Department of Computer Science and Engineering

Subject Name: Digital Principles and System Design

Syllabus

CS 34 DIGITAL PRINCIPLES AND SYSTEM DESIGN
(Common to CSE & IT)

AIM

To provide an in-depth knowledge of the design of digital circuits and the use of Hardware Description Language in digital system design.

OBJECTIVES

To understand different methods used for the simplification of Boolean functions
To design and implement combinational circuits
To design and implement synchronous sequential circuits
To design and implement asynchronous sequential circuits
To study the fundamentals of VHDL / Verilog HDL

UNIT I BOOLEAN ALGEBRA AND LOGIC GATES
Review of binary number systems - Binary arithmetic – Binary codes – Boolean algebra and theorems - Boolean functions – Simplifications of Boolean functions using Karnaugh map and tabulation methods – Implementation of Boolean functions using logic gates.

UNIT II COMBINATIONAL LOGIC
Combinational circuits – Analysis and design procedures - Circuits for arithmetic operations - Code conversion – Introduction to Hardware Description Language (HDL)

UNIT III DESIGN WITH MSI DEVICES
Decoders and encoders - Multiplexers and demultiplexers - Memory and programmable logic - HDL for combinational circuits

UNIT IV SYNCHRONOUS SEQUENTIAL LOGIC
Sequential circuits – Flip flops – Analysis and design procedures - State reduction and state assignment - Shift registers – Counters – HDL for Sequential Circuits.

UNIT V ASYNCHRONOUS SEQUENTIAL LOGIC
Analysis and design of asynchronous sequential circuits - Reduction of state and flow tables – Race-free state assignment – Hazards, ASM Chart.

TEXT BOOKS

1. M.Morris Mano, “Digital Design”, 3rd edition, Pearson Education, 2007.

REFERENCES

1. Charles H.Roth, Jr. “Fundamentals of Logic Design”, 4th Edition, Jaico Publishing House, Cengage Earning, 5th ed, 2005.
2. Donald D.Givone, “Digital Principles and Design”, Tata McGraw-Hill, 2007

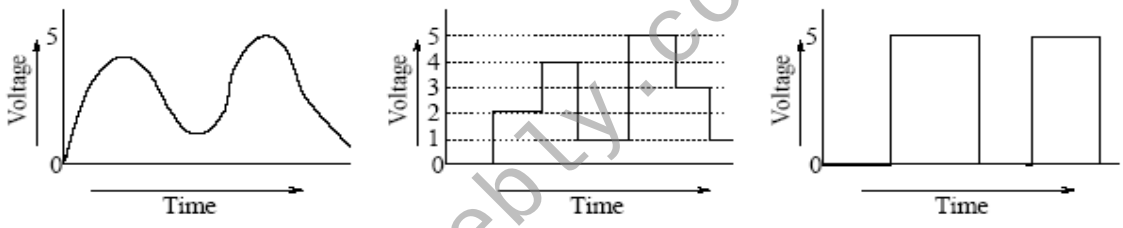
OBJECTIVES

- To understand different methods used for the simplification of Boolean functions
- To understand and implement Logic Gates
- Introduction to Combinational Logic
    - *Motivation*
    - *Switches and logic gates*
    - *Logic functions, truth tables and variables*
    - *Boolean axioms and laws, sum of products, product of sums*
    - *Simple algebraic minimization - making things cheaper*
- To design and implement Multiplexer and Demultiplexer
- To design and implement Decoder and Encoder
- To design and implement Magnitude Comparator
- To design and implement BCD adder
- To design and implement Memory
- To design and implement PLD's
- Sequential Logic
    - *Distinction between combinatorial and sequential logic (gates and flip-flops)*
    - *SR Flip-flop*
    - *Clocked D Flip-flop*
    - *Master-Slave JK flip-flop*
    - *Clocked T Flip-flop*
    - *Shift registers*
    - *Counters*
- To design and implement synchronous counters
- To design and implement Asynchronous counters
- To design and implement synchronous sequential circuits
    - *How logic is controlled*
    - *Mealy/Moore state machines*
    - *State diagrams & state tables*
- To design and implement Asynchronous sequential circuits
    - *Comparison of synchronous/asynchronous circuits*
    - *Dealing with asynchronous inputs*
    - *Asynchronous hazards*
- To study the fundamental of Races
- To study the fundamental of ASM Chart
- To study the fundamentals of VHDL / Verilog HDL

Digital System

- The term digital refers to any process that is accomplished using discrete units
- Digital computer is the best example of a digital system.
- Basically deal with two types of signals in electronics
  - i) Analog
  - ii) Digital

Digital versus Analog (contd.)



Analog Signal | Digital Signal | Logic (Binary) Signal

- Almost all digital circuits are really logic ckts–WHY?
- Answers:
  1. It is much easier to manipulate logic values (i.e., 0/1 or low/high values) than to manipulate and process multiple (discrete) voltage levels
  2. One can use the formal laws of logic to design logic ckts easily and systematically—these laws are called *Boolean algebra* or *switching algebra*

Types of Number Systems are

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

Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. There are two types of complements

- i) r's complement
- ii) (r-1)'s complement.

Binary Codes

Dec	Hex	Oct	Bin
0	0	000	00000000
1	1	001	00000001
2	2	002	00000010
3	3	003	00000011
4	4	004	00000100
5	5	005	00000101
6	6	006	00000110
7	7	007	00000111
8	8	010	00001000
9	9	011	00001001
10	A	012	00001010
11	B	013	00001011
12	C	014	00001100
13	D	015	00001101
14	E	016	00001110
15	F	017	00001111

Dec	Hex	Oct	Bin
16	10	020	00010000
17	11	021	00010001
18	12	022	00010010
19	13	023	00010011
20	14	024	00010100
21	15	025	00010101
22	16	026	00010110
23	17	027	00010111
24	18	030	00011000
25	19	031	00011001
26	1A	032	00011010
27	1B	033	00011011
28	1C	034	00011100
29	1D	035	00011101
30	1E	036	00011110
31	1F	037	00011111

Binary to Decimal

Binary	Decimal
11011_2	
$2^4+2^3+0^1+2^1+2^0$	=16+8+0+2+1
Result	27_{10}

Decimal 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

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

Decimal to octal

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		$= 010110001_2$

Decimal to Hexadecimal

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

Hexadecimal to binary

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

Hexadecimal to octal

Hexadecimal	Binary/Octal
5A816	= **0101** 1010 **1000** (Binary)
	= **010** 110 **101** 000 (Binary)
Result	= 2 6 5 0 (Octal)

Binary Arithmetic

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

For example,

`00011010 + 00001100 = 00100110`

```
     1 1                    carries
   0 0 0 1 1 0 1 0 = 26(base 10)
+  0 0 0 0 1 1 0 0 = 12(base 10)
-----------------
   0 0 1 0 0 1 1 0 = 38(base 10)
```

`00010011 + 00111110 = 01010001`

```
   1 1 1 1 1              carries
   0 0 0 1 0 0 1 1 = 19(base 10)
+  0 0 1 1 1 1 1 0 = 62(base 10)
-----------------
   0 1 0 1 0 0 0 1 = 81(base 10)
```

Note: The rules of binary addition (without carries) are the same as the truths of the **XOR** gate.

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`

For example,

`00100101 - 00010001 = 00010100`

```
     0                      borrows
   0 0 ~~1~~ 1 0 0 1 0 1 = 37(base 10)
-  0 0 0 1 0 0 0 1 = 17(base 10)
-----------------
   0 0 0 1 0 1 0 0 = 20(base 10)
```

`00110011 - 00010110 = 00011101`

```
     0 1 0 1                borrows
   0 0 ~~1~~ ~~1~~ ~~0~~ 1 0 1 1 = 51(base 10)
-  0 0 0 1 0 1 1 0 = 22(base 10)
-----------------
   0 0 0 1 1 1 0 1 = 29(base 10)
```

Rules of Binary Multiplication

- `0 x 0 = 0`
- `0 x 1 = 0`

- 1 x 0 = 0
- 1 x 1 = 1, and no carry or borrow bits

For example,

00101001 × 00000110 = 11110110

		0	0	1	0	1	0	0	1	= $41_{(base\ 10)}$
	×	0	0	0	0	0	1	1	0	= $6_{(base\ 10)}$
		0	0	0	0	0	0	0	0	
	0	0	1	0	1	0	0	1		
0	0	1	0	1	0	0	1			
0	0	1	1	1	1	0	1	1	0	= $246_{(base\ 10)}$

00010111 × 00000011 = 01000101

		0	0	0	1	0	1	1	1	= $23_{(base\ 10)}$
	×	0	0	0	0	0	0	1	1	= $3_{(base\ 10)}$
					1	*1*	*1*	*1*	*1*	*carries*
		0	0	0	1	0	1	1	1	
	0	0	0	1	0	1	1	1		
0	0	1	0	0	0	1	0	1		= $69_{(base\ 10)}$

Note: The rules of binary multiplication are the same as the truths of the **AND** gate.

Another Method: Binary multiplication is the same as repeated binary addition; add the multicand to itself the multiplier number of times.

For example,

00001000 × 00000011 = 00011000

	1								*carries*
	0	0	0	0	1	0	0	0	= $8_{(base\ 10)}$
	0	0	0	0	1	0	0	0	= $8_{(base\ 10)}$
+	0	0	0	0	1	0	0	0	= $8_{(base\ 10)}$
	0	0	0	1	1	0	0	0	= $24_{(base\ 10)}$

Binary Division

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

For example,

00101010 ÷ 00000110 = 1 1 1 = $7_{(base\ 10)}$

```
= 00000111

                _________________
          1 1 0 ) 0 0 1 10 1 0 1 0    =  42(base 10)
                    -   1  1 0        =   6(base 10)
                  ___________
                        1                 borrows
                      1 0 10 1
                    -   1  1 0
                    ____________
                           1 1 0
                         - 1 1 0
                         _______
                               0

10000111 ÷ 00000101                 1 1 0 1 1  =  27(base 10)
         = 00011011
                _________________
          1 0 1 ) 1 0 0 10 0 1 1 1  =  135(base 10)
                - 1 0 1             =    5(base 10)
                ________
                    1 1 10
                  - 1 0 1
                  _______
                        1 1
                      -   0
                      _____
                        1 1 1
                      - 1 0 1
                      _______
                          1 0 1
                        - 1 0 1
                        _______
                              0
```


Binary Number System

System Digits: 0 and 1

Bit (short for *b*inary dig*it*): A single binary digit

LSB (least significant bit): The rightmost bit

MSB (most significant bit): The leftmost bit

Upper Byte (or nybble): The right-hand byte (or nybble) of a pair

Lower Byte (or nybble): The left-hand byte (or nybble) of a pair

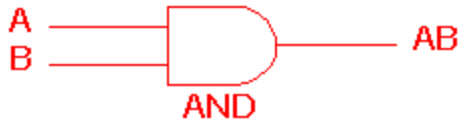
Binary Equivalents

1 Nybble (or nibble) = 4 bits
1 Byte = 2 nybbles = 8 bits
1 Kilobyte (KB) = 1024 bytes
1 Megabyte (MB) = 1024 kilobytes = 1,048,576 bytes
1 Gigabyte (GB) = 1024 megabytes = 1,073,741,824 bytes

Logic gates

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

AND gate



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

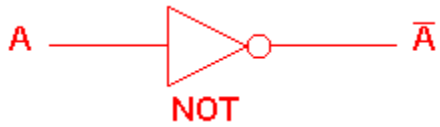
The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB

OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.

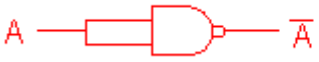
NOT gate

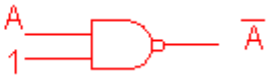


NOT gate	
A	$\overline{A}$
0	1
1	0

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the

inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The diagrams below show two ways that the NAND logic gate can be configured to produce a NOT gate. It can also be done using NOR logic gates in the same way.





NAND gate

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

NOR gate

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

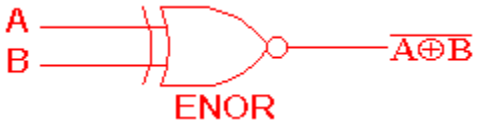
This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high.
The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

EXOR gate

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EOR operation.

EXNOR gate



2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

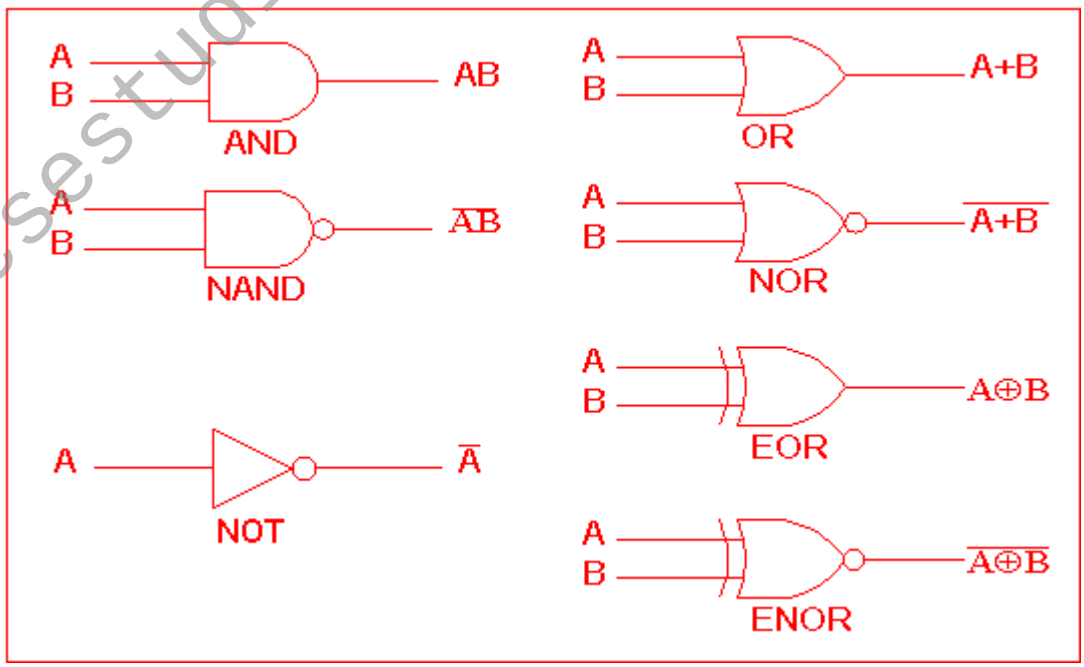
The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.

Note:

A function in *sum of products* form can be implemented using NAND gates by replacing all AND and OR gates by NAND gates.

A function in *product of sums* form can be implemented using NOR gates by replacing all AND and OR gates by NOR gates.

Table 1: Logic gate symbols



Types of binary codes

Binary codes are codes which are represented in binary system with modification from the original ones. Below we will be seeing the following: Weighted codes and Non-Weighted codes

Weighted binary codes

Weighted binary codes are those which obey the positional weighting principles, each position of the number represents a specific weight. The binary counting sequence is an example.

0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0010	0011	0101
3	0011	0011	0101	0110
4	0100	0100	0111	0111
5	0101	1011	1000	1000
6	0110	1100	1010	1001
7	0111	1101	1100	1010
8	1000	1110	1110	1011
9	1001	1111	1111	1100

8421 code/BCD code

The BCD (Binary Coded Decimal) is a straight assignment of the binary equivalent. It is possible to assign weights to the binary bits according to their positions. The weights in the BCD code are 8,4,2,1.

Example: The bit assignment 1001, can be seen by its weights to represent the decimal 9 because 1x8+0x4+0x2+1x1 = 9

2421 code

This is a weighted code; its weights are 2, 4, 2 and 1. A decimal number is represented in 4-bit form and the total four bits weight is 2 + 4 + 2 + 1 = 9. Hence the 2421 code represents the decimal numbers from 0 to 9.

5211 code

This is a weighted code; its weights are 5, 2, 1 and 1. A decimal number is represented in 4-bit form and the total four bits weight is 5 + 2 + 1 + 1 = 9. Hence the 5211 code represents the decimal numbers from 0 to 9.

Reflective code

A code is said to be reflective when code for 9 is complement for the code for 0, and so is for 8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not.

Sequential code

A code is said to be sequential when two subsequent codes, seen as numbers in binary representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

Non-Weighted code

Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value.

Excess-3 code

Excess-3 is a non weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3).

Example: 1000 of 8421 = 1011 in Excess-3

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. The Gray code is non-weighted code, as the position of bit does not contain any weight. 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. In digital Gray code has got a special place.

Decimal Number	Binary Code	Gray Code
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

Error detecting and correcting codes

For reliable transmission and storage of digital data, error detection and correction is required. Below are a few examples of codes which permit error detection and error correction after detection.

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.

Parity bit

In parity codes, every data byte, or nibble (according to how user wants to use it) is checked if they have even number of ones or even number of zeros. Based on this information an additional bit is appended to the original data. Thus if we consider 8-bit data, adding the parity bit will make it 9 bit long.

At the receiver side, once again parity is calculated and matched with the received parity (bit 9), and if they match, data is ok, otherwise data is corrupt.

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 ones 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 number of ones is even then parity bit is set to 1.

Error correcting codes

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

The binary codes that can be used to represent all the letters of the alphabet, numbers and mathematical symbols, punctuation marks, are known as alphanumeric codes or character codes. These codes enable us to interface the input-output devices like the keyboard, printers, video displays with the computer.

ASCII codes

ASCII stands for American Standard Code for Information Interchange. It has become a world standard alphanumeric code for microcomputers and computers. It is a 7-bit code representing 2^7 = 128 different characters. These characters represent 26 upper case letters (A to Z), 26 lowercase letters (a to z), 10 numbers (0 to 9), 33 special characters and symbols and 33 control characters.

EBCDIC codes

EBCDIC stands for Extended Binary Coded Decimal Interchange. It is mainly used with large computer systems like mainframes. EBCDIC is an 8-bit code and thus accommodates up to 256 characters. An EBCDIC code is divided into two portions: 4 zone bits (on the left) and 4 numeric bits (on the right).

Simplify the Boolean functions using

i) Theorems and postulates.
ii) K-Map
iii) Tabulation methods

Boolean Algebra theorems and postulates.

Principle of duality

1. Interchanging the OR and AND operations of the expression.
2. Interchanging the 0 and 1 elements of the expression.
3. Not changing the form of the variables.

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
(a) $(\overline{A}) = \overline{A}$
(b) $(\overline{\overline{A}}) = A$

T6: Redundance Law
(a) $A + A\,B = A$
(b) $A\,(A + B) = A$

T7:
(a) $0 + A = A$
(b) $1\,A = A$
(c) $1 + A = 1$
(d) $0\,A = 0$

T8 :
(a) $\overline{A} + A = 1$
(b) $\overline{A}\,A = 0$

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

T10 : De Morgan's Theorem
(a) $(\overline{A + B}) = \overline{A}\ \overline{B}$
(b) $(\overline{A\,B}) = \overline{A} + \overline{B}$

Using theorems,

$$
\begin{aligned}
A + \overline{A}\,B &= A\,1 + \overline{A}\,B && \text{T7(a)}\\
&= A\,(1 + B) + \overline{A}\,B && \text{T7(c)}\\
&= A + A\,B + \overline{A}\,B && \text{T3(a)}\\
&= A + B\,(A + \overline{A}) && \text{T3(a)}\\
&= A + B && \text{T(8)}
\end{aligned}
$$

Using truth table,

A	B	$A+B$	$\overline{A}\,B$	$A+\overline{A}\,B$
0	0	**0**	0	**0**
0	1	**1**	1	**1**
1	0	**1**	0	**1**
1	1	**1**	0	**1**

What is a Karnaugh map?

A Karnaugh map provides a pictorial method of grouping together expressions with common factors and therefore eliminating unwanted variables. The Karnaugh map can also be described as a special arrangement of a truth table.

The diagram below illustrates the correspondence between the Karnaugh map and the truth table for the general case of a two variable problem.

A	B	F
0	0	a
0	1	b
1	0	c
1	1	d

Truth Table.

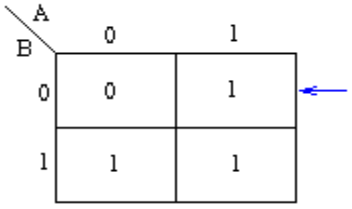
B \ A	0	1
0	a	b
1	c	d

F.

The values inside the squares are copied from the output column of the truth table, therefore there is one square in the map for every row in the truth table. Around the edge of the Karnaugh map are the values of the two input variable. A is along the top and B is down the left hand side. The diagram below explains this:

A	B	F
0	0	0
0	1	1
1	0	1 ←
1	1	1

Truth Table.

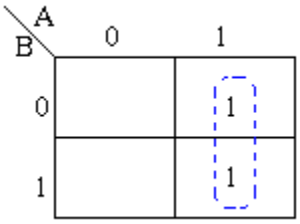


F.

The values around the edge of the map can be thought of as coordinates. So as an example, the square on the top right hand corner of the map in the above diagram has coordinates A=1 and B=0. This square corresponds to the row in the truth table where A=1 and B=0 and F=1. Note that the value in the F column represents a particular function to which the Karnaugh map corresponds.

Example 1:

Consider the following map. The function plotted is: $Z = f(A,B) = A\overline{B} + AB$



- Note that values of the input variables form the rows and columns. That is the logic values of the variables A and B (with one denoting true form and zero denoting false form) form the head of the rows and columns respectively.
- Bear in mind that the above map is a one dimensional type which can be used to simplify an expression in two variables.
- There is a two-dimensional map that can be used for up to four variables, and a three-dimensional map for up to six variables.

Using algebraic simplification,

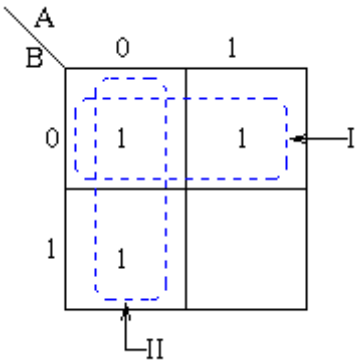
$Z = A\overline{B} + AB$
$Z = A(\overline{B} + B)$
$Z = A$

Variable B becomes redundant due to Boolean Theorem T9a.

Referring to the map above, the two adjacent 1's are grouped together. Through inspection it can be seen that variable B has its true and false form within the group. This eliminates variable B leaving only variable A which only has its true form. The minimised answer therefore is Z = A.

Example 2:

Consider the expression Z = f(A,B) = $\overline{A}\overline{B}$+ A $\overline{B}$+ $\overline{A}$B plotted on the Karnaugh map:



Pairs of 1's are *grouped* as shown above, and the simplified answer is obtained by using the following steps:

Note that two groups can be formed for the example given above, bearing in mind that the largest rectangular clusters that can be made consist of two 1s. Notice that a 1 can belong to more than one group.
The first group labelled I, consists of two 1s which correspond to A = 0, B = 0 and A = 1, B = 0. Put in another way, all squares in this example that correspond to the area of the map where B = 0 contains 1s, independent of the value of A. So when B = 0 the output is 1. The expression of the output will contain the term $\overline{B}$

For group labeled II corresponds to the area of the map where A = 0. The group can therefore be defined as $\overline{A}$. This implies that when A = 0 the output is 1. The output is therefore 1 whenever B = 0 and A = 0
Hence the simplified answer is Z = $\overline{A}$+ $\overline{B}$

The tabular method which is also known as the Quine-McCluskey method is particularly useful when minimising functions having a large number of variables, e.g. The six-variable functions. Computer programs have been developed employing this algorithm. The method reduces a function in standard sum of products form to a set of prime implicants from which as many variables are eliminated as possible. These prime implicants are then examined to see if some are redundant.

The tabular method makes repeated use of the law A + $\overline{A}$= 1. Note that Binary notation is used for the function, although decimal notation is also used for the functions. As usual a variable in true form is denoted by 1, in inverted form by 0, and the abscence of a variable by a dash (-).

Rules of Tabular Method

Consider a function of three variables f(A, B, C):

$\overline{A}\ B\ C$ is represented by 011 ← Binary notation, where A = 0, B= 1 and C= 1
$A\ \overline{B}\ \overline{C}$ is represented by 100
$A\ \overline{C}$ is represented by 1–0
$B\ C$ is represented by –11

Consider the function:

$$f(A, B, C, D) = \sum(1110, 1111) = A\ B\ C\ \overline{D} + A\ B\ C\ D = A\ B\ C$$

Listing the two minterms shows they can be combined

```
A B C D
1 1 1 0 ←┐
1 1 1 1 ←┘ Can combine (Differs in one digit position)
1 1 1 –
```

Now consider the following:

$$f(A, B, C, D) = \sum(1101, 1110) = A\ B\ \overline{C}\ D + A\ B\ C\ \overline{D}$$

Note that these variables cannot be combined

```
A B C D
1 1 0 1 ←┐
1 1 1 0 ←┘ Cannot combine (Differs in more than one digit position)
    ?
```

This is because the *FIRST RULE* of the Tabular method for two terms to combine, and thus eliminate one variable, is that they must differ in only **one digit** position.

Bear in mind that when two terms are combined, one of the combined terms has one digit more at logic 1 than the other combined term. This indicates that the number of 1's in a term is significant and is referred to as its index.

For example: f(A, B, C, D)

0000.....Index 0
0010, 1000.....Index 1
1010, 0011, 1001.....Index 2
1110, 1011.....Index 3
1111.....Index 4

The necessary condition for combining two terms is that the indices of the two terms must differ by one logic variable which must also be the same.

