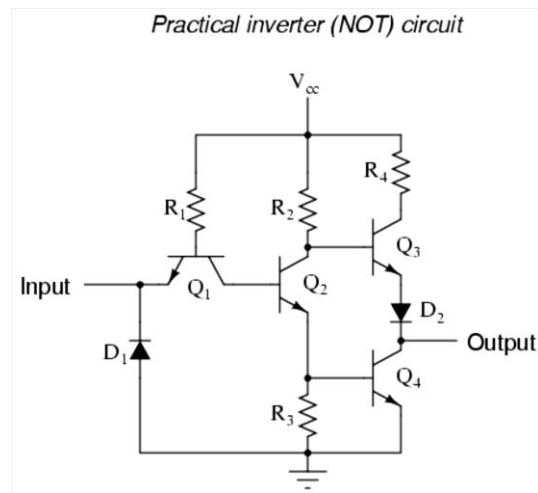
54. State the advantages of CMOS logic.

(Dec 2019 [April/May-2015])

- Consumes less power.
- Operated at high voltages, resulting in improved noise immunity.
- Fan-out is more.
- Better noise margin.

55. Draw the TTL Inverter (NOT) Circuit.

[April/May-2012]



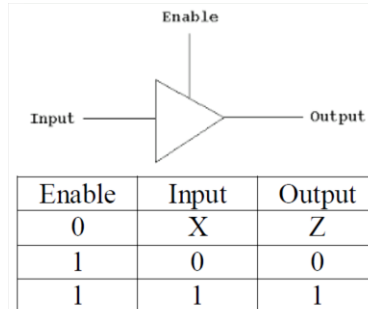
56. What is a totem pole output?

[April/May-2011]

Totem pole output is a standard output of a TTL gate. It is specifically designed to reduce the propagation delay in the circuit and to provide sufficient output power for high fan-out.

57. Draw an active-high tri-state buffer and write its truth table.

[April/May-2010]



58. What is TTL logic?

It is a logic family implemented with bipolar process technology that combines or integrates NPN transistors, PN junction diodes and diffused resistors in a single monolithic structure to get the desired logic function.

59. Interpret Read and Write Operation.

(Dec 2018)

The process of storing new information into memory is referred to as a memory “write” operation.

The process of transferring the stored information out of memory is referred to as a memory “read” operation.

- 60. A Standard TTL gate has the following current specifications: $I_{OH} = 400\mu A$, $I_{IH} = 40\mu A$, $I_{OH} = 16mA$, $I_{IL} = 1.6mA$. Calculate the fanout. (May 2019)**

Sol:

$$(\text{Fanout})_h = \frac{I_{OH}}{I_{IH}} = \frac{400\mu A}{40\mu A} = 10$$

$$(\text{Fanout})_i = \frac{I_{OL}}{I_{IL}} = \frac{16mA}{1.6mA} = 10$$

- 61. A DRAM chip uses two dimensional address multiplexing. It has 13 common address pins with the row address having one bit more than the column address. What is the capacity of the memory? [NOV/DEC 2021] (May 2019)**

Sol:

n = 13 address pins

Memory capacity = $2^n = 2^{13} = 8192$ bytes

- 62. What do you mean by propagation delay and noise margin ? [NOV 2020]**

Noise immunity or the noise margin is the limit of noise voltage that may appear at the input of the logic gate without any impairment of its proper logic operation.

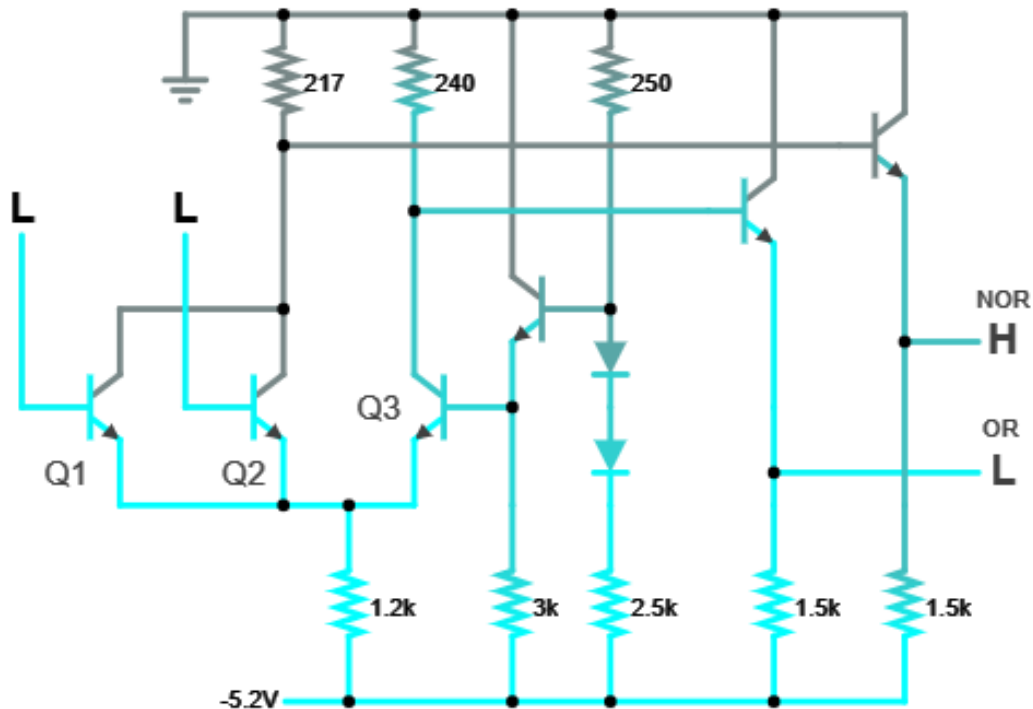
Propagation delay is defined as the time taken for the output of a logic gate to change after the inputs have changed.

- 63. Define the term 'fan-in'. [NOV/DEC 2021]**

The maximum number of inputs that can be connected to a logic gate without any impairment of its normal operation is referred to as fan in.

Explain the working principle of ECL OR/NOR gate with circuit diagram. [NOV 2020]

ECL OR/NOR gate:



This is a NOR/OR gate using emitter-coupled logic, a high-speed type of logic using transistors. The two inputs are shown at left. If either one of them is high (-700 mV), then the OR output is high, and the NOR output is low. If they are both low (-1.4V), then the OR is low, and NOR is high.

Q3's base voltage is fixed at a level where there is enough base current to get Q3 to conduct. This brings Q3's collector down to about 740 mV, which brings the OR output low (through a follower attached to Q3's collector). Q3's emitter is high enough relative to Q2's base that Q2 can't conduct, so Q2's collector stays at ground. This keeps the NOR output high (through a follower).

If either of the two inputs is high, then the corresponding transistor conducts. This brings Q1/Q2's collector low, which brings the NOR output low. It also brings Q1/Q2's emitter high enough so that Q3 can't conduct, which brings the OR output high.

The advantage of ECL is speed, because the transistors are never in saturation. They are either in cutoff or forward-active mode; transistors can switch between these two states quickly. The disadvantage is that there is always a lot of current, and therefore power consumption.

Analyze the working principle and characteristics of CMOS (i) inverter (ii) NAND gate (iii) NOR gate with circuit diagram. [NOV 2020]

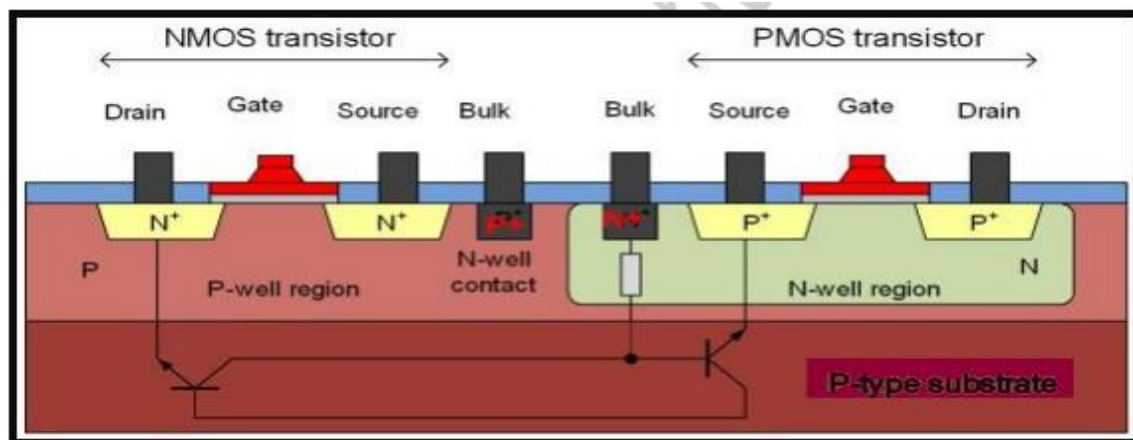
CMOS Working Principle and Applications

- The term CMOS stands for “Complementary Metal Oxide Semiconductor”.
- CMOS technology is one of the most popular technology in the computer chip design industry and broadly used today to form integrated circuits in numerous and varied applications.
- Today's computer memories, CPUs and cell phones make use of this technology due to several key advantages.
- This technology makes use of both P channel and N channel semiconductor devices. One of the most popular MOSFET technologies available today is the Complementary MOS or CMOS technology.
- This is the dominant semiconductor technology for microprocessors, microcontroller chips, memories like RAM, ROM, EEPROM and application specific integrated circuits (ASICs).