Examples

Example 1:

Consider the function: Z = f(A,B,C) = $\overline{A}\overline{B}\overline{C}$+ $\overline{A}\overline{B}$C + A$\overline{B}\overline{C}$ + A$\overline{B}$C

To make things easier, change the function into binary notation with index value and decimal value.

f(A, B , C) = $\sum$ (000, 001, 100, 101)————Binary notation

0 1 1 2————Index

0 1 4 5————Decimal value

Tabulate the index groups in a colunm and insert the decimal value alongside.

	First List				Second List				Third List			
		A	B	C		A	B	C		A	B	C
Index 0	0	0	0	0 ✓	0,1	0	0	– ✓	0,1,4,5	–	0	–
	1	0	0	1 ✓	0,4	–	0	0 ✓	0,4,1,5	–	0	–
Index 1	4	1	0	0 ✓	1,5	–	0	1 ✓				
Index 2	5	1	0	0 ✓	4,5	1	0	– ✓				

From the first list, we combine terms that differ by 1 digit only from one index group to the next. These terms from the first list are then seperated into groups in the second list. Note that the ticks are just there to show that one term has been combined with another term. From the second list we can see that the expression is now reduced to: Z = $\overline{A}\overline{B}$+ $\overline{B}$ $\overline{C}$+ $\overline{B}$C + A$\overline{B}$

From the second list note that the term having an index of 0 can be combined with the terms of index 1. Bear in mind that the dash indicates a missing variable and **must** line up in order to get a third list. The final simplified expression is: Z = $\overline{B}$

Bear in mind that any unticked terms in any list must be included in the final expression (none occured here except from the last list). Note that the only prime implicant here is Z = $\overline{B}$.

The tabular method reduces the function to a set of prime implicants.

Note that the above solution can be derived algebracially. Attempt this in your notes.

Example 2:

Consider the function f(A, B, C, D) = $\sum$(0,1,2,3,5,7,8,10,12,13,15), note that this is in decimal form.

$\sum$(0000,0001,0010,0011,0101,0111,1000,1010,1100,1101,1111) in binary form.

(0,1,1,2,2,3,1,2,2,3,4) in the index form.

First List

	A	B	C	D	
0	0	0	0	0	✓
1	0	0	0	1	✓
2	0	0	1	0	✓
8	1	0	0	0	✓
3	0	0	1	1	✓
5	0	1	0	1	✓
10	1	0	1	0	✓
12	1	1	0	0	✓
7	0	1	1	1	✓
13	1	1	1	1	✓
15	1	1	1	1	✓

Second List

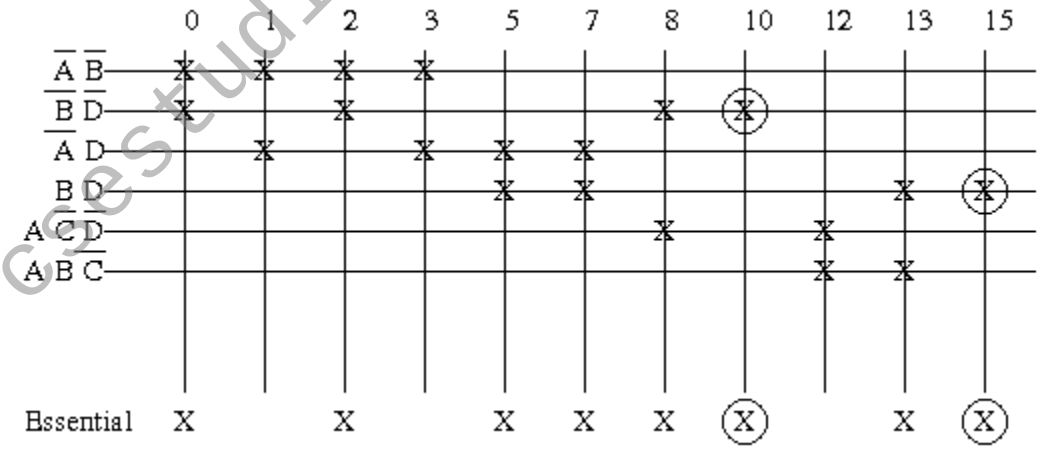
	A	B	C	D	
0,1	0	0	0	–	✓
0,2	0	0	–	0	✓
0,8	–	0	0	0	✓
1,3	0	0	–	1	✓
1,5	0	–	0	1	✓
2,3	0	0	1	–	✓
2,10	–	0	1	0	✓
8,10	1	0	–	0	✓
8,12	1	–	0	0	$A\overline{C}\overline{D}$
3,7	0	–	1	1	✓
5,7	0	1	–	1	✓
5,13	–	1	0	1	✓
12,13	1	1	0	–	$AB\overline{C}$
7,15	–	1	1	1	✓
13,15	1	1	–	1	✓

Third List

	A	B	C	D	
0, 1 2 3	0	0	–	–	$\overline{A}\,\overline{B}$
0, 2 1 3	0	0	–	–	
0, 2, 8, 10	–	0	–	0	$\overline{B}\,\overline{D}$
0, 8, 2, 10	–	0	–	0	
1, 3, 5, 7	0	–	–	1	$\overline{A}D$
1, 5, 3, 7	0	–	–	1	
5, 7, 13, 15	–	1	–	1	BD
5, 13, 7, 15	–	1	–	1	

The prime implicants are: $\overline{A}\overline{B}+\overline{B}\overline{D}+\overline{A}D+BD+A\overline{C}\overline{D}+AB\overline{C}$

The chart is used to remove redundant prime implicants. A grid is prepared having all the prime implicants listed at the left and all the minterms of the function along the top. Each minterm covered by a given prime implicant is marked in the appropriate position.



From the above chart, BD is an essential prime implicant. It is the only prime implicant that covers the minterm decimal 15 and it also includes 5, 7 and 13. $\overline{B}\overline{D}$ is also an essential prime implicant. It is the only prime implicant that covers the minterm denoted by decimal 10 and it also includes the terms 0, 2 and 8. The other minterms of the

function are 1, 3 and 12. Minterm 1 is present in $\overline{A}\overline{B}$and $\overline{A}$D. Similarly for minterm 3. We can therefore use either of these prime implicants for these minterms. Minterm 12 is present in A$\overline{C}\overline{D}$ and AB$\overline{C}$, so again either can be used.

Thus, one minimal solution is: Z = $\overline{B}\overline{D}$+ BD + $\overline{A}\overline{B}$+ A$\overline{C}\overline{D}$

Combinational Circuits

A combinational circuit neither contains a periodic clock signal nor has any provisions for storage. There are no feedbacks involved and the output at all time is dependent on the inputs provided. The name combinational is derived from the combinations of logic gates used for such circuits.

A sequential circuit involves feedback and has memory (so it is employed for designing RAM). It also has a periodic clock signal and hence the output is also a function of time in addition to being a function of inputs and previous outputs. The name sequential is derived as the output is produced in sequences as the clock circuit enables and disables the functioning. (A latch is also a sequential circuit but has no clock signal and hence is a special case. It is also the basic building block of any sequential circuit.)

Designing Combinational Circuits

In general we have to do following steps:

1. **Problem description**
2. **Input/output of the circuit**
3. **Define truth table**
4. **Simplification for each output**
5. **Draw the circuit**

ADDERS

Half adder

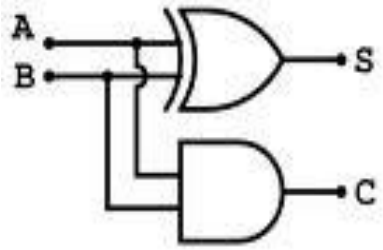
A **half adder** is a logical circuit that performs an addition operation on two one-bit binary numbers often written as A and B. The half adder output is a sum of the two inputs usually represented with the signals C_{out} and S where . Following is the logic table for a half adder:

Inputs Outputs

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

S=A (XOR) B

C=AB

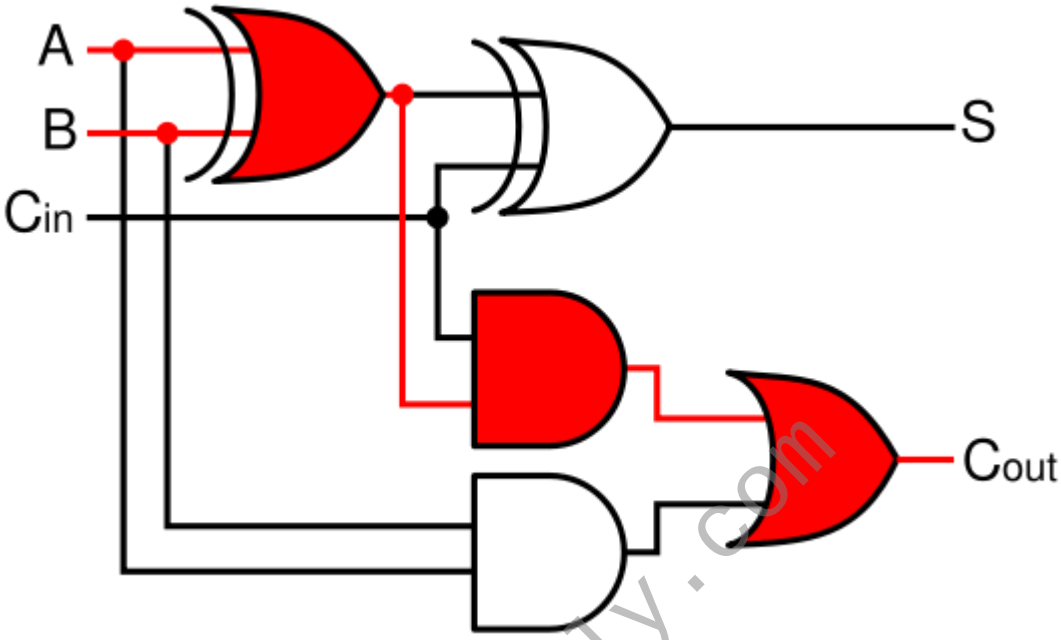


Full Adder

A **full adder** is a logical circuit that performs an addition operation on three one-bit binary numbers often written as A, B, and C_{in}. The full adder produces a two-bit output sum typically represented with the signals C_{out} and S.

Inputs Outputs

A	*B*	C_i	C_o	*S*
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

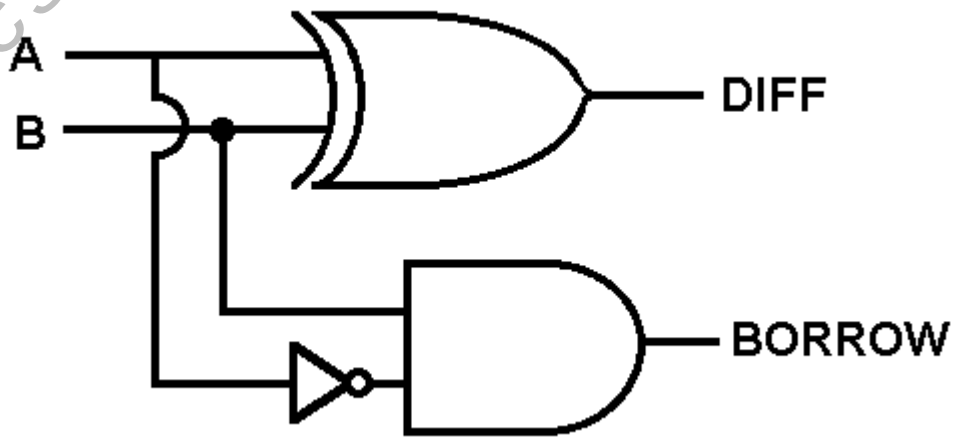


SUBTRACTOR

Half subtractor

The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow).

X Y D B
0 0 0 0
0 1 1 1
1 0 1 0
1 1 0 0

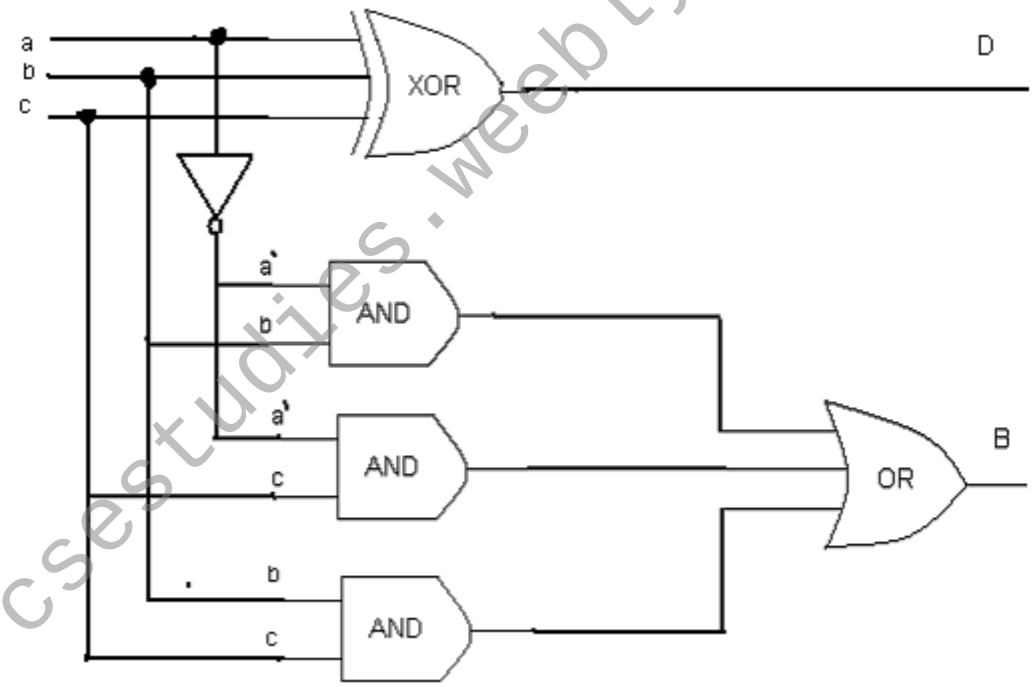


Full subtractor

The Full_subtractor is a combinational circuit which is used to perform subtraction of three bits. It has three inputs, X (minuend) and Y (subtrahend) and Z (subtrahend) and two outputs D (difference) and B (borrow).

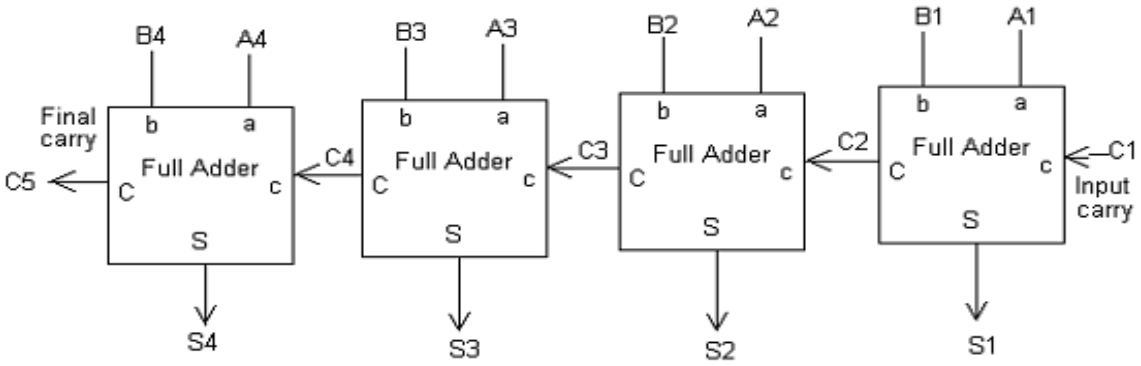
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

B=A'B + A'C + BC
C=A xor B xor C



Parallel adder: Parallel adder is the one where we input the all the bits of two given numbers and we don't need any memory element.

Binary Adder: In this adder we need n full adders for n bit adder. In this adder we use the n full adders in cascaded from to implement the ripple carry adder. This type of adder is also called carry propagation adder. The circuit for 4-bit parallel adder is as follow:



- For example to add A= 1011 and B= 0011

subscript i:	3	2	1	0	
Input carry:	0	1	1	0	Ci
Augend:	1	0	1	1	Ai
Addend:	0	0	1	1	Bi
	---	---	---	---	
Sum:	1	1	1	0	Si
Output carry:	0	0	1	1	Ci+1

Binary Subtractor

- The subtrcation A – B can be done by taking the 2's complement of B and adding it to A because A- B = A + (-B)
- It means if we use the inveters to make 1's complement of B (connecting each Bi to an inverter) and then add 1 to the least significant bit (by setting carry C0 to 1) of binary adder, then we can make a binary subtractor.

Adder Subtractor

- The addition and subtraction can be combined into one circuit with one common binary adder (see next slide).
- The mode M controls the operation. When M=0 the circuit is an adder when M=1 the circuit is subtractor. It can be don by using exclusive-OR for each Bi and M. Note that 1 $\oplus$x = x' and 0 $\oplus$x = x

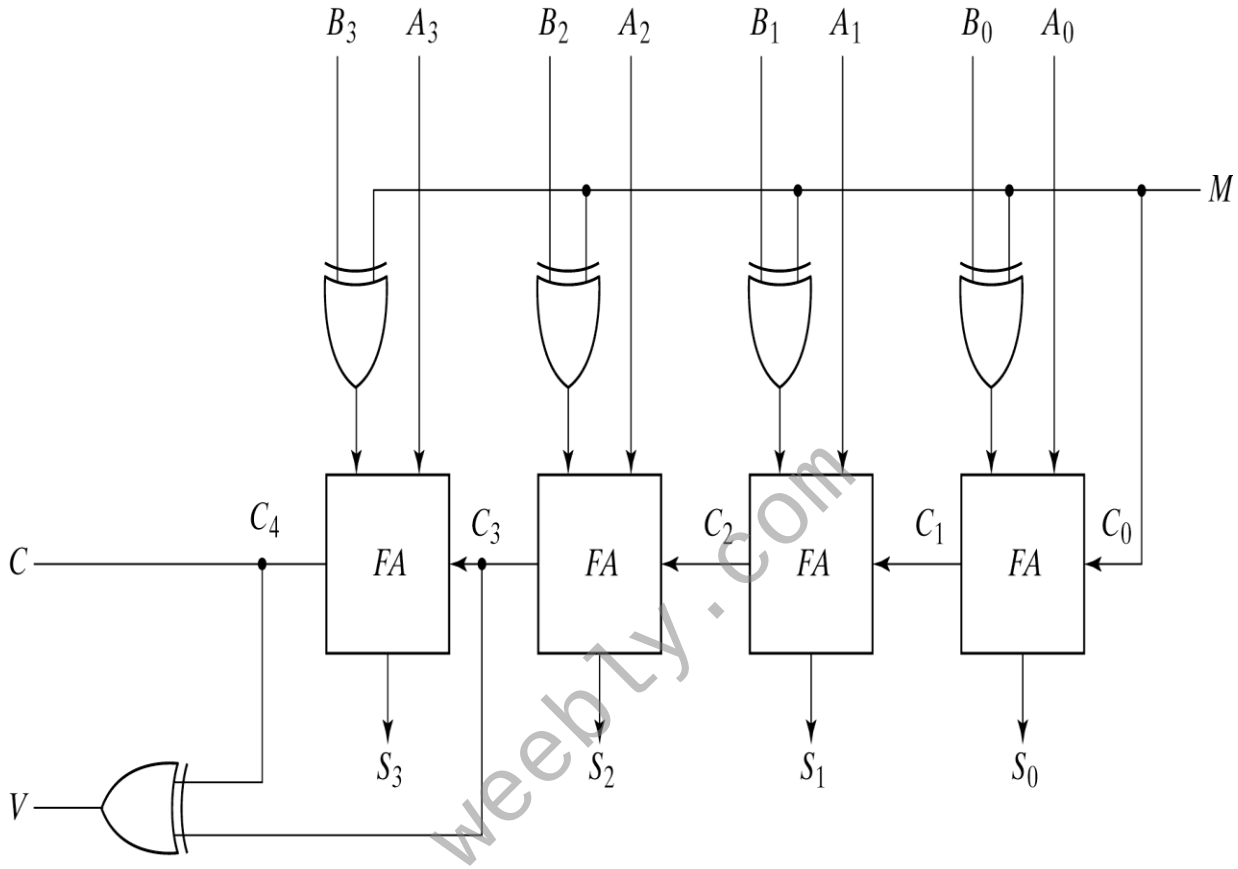


Fig. 4-13 4-Bit Adder Subtractor

Binary to Gray converter

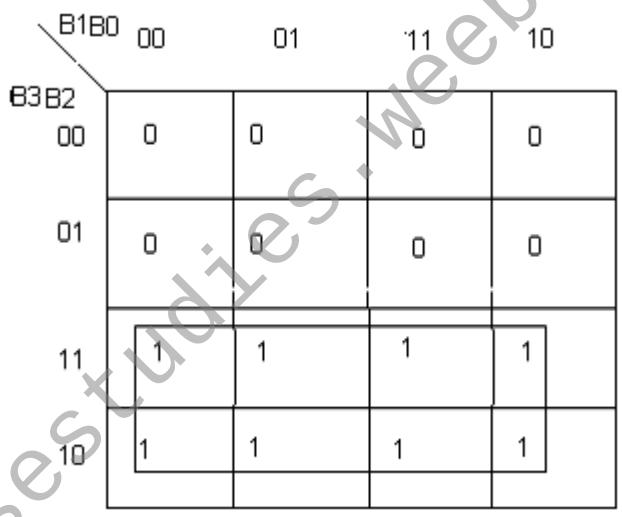
In this circuit we'll convert BINARY numbers to GRAY numbers. Following is the truth table for it:

	B3	B2	B1	B0	G3	G2	G1	G0
0.	0	0	0	0	0	0	0	0
1.	0	0	0	1	0	0	0	1
2.	0	0	1	0	0	0	1	1
3.	0	0	1	1	0	0	1	0
4.	0	1	0	0	0	1	1	0
5.	0	1	0	1	0	1	1	1
6.	0	1	1	1	0	1	0	0

8.	1	0	0	0	1	1	0	0
9.	1	0	0	1	1	1	0	1
10.	1	0	1	0	1	1	1	1
11.	1	0	1	1	1	1	1	0
12.	1	1	0	0	1	0	1	0
13.	1	1	0	1	1	0	1	1
14.	1	1	1	0	1	0	0	1
15.	1	1	1	1	1	0	0	0

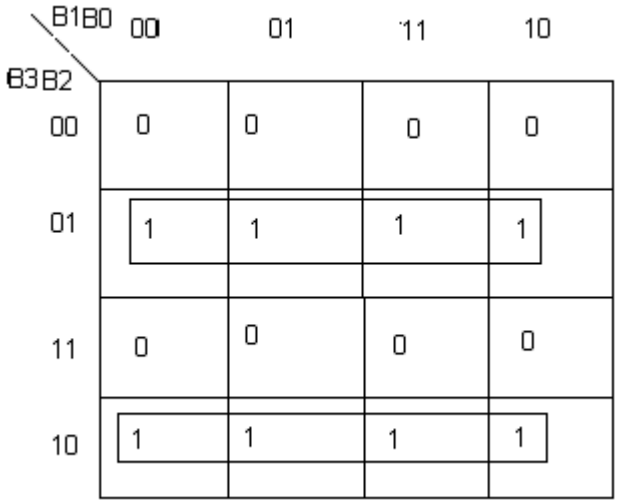
K-MAPS:

K-MAP FOR G3:



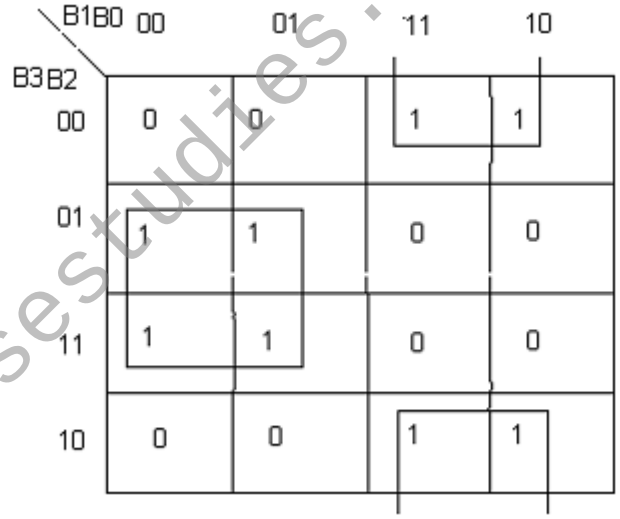
Equation for G3= B3

K-MAP FOR G2:



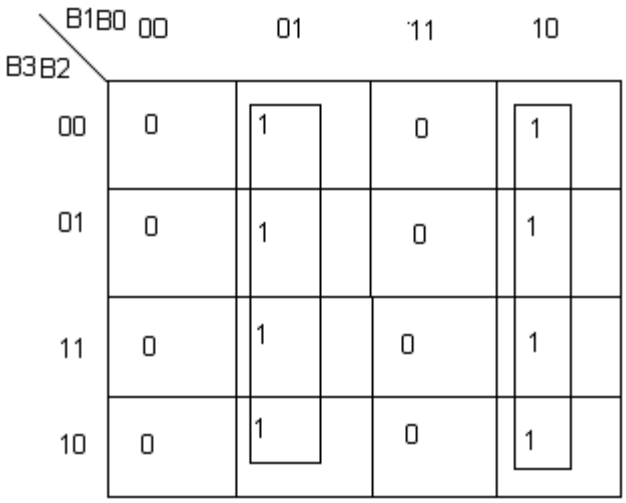
Equation for G2= B3' B2 + B3 B2'= B3 XOR B2

K-MAP FOR G1:

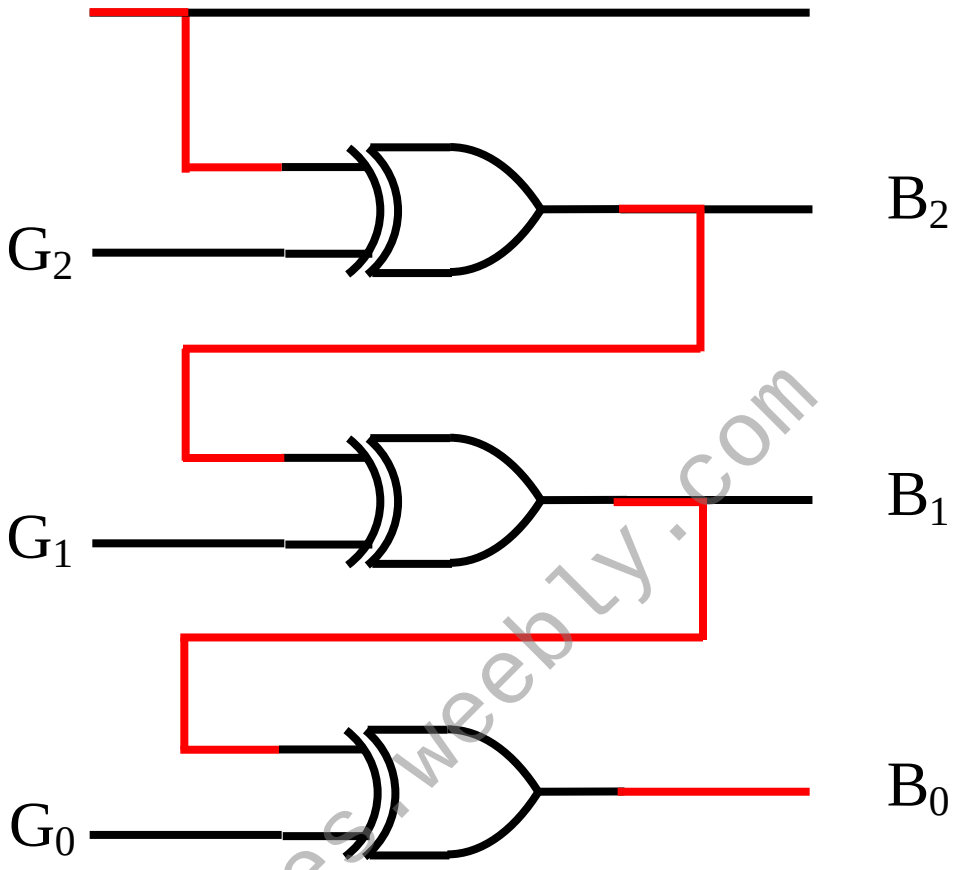


Equation for G1= B1' B2 + B1 B2'= B1 XOR B2

K-MAP FOR G0:



Equation for G0= B1' B0 + B1 B0'= B1 XOR B0



COMPARATORS

- It is a combinational circuit that compares to numbers and determines their relative magnitude
- The output of comparator is usually 3 binary variables indicating:

A>B

A=B

A<B

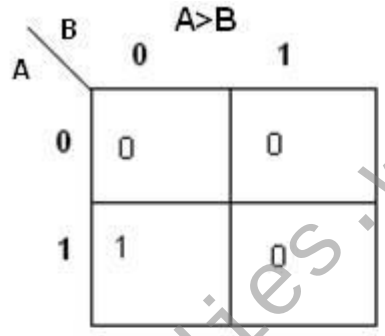
- For example to design a comparator for 2 bit binary numbers A (A1A0) and B (B1B0) we do the following steps:

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.

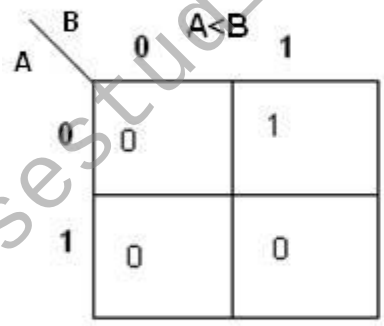
If we list all the input combinations at the input then we get the following table describing the corresponding outputs.

A	B	f (A>B)	f (A=B)	f (A<B)
0	**0**	**0**	**1**	**0**
1	**0**	**1**	**0**	**0**
0	**1**	**0**	**0**	**1**
1	1	0	1	0

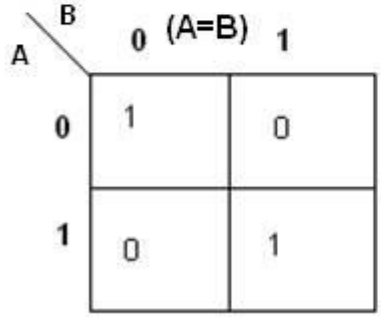
And now we find the equations using K-maps each for f (A>B), f (A=B) and f (A<B) as follow:



Equation is $A>B = A.\overline{B}$



Equation is $A<B = \overline{A}.B$



The equation is $f(A=B) = \overline{A}.\overline{B} + A.B$
$= A \text{ XNOR } B$

or we can write the equation for $f(A=B)$ as $\overline{A.\overline{B} + \overline{A}.B} = \overline{f(A>B)+f(A<B)}$

- For a 2-bit comparator we have four inputs A1A0 and B1B0 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 KMAP the result is
- E= A'1A'0B'1B'0 + A'1A0B'1B0 + A1A0B1B0 + A1A'0B1B'0

or E=((A0 ⊕B0) + (A1 ⊕B1))'

G = A1B'1 + A0B'1B'0 + A1A0B'0

L= A'1B1 + A'1A'0B0 + A'0B1B0

4-Bit magnitude comparator

It means X0 = A0B0 + A'0B'0 and

X1= A1B1 + A'1B'1

(A= B) = x3x2x1x0

(A> B) = A3B'3 + x3A2B'2 + x3x2A1B'1+ x3x2x1A0B'0

(A< B) = A'3B3 + x3A'2B2 + x3x2A'1B1+ x3x2x1A'0B0

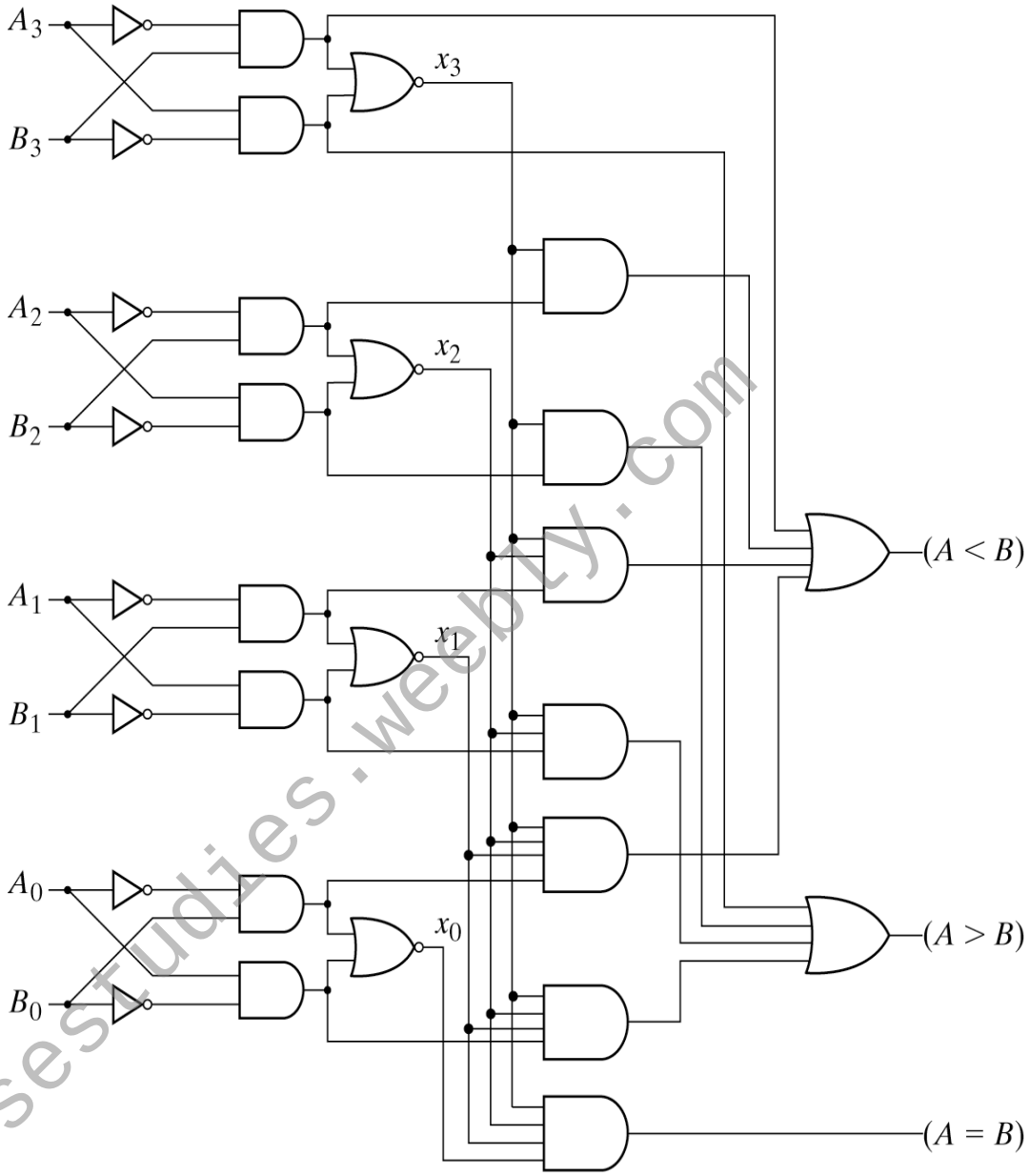


Fig. 4-17 4-Bit Magnitude Comparator

BCD Adder

Number	C	S8	S4	S2	S1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0

3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
Number	C	S8	S4	S2	S1
10	1	0	0	0	0
11	1	0	0	0	1
12	1	0	0	1	0
13	1	0	0	1	1
14	1	0	1	0	0
15	1	0	1	0	1
16	1	0	1	1	0
17	1	0	1	1	1
18	1	1	0	0	0
19	1	1	0	0	1

Binary Sum

Number	K	Z8	Z4	Z2	Z1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

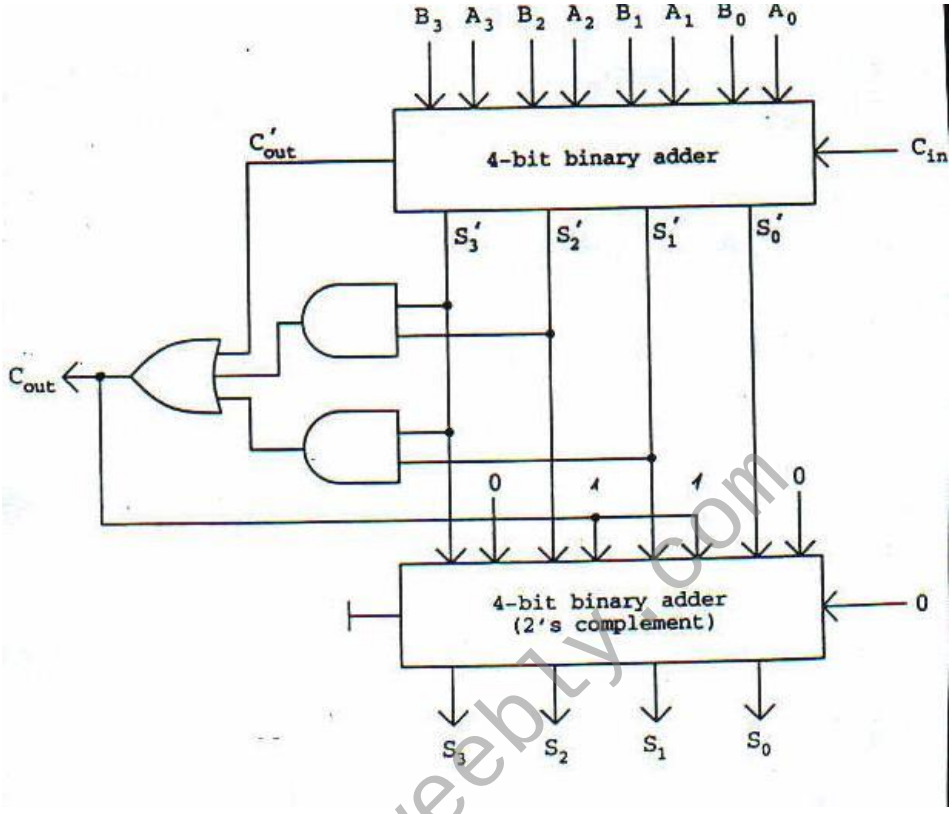
Number	K	Z8	Z4	Z2	Z1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

Number	K	Z8	Z4	Z2	Z1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

C= K +

C= K + Z8*Z4+

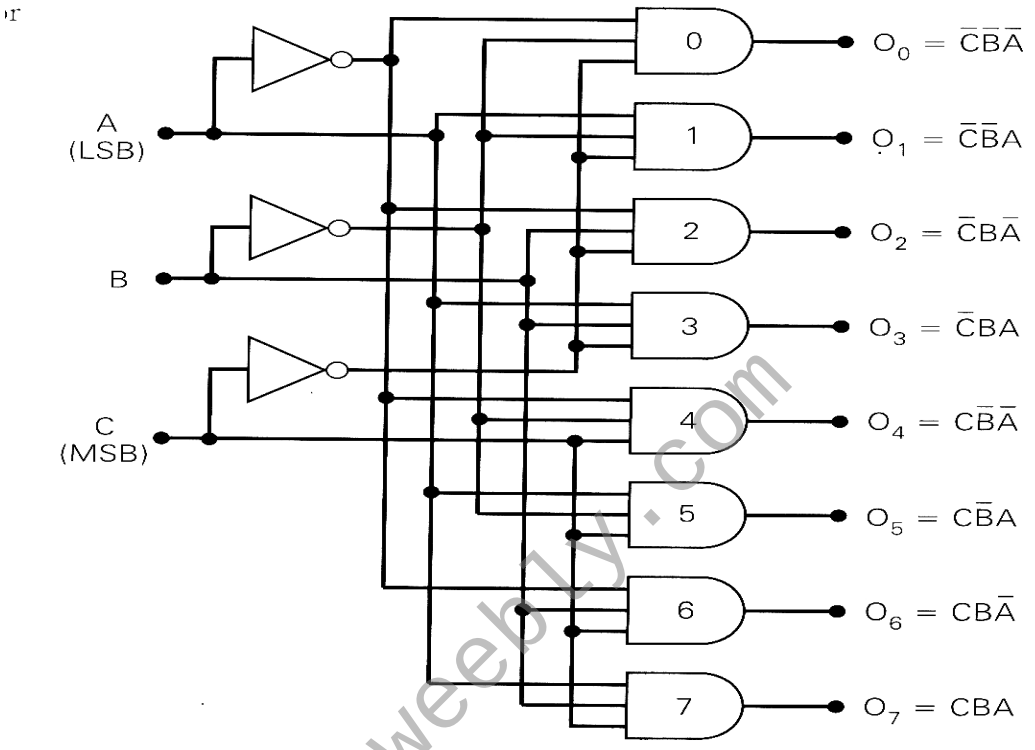
C = K + Z8*Z4+Z8*Z2



Decoders

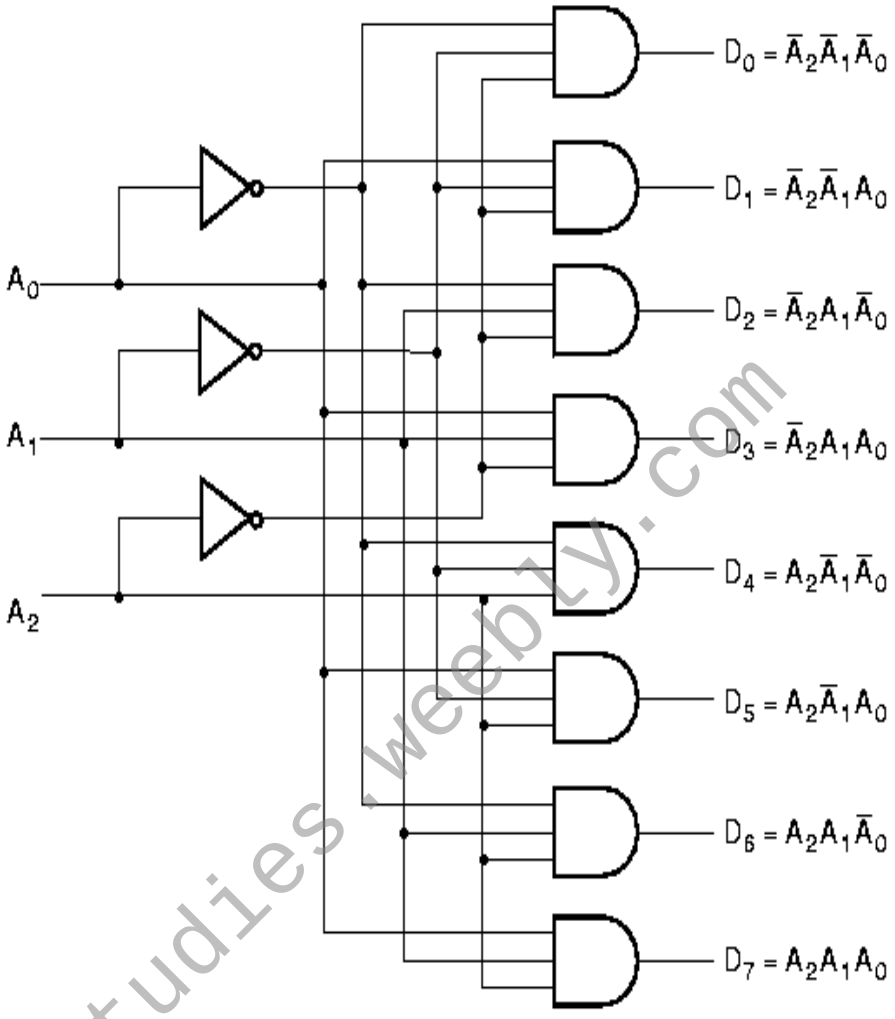
- **Accepts a value and decodes it**
  - **Output corresponds to value of *n* inputs**

- **Consists of:**
  - Inputs (n)
  - Outputs (2n , numbered from 0 → 2n - 1)
  - Selectors / Enable (active high or active low)



C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Octal Numbar	D0	D1	D2	D3	D4	D5	D6	D7	a	b	c
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1



Encoder

- Perform the inverse operation of a decoder
- 2n (or less) input lines and n output lines

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

- Can be implemented with 3 OR gates

A0 = D1 + D3 + D5 + D7;

A1 = D2 + D3 + D6 + D7;

A2 = D4 + D5 + D6 + D7;

If more than 2 inputs are active we need to use priority encoder (priority for inputs)

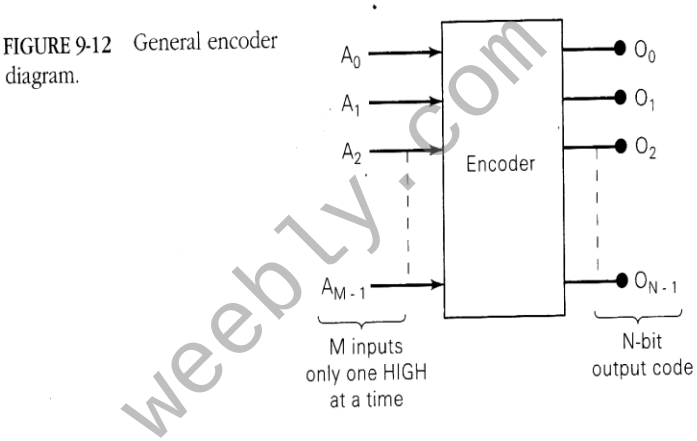
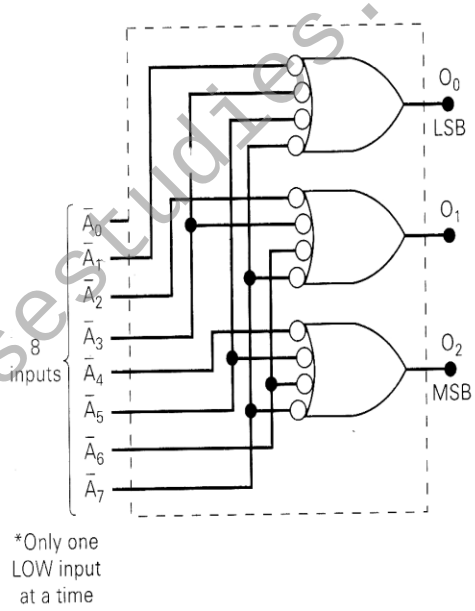


FIGURE 9-12 General encoder diagram.

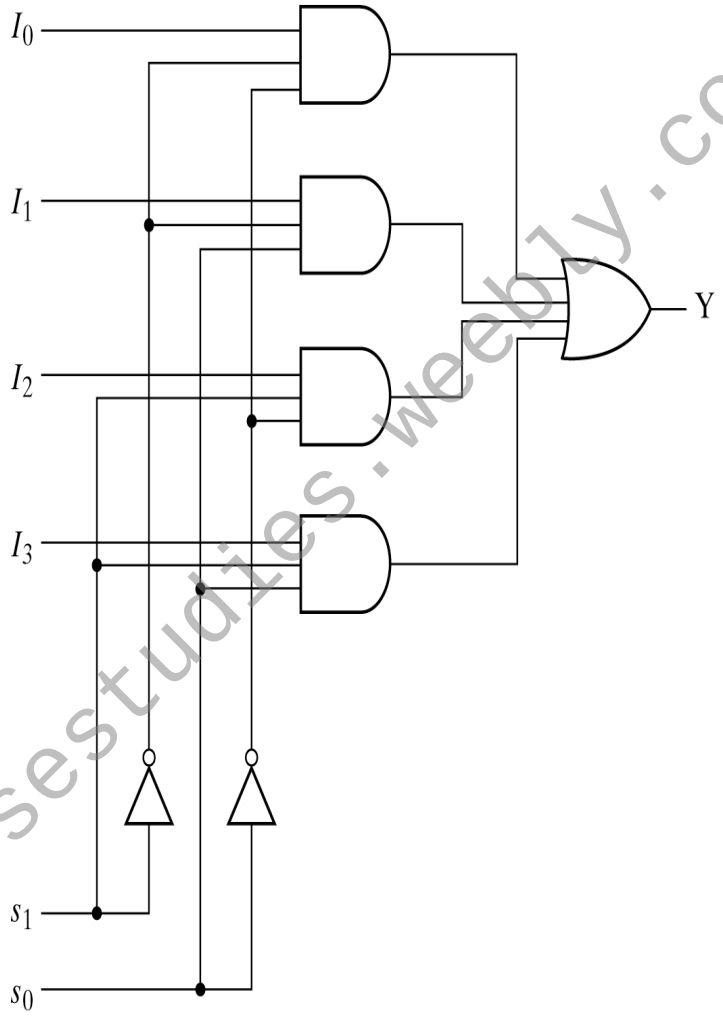


Inputs								Outputs		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

not corrected.