CMOS (Complementary Metal Oxide Semiconductor)

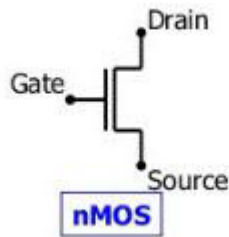
- The main advantage of CMOS over NMOS and BIPOLAR technology is the much smaller power dissipation.

- Unlike NMOS or BIPOLAR circuits, a Complementary MOS circuit has almost no static power dissipation.
- Power is only dissipated in case the circuit actually switches.
- This allows integrating more CMOS gates on an IC than in NMOS or bipolar technology, resulting in much better performance.
- Complementary Metal Oxide Semiconductor transistor consists of P-channel MOS (PMOS) and N-channel MOS (NMOS).



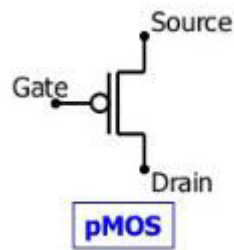
NMOS

- NMOS is built on a p-type substrate with n-type source and drain diffused on it.
- In NMOS, the majority carriers are electrons.
- When a high voltage is applied to the gate, the NMOS will conduct.
- Similarly, when a low voltage is applied to the gate, NMOS will not conduct.
- NMOS are considered to be faster than PMOS, since the carriers in NMOS, which are electrons, travel twice as fast as the holes.



PMOS

- P- channel MOSFET consists P-type Source and Drain diffused on an N-type substrate.
- Majority carriers are holes. When a high voltage is applied to the gate, the PMOS will not conduct.
- When a low voltage is applied to the gate, the PMOS will conduct.
- The PMOS devices are more immune to noise than NMOS devices.

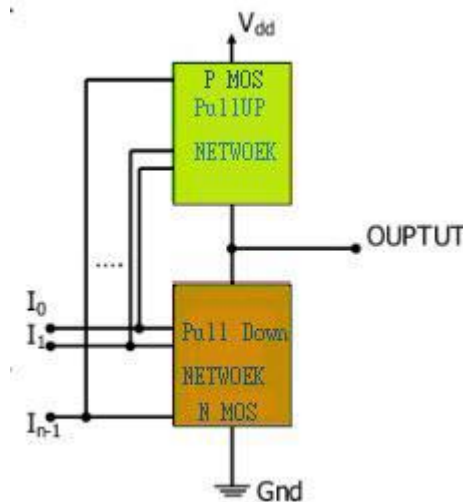


CMOS Working Principle

- In CMOS technology, both N-type and P-type transistors are used to design logic functions.
- The same signal which turns ON a transistor of one type is used to turn OFF a transistor of the other type.
- This characteristic allows the design of logic devices using only simple switches, without the need for a pull-up resistor.
- In CMOS logic gates a collection of n-type MOSFETs is arranged in a pull-down network between the output and the low voltage power supply rail (V_{ss} or quite often ground). Instead of

the load resistor of NMOS logic gates, CMOS logic gates have a collection of p-type MOSFETs in a pull-up network between the output and the higher-voltage rail (often named V_{dd}).

- Thus, if both a p-type and n-type transistor have their gates connected to the same input, the p-type MOSFET will be ON when the n-type MOSFET is OFF, and vice-versa.
- The networks are arranged such that one is ON and the other OFF for any input pattern as shown in the figure below.