Multiplexer

- It is a combinational circuit that selects binary information from one of the input lines and directs it to a single output line
- Usually there are 2n input lines and n selection lines whose bit combinations determine which input line is selected
- For example for 2-to-1 multiplexer if selection S is zero then I0 has the path to output and if S is one I1 has the path to output



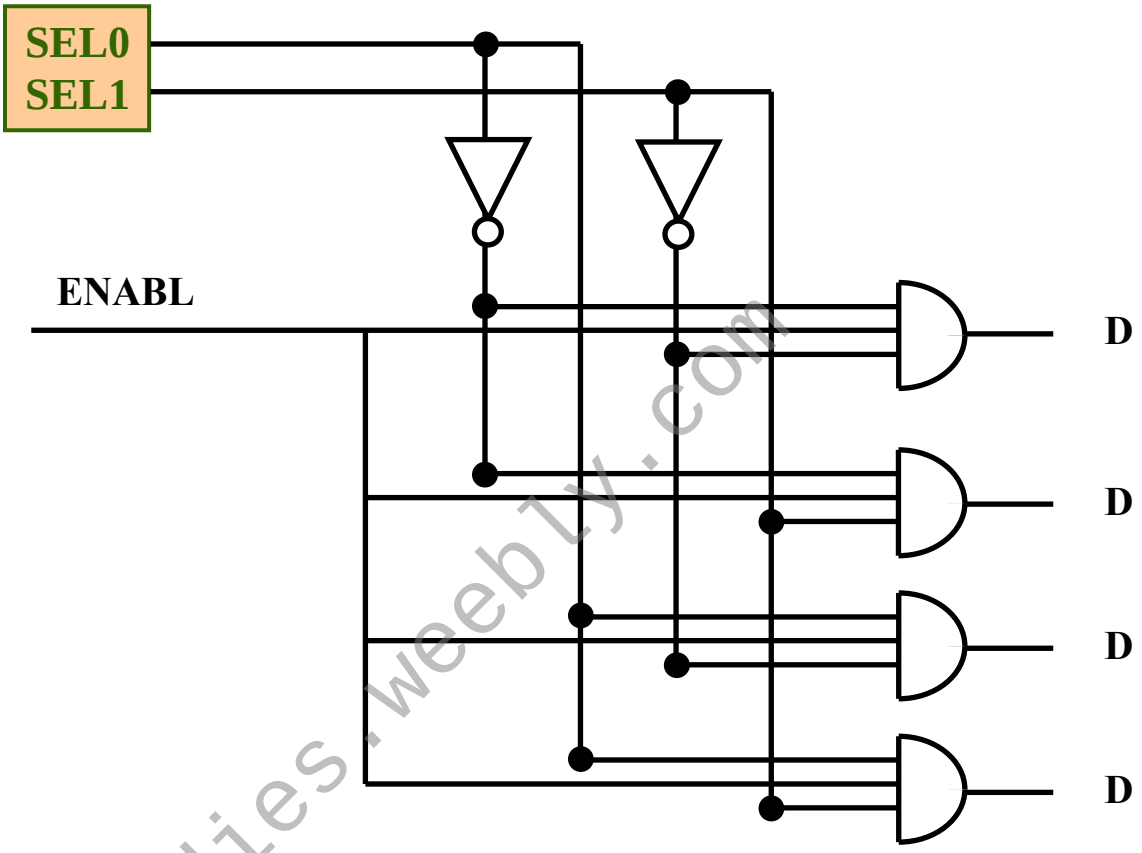
(a) Logic diagram

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

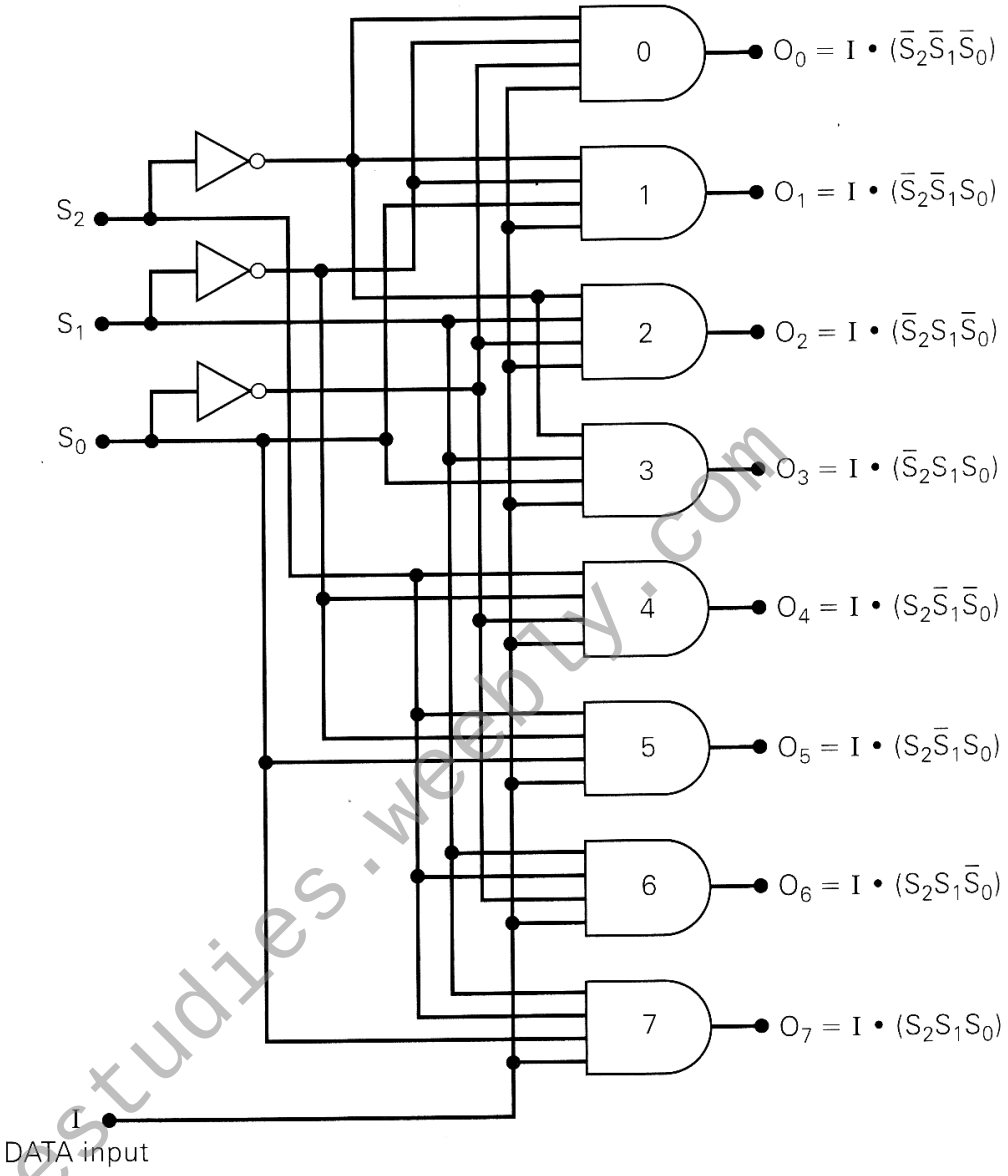
Fig. 4-25 4-to-1-Line Multiplexer

Demultiplexer



ENBL	SEL1	SEL2	D00	D01	D10	D11
0	X	X	0	0	0	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	0	1

1 to 8 line Demultiplexer



SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Note: I is the data input

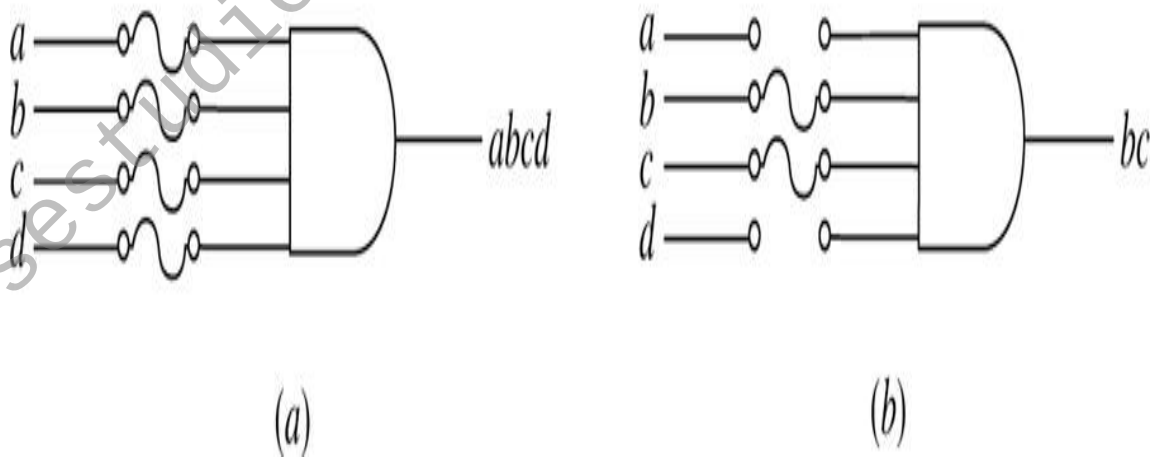
Programmable Logic Devices (PLD's)

- ***Programmable devices have their functionality programmed before they are first used.***
- ***Range in complexity from 100's to 10,000's of logic gates.***

- ***Types of Programmable Logic Devices***
- PLDs (Programmable Logic Devices)
  - ROM (Read-Only Memory)
  - PLA (Programmable Logic Array)
  - PAL (Programmable Array Logic)

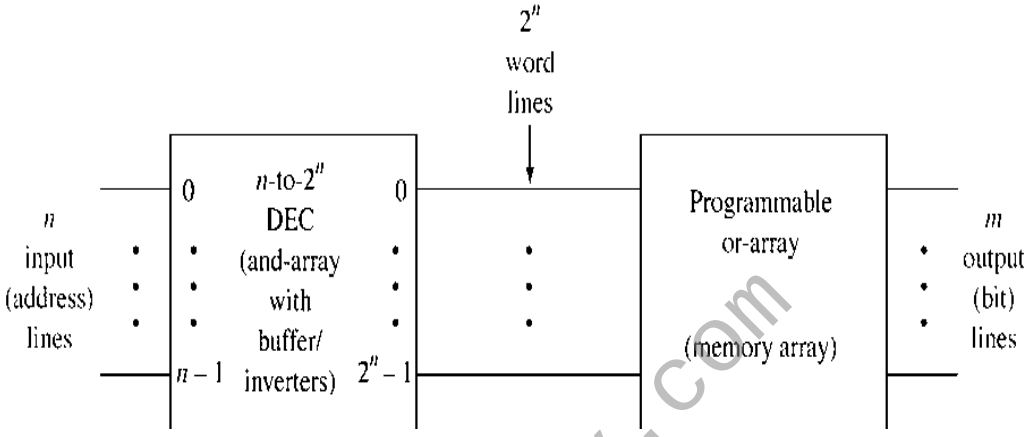
Device	AND-array	OR-array
PROM	Fixed	Programmable
PLA	Programmable	Programmable
PAL	Programmable	Fixed
GAL	Programmable	Fixed

Programming by blowing fuses.

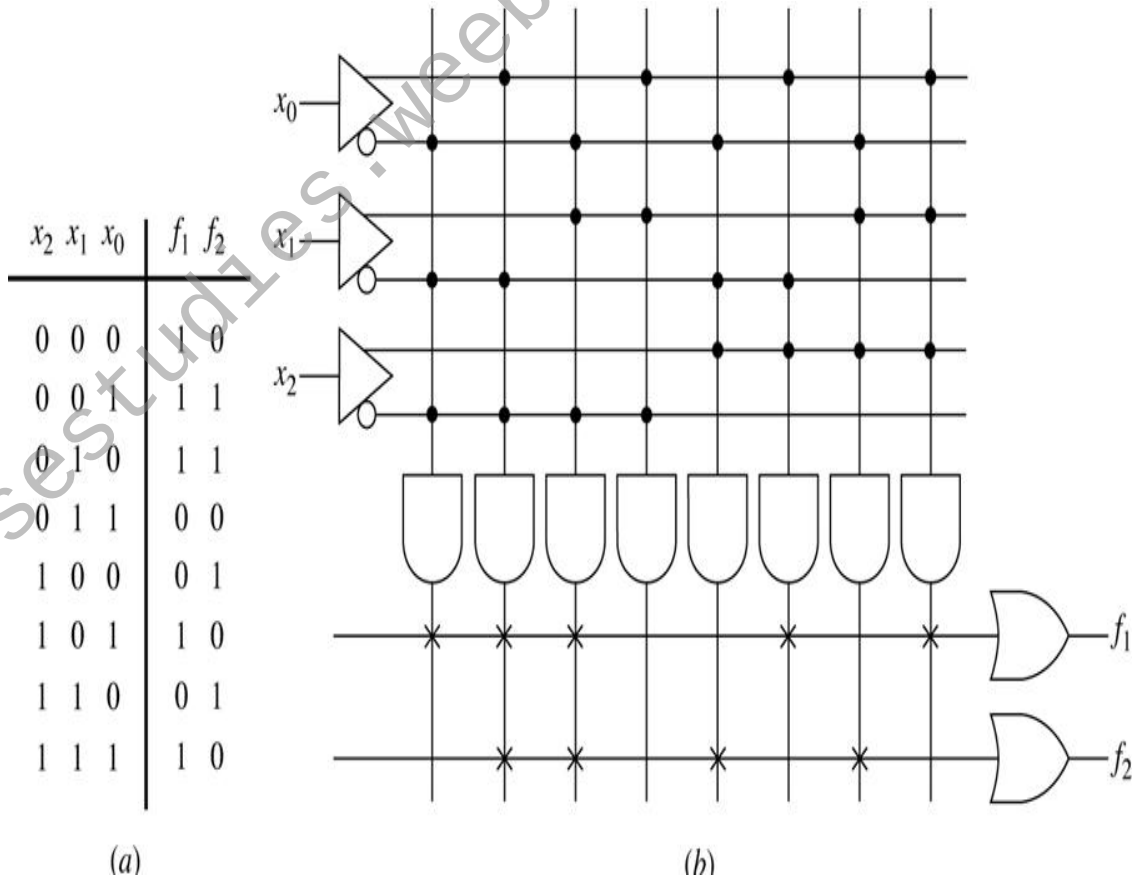


(*a*) Before programming. **(*b*) After Programming.**

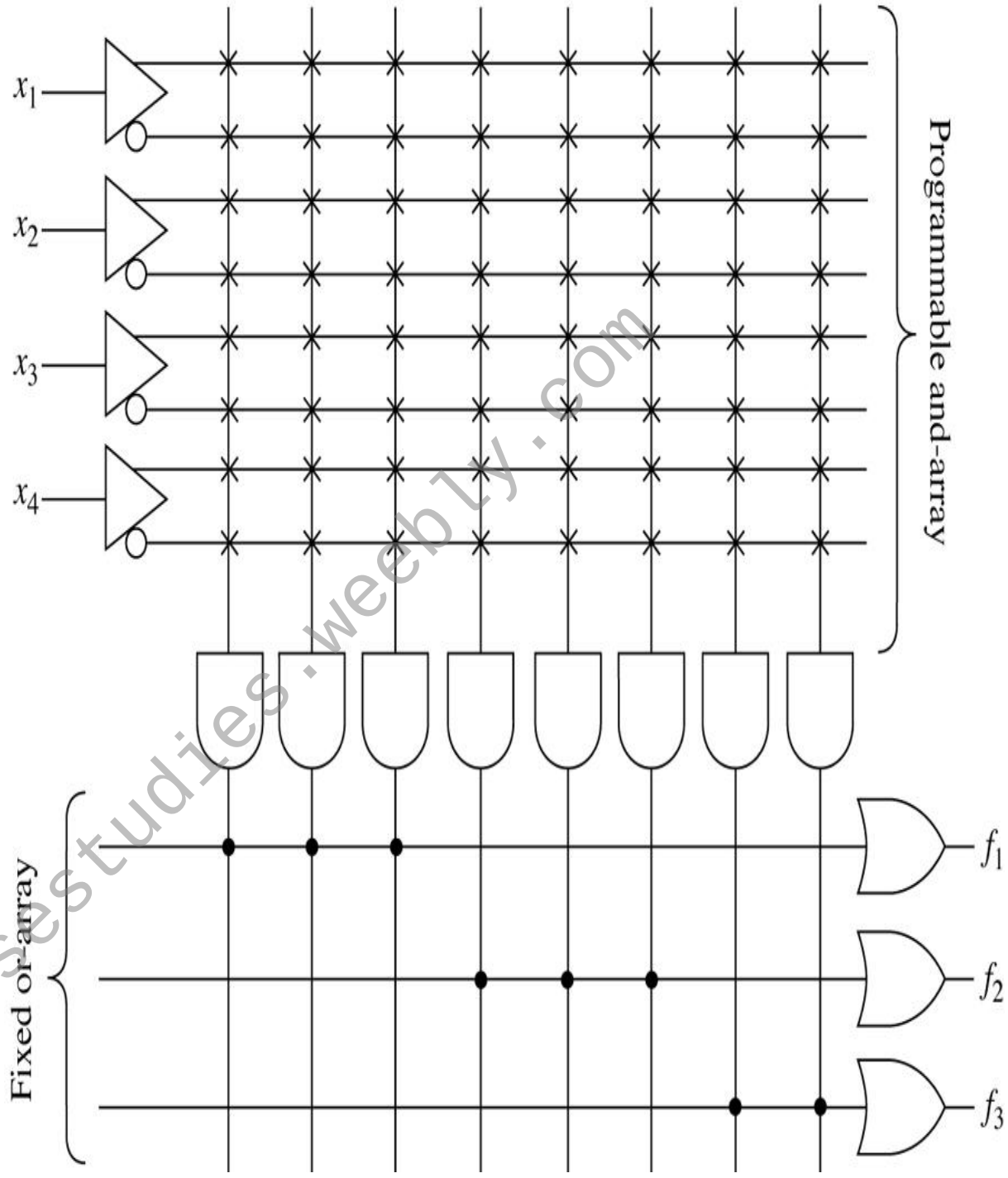
PROM Notation



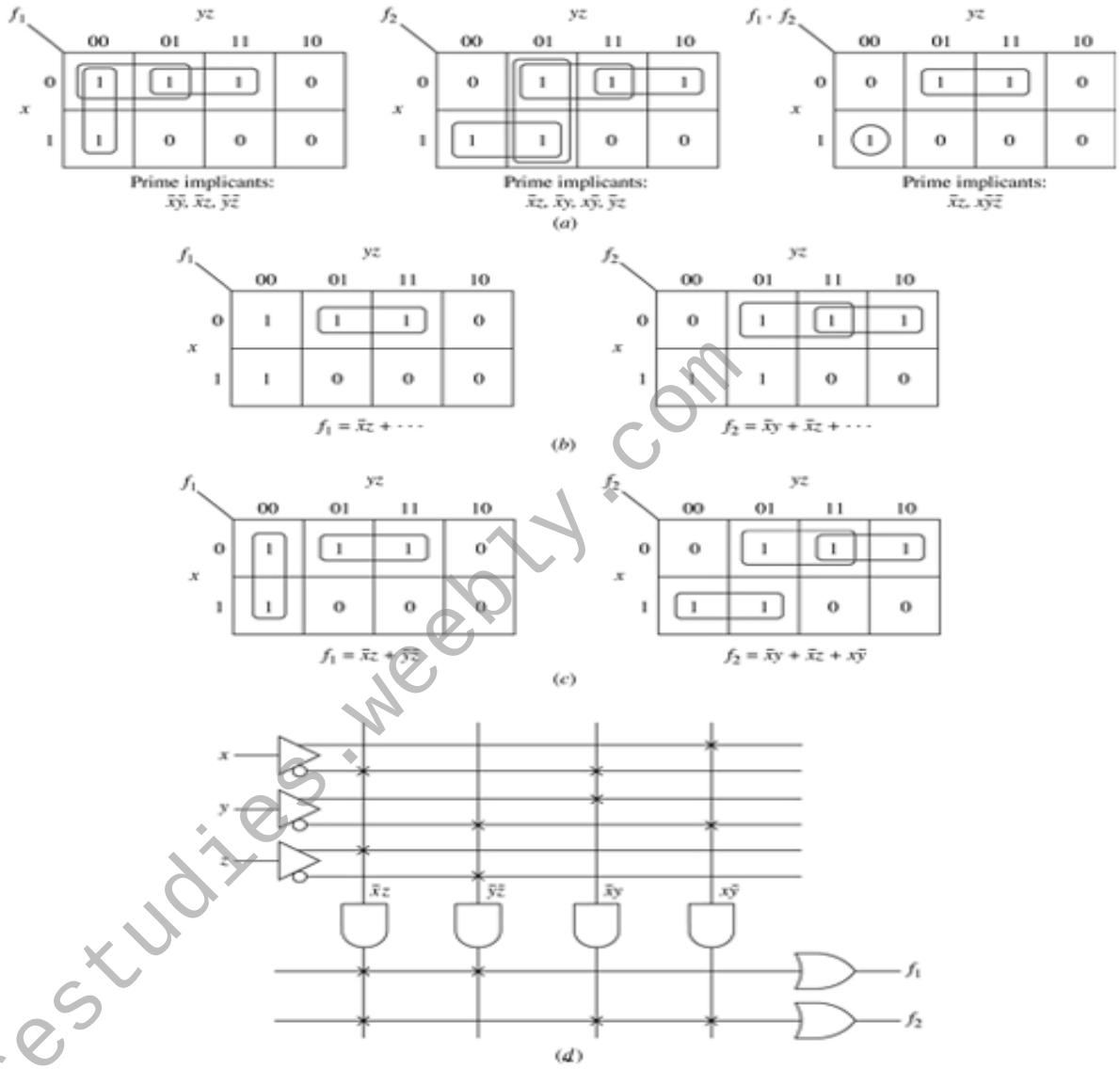
Using a PROM for logic design



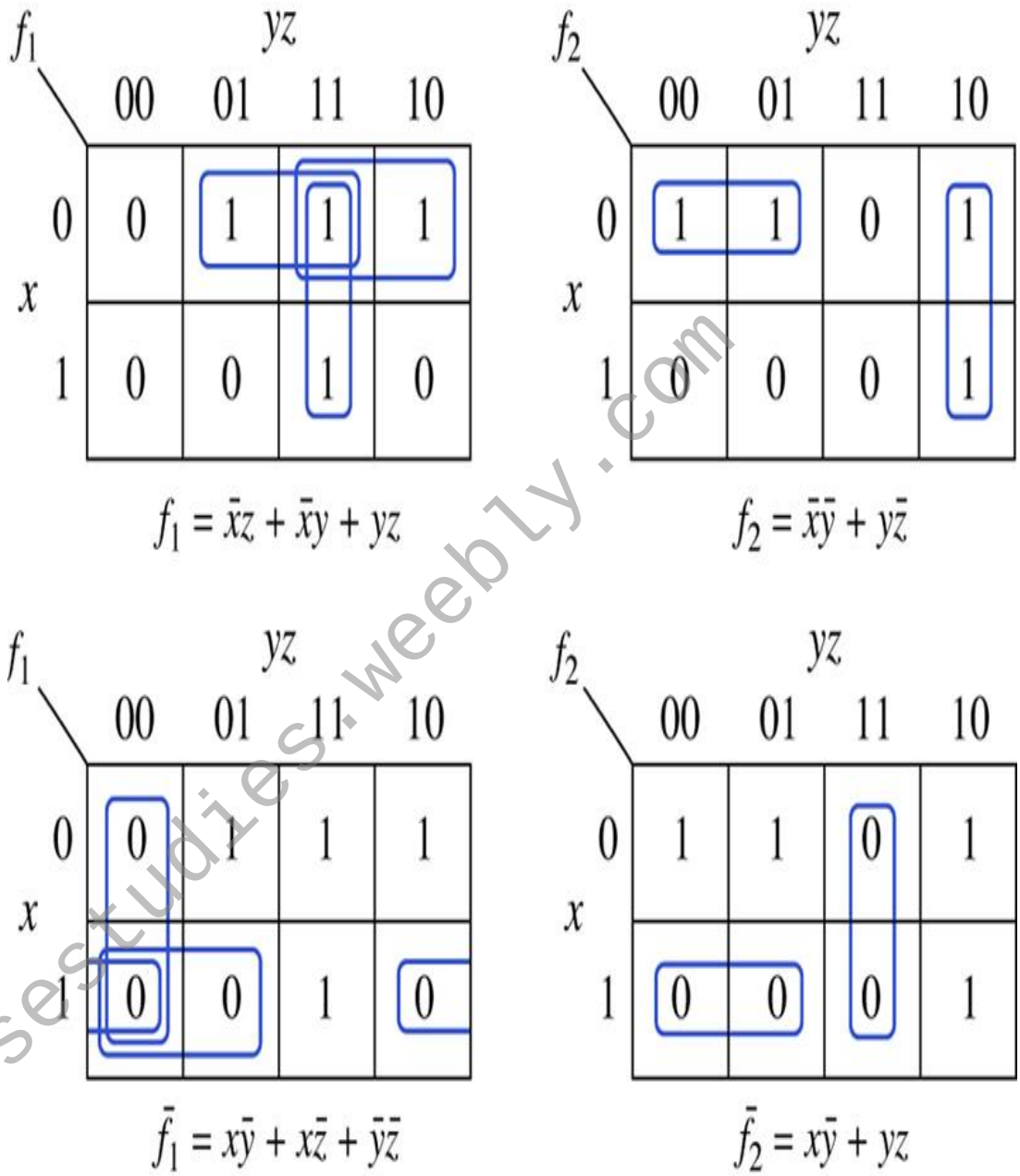
A simple four-input, three-output PAL device.



Example of combinational logic design using a PLA.

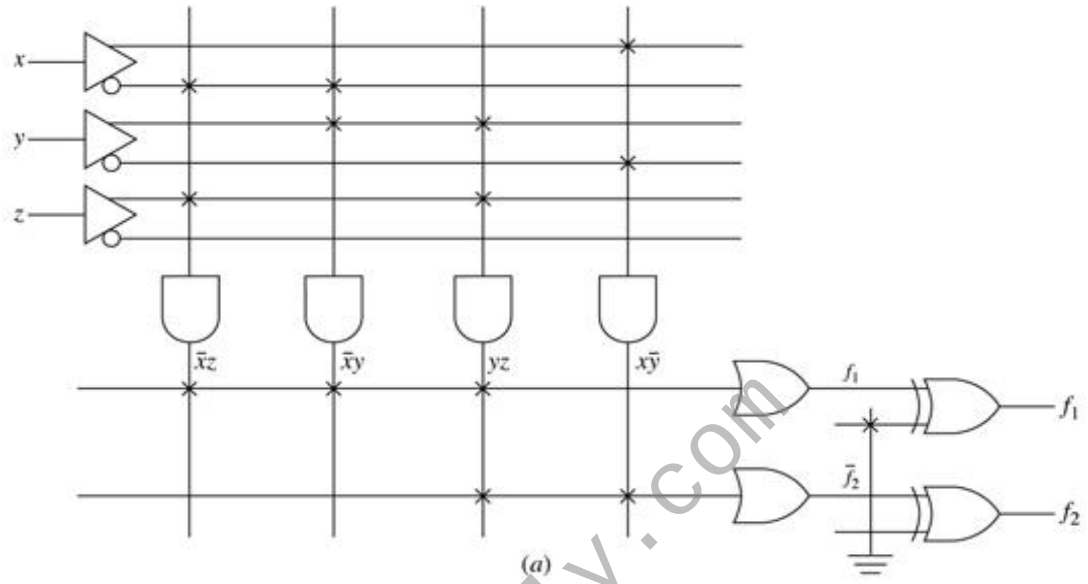


Karnaugh maps for the functions *f*1(*x*,*y*,*z*) = Σ*m*(1,2,3,7) and *f*2(*x*,*y*,*z*) = Σ*m*(0,1,2,6)

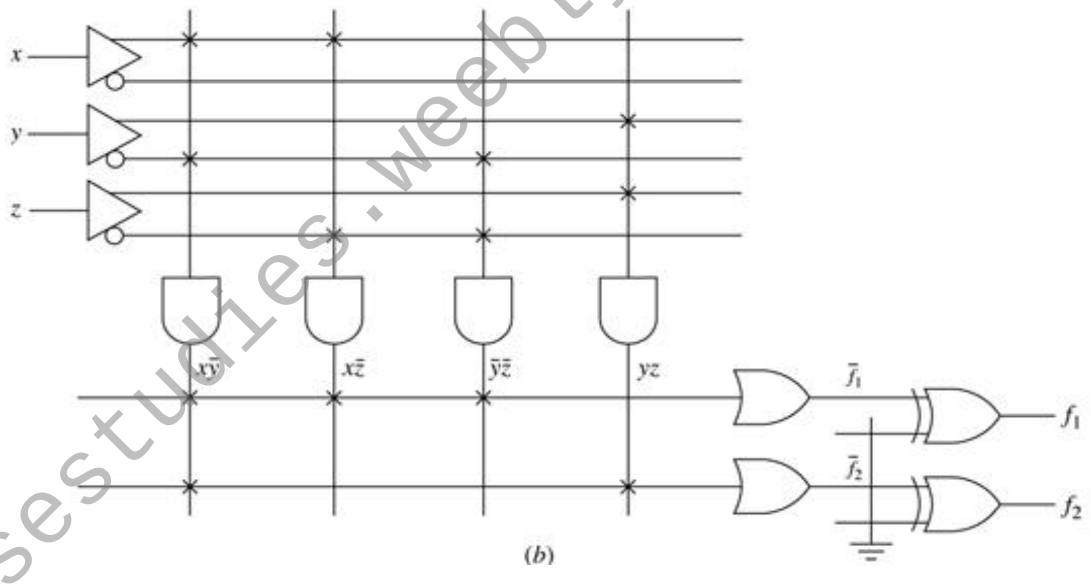


Two realizations of *f*1(*x*,*y*,*z*) = Σ*m*(1,2,3,7) and *f*2(*x*,*y*,*z*) = Σ*m*(0,1,2,6).

(*a*) Realization based on *f*1 and 2 (*b*) Realization based on 1 and 2

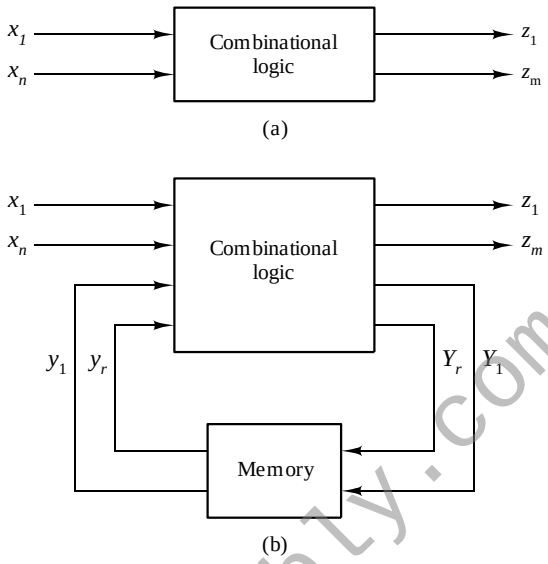


(a)



(b)

Sequential Circuits

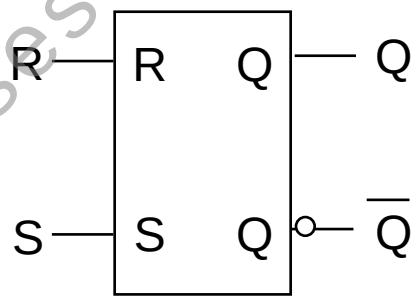


Flip Flop

A circuit that changes from 1 to 0 or from 0 to 1 when current is applied. It is one bit storage location.

SR Flip-Flop

Graphical Symbol



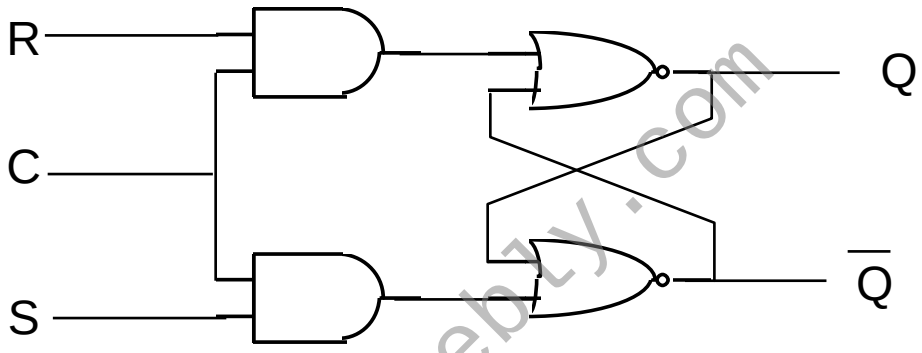
Characteristic table

C	D	Q
0	x	No change
1	0	0 (reset)
1	1	1 (set)

Characteristic Equation

$Q(t+1)=S+R^{1}Q$

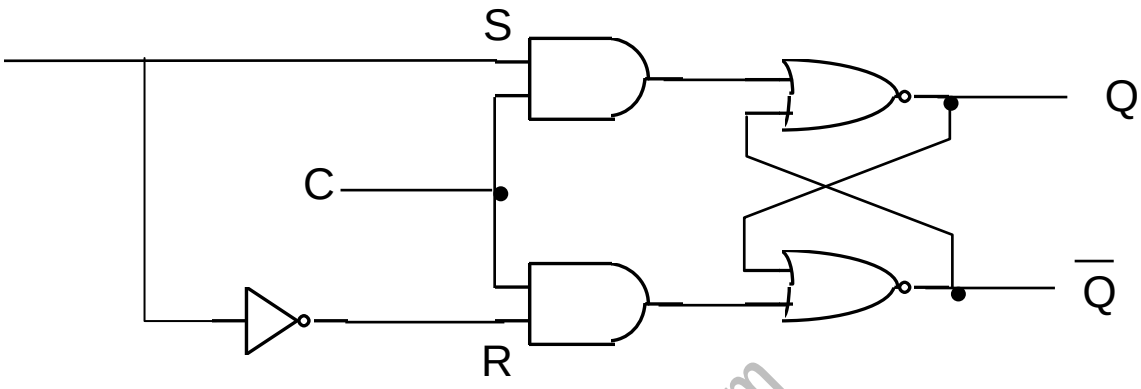
Logic diagram



EXCITATION TABLE

Q	$Q_{(next)}$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

D Flip-Flop



D

Characteristic table

D	Q(t+1)	Operation
0	0	Reset
1	1	Set

Characteristic Equation

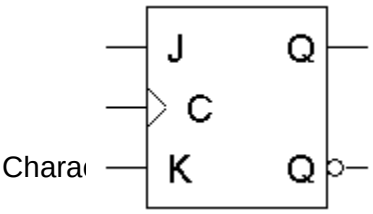
Q(t+1) = D

EXCITATION TABLE

Q	$Q_{(next)}$	D
0	0	0
0	1	1
1	0	0
1	1	1

JK Flip-Flop

Graphical Symbol



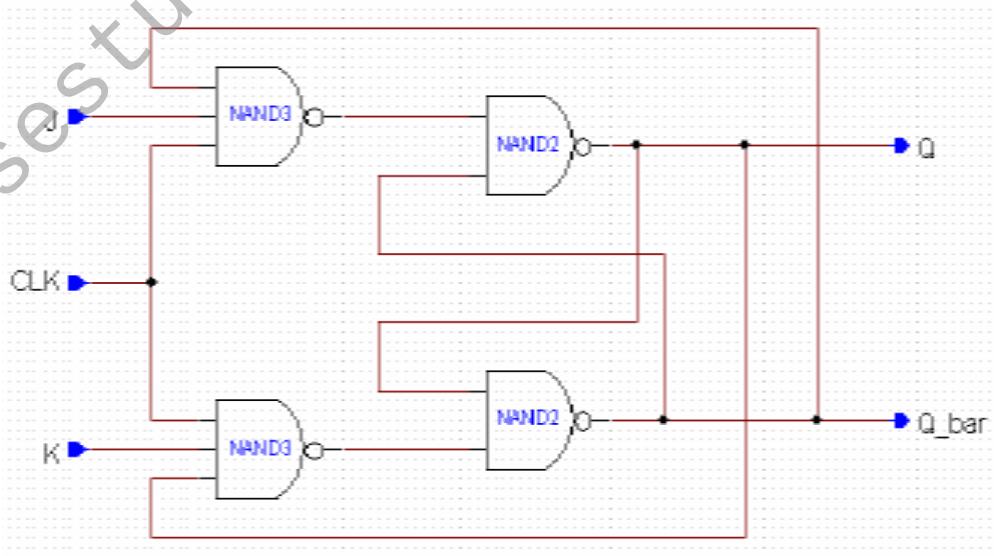
Chara

J	K	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

Characteristic Equation

Q(t+1) = K'Q(t) + JQ'(t)

Logic Diagram

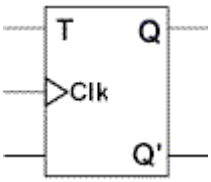


EXCITATION TABLE

Q	$Q_{(next)}$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

T Flip-Flop

Graphical Symbol



Characteristic Table

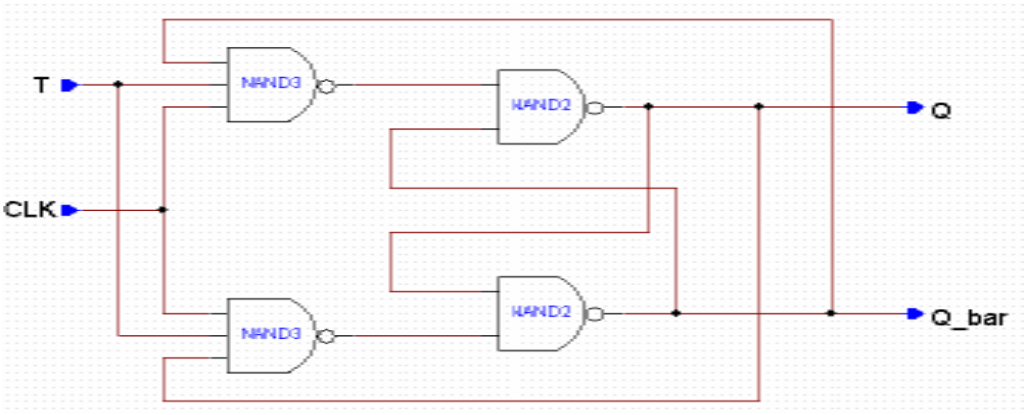
T	Q(t+1)	Operation
0	Q(t)	No change
1	Q'(t)	Complement

Characteristic Equation

Q(t+1) = T'Q(t) + TQ'(t)

= T ⊕ Q(t)

Logic Diagram

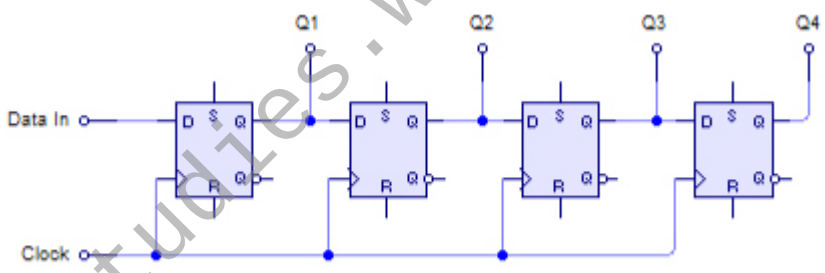


Q	$Q_{(next)}$	T
0	0	0
0	1	1
1	0	1
1	1	0

EXCITATION TABLE

Shift Register

A **shift register** is a cascade of Flip flops, sharing the same clock, which has the output of any one but the last flip-flop connected to the "data" input of the next one in the chain, resulting in a circuit that shifts by one position the one-dimensional "bit array" stored in it, *shifting in* the data present at its input and *shifting out* the last bit in the array, when enabled to do so by a transition of the clock input. More generally, a **shift register** may be multidimensional, such that its "data in" input and stage outputs are themselves bit arrays: this is implemented simply by running several shift registers of the same bit-length in parallel.



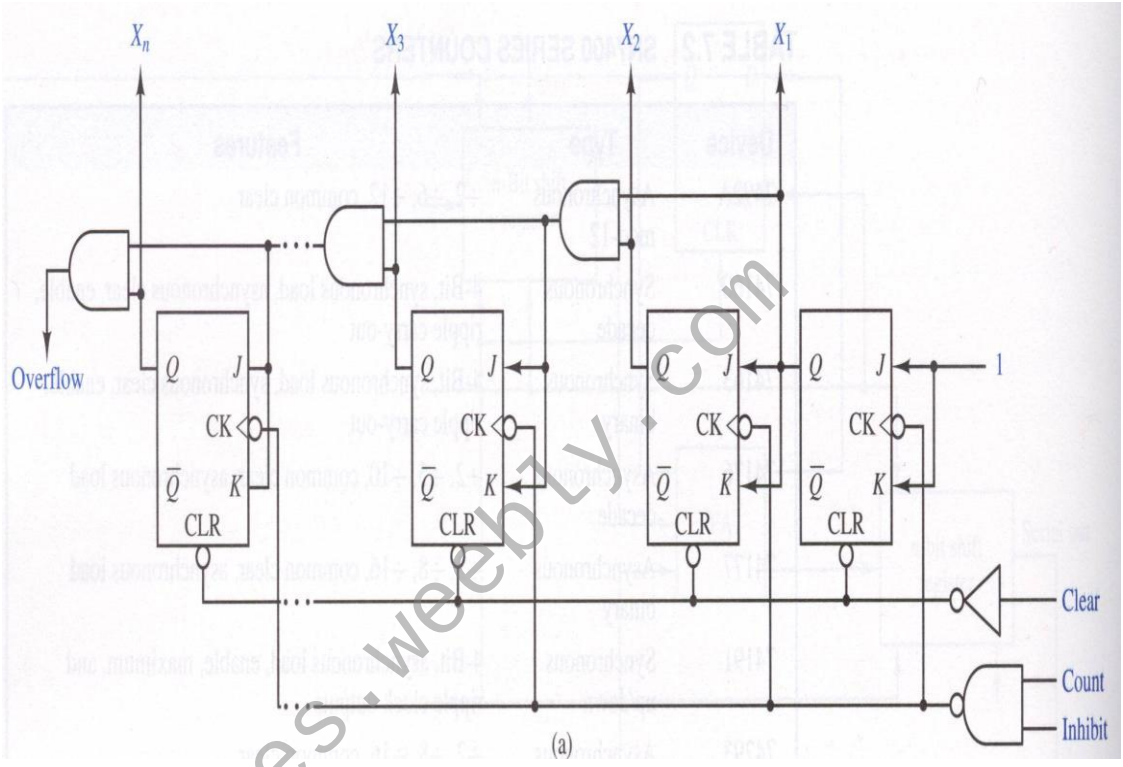
Counters

In digital logic and computing, a **counter** is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.

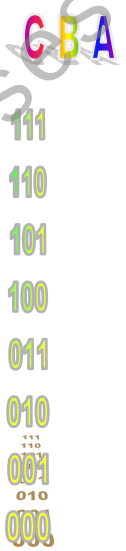
- Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops
- Synchronous counter – all state bits change under control of a single clock
- Decade counter – counts through ten states per stage
- Up–down counter – counts both up and down, under command of a control input
- Ring counter – formed by a shift register with feedback connection in a ring

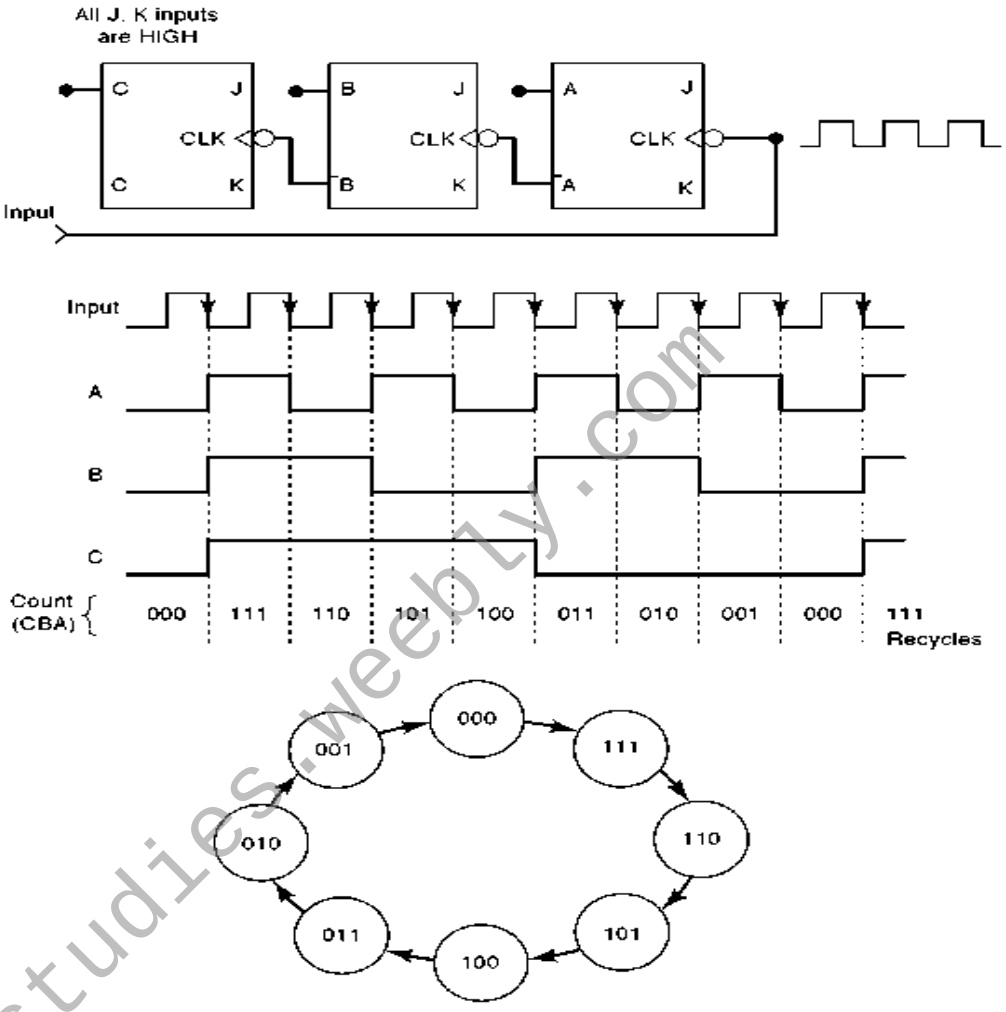
- Johnson counter – a *twisted* ring counter
- Cascaded counter

Asynchronous Up-counter (Ripple Counter)

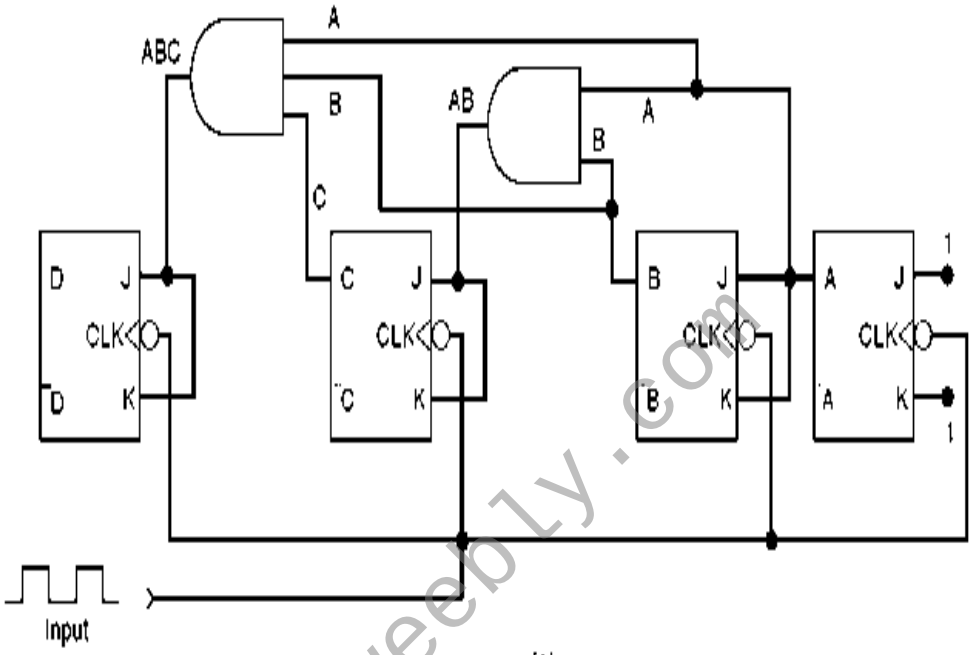


Asynchronous Down Counter

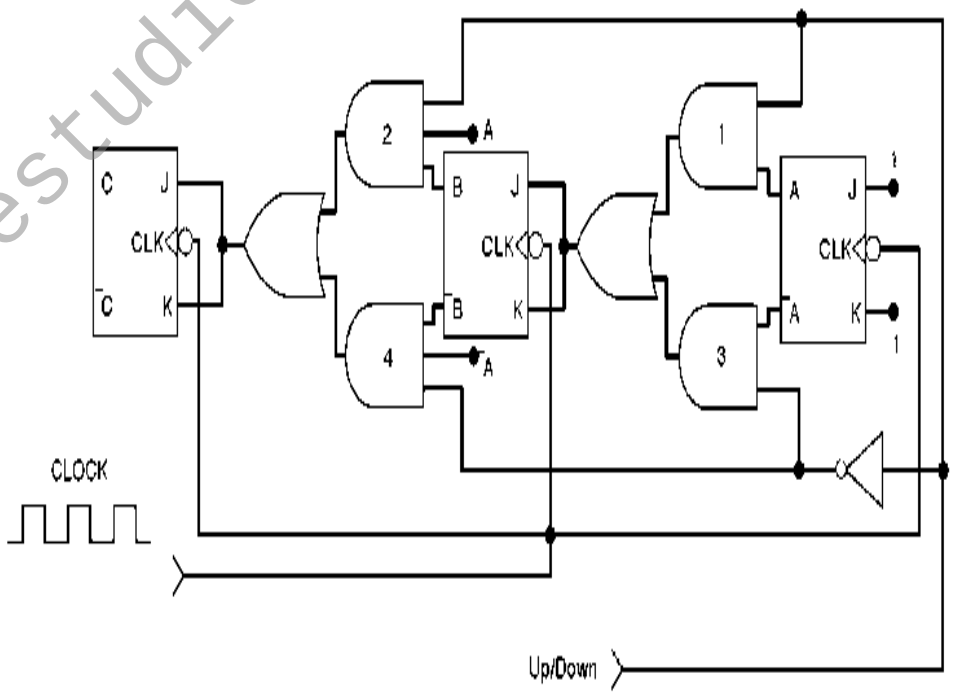




Synchronous (Parallel) Counter



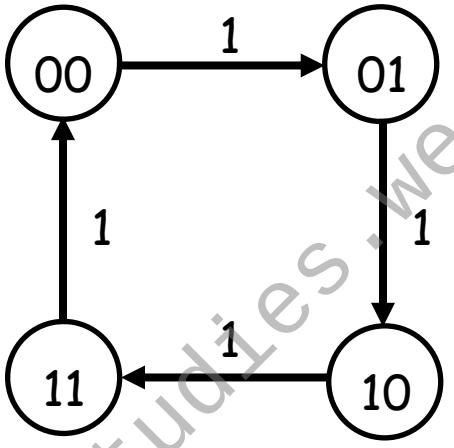
(a)



(a)

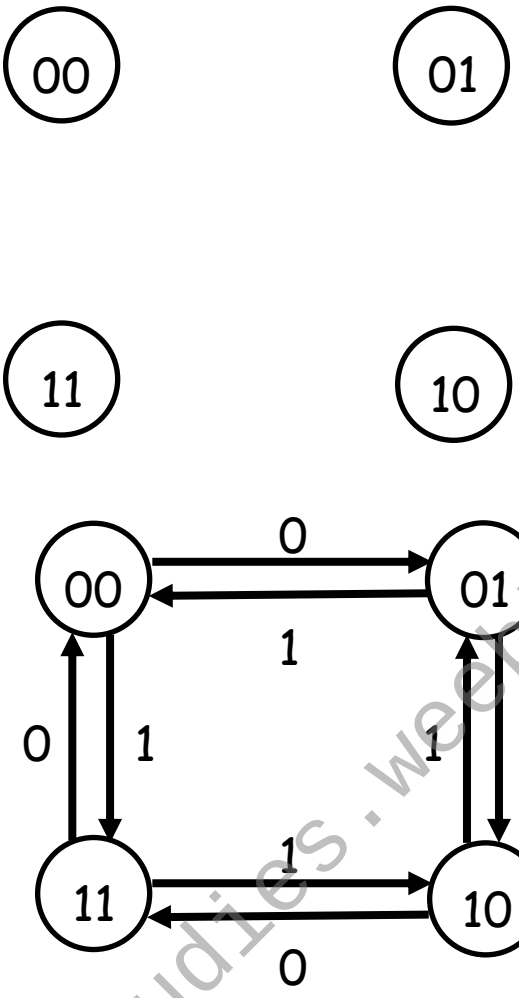
- Counters are a specific type of sequential circuit.
- Like registers, the state, or the flip-flop values themselves, serves as the "output."
- The output value increases by one on each clock cycle.
- After the largest value, the output "wraps around" back to 0.
- Using two bits, we'd get something like this:

Present State		Next State	
A	B	A	B
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0



Counter Design

- Let's try to design a slightly different two-bit counter:
  - Again, the counter outputs will be 00, 01, 10 and 11.
  - Now, there is a single input, X. When X=0, the counter value should *increment* on each clock cycle. But when X=1, the value should *decrement* on successive cycles.
- We'll need two flip-flops again. Here are the four possible states:



Present State Q_1	Present State Q_0	Inputs X	Next State Q_1	Next State Q_0
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

- If we use D flip-flops, then the D inputs will just be the same as the desired next states.
- Equations for the D flip-flop inputs are shown at the right.

- Why does D0 = Q0' make sense?

Present State		Inputs	Next State	
Q_1	Q_0	X	Q_1	Q_0
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

			Q_0	
	0	1	0	1
Q_1	1	0	1	0
		X		

$D1 = Q1 \oplus Q0 \oplus X$

			Q_0	
	1	1	0	0
Q_1	1	1	0	0
		X		

$$D_0 = Q_0'$$

- If we use JK flip-flops instead, then we have to compute the JK inputs for each flip-flop.
- Look at the present and desired next state, and use the excitation table on the right.

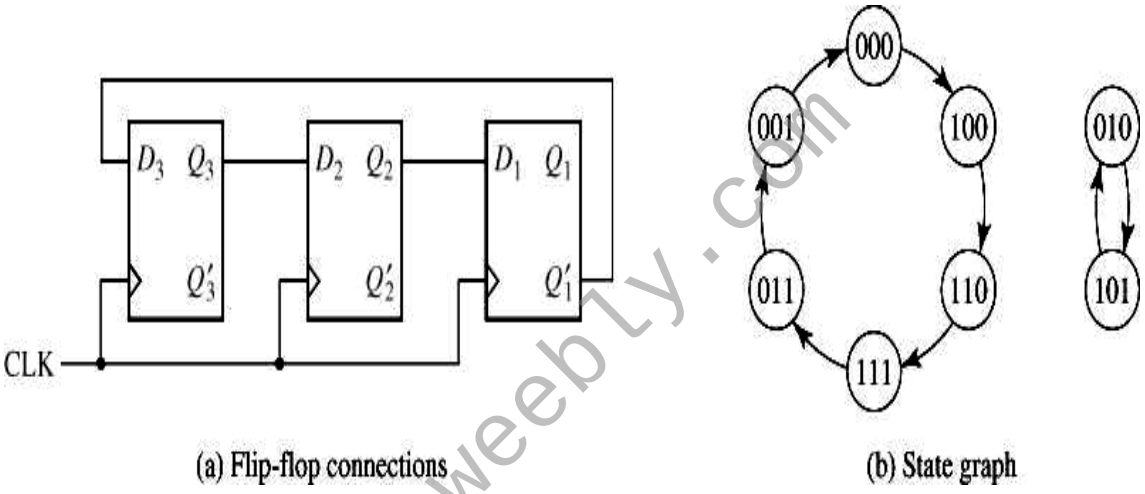
Present State		Inputs	Next State		Flip flop inputs			
Q_1	Q_0	X	Q_1	Q_0	J_1	K_1	J_0	K_0
0	0	0	0	1	0	x	1	x
0	0	1	1	1	1	x	1	x
0	1	0	1	0	1	x	x	1
0	1	1	0	0	0	x	x	1
1	0	0	1	1	x	0	1	x
1	0	1	0	1	x	1	1	x
1	1	0	0	0	x	1	x	1
1	1	1	1	0	x	0	x	1

We can then find equations for all four flip-flop inputs, in terms of the present state and inputs. Here, it turns out J1 = K1 and J0 = K0.

J1 = K1 = Q0' X + Q0 X'

J0 = K0 = 1

Johnson Counter



(a) Flip-flop connections (b) State graph

State Table

The state table representation of a sequential circuit consists of three sections labelled *present state*, *next state* and *output*. The present state designates the state of flip-flops before the occurrence of a clock pulse. The next state shows the states of flip-flops after the clock pulse, and the output section lists the value of the output variables during the present state.

State Diagram

In addition to graphical symbols, tables or equations, flip-flops can also be represented graphically by a state diagram. In this diagram, a state is represented by a circle, and the transition between states is indicated by directed lines (or arcs) connecting the circles. An example of a state diagram is shown in Figure below.

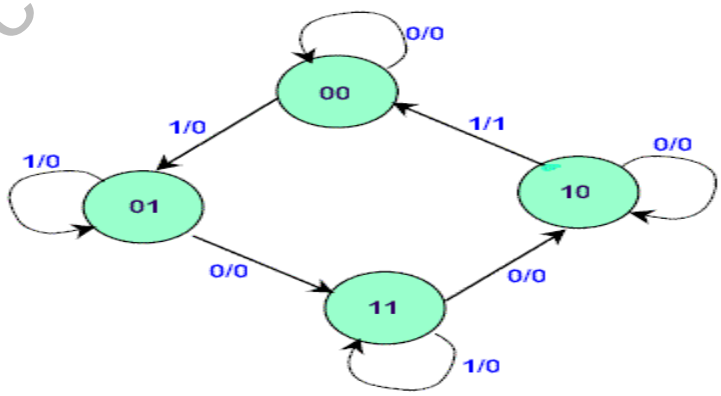


Figure . State Diagram

State Table

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
00	11	01	0	0
01	11	00	0	0
10	10	11	0	1
11	10	10	0	1

State reduction

In the design of sequential circuits, we need to reduce the number of flip flops and the number of logic gates used in the combinational circuit part. Reduction of the number of flip-flops may result from the reduction of the number of states in the circuit. This is possible if we are interested in the input output relationship of the circuit and not in the outputs of the flip-flops.

State Assignment

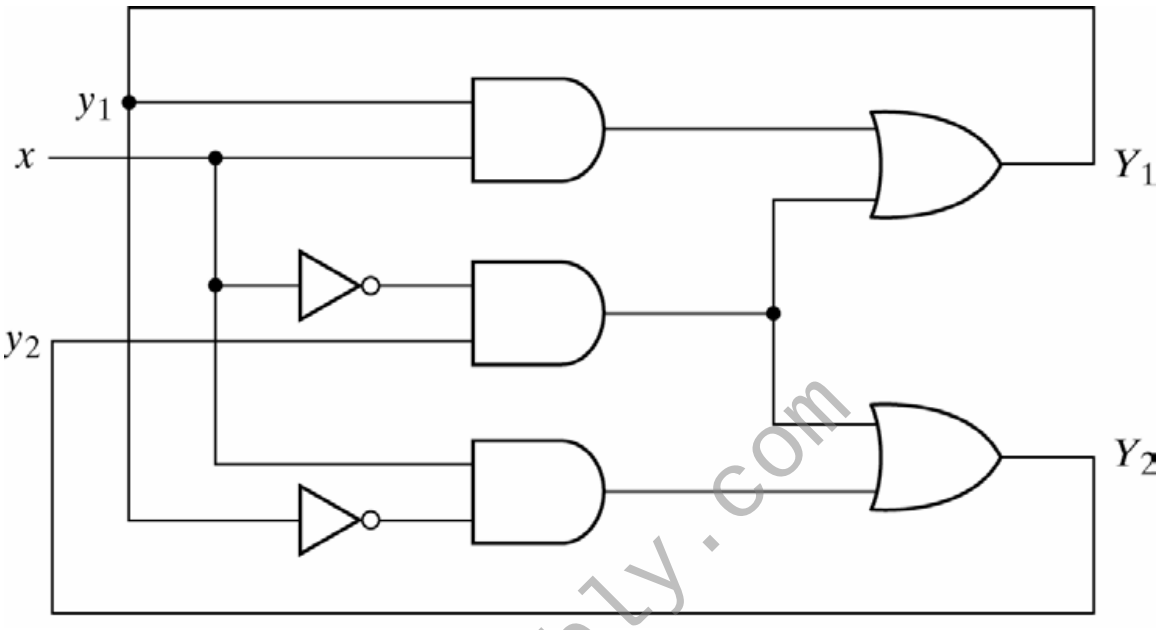
State Assignment procedures are concerned with methods for assigning binary values to states in such a way as to reduce the cost of combinational circuit that drives the flipflop.

State	Assignment 1	Assignment 2
A	001	000
B	000	010
C	010	101
D	110	111

Races

A **race condition** or **race hazard** is a flaw in an electronic system or process whereby the output and/or result of the process is unexpectedly and critically dependent on the sequence or timing of other events. The term originates with the idea of two signals racing each other to influence the output first.

Transition Table



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.
The next step is to plot the *Y*1 and *Y*2 functions in a map:

$y_1 y_2$ \ x	0	1
00	0	0
01	1	0
11	1	1
10	0	1

(a) Map for $Y_1 = xy_1 + x'y_2$

$y_1 y_2$ \ x	0	1
00	0	1
01	1	1
11	1	0
10	0	0

(b) Map for $Y_2 = xy'_1 + x'y_2$

Combining the binary values in corresponding squares the following *transition table* is obtained:

$y_1 y_2$ \ x	0	1
00	(00)	01
01	11	(01)
11	(11)	10
10	00	(10)

Primitive flow table: one that has only one stable state in each row.
To obtain the circuit described by a flow table, it is necessary to assign to each state a distinct binary value, which converts the flow table into a transition table.

	x = 0	x = 1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

(a) Four states with one input

	$x_1 x_2$ = 00	01	11	10
a	(a), 0	(a), 0	(a), 0	b , 0
b	a , 0	a , 0	(b), 1	(b), 0

(b) Two states with two inputs and one output

Races

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*. Examples of noncritical races are illustrated in the transition tables below:

$y_1 y_2$ \ x	0	1
00	(00)	11
01		11
11		(11)
10		11

(a) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10 \rightarrow 11$

$y_1 y_2$ \ x	0	1
00	(00)	11
01		(01)
11		01
10		11

(b) Possible transitions:

$00 \rightarrow 11 \rightarrow 01$
$00 \rightarrow 01$
$00 \rightarrow 10 \rightarrow 11 \rightarrow 01$

The transition tables below illustrate critical races:

$y_1 y_2$ \ x	0	1
00	(00)	11
01		(01)
11		(11)
10		(10)

(a) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01$
$00 \rightarrow 10$

$y_1 y_2$ \ x	0	1
00	(00)	11
01		11
11		(11)
10		(10)

(b) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10$

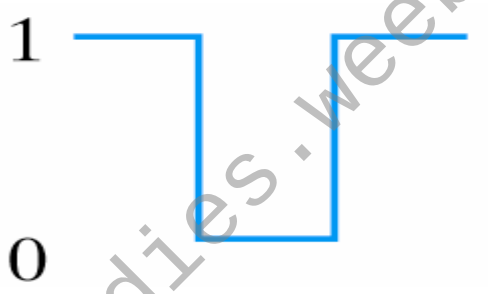
Hazard

In digital logic, a **hazard** in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as **static**, **dynamic** and **function hazards**.

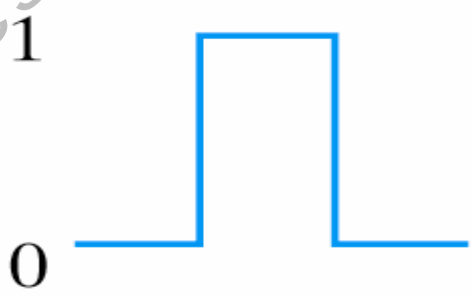
Hazards are a temporary problem, as the logic circuit will eventually settle to the desired function. However, despite the logic arriving at the correct output, it is imperative that hazards be eliminated as they can have an effect on other connected systems.

A **static hazard** is the situation where, when one input variable changes, the output changes momentarily before stabilizing to the correct value. There are two types of static hazards:

- Static-1 Hazard: the output is currently 1 and after the inputs change, the output momentarily changes to 0 before settling on 1

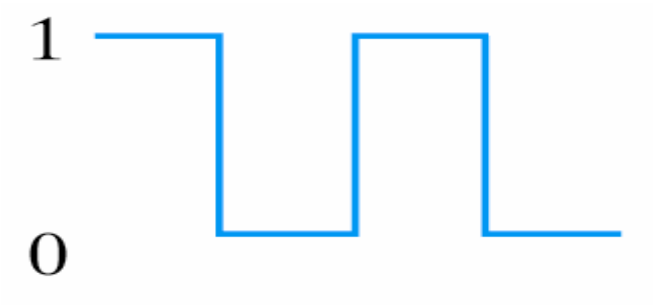


- Static-0 Hazard: the output is currently 0 and after the inputs change, the output momentarily changes to 1 before settling on 0

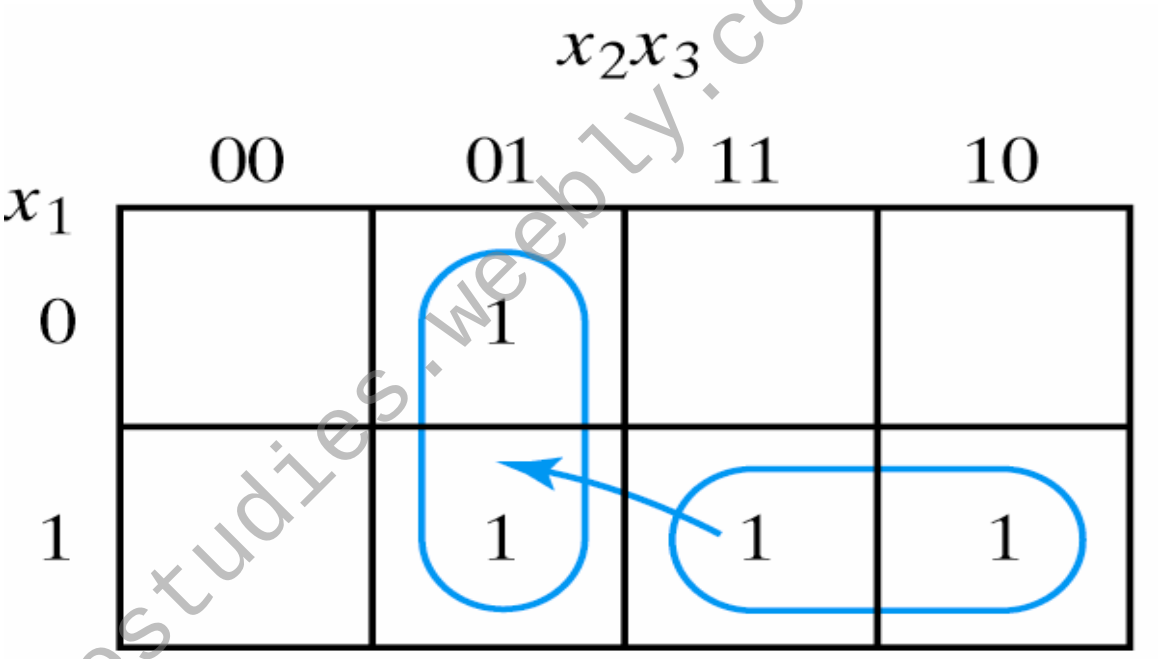


A **dynamic hazard** is the possibility of an output changing more than once as a result of a single input change. Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input). If each route has a different delay, then it quickly becomes clear that there is the potential for changing output values that differ from the required / expected output. e.g. A logic circuit is meant to change output state

from **1** to **0**, but instead changes from **1** to **0** then **1** and finally rests at the correct value **0**. This is a dynamic hazard.



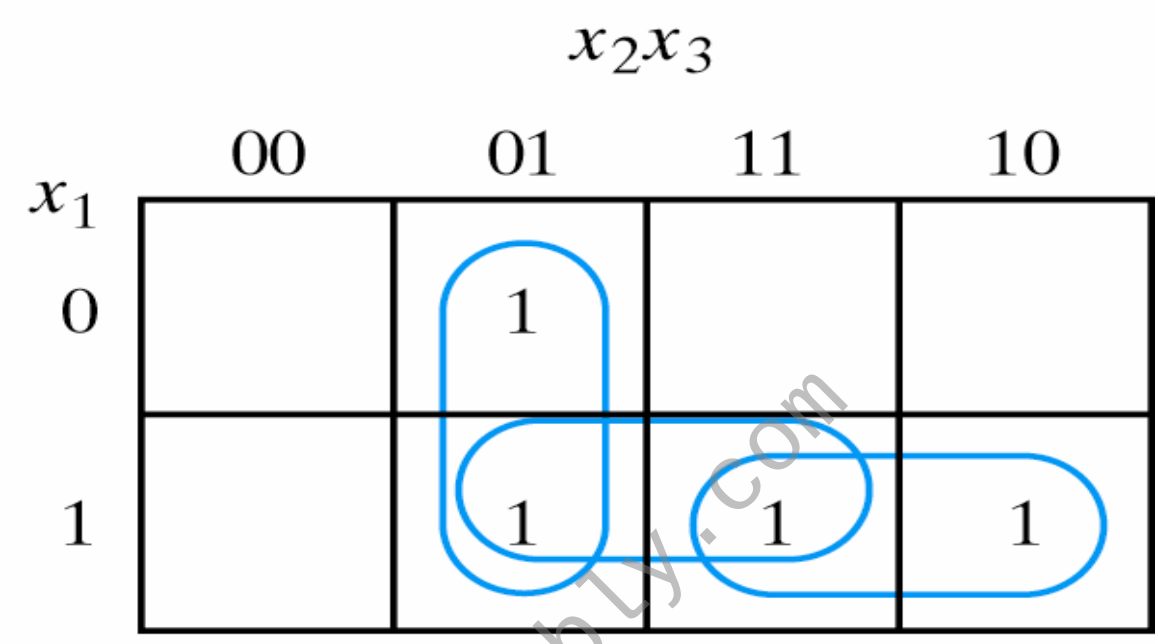
Example of Hazard free circuit is,



Normal K'Map answer is

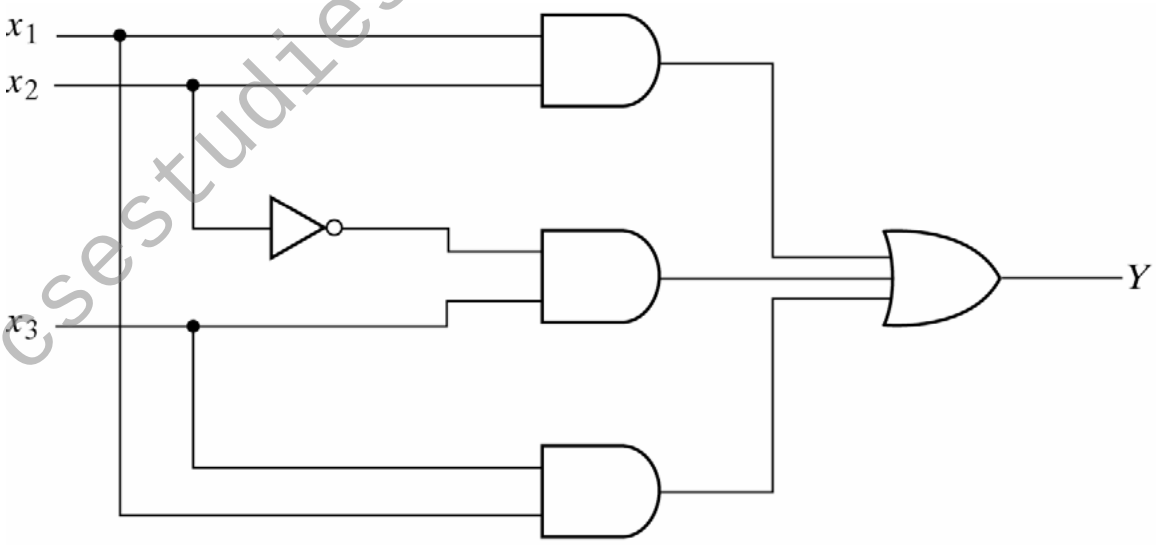
$$Y = x1x2 + x2'\,x3$$

Hazard free map is



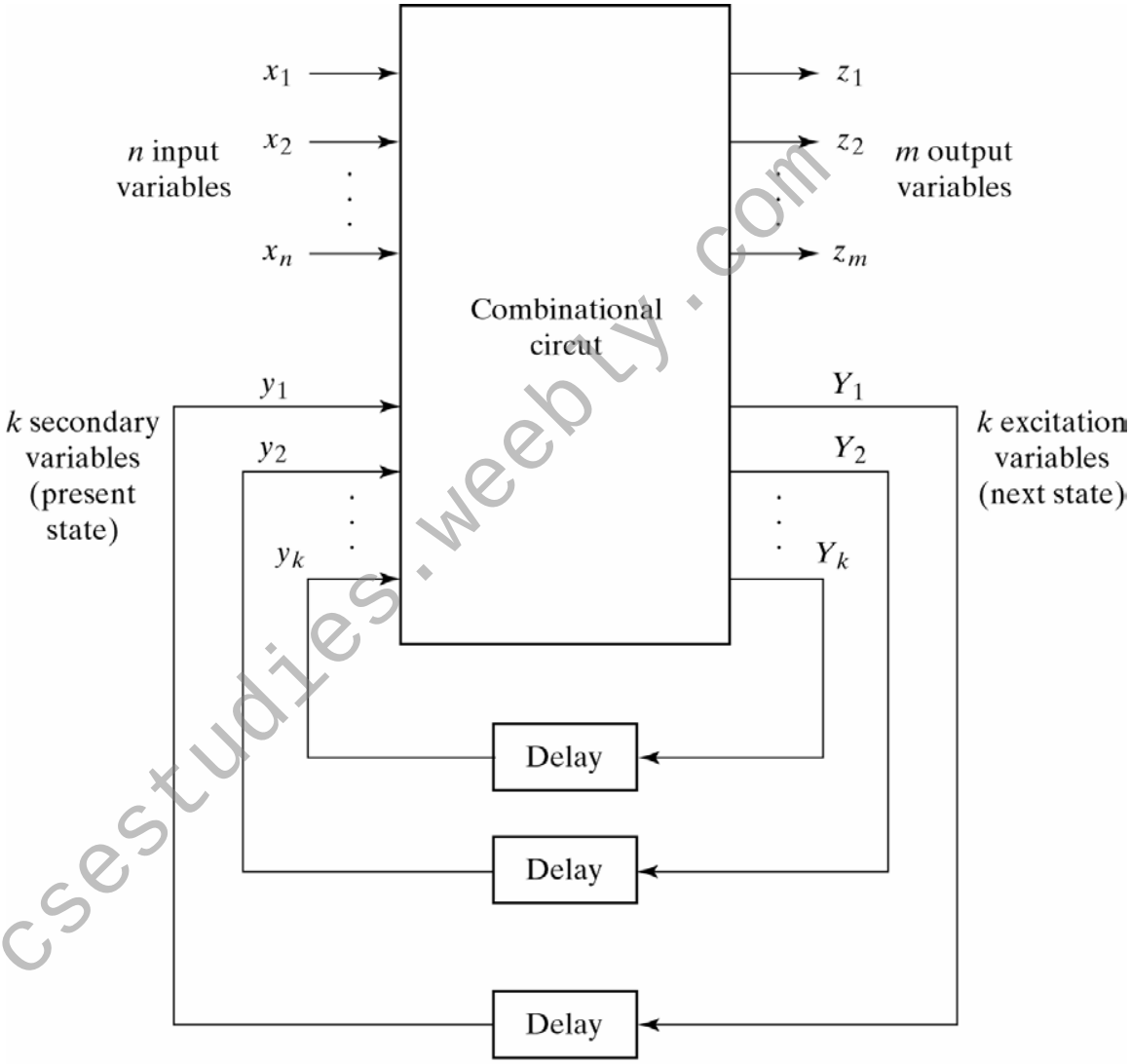
$$Y = x1x2 + x2'\,x3 + x1x3$$

Hazard Free Logic diagram is



Synchronous sequential logic

Nearly all sequential logic today is 'clocked' or 'synchronous' logic: there is a 'clock' signal, and all internal memory (the 'internal state') changes only on a clock edge. The basic storage element in sequential logic is the flip-flop.



The main advantage of synchronous logic is its simplicity. Every operation in the circuit must be completed inside a fixed interval of time between two clock pulses, called a 'clock cycle'. As long as this condition is met (ignoring certain other details), the circuit is guaranteed to be reliable. Synchronous logic also has two main disadvantages, as follows.

1. The clock signal must be distributed to every flip-flop in the circuit. As the clock is usually a high-frequency signal, this distribution consumes a relatively large amount of

power and dissipates much heat. Even the flip-flops that are doing nothing consume a small amount of power, thereby generating waste heat in the chip.

2. The maximum possible clock rate is determined by the slowest logic path in the circuit, otherwise known as the critical path. This means that every logical calculation, from the simplest to the most complex, must complete in one clock cycle. One way around this limitation is to split complex operations into several simple operations, a technique known as 'pipelining'. This technique is prominent within microprocessor design, and helps to improve the performance of modern processors.

In digital circuit theory, **sequential logic** is a type of logic circuit whose output depends not only on the present input but also on the history of the input. This is in contrast to *combinational logic*, whose output is a function of, and only of, the present input. In other words, sequential logic has *storage* (*memory*) while combinational logic does not.

Sequential logic is therefore used to construct some types of computer memory, other types of delay and storage elements, and finite state machines. Most practical computer circuits are a mixture of combinational and sequential logic.

There are two types of finite state machine that can be built from sequential logic circuits:

- Moore machine: the output depends only on the internal state. (Since the internal state only changes on a clock edge, the output only changes on a clock edge too).
- Mealy machine: the output depends not only on the internal state, but also on the inputs.

Depending on regulations of functioning, digital circuits are divided into synchronous and asynchronous. In accordance with this, behavior of devices obeys synchronous or asynchronous logic.

Asynchronous sequential logic

Asynchronous sequential logic expresses memorizing effect by fixing moments of time, when digital device changes its state. These moments are represented not in explicit form, but taking into account principle “before/after” in temporal relations of logical values. For asynchronous logic it is sufficient to determine a sequence of switchings irrespective of any connections of the corresponding moments with real or virtual time.
Theoretical apparatus of sequential logic consists of mathematical instruments of sequention and venjunction as well as of logic-algebraic equations on their basis.

An asynchronous sequential circuit is a sequential circuit whose behavior depends only on the order in which its input signals change and can be affected at any instant of time.

Memory (delay) elements are either latches (unclocked) or time-delay elements (instead of clocked FFs as in a synchronous sequential circuit).

+ An asynchronous sequential circuit quite often resembles a combinational circuit with feedback.

+ Faster and often cheaper than synchronous ones, but more difficult to design, verify, or test (due to possible timing problems involved in the feedback path).

Analysis Procedure

The analysis consists of obtaining a table or a diagram that describes the sequence of internal states and outputs as a function of changes in the input variables.

1. Determine all feedback loops.
2. Designate each feedback-loop output with Yi and its corresponding input with yi for i = 1; : : : ; k, where k is the number of feedback loops.
3. Derive the boolean functions for all Y 's.
4. Plot the transition table from the equations.

Design Procedure

1. Obtain a primitive flow table.
2. Reduce the flow table.
3. Assign binary state variables to obtain the transition table.
4. Assign output values to the dashes to obtain the output maps.
5. Simplify the excitation and output functions.
6. Draw the logic diagram.

Example problem

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*.

Solution

State table

State	Inputs		Output
	D	G	Q
a	0	1	0
b	1	1	1
c	0	0	0
d	1	0	0
e	1	0	1
f	0	0	1

Primitive Flow table

	DG = 00	DG = 01	DG = 11	DG = 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* , –

Informal Merging

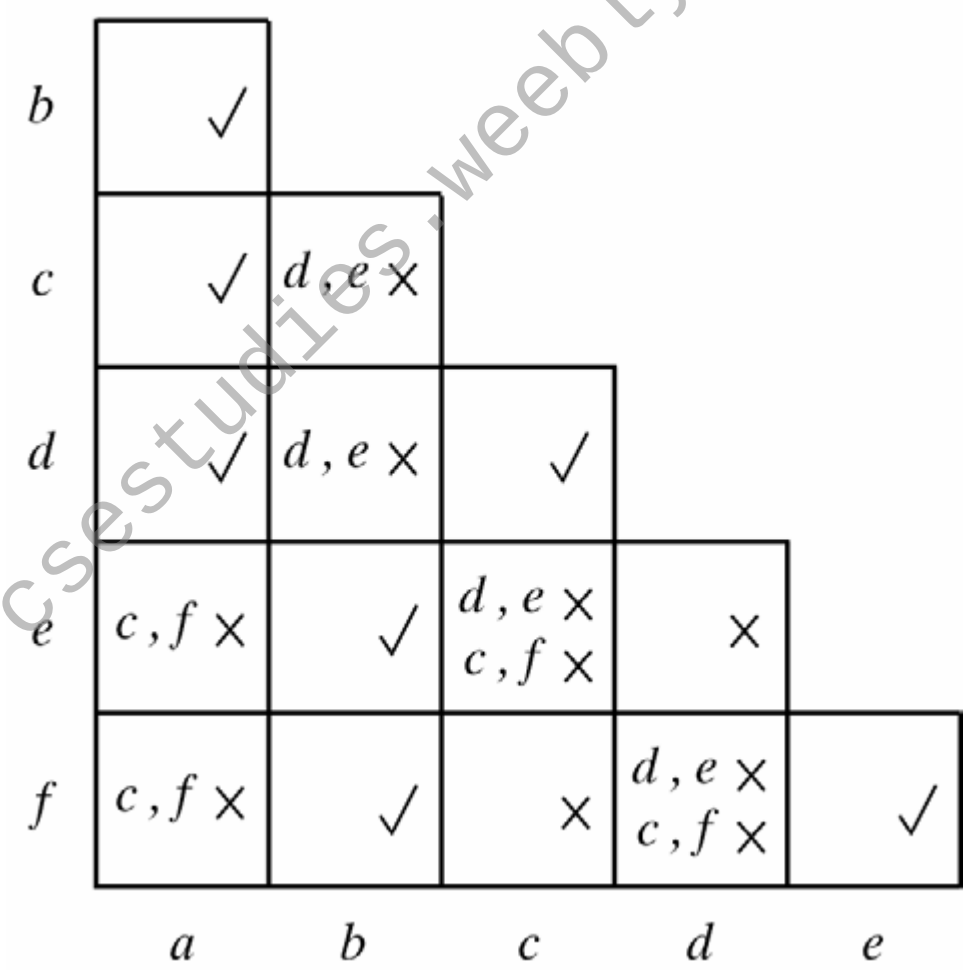
	DG 00	01	11	10
a, c, d	(*c*), 0	(*a*), 0	*b* , –	(*d*), 0
b, e, f	(*f*), 1	*a* , –	(*b*), 1	(*e*), 1

	DG 00	01	11	10
a	(*a*), 0	(*a*), 0	*b* , –	(*a*), 0
b	(*b*), 1	*a* , –	(*b*), 1	(*b*), 1

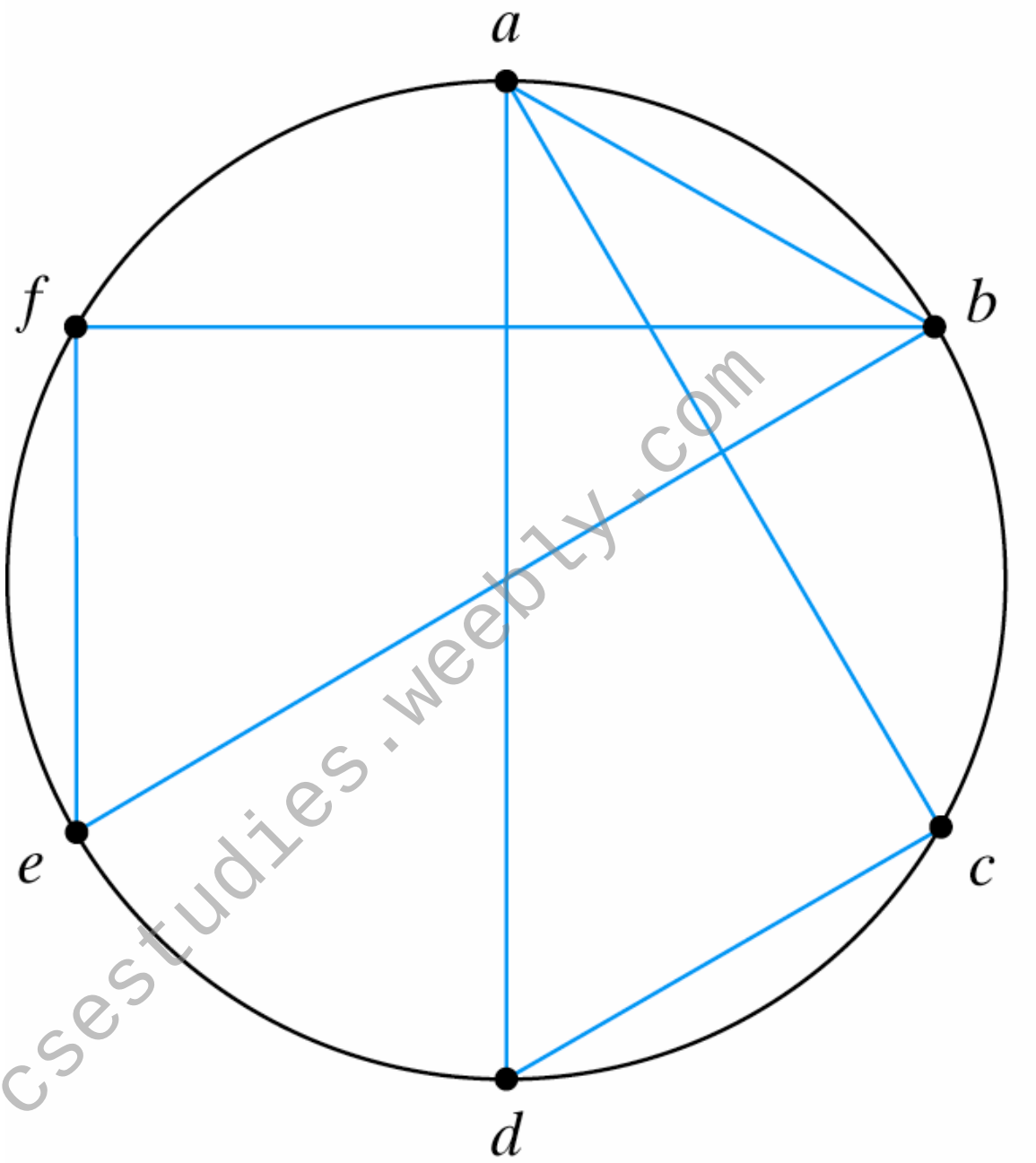
(b) Reduced table (two alternatives)

Formal Merging

Compatible Pairs



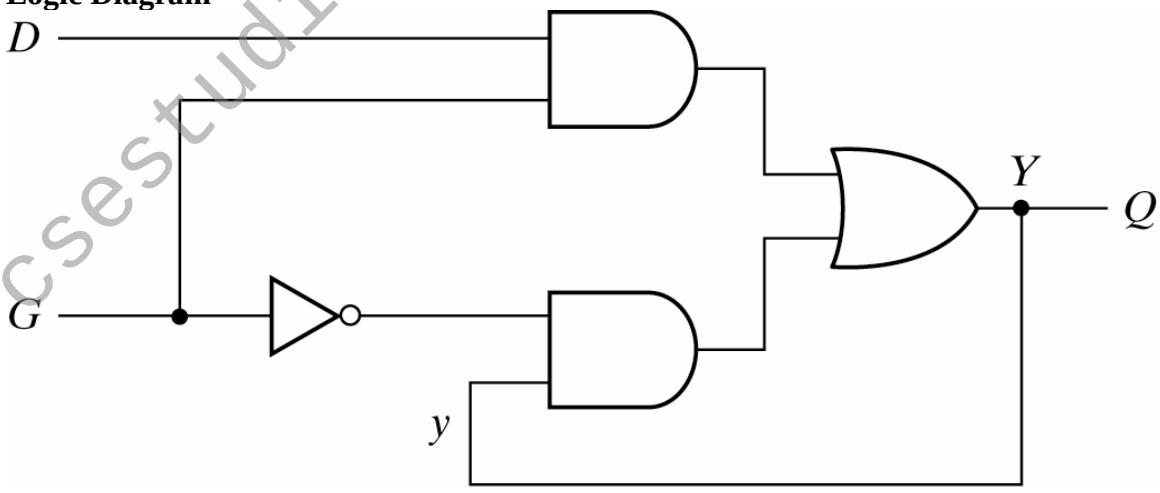
Maximal Compatibles



Reduced Table

	DG = 00	DG = 01	DG = 11	DG = 10
a	(a), 0	(a), 0	b, –	(a), 0
b	(b), 1	a, –	(b), 1	(b), 1

Logic Diagram



The same problem using SR Latch

Lists the required inputs *S* and *R* for each of the possible transitions from the secondary variable *y* to the excitation variable *Y*.

y	*Y*	*S*	*R*
0	0	0	*X*
0	1	1	0
1	0	0	1
1	1	*X*	1

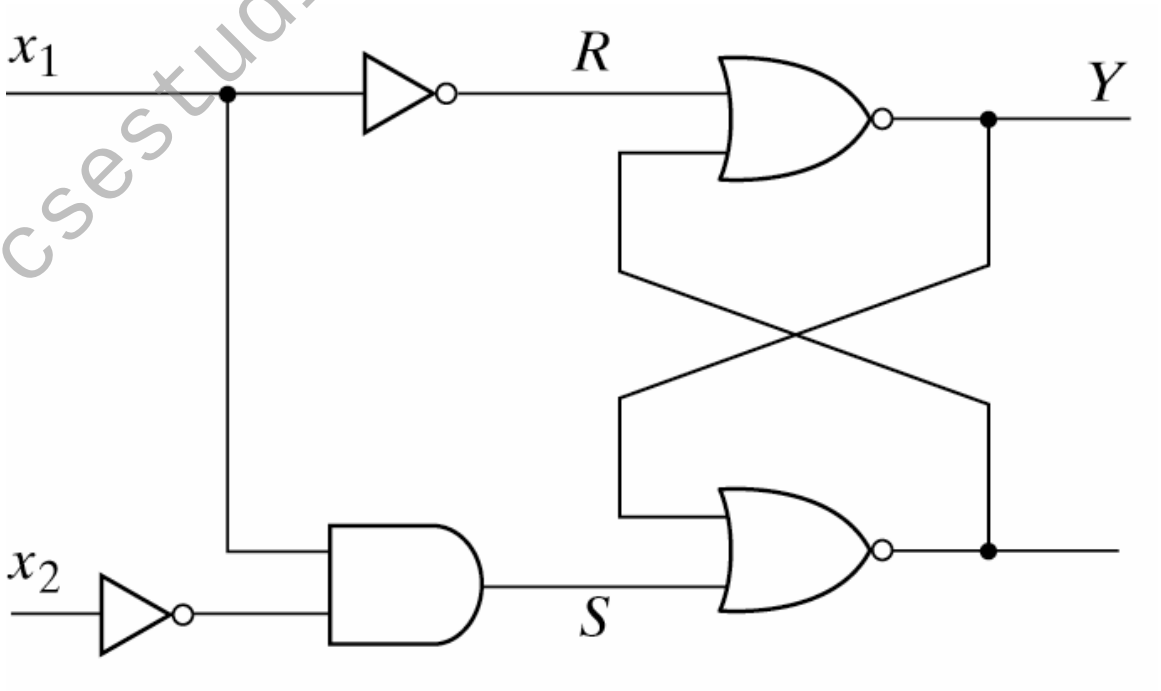
y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	*X*	*X*

(c) Map for $S = x_1x'_2$

y \ x_1x_2	00	01	11	10
0	*X*	*X*	*X*	0
1	1	1	0	0

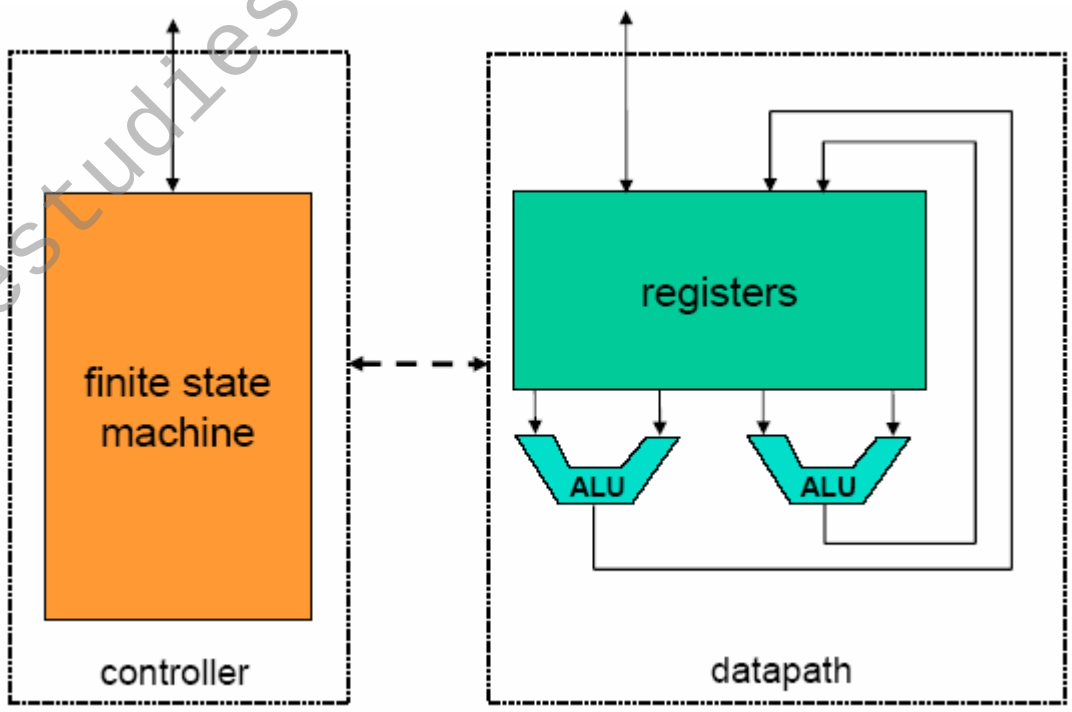
(d) Map for $R = x'_1$

SR Latch Logic Diagram



HDL

- Basic idea is a programming language to describe hardware
- The module starts with **module** keyword and finishes with **endmodule**.
- Internal signals are named with **wire**.
- Comments follow //
- **input** and **output** are ports. These are placed at the start of the module definition.
- Each statement ends with a semicolon, except **endmodule**.
- Design entry using both structure and behaviour
- Simulation modelling
- Testing
- It is a standard language
- *Register* - Retains the last value assigned to it. Often used to represent storage elements.
- 'wire' equivalent; when there are multiple drivers driving them, the outputs of the drivers are shorted together.
- Arithmetic operators - *, /, +, -, %
- Logical operators - ! □ logical negation && □ logical AND | | □ logical OR
- Relational operators >, <, >=, <=, ==, !=
- Bitwise operators ~, &, |, ^, ~^
- Reduction operators (operate on all the bits within a word) &, ~&, |, ~|, ^, ~^
- □ accepts a single word operand and produces a single bit as output
- Shift operators >>, <<
- Concatenation { }
- Replication { n { } }
- Conditional <condition> ? <expression1> : <expression2>



Basic Structure

```
module module_name (list_of_ports);
    input/output declarations;
    local net declarations;
    parallel statements;
endmodule
```

Simple AND circuit for HDL

```
module simpleand (f, x, y);
    input x, y;
    output f;
    assign f = x & y;
endmodule
```

Simple circuit diagram

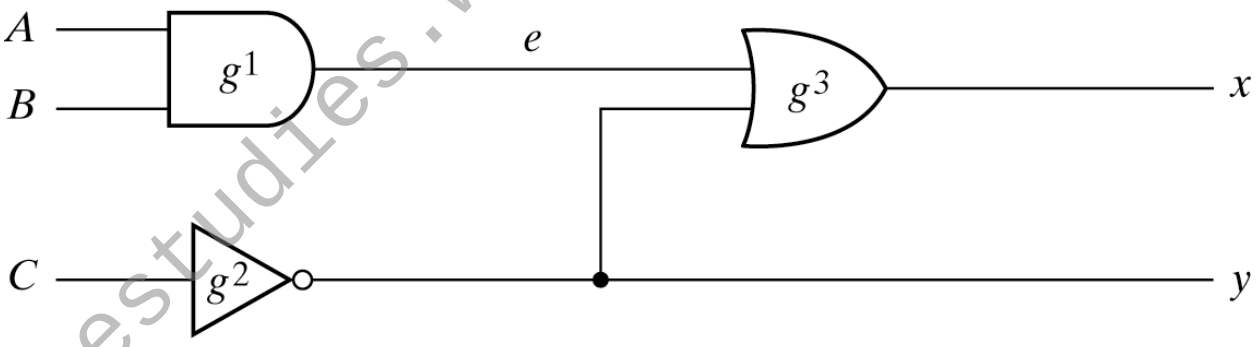


Fig. 3-37 Circuit to Demonstrate HDL

Simple HDL

```
module smpl_circuit(A,B,C,x,y);
  input A,B,C;
  output x,y;
  wire e;
  and g1(e,A,B);
```

```
   not g2(y, C);
   or  g3(x,e,y);
endmodule
```

Simple circuit with delay

```
module circuit_with_delay (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```


Effect of Delay

Time (ns)	Input A B C	Output y e x
<0	0 0 0	1 0 1
0	1 1 1	1 0 1
10	1 1 1	0 0 1
20	1 1 1	0 0 1

30	1 1 1	0 1 0
40	1 1 1	0 1 0
50	1 1 1	0 1 1

Simple circuit for Boolean functions

$x = A.B + C^1$
$y = C^1$

```
//Circuit specified with Boolean equations
module circuit_bln (x,y,A,B,C);
  input A,B,C;
  output x,y;
  assign x = A | (B & ~C);
  assign y = ~C ;
endmodule
```

Dataflow description of 2-input Mux

```
module mux2x1_df (A,B,select,OUT);
  input A,B,select;
  output OUT;
  assign OUT = select ? A : B;
endmodule
```

Behavioral description of 2-input mux

```
module mux2x1_bh(A,B,select,OUT);
  input A,B,select;
  output OUT;
  reg OUT;
  always @ (select or A or B)
      if (select == 1) OUT = A;
      else OUT = B;
endmodule
```

```
//D Latch
module DLatch(D, Control, Q, notQ);
        input D, Control;
        output Q, Qnot;
        reg Q, Qnot;
        always @ (Control or D)
        begin
                if (Control) Q = D;
                Qnot = ~Q;
        end
endmodule

//D flip-flop
module DFF(D, clock, Q, Qnot, Reset);
input D, clock, Reset;
output Q, Qnot;
reg Q, Qnot;
always @ (posedge clock or negedge Reset)
if (~Reset) {Q,Qnot} = 2'b01;
else {Q,Qnot} = {D, ~D};
endmodule
```

2 Marks Questions and Answers

1. Define the term digital.
   The term digital refers to any process that is accomplished using discrete units

2. What is meant by bit?
   A binary digit is called bit

3. What is the best example of digital system?
   Digital computer is the best example of a digital system.

4. Define byte?
   A group of 8 bits.

5. List the number systems?
   v) Decimal Number system
   vi) Binary Number system
   vii) Octal Number system
   viii) Hexadecimal Number system

6. State the sequence of operator precedence in Boolean expression?
   i) Parenthesis
   ii) AND
   iii) OR

7. What is the abbreviation of ASCII and EBCDIC code?
   ASCII-American Standard Code for Information Interchange.
   EBCDIC-Extended Binary Coded Decimal Information Code.

8. What are the universal gates?
   NAND and NOR

9. What are the different types of number complements?
   i) r's Complement
   ii) (r-1)'s Complement.

10. Why complementing a number representation is needed?
    Complementing a number becomes as in digital computer for simplifying the subtraction operation and for logical manipulation complements are used.

11. How to represent a positive and negative sign in computers?
    Positive (+) sign by 0
    Negative (-) sign by 1.

12. What is meant by Map method?

The map method provides a simple straightforward procedure for minimizing Boolean function.

13. What is meant by two variable map?

Two variable map have four minterms for two variables, hence the map consists of four squares, one for each minterm

14. What is meant by three variable map?

Three variable map have 8 minterms for three variables, hence the map consists of 8 squares, one for each minterm

15. Which gate is equal to AND-inverter Gate?

NAND gate.

16. Which gate is equal to OR-inverter Gate?

NOR gate.

17. Bubbled OR gate is equal to----------

NAND gate

18. Bubbled AND gate is equal to----------

NOR gate

19. What is the use of Don't care conditions?

Any digital circuit using this code operates under the assumption that these unused combinations will never occur as long as the system

.

20. Express the function f(x, y, z)=1 in the sum of minterms and a product of maxterms?

Minterms=$\sum(0,1,2,3,4,5,6,7)$

Maxterms=Nomaxterms.

21. What is the algebraic function of Exclusive-OR gate and Exclusive-NOR gate?

$F=xy^1 + x^1y$

$F=xy + x^1y^1$

22. What are the methods adopted to reduce Boolean function?

i) Karnaugh map

ii) Tabular method or Quine mccluskey method

iii) Variable entered map technique.

23. Why we go in for tabulation method?

This method can be applied to problems with many variables and has the advantage of being suitable for machine computation.

24. State the limitations of karnaugh map.

i) Generally it is limited to six variable map (i.e.) more then 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.

25. What is tabulation method?

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

26. What are prime-implicants?

The terms remained unchecked are called prime-implecants. They cannot be reduced further.

27. Explain or 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.

28. List out the advantages and disadvantages of Quine-Mc Cluskey method?

The advantages are,

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

The disadvantages are,

a. Lengthy procedure than K-map.
b. Requires several grouping and steps as compared to K-map.
c. It is much slower.
d. No visual identification of reduction process.
e. The Quine Mc Cluskey method is essentially a computer reduction method.

29. Define Positive Logic.

When high voltage or more positive voltage level is associated with binary '1' and while the low or less positive level is associated with binary '0' then the system adhering to this is called positive logic.

30. Define Negative Logic.

When high voltage level is associated with binary '0' and while the low level is associated with binary '1' then the system adhering to this is called negative logic

31. List the characteristics of digital Ics

i) propagation delay
ii) power dissipation
iii) Fan-in
iv) Fan-out
v) Noise margin

32. What is propagation delay?

It is the average transition delay time for the signal to propagate from input to output when the signals change in value.

33. What is Noise margin?

It is the limit of a noise voltage, which may be present with out impairing the proper operation of the circuit.

34. What is power dissipation?

It is the power consumed by the gate, which must be available from the power supply.

35. Why parity checker is needed?

Parity checker is required at the receiver side to check whether the expected parity is equal to the calculated parity or not. If they are not equal then it is found that the received data has error.

36. What is meant by parity bit?

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 and for errors.

37. Why parity generator necessary?

Parity generator is essential to generate parity bit in the transmitter.

38. What is IC?

An integrated circuit is a small silicon semiconductor crystal called a chip containing electrical components such as transistors, diodes, resistors and capacitors. The various components are interconnected inside the chip to form an electronic circuit.

39. What are the needs for binary codes?

a. Code is used to represent letters, numbers and punctuation marks.
b. Coding is required for maximum efficiency in single transmission.
c. Binary codes are the major components in the synthesis (artificial generation) of speech and video signals.
d. By using error detecting codes, errors generated in signal transmission can be detected.
e. Codes are used for data compression by which large amounts of data are transmitted in very short duration of time.

40.Mention the different type of binary codes?

The various types of binary codes are,

f. BCD code (Binary Coded decimal).
g. Self-complementing code.
h. The excess-3 (X's-3) code.
i. Gray code.
j. Binary weighted code.
k. Alphanumeric code.
l. The ASCII code.
m. Extended binary-coded decimal interchange code (EBCDIC).
n. Error-detecting and error-correcting code.
o. Hamming code.

41.List the advantages and disadvantages of BCD code?

The advantages of BCD code are

a. Any large decimal number can be easily converted into corresponding binary number
b. A person needs to remember only the binary equivalents of decimal number from 0 to 9.
c. Conversion from BCD into decimal is also very easy.

The disadvantages of BCD code are

a. The code is least efficient. It requires several symbols to represent even small numbers.
b. Binary addition and subtraction can lead to wrong answer.
c. Special codes are required for arithmetic operations.
d. This is not a self-complementing code.
e. Conversion into other coding schemes requires special methods.

42. What is meant by self-complementing code?

A self-complementing code is the one in which the members of the number system complement on themselves. This requires the following two conditions to be satisfied.

a. The complement of the number should be obtained from that number by replacing 1s with 0s and 0s with 1s.
b. The sum of the number and its complement should be equal to decimal 9. Example of a self-complementing code is

i. 2-4-2-1 code.
ii. Excess-3 code.

43. Mention the advantages of ASCII code?

The following are the advantages of ASCII code

a. There are 2^7 =128 possible combinations. Hence, a large number of symbols, alphabets etc.., can be easily represented.
b. There is a definite order in which the alphabets, etc.., are assigned to each code word.
c. The parity bits can be added for error-detection and correction.

44. What are the disadvantages of ASCII code?

The disadvantages of ASCII code are

a. The length of the code is larger and hence more bandwidth is required for transmission.
b. With more characters and symbols to represent, this is not completely sufficient.

45. What is the truth table?

A truth table lists all possible combinations of inputs and the corresponding outputs.

46. Define figure of merit?

Figure of merits is defined as the product of speed and power. The speed is specified in terms of propagation delay time expressed in nano seconds.

Figure of merits=Propagation delay time (ns)*
Power (mw)

It is specified in pico joules (ns*mw=PJ).

47. What are the two types of logic circuits for digital systems?

Combinational and sequential

48. Define Combinational circuit.

A combinational circuit consist of logic gates whose outputs at anytime are determined directly from the present combination of inputs without regard to previous inputs.

49. Define sequential circuits.

Their outputs are a function of the inputs and the state of memory elements. The state of memory elements, in turn, is a function of previous inputs.

50. What is a half-adder?

The combinational circuit that performs the addition of two bits are called a half-adder.

51. What is a full-adder?

The combinational circuit that performs the addition of three bits are called a half-adder.

52. What is half-subtractor?

The combinational circuit that performs the subtraction of two bits are called a half-sub tractor.

53. What is a full-subtractor?

The combinational circuit that performs the subtraction of three bits are called a half- sub tractor.

54. What is Binary parallel adder?

A binary parallel adder is a digital function that produces the arithemetic sum of two binary numbers in parallel.

55. What is BCD adder?

A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit also in BCD.

56. What is Magnitude Comparator?

A Magnitude Comparator is a combinational circuit that compares two numbers, A and B and determines their relative magnitudes.

57. What is decoder?

A decoder is a combinational circuit that converts binary information from 'n' input lines to a maximum of 2^n unique output lines.

58. What is encoder?

A decoder is a combinational circuit that converts binary information from 2^nInput lines to a maximum of 'n' unique output lines.

59. Define Multiplexing?

Multiplexing means transmitting a large number of information units over a smaller number of channels or lines.

60. What is Demultiplexer?

A Demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines

.

61. Give the truth table for a half adder.

Input		Output	
X	Y	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0

1	1	0	1

62. Give the truth table for a half Subtractor.

Input		Output	
X	Y	Borrow(B)	Diffe (D)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

63.From the truth table of a half adder derive the logic equation

$S = X \oplus Y$

$C = X . Y$

64. From the truth table of a half subractor derive the logic equation

$D = X \oplus Y$

$B = X^1 . Y$

65. From the truth table of a full adder derive the logic equation

$S = X \oplus Y \oplus Z$

$C = XY + YZ + XZ$

66. What is code conversion?

If two systems working with different binary codes are to be synchronized in operation, then we need digital circuit which converts one system of codes to the other. The process of conversion is referred to as code conversion.

67. What is code converter?

It is a circuit that makes the two systems compatible even though each uses a different binary code. It is a device that converts binary signals from a source code to its output code. One example is a BCD to Xs3 converter.

68. What do you mean by analyzing a combinational circuit?

The reverse process for implementing a Boolean expression is called as analyzing a combinational circuit. (ie) the available logic diagram is analyzed step by step and finding the Boolean function

69. 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.

70. Mention the uses of Demultiplexer.

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.

71. Give other name for Multiplexer and Demultiplexer.

Multiplexer is other wise called as Data selector.

Demultiplexer is otherwise called as Data distributor.

72. 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.

73. Give the truth table for a full Subtractor.

Input			Output	
X	Y	Z	Borrow (B)	Diffe (D)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

74. Give the truth table for a full adder.

Input			Output	
X	Y	Z	Sum (S)	Carry (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

75. From the truth table of a full subtractor derive the logic equation

$S = X \oplus Y \oplus Z$

$C = X^1Y + YZ + X^1Z$

76. What is priority encoder?

A priority encoder is an encoder that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

77. Can a decoder function as a Demultiplexer?

Yes. A decoder with enable can function as a Demultiplexer if the enable line E is taken as a data input line A and B are taken as selection lines.

78. List out the applications of multiplexer?

The various applications of multiplexer are

a. Data routing.
b. Logic function generator.
c. Control sequencer.
d. Parallel-to-serial converter.

79. List out the applications of decoder?

The applications of decoder are

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.

80. List out the applications of comparators?

The following are the applications of comparator

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.

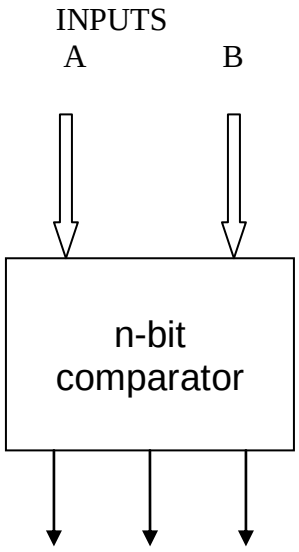
81. What are the applications of seven segment displays?

The seven segment displays are used in

a. LED displays
b. LCD displays

82. What is digital comparator?

A comparator is a special combinational circuit designed primarily to compare the relative magnitude of two binary numbers.



A>B A=B A<B

OUTPUTS

Block diagram of n-bit comparator

83. List the types of ROM.

i) Programmable ROM (PROM)
ii) Erasable ROM (EPROM)
iii) Electrically Erasable ROM (EEROM)

84. Differentiate ROM & PLD's

ROM (Read Only Memory)	PLD's (Programmable Logic Array)
1.It is a device that includes both the decoder and the OR gates with in a single IC package	1.It is a device that includes both AND and OR gates with in a single IC package
2.ROM does not full decoding of the variables and does generate all the minterms	2.PLD's does not provide full decoding of the variable and does not generate all the minterms

85. What are the different types of RAM?

The different types of RAM are

a. NMOS RAM (Nitride Metal Oxide Semiconductor RAM)
b. CMOS RAM (Complementary Metal Oxide Semiconductor RAM)
c. Schottky TTL RAM
d. ELL RAM.

86. What are the types of arrays in RAM?

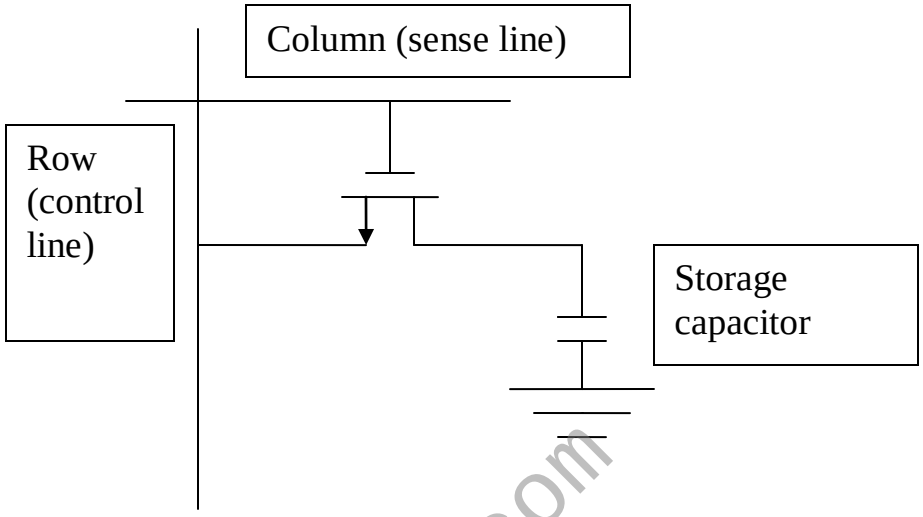
RAM has two type of array namely,

a. Linear array
b. Coincident array

87. Explain DRAM?

The dynamic RAM (DRAM) is an operating mod, which stores the binary information in the form of electric charges on capacitors.

The capacitors are provided inside the chip by MOS transistors.



DRAM cell

The stored charges on the capacitors tend to discharge with time and the capacitors must be tending to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.

DRAM offers reduced power consumption and larger storage capacity in a single memory chip.

88. Explain SRAM?

Static RAM (SRAM) consists of internal latches that store the binary information. The stored information remains valid as long as the power is applied to the unit.

SRAM is easier to use and has shorter read and write cycle.

The memory capacity of a static RAM varies from 64 bit to 1 mega bit.

89. Differentiate volatile and non-volatile memory?

Volatile memory	Non-volatile memory
They are memory units which lose stored information when power is turned off. E.g. SRAM and DRAM	It retains stored information when power is turned off. E.g. Magnetic disc and ROM

90. What are the terms that determine the size of a PAL?

The size of a PLA is specified by the

a. Number of inputs
b. Number of products terms
c. Number of outputs

91. What are the advantages of RAM?

The advantages of RAM are

a. Non-destructive read out

b. Fast operating speed
c. Low power dissipation
d. Compatibility
e. Economy

92. What is VHDL?

VHDL is a hardware description language that can be used to model a digital system at many level of abstraction, ranging from the algorithmic level to the gate level.

The VHDL language as a combination of the following language.

a. Sequential language
b. Concurrent language
c. Net-list language
d. Timing specification
e. Waveform generation language.

93. What are the features of VHDL?

The features of VHDL are

a. VHDL has powerful constructs.
b. VHDL supports design library.
c. The language is not case sensitive.

94. Define entity?

Entity gives the specification of input/output signals to external circuitry. An entity is modeled using an entity declaration and at least one architecture body. Entity gives interfacing between device and others peripherals.

95. List out the different elements of entity declaration?

The different elements of entity declaration are:

1. entity_name
2. signal_name
3. mode
4. in:
5. out:
6. input
7. buffer
8. signal_type

96. Give the syntax of entity declaration?

```
ENTITY      entity_name is
PORT  (signal_name: mode signal_type;
            signal_names: mode signal_type;
                :
                :
            signal_names: mode signal_type;
```

END entity_name;

97. What do you meant by concurrent statement?

Architecture contains only concurrent statements. It specifies behavior, functionality, interconnections or relationship between inputs and outputs.

98. What are operates used in VDHL language?

There are different types of operators used in VHDL language

Logical operators : AND, OR, NOT, XOR, etc.,
Relational operator : equal to, <less than etc.,
Shift operators : SLL- Shift Left Logical,
ROR- Rotate Right Logical etc.,
Arithmetic operators: Addition, subtraction etc.,
Miscellaneous operators: <= assign to etc.,

99. Define VHDL package?

A VHDL, package is a file containing definitions of objects which can be used in other programs. A package may include objects such as signals, type, constant, function, procedure and component declarations

100. What is meant by memory decoding?

The memory IC used in a digital system is selected or enabled only for the range of addresses assigned to it .

101. What is access and cycle time?

The access time of the memory is the time to select word and read it. The cycle time of a memory is a time required to complete a write operation.

102. What is sequential circuit?

Sequential circuit is a broad category of digital circuit whose logic states depend on a specified time sequence. A sequential circuit consists of a combinational circuit to which memory elements are connected to form a feedback path.

103. List the classifications of sequential circuit.

i) Synchronous sequential circuit.
ii) Asynchronous sequential circuit.

104. What is Synchronous sequential circuit?

A Synchronous sequential circuit is a system whose behavior can be defined from the knowledge of its signal at discrete instants of time.

105. What is clocked sequential circuits?

Synchronous sequential circuit that use clock pulses in the inputs of memory elements are called clocked sequential circuit. One advantage as that they don't cause instability problems.

106. What is called latch?

Latch is a simple memory element, which consists of a pair of logic gates with their inputs and outputs inter connected in a feedback arrangement, which permits a single bit to be stored.

107. List different types of flip-flops.

i) SR flip-flop
ii) Clocked RS flip-flop
iii) D flip-flop
iv) T flip-flop
v) JK flip-flop
vi) JK master slave flip-flop

108. What do you mean by triggering of flip-flop.

The state of a flip-flop is switched by a momentary change in the input signal. This momentary change is called a trigger and the transition it causes is said to trigger the flip-flop

109. What is an excitation table?

During the design process we usually know the transition from present state to next state and wish to find the flip-flop input conditions that will cause the required transition. A table which lists the required inputs for a given chance of state is called an excitation table.

110. Give the excitation table of a JK flip-flop

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

111. Give the excitation table of a SR flip-flop

Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

112. Give the excitation table of a T flip-flop

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

113. Give the excitation table of a D flip-flop

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	0
1	1	1

114. What is a characteristic table?

A characteristic table defines the logical property of the flip-flop and completely characteristic its operation.

115. Give the characteristic equation of a SR flip-flop.

$Q(t+1)=S+R^1Q$

116. Give the characteristic equation of a D flip-flop.

$Q(t+1)=D$

117. Give the characteristic equation of a JK flip-flop.

$Q(t+1)=JQ^1+K^1Q$

118. Give the characteristic equation of a T flip-flop.

$Q(t+1)=TQ^1+T^1Q$

119. What is the difference between truth table and excitation table.

i) An excitation table is a table that lists the required inputs for a given change of state.

ii) A truth table is a table indicating the output of a logic circuit for various input states.

120. What is counter?

A counter is used to count pulse and give the output in binary form.

121. What is synchronous counter?

In a synchronous counter, the clock pulse is applied simultaneously to all flip-flops. The output of the flip-flops change state at the same instant. The speed of operation is high compared to an asynchronous counter

122. What is Asynchronous counter?

In a Asynchronous counter, the clock pulse is applied to the first flip-flops. The change of state in the output of this flip-flop serves as a clock pulse to the next flip-flop and so on. Here all the flip-flops do not change state at the same instant and hence speed is less.

123. What is the difference between synchronous and asynchronous counter?

Sl.No.	Synchronous counter	Asynchronous counter
1.	Clock pulse is applied simultaneously	Clock pulse is applied to the first flip-flop, the change of output is given as clock to next flip-flop
2.	Speed of operation is high	Speed of operation is low.

124. Name the different types of counter.

a) Synchronous counter
b) Asynchronous counter
   i) Up counter
   ii) Down counter
   iii) Modulo – N counter
   iv) Up/Down counter

125. What is up counter?

A counter that increments the output by one binary number each time a clock pulse is applied.

126. What is down counter?

A counter that decrements the output by one binary number each time a clock pulse is applied.

127. What is up/down counter?

A counter, which is capable of operating as an up counter or down counter, depending on a control lead.

128. What is a ripple counter?

A ripple counter is nothing but an asynchronous counter, in which the output of the flip-flop change state like a ripple in water.

129. What are the uses of a counter?

i) The digital clock
ii) Auto parking control
iii) Parallel to serial data conversion.

130. What is meant by modulus of a counter?

By the term modulus of a counter we say it is the number of states through which a counter can progress.

131. What is meant by natural count of a counter?

By the term natural count of a counter we say that the maximum number of states through which a counter can progress.

132.A ripple counter is a ---------- sequential counter.

Ans: Synchronous.

133. What is a modulo counter?

A counter that counts from 0 to T is called as modulo counter.

134. A counter that counts from to T is called a modulo counter. True or False.

Ans: True

135. The number of flip-flops required for modulo-18 counter is -------

Ans: five.

136. Form the truth table for 3-bit binary down counter.

Clk	Q2	Q1	Q0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
1	1	1	1

137. What is a ring counter?

A counter formed by circulating a 'bit' in a shift register whose serial output has been connected to its serial input.

138. What is BCD counter?

A BCD counter counts in binary coded decimal from 0000 to 1001 and back to 0000. Because of the return to 0000 after a count of 1001, a BCD counter does not have a regular pattern as in a straight binary counter.

139. What are the uses of a ring counter?

i) Control section of a digital system.

ii) Controlling events, which occur in strict time sequence.

140. What is a register?

Memory elements capable of storing one binary word. It consists of a group of flip-flops, which store the binary information.

141. What is Johnson counter?

It is a ring counter in which the inverted output is fed into the input. It is also know as a twisted ring counter.

142. What is a shift register?

In digital circuits, datas are needed to be moved into a register (shift in) or moved out of a register (shift out). A group of flip-flops having either or both of these facilities is called a shift register.

143. What is serial shifting?

In a shift register, if the data is moved 1 bit at a time in a serial fashion, then the technique is called serial shifting.

144. What is parallel shifting?

In a shift register all the data are moved simultaneously and then the technique is called parallel shifting.

145. Write the uses of a shift register.

i) Temporary data storage
ii) Bit manipulations.

146. What is a cycle counter?

A cycle counter is a counter that outputs a stated number of counts and then stops.

147. Define state of sequential circuit?

The binary information stored in the memory elements at any given time defines the “state” of sequential circuits.

148. Define state diagram.

A graphical representation of a state table is called a state diagram.

149. 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.

150. What is state table?

A table, which consists time sequence of inputs, outputs and flip-flop states, is called state table. Generally it consists of three section present state, next state and output.

151. 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.

152. What is meant by race around condition?

In JK flip-flop output is fed back to the input, and therefore changes in the output results change in the input. Due to this in the positive half of the clock pulse if J and K are both high then output toggles continuously. This condition is known as race around condition.

153. How many bits would be required for the product register if the multiplier has 6 bits and the multiplicand has 8 bits?

The product register is 14-bit width with extra bit at the left end indicating a temporary storage for any carry, which is generated when the multiplicand is added to the accumulator.

154. What is SM chart?

Just as flow charts are useful in software design, flow charts are useful in the hardware design of digital systems. These flow charts are called as State Machine Flow Charts or SM charts. SM charts are also called as ASMC (Algorithmic State machine chart). ASM chart describes the sequential operation in a digital system.

155. What are the three principal components of SM charts?

The 3 principal components of SM charts are state box, decision box & Conditional output box.

156. What is decision box?

A diamond shaped symbol with true or false branches represents a decision box. The condition placed in the box is a Boolean expression that is evaluated to determine which branch to take in SM chart.

157. What is link path? How many entrance paths & exit paths are there in SM block?

A path through an SM block from entrance to exit is referred to as link path. An SM block has one entrance and exit path.

158. Differentiate ASM chart and conventional flow chart?

A conventional Flow chart describes the sequence of procedural steps and decision paths for an algorithm without concern for their time relationship.

The ASM chart describes the sequence of events as well as the timing relationships between the states of a sequential controller and the events that occur while going from one state to the next.

159. What is flow table?

During the design of synchronous sequential circuits, it is more convenient to name the states by letter symbols without making specific reference to their binary values. Such table is called Flow table.

160. What is primitive flow table?

A flow table is called Primitive flow table because it has only one stable state in each row.

161. Define race condition.

A race condition is said to exist in a synchronous sequential circuit when two or more binary state variables change, the race is called non-critical race.

162. Define critical & non-critical race with example.

The final stable state that the circuit reaches does not depend on the order in which the state variables change, the race is called non-critical race.

The final stable state that the circuit reaches depends on the order in which the state variables change, the race is called critical race.

163. How can a race be avoided?

Races can be avoided by directing the circuit through intermediate unstable states with a unique state – variable change.

164. Define cycle and merging?

When *a* circuit goes through a u*n*ique sequence of unstable states, *i*t is said to have a cyc*l*e.

The *g*rouping of stab*l*e states from s*e*parate rows into o*n*e common row is called merging.

165. Give state – reduction procedure.

The state – reduction procedure for co*m*pletely specified st*a*te tables is base*d* on t*h*e algorithm that two stat*e*s in a state table can be combined in to one if they can be shown to be equi*va*le*n*t.

166. Define hazards.

Hazards are unwanted switching transients that may appear at the output of a circuit because different paths exhibit different propagation delays.

167. Does Hazard occur in sequential circuit? If so what is the problem caused?

Yes, Hazards occur in sequential circuit that is Asynchronous sequential circuit. It may result in a transition to a wrong state.

168. Give the procedural steps for determining the compatibles used for the purpose of merging a flow table.

The purpose that must be applied in order to find a suitable group of compatibles for the purpose of merging a flow table can be divided into 3 procedural steps.

i. Determine all compatible pairs by using the implication table.
ii. Find the maximal compatibles using a Merger diagram
iii. Find a minimal collection of compatibles that covers all the states and is closed.

169. What are the types of hazards?

The 3 types of hazards are 1) Static – 0 hazards
2) Static – 1 hazard
3) Dynamic hazards

170. What is mealy and Moore circuit?

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.

171. Differentiate Moore circuit and Mealy circuit?

Moore circuit	Mealy circuit
a. It is output is a function of present state only. b. Input changes do not affect the output. c. Moore circuit requires more number of states for implementing same function.	a. It is output is a function of present state as well as the present input. b. Input changes may affect the output of the circuit. c. It requires less numbers of states for implementing same function.

172. How can the hazards in combinational circuit be removed?

Hazards in the combinational circuits can be removed by covering any two min terms 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.

173. How does an essential hazard occur?

An essential hazard occurs due to 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 causes essential hazard.

174. What is Timing diagram?

Timing diagrams are frequently used in the analysis of sequential network. These diagrams show various signals in the network as a function of time.

175. What is setup and hold time?

The definite time in which the input must be maintained at a constant value prior to the application of the pulse is setup time

The definite time is which the input must not chance after the application of the positive or negative going transition of the pulse based on the triggering of the pulse.

176. Define bit time and word time.

The time interval between clock pulses is called bit time.

The time required to shift the entire contents of a shift register is called word time.

177. 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.

178. Define equivalent state.

If a state machine is started from either of two states and identical output sequences are generated from every possible set of sequences, then the two states are said to be equivalent.

179. If a shift register can be operated in all possible ways then it is called as----------

Ans: Univerasal register: It can be operated in all possible modes with bi-directional shift facility.

180. What is gate delay?

If the change in output is delayed by a time ε with respect to the input. We say that the gate has a propagation delay of ε. Normally propagation delay for 0 to 1 output (ε1) may be different than the delay for 1 to 0 changes (ε2).

181. Define state reduction algorithm.

State reduction algorithm is stated as "Two states are said to be equivalent if, for each member of the set of inputs they give the same output and send the circuit either to the same state or to an equivalent state. When two states are equivalent, one of them can be removed without altering the input-output relation.

182. What is meant by level triggering?

In level triggering the output of the flip-flop changes state or responds only when the clock pulse is present.

183. Write the uses of a shift register.

i) Temporary data storage.

ii) Bit manipulations.

184. What is meant by flow table?

During the design of asynchronous sequential circuits, it is more convenient to name the states by letter symbols without making specific reference to their binary values. Such a table is called a flow table.

185. What are the problems involved in asynchronous circuits?

The asynchronous sequential circuits have three problems namely,

a. Cycles
b. Races
c. Hazards

186. Define cycles?

If an input change includes a feedback transition through more than unstable state then such a situation is called a cycle.

187. Define primitive flow table?

A primitive flow table is a flow table with only one stable total state in each row. Remember that a total state consists of the internal state combined with the input.

188. Define merging?

The primitive flow table has only one stable state in each row. The table can be reduced to a smaller numbers of rows if two or more stable states are placed in the same row of the flow table. The grouping of stable states from separate rows into one common row is called merging.

189. What are the types of Registers?

i) Buffer Register
ii) Shift Register
iii) Uni directional Shift Register
iv) Bidirectional Shift Register