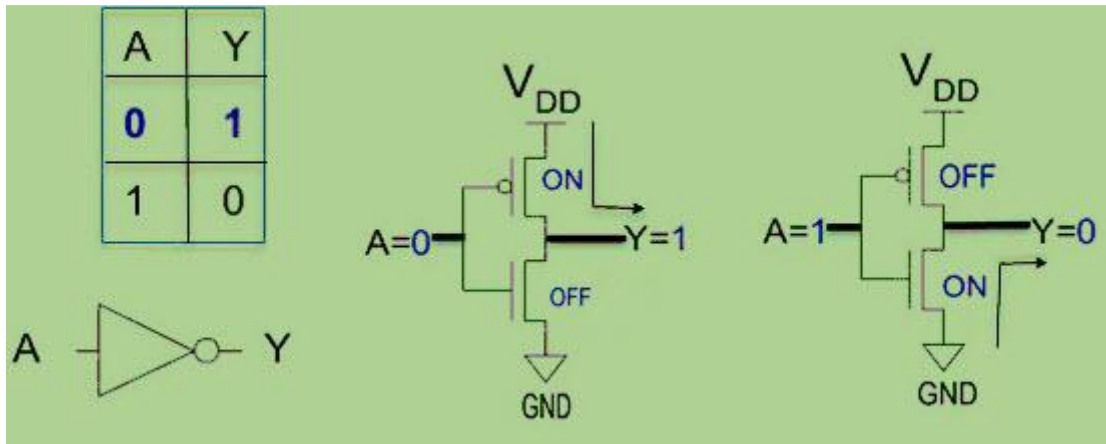


CMOS Logic Gate using Pull-Up and Pull-Down Networks

- CMOS offers relatively high speed, low power dissipation, high noise margins in both states, and will operate over a wide range of source and input voltages (provided the source voltage is fixed).
- Furthermore, for the better understanding of the Complementary Metal Oxide Semiconductor working principle, we need to discuss in brief about CMOS logic gates as explained below.

CMOS Inverter

- The inverter circuit as shown in the figure below. It consists of PMOS and NMOS FET. The input A serves as the gate voltage for both transistors.



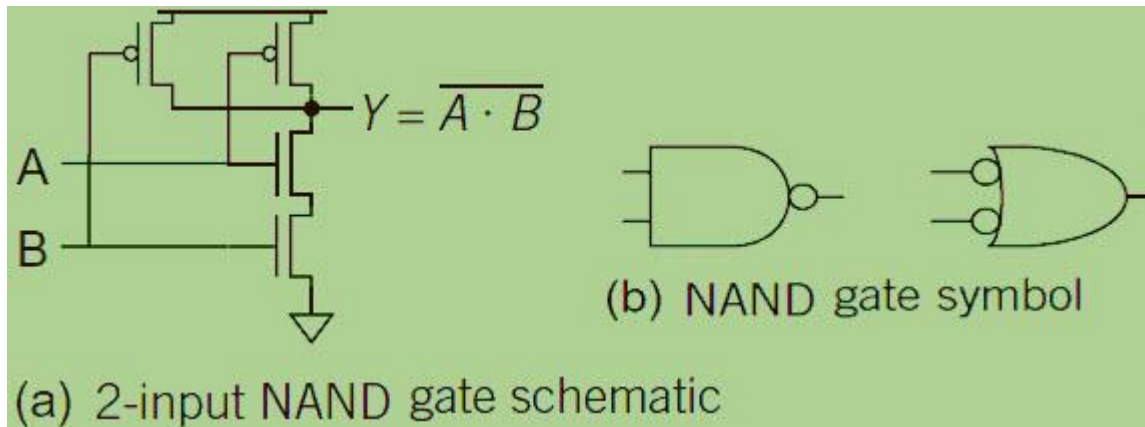
CMOS Inverter

- The NMOS transistor has an input from V_{ss} (ground) and PMOS transistor has an input from V_{dd}. The terminal Y is output. When a high voltage ($\sim V_{dd}$) is given at input terminal (A) of the inverter, the PMOS becomes open circuit and NMOS switched OFF so the output will be pulled down to V_{ss}.
- When a low-level voltage ($< V_{dd}$, $\sim 0v$) applied to the inverter, the NMOS switched OFF and PMOS switched ON. So the output becomes V_{dd} or the circuit is pulled up to V_{dd}.

INPUT	LOGIC INPUT	OUTPUT	LOGIC OUTPUT
0 v	0	V _{dd}	1
V _{dd}	1	0 v	0

CMOS NAND Gate

The below figure shows a 2-input Complementary MOS NAND gate. It consists of two series NMOS transistors between Y and Ground and two parallel PMOS transistors between Y and V_{DD}.



CMOS NAND Gate

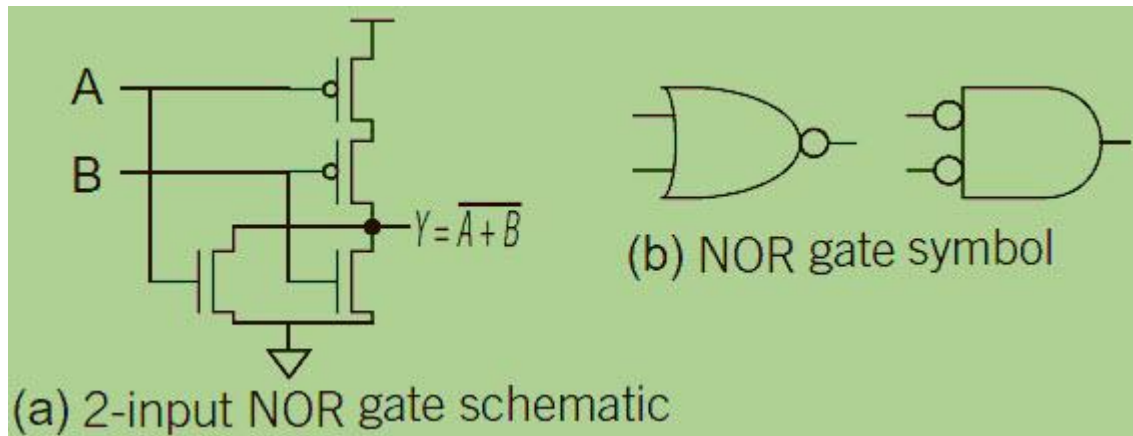
If either input A or B is logic 0, at least one of the NMOS transistors will be OFF, breaking the path from Y to Ground. But at least one of the pMOS transistors will be ON, creating a path from Y to VDD.

Hence, the output Y will be high. If both inputs are high, both of the nMOS transistors will be ON and both of the pMOS transistors will be OFF. Hence, the output will be logic low. The truth table of NAND logic gate is given in below table.

A	B	Pull-Down Network	Pull-up Network	OUTPUT Y
0	0	OFF	ON	1
0	1	OFF	ON	1
1	0	OFF	ON	1
1	1	ON	OFF	0

CMOS NOR Gate

A 2-input NOR gate is shown in the figure below. The NMOS transistors are in parallel to pull the output low when either input is high. The PMOS transistors are in series to pull the output high when both inputs are low, as given in below table. The output is never left floating.



Complementary MOS NOR Gate

The truth table of NOR logic gate given in below table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

CMOS Applications

Complementary MOS processes were widely implemented and have fundamentally replaced NMOS and bipolar processes for nearly all digital logic applications. The CMOS technology has been used for the following digital IC designs.

- Computer memories, CPUs
- Microprocessor designs
- Flash memory chip designing
- Used to design application-specific integrated circuits (ASICs)

Derive the PLA programming table for the four Boolean functions listed below. Minimize the numbers of product terms,

$$A(x, y, z) = \Sigma(1, 2, 4, 6)$$

$$B(x, y, z) = \Sigma(0, 1, 6, 7)$$

$$C(x, y, z) = \Sigma(2, 6)$$

$$D(x, y, z) = \Sigma(1, 2, 3, 5, 7)$$

[NOV/DEC 2021]

		y			
		00	01	11	10
x	yz				
	0	m_0 0	m_1 1	m_3 0	m_2 1
1	1	m_4 1	m_5 0	m_7 0	m_6 1

$$A = yz' + xz' + x'y'z$$

$$A' = yz + xz + x'y'z'$$

		y			
		00	01	11	10
x	yz				
	0	m_0 1	m_1 1	m_3 0	m_2 0
1	1	m_4 0	m_5 0	m_7 1	m_6 1

$$B = xy + x'y'$$

$$B' = x'y' + x'y$$

		y			
		00	01	11	10
x	yz				
	0	m_0 0	m_1 0	m_3 0	m_2 1
1	1	m_4 0	m_5 0	m_7 0	m_6 1

$$C = yz'$$

$$C' = y' + z$$

		y			
		00	01	11	10
x	yz				
	0	m_0 0	m_1 1	m_3 1	m_2 1
1	1	m_4 0	m_5 1	m_7 1	m_6 0

$$D = z + x'y$$

$$D' = y'z' + xz'$$

Product Inputs			Outputs				
term	x	y	z	A	B	C	D
yz'	1	-	1	0	1	-	-
xz'	2	1	-	0	1	-	-
$x'y'z$	3	0	0	1	-	-	-
xy'	4	1	0	-	1	-	-
$x'y$	5	0	1	-	1	-	1
z	6	-	-	1	-	-	1
				T	C	T	T

PLA programming table is given below:

S.No.	Product Term	Inputs			Outputs			
		x	y	z	A	B	C	D
1	$\bar{x}\bar{y}z$	0	0	1	1	-	-	-
2	$x\bar{z}$	1	-	0	1	-	-	1
3	$y\bar{z}$	-	1	0	1	-	1	-
4	$\bar{x}\bar{y}$	0	0	-	-	1	-	-
5	xy	1	1	-	-	1	-	-
6	$\bar{y}\bar{z}$	-	0	0	-	-	-	1
					T	T	T	C

PLA implementation is shown below:

