B.E Electronics and Communication Engineering **2019-2020**

Semester-III

19BEEC303 **DIGITAL SYSTEM DESIGN** **3H-3C**

Instruction Hours/week: L: 3 T:0 P: 0 **Marks:** Internal**:40** External**:60** Total**:100**

End Semester Exam:3 Hours

Course Objective

- To introduce the theoretical and circuit aspects of digital electronics, which is the back bone for the basics of the hardware aspect of digital computers

Course Outcomes

At the end of this course students will demonstrate the ability to

- Design and analyze combinational logic circuits
- Design & analyze modular combinational circuits with MUX/DEMUX, Decoder, Encoder
- Design & analyze synchronous sequential logic circuits
- Differentiate different logical families
- Gain knowledge about various memory devices and implement using PLDs
- Use HDL & appropriate EDA tools for digital logic design and simulation

UNIT I BOOLEAN ALGEBRA

Number system, Logic Simplification and Combinational Logic Design: Review of Boolean Algebra and De-Morgan's Theorem, SOP & POS forms, Canonical forms, Karnaugh maps up to 6 variables, Binary codes, Code Conversion.

UNIT II COMBINATIONAL CIRCUITS

Logic gates, AND & NOR implementation, MSI devices - Comparators, Multiplexers, Encoder, Decoder, Driver & Multiplexed Display, Half and Full Adders, Subtractors, Serial and Parallel Adders, BCD Adder, Barrel shifter and ALU.

UNIT III SEQUENTIAL LOGIC DESIGN

D,S-R, JK FF and Master-Slave JK FF, Edge triggered FF, Ripple and Synchronous counters, Shift registers, Finite state machines, Design of synchronous FSM, Algorithmic State Machines charts. Designing synchronous circuits like Pulse train generator, Pseudorandom Binary Sequence generator, Clock generation

UNIT IV LOGIC FAMILIES AND SEMICONDUCTOR MEMORIES

TTL NAND gate, Specifications, Noise margin, Propagation delay, fan-in, fan-out, Tristate TTL, ECL, CMOS families and their interfacing, Memory elements, Concept of Programmable logic devices like FPGA. Logic implementation using Programmable Devices.

UNIT V INTRODUCTION TO VHDL

VLSI Design flow: Design entry: Schematic, FSM & HDL, different modeling styles in VHDL, Data types and objects, Dataflow, Behavioral and Structural Modeling, Synthesis and Simulation VHDL constructs and codes for combinational and sequential circuits.

Suggested Readings

1. R.P. Jain, “Modern digital Electronics”, Tata McGraw Hill, 4th edition, 2009.
2. Douglas Perry, “VHDL”, Tata McGraw Hill, 4th edition, 2002.
3. W.H. Gothmann, “Digital Electronics- An introduction to theory and practice”, PHI, 2nd edition ,2006.
4. D.V. Hall, “Digital Circuits and Systems”, Tata McGraw Hill, 2004
5. Charles Roth, “Digital System Design using VHDL”, Tata McGraw Hill 2nd edition 2012.

Karpagam Academy of Higher Education

(Established under Section 3 of UGC Act 1956)

Eachanari, Coimbatore-641 021. INDIA

Department of Electronics and Communication Engineering
Faculty of Engineering

DIGITAL ELECTRONICS

Lecture Notes –Unit I

PREPARED BY
Dr.S.Bhavani,HOD/ECE

UNIT-I NUMBER SYSTEMS

Binary, Octal, Decimal, Hexadecimal-Number base conversions – complements – signed Binary numbers. Binary Arithmetic- Binary codes: Weighted –BCD-2421-Gray code-Excess 3 code-ASCII –Error detecting code – conversion from one code to another-Boolean postulates and laws –De-Morgan's Theorem- Principle of Duality- Boolean expression – Boolean function-Minimization of Boolean expressions – Sum of Products (SOP) –Product of Sums (POS)-Minterm- Maxterm- Canonical forms – Conversion between canonical forms –Karnaugh map Minimization – Don't care conditions.

UNIT-I NUMBER SYSTEMS

1. Introduction to Digital System

Any system has input, output and processing unit. What processing it does depend on the type of the system For example if it is an adder system, we will have inputs, the outputs will be sum and carry and the processing unit will have units to perform addition operation on inputs to generate the required output. The term digital refers to any process that is accomplished in discrete manner. A best example of a digital system is Digital computer.

The Electronic circuits are basically classified into two types namely

i) Analog
ii) Digital

Almost all digital circuits are logical circuits because

1. It is easier to manipulate logical values (i.e., 0/1 or low/high values) than to manipulate and process multiple (Discrete) voltage levels.

2. Everyone can use the formal laws of Boolean logic to design logic circuits easily and systematically. Hence these laws which are used to formulate digital logic are called as Boolean algebra or Switching algebra

1.1Types of Number Systems

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

Complements

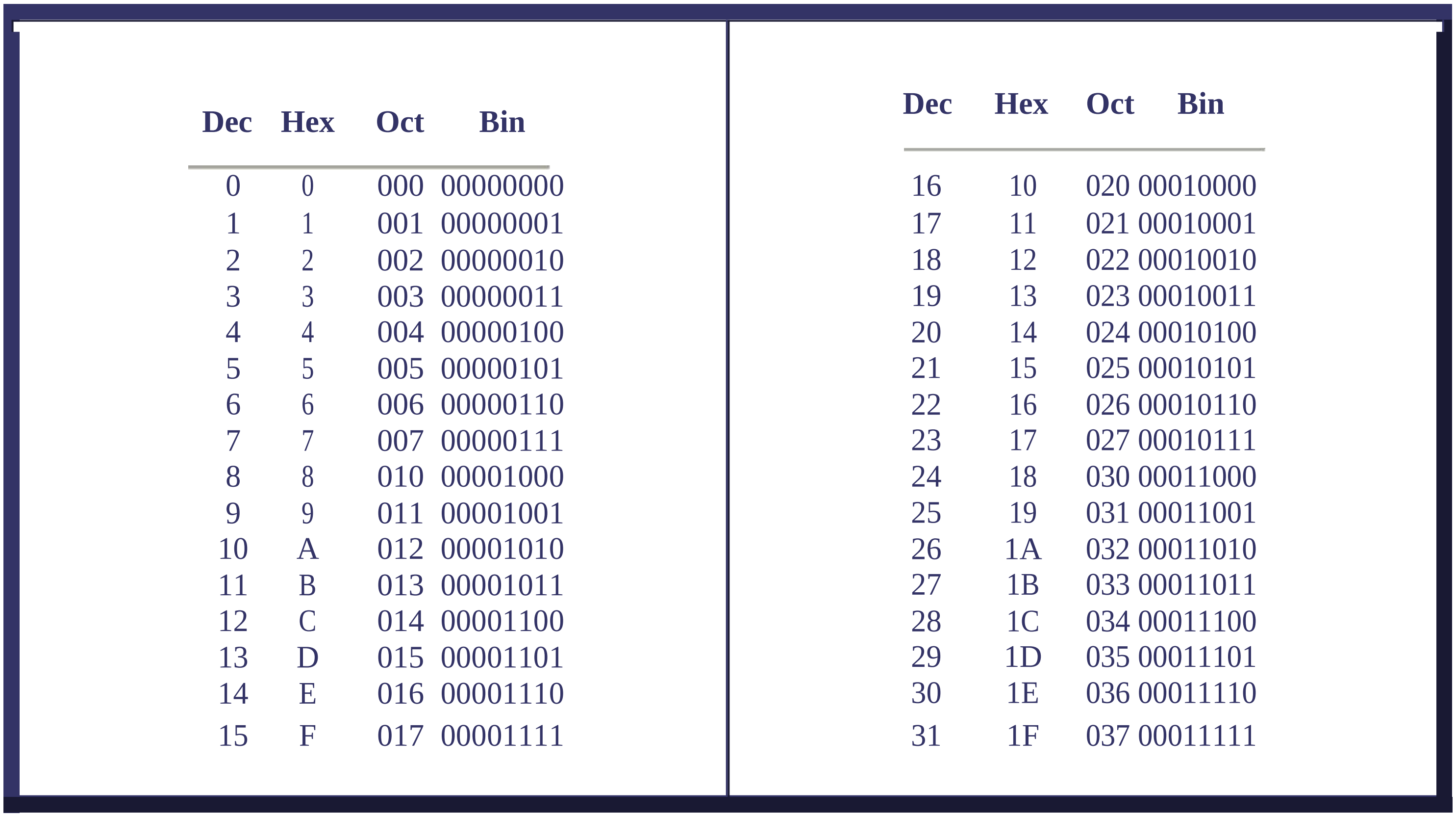
Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. For any numbering system, there are two types of complements available, which are

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

In general complements are used to have reduction in computation time and complexity. When we go for arithmetic operations we will find the subtraction operation in simple way with less computation in 2’s complement compared to I’s complement.

For Binary numbering system we have 2’s complement and 1’s complement.
For Decimal numbering system we have 10’s complement and 9’s complement
For Octal numbering system we have 8’s complement and 7’s complement
For Hexadecimal numbering system we have 16’s complement and 15’s complement

Binary Codes



Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
0	0	000	00000000	16	10	020	00010000
1	1	001	00000001	17	11	021	00010001
2	2	002	00000010	18	12	022	00010010
3	3	003	00000011	19	13	023	00010011
4	4	004	00000100	20	14	024	00010100
5	5	005	00000101	21	15	025	00010101
6	6	006	00000110	22	16	026	00010110
7	7	007	00000111	23	17	027	00010111
8	8	010	00001000	24	18	030	00011000
9	9	011	00001001	25	19	031	00011001
10	A	012	00001010	26	1A	032	00011010
11	B	013	00001011	27	1B	033	00011011
12	C	014	00001100	28	1C	034	00011100
13	D	015	00001101	29	1D	035	00011101
14	E	016	00001110	30	1E	036	00011110
15	F	017	00001111	31	1F	037	00011111

Conversion of codes from one form to another:

1. 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}

2. 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

3.Binary to Octal

100 111 0102 = (100) (111) (010)2 = 4 7 28

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}

4. 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}

5. Hexadecimal to binary

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

6. Hexadecimal to octal

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

1.2 Binary Arithmetic Operations:

1.2.1 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)

1.2.2 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)

1.2.3 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	=	$41_{(\text{base } 10)}$
		× 0	0	0	0	0	1	1	0	=	$6_{(\text{base } 10)}$
		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	=	$246_{(\text{base } 10)}$

00010111 × 00000011 = 01000101

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

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

Binary Division

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

1.2.4 Rules of Binary Divison

1/1= 1
0/0=0

For example,

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

```
= 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   1 0
                          -   1   0   1
                          _____________
                                      1   1
                                  -       0
                                  _____________
                                      1   1   1
                                  -   1   0   1
                                  _____________
                                          1   0   1
                                      -   1   0   1
                                          _____________
                                                      0
```

1.3 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

1.3.1 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

1. 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

2. 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.

3. 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.

1.3.2. 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.

1. 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

2. 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

3. 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.

1. **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.

1.3.3. 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.

1. **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.

2. **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).

Simplification of Boolean functions can be done by using

i) Boolean theorems and postulates.
ii) Karnaugh Map(K-Map)
iii) Prime Implicant Method or Tabulation methods or Quine Mc-Cluskey Method

1.4 Boolean Algebra theorems and postulates.

Principle of duality

Principle of duality states that a dual expression can be obtainbed by Interchanging the OR and AND operatio expression, Interchanging the 0 and 1 elements of the expression and 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: Idempotent Law for Addition

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

T5: Negation Law

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

(b) (A)’’= A

T6: Redundance Law

(a) $A + A B = A$

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

T7: Law of Identity

(a) $0 + A = A$

(b) $1 A = A$

(c) $1 + A = 1$

(d) $0 A = 0$

T8 : Idempotent law for Multiplication

(a) $\overline{A} + A = 1$

(b) $\overline{A}\ A = 0$

T9 : Absorption Law

(a) $A + \overline{A}\ B = A + B$

(b) $A\ (\overline{A} + B) = A\ B$

T10 : De Morgan's Theorem

(a) (A+B)’= A’ . B’

(b) (A.B)’= A’+B’

1.5 Canonical Form of Boolean Expressions

An expanded form of Boolean expression, where each term contains all Boolean variables in their true or complemented form, is also known as the canonical form of the expression. As an illustration, $f(A.B, C) = \overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + A.B.C$ is a Boolean function of three variables expressed in canonical form. This function after simplification reduces to $\overline{A}.\overline{B} + A.B.C$ and loses its canonical form

1.5.1 MIN TERMS AND MAX TERMS

Any boolean expression may be expressed in terms of either minterms or maxterms. To do this we must first define the concept of a literal. A literal is a single variable within a term which may or may not be complemented. For an expression with N variables, minterms and maxterms are defined as follows :

- A minterm is the product of N distinct literals where each literal occurs exactly once.
- A maxterm is the sum of N distinct literals where each literal occurs exactly once.

Product-of-Sums Expressions

1.5.2 Standard Forms

A product-of-sums expression contains the product of different terms, with each term being either a single literal or a sum of more than one literal. It can be obtained from the truth table by considering those input combinations that produce a logic _0‘ at the output. Each such input combination gives a term, and the product of all such terms gives the expression.

Different terms are obtained by taking the sum of the corresponding literals. Here, _0‘ and _1‘ respectively mean the un complemented and complemented variables, unlike sum-of-products expressions where _0‘ and _1‘ respectively mean complemented and un complemented variables.

Since each term in the case of the product-of-sums expression is going to be the sum of literals, this implies that it is going to be implemented using an OR operation. Now, an OR gate produces a logic _0‘ only when all its inputs are in the logic _0‘ state, which means that the first term corresponding to the second row of the truth table will be A+B+C. The product-of-sums Boolean expression for this truth table is given by Transforming the given product-of-sums expression into an equivalent sum-of-products expression is a straightforward process. Multiplying out the given expression and carrying out the obvious simplification provides the equivalent sum-of-products expression:A given sum-of-products expression can be transformed into an equivalent product-of-sums expression by (a) taking the dual of the given expression, (b) multiplying out different terms to get the sum-of products form, (c) removing redundancy and (d) taking a dual to get the equivalent product-of-sums expression. As an illustration, let us find the equivalent product-of-sums expression of the sum-of products expression

$A.B+\overline{A}.\overline{B}$

The dual of the given expression = $(A+B).(\overline{A}+\overline{B})$:

$(A+B).(\overline{A}+\overline{B}) = A.\overline{A}+A.\overline{B}+B.\overline{A}+B.\overline{B} = 0+A.\overline{B}+B.\overline{A}+0 = A.\overline{B}+\overline{A}.B$

The dual of $(A.\overline{B}+\overline{A}.B) = (A+\overline{B}).(\overline{A}+B)$. Therefore

$$A.B+\overline{A}.\overline{B} = (A+\overline{B}).(\overline{A}+B)$$

1.6 Minimization Technique

The primary objective of all simplification procedures is to obtain an expression that has the minimum number of terms. Obtaining an expression with the minimum number of literals is usually the secondary objective. If there is more than one possible solution with the same number of terms, the one having the minimum number of literals is the choice.

There are several methods for simplification of Boolean logic expressions. The process is usually called logic minimization‖ and the goal is to form a result which is efficient. Two methods we will discuss are algebraic minimization and Karnaugh maps. For very complicated problems the former method can be done using special software analysis programs. Karnaugh maps are also limited to problems with up to 4 binary inputs. The Quine–McCluskey tabular method is used for more than 4 binary inputs.

1.6.1 Karnaugh Map Method

Maurice Karnaugh, a telecommunications engineer, developed the Karnaugh map at Bell Labs in 1953 while designing digital logic based telephone switching circuits.Karnaugh maps reduce logic functions more quickly and easily compared to Boolean algebra. By reduce we mean simplify, reducing the number of gates and inputs. We like to simplify logic to a lowest cost form to save costs by elimination of components. We define lowest cost as being the lowest number of gates with the lowest number of inputs per gate. A Karnaugh map is a graphical representation of the logic system. It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions. Drawing a Karnaugh map from the truth table involves an additional step of writing the minterm or maxterm expression depending upon whether it is desired to have a minimized sum-of-products or a minimized product of-sums expression

Construction of a Karnaugh Map

An n-variable Karnaugh map has 2n squares, and each possible input is allotted a square. In the case of a minterm Karnaugh map, _1‗ is placed in all those squares for which the output is _1‗, and _0‗ is placed in all those squares for which the output is _0‗. 0s are omitted for simplicity. An _X‗ is placed in squares corresponding to _don‗t care‗ conditions. In the case of a maxterm Karnaugh map, a _1‗ is placed in all those squares for which the output is _0‗, and a _0‗ is placed for input entries corresponding to a _1‗ output. Again, 0s are omitted for simplicity, and an _X‗ is placed in squares corresponding to _don‗t care‗ conditions. The choice of terms identifying different rows and columns of a Karnaugh map is not unique for a given number of variables. The only condition to be satisfied is that the designation of adjacent rows and adjacent columns should be the same except for one of the literals being complemented. Also, the extreme rows and extreme columns are considered adjacent.Some of the possible designation styles for two-, three- and four-variable minterm Karnaugh maps are shown in the figure below.

The style of row identification need not be the same as that of column identification as long as it meets the basic requirement with respect to adjacent terms. It is, however, accepted practice to adopt a uniform style of row and column identification. Also, the style shown in the figure below is more commonly used. A similar discussion applies for maxterm Karnaugh maps. Having drawn the Karnaugh map, the next step is to form groups of 1s as per the following guidelines:

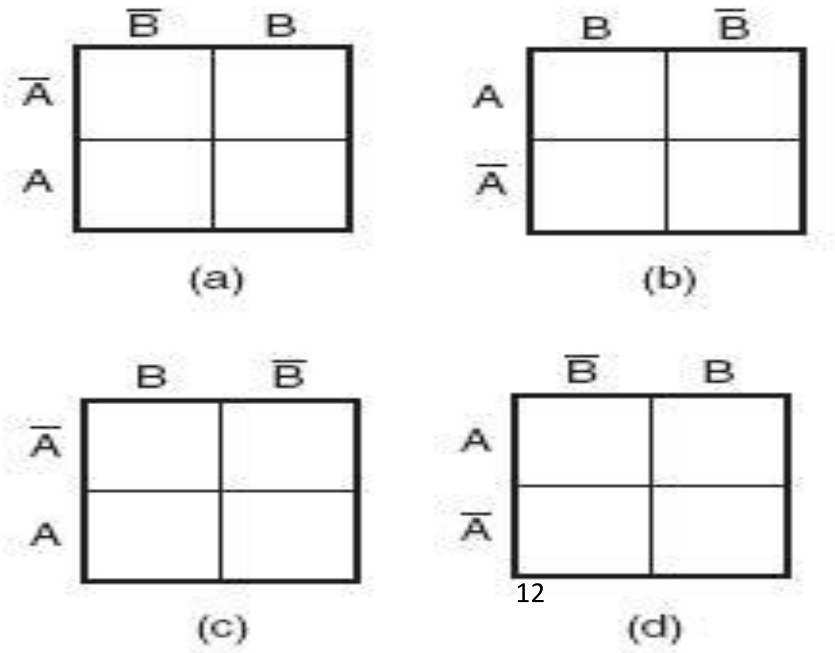
Each square containing a _1‗ must be considered at least once, although it can be considered as often as desired.

The objective should be to account for all the marked squares in the minimum number of groups.

The number of squares in a group must always be a power of 2, i.e. groups can have 1,2, 4_ 8, 16, squares.

Each group should be as large as possible, which means that a square should not be accounted for by itself if it can be accounted for by a group of two squares; a group of two squares should not be made if the involved squares can be included in a group of four squares and so on.

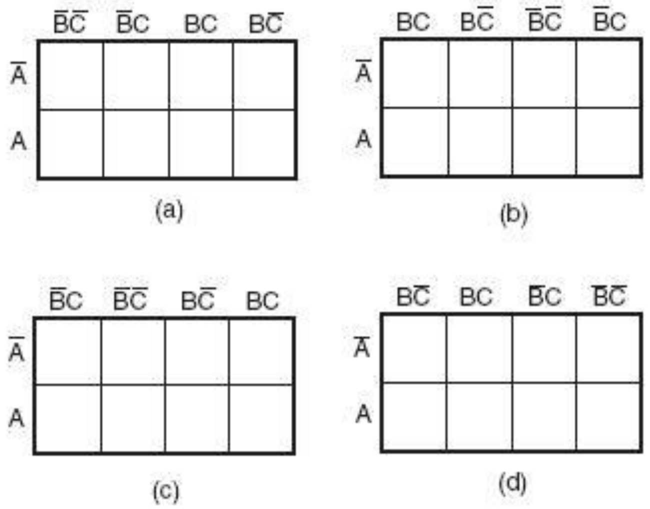
_Don‗t care‗ entries can be used in accounting for all of 1-squares to make optimum groups. They are marked _X‗ in the corresponding squares. It is, however, not necessary to account for all _don‗t care‗ entries. Only such entries that can be used to advantage should be used.



(a) (b)

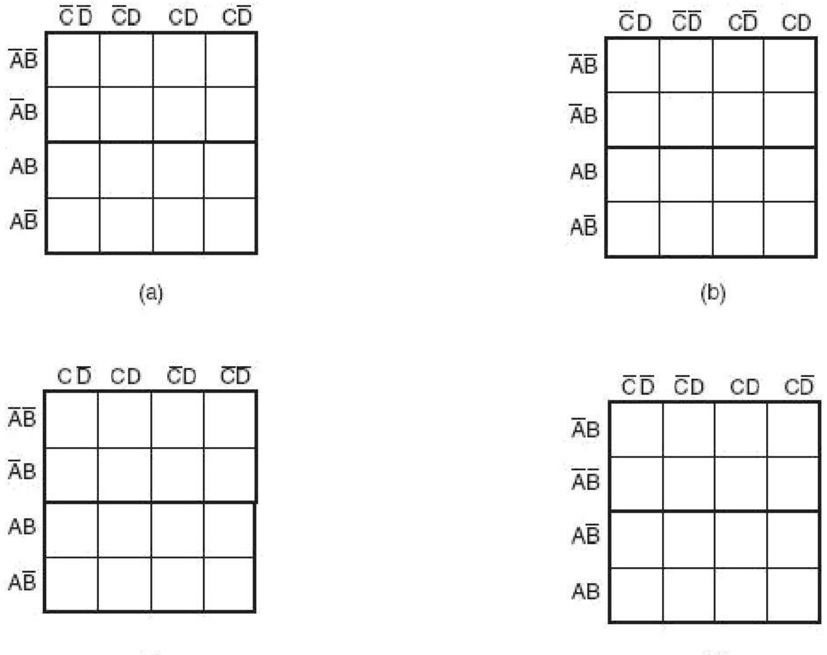
(c) (d)

Two variable K Map



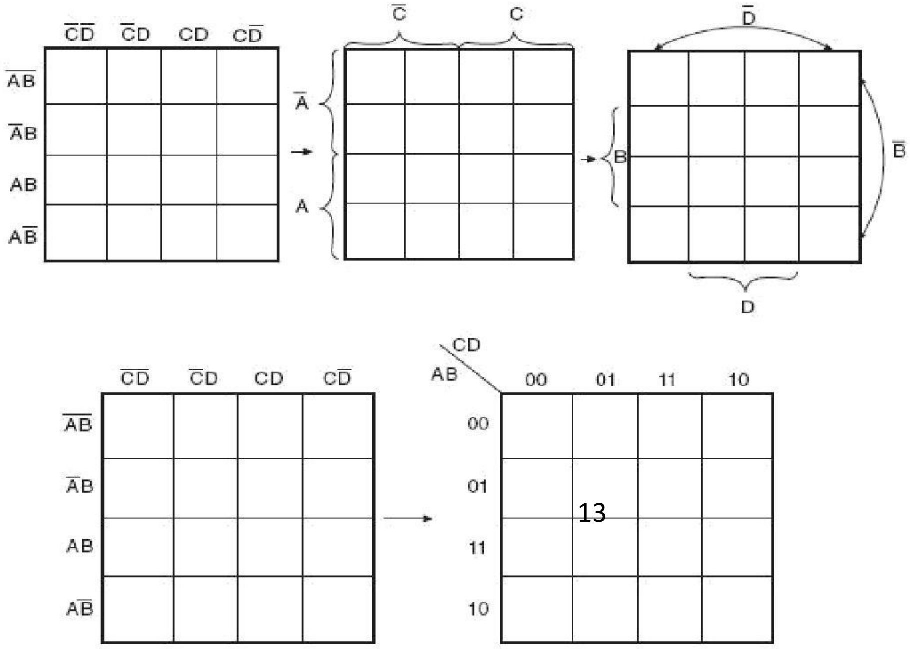
$\overline{B}\overline{C}$ $\overline{B}C$ BC $B\overline{C}$
$\overline{A}$
A
(a)
BC $B\overline{C}$ $\overline{B}\overline{C}$ $\overline{B}C$
$\overline{A}$
A
(b)
$\overline{B}C$ $\overline{B}\overline{C}$ $B\overline{C}$ BC
$\overline{A}$
A
(c)
$B\overline{C}$ BC $\overline{B}C$ $\overline{B}\overline{C}$
$\overline{A}$
A
(d)

Three variable K Map



$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
(a)
$\overline{C}D$ $\overline{C}\overline{D}$ $C\overline{D}$ CD
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
(b)
$C\overline{D}$ CD $\overline{C}D$ $\overline{C}\overline{D}$
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
(c)
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}B$
$\overline{A}\overline{B}$
$A\overline{B}$
AB
(d)

Four variable K Map



$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
$\overline{C}$
C
$\overline{A}$
A
$\overline{D}$
B
$\overline{B}$
D
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
CD
AB
00 01 11 10
00
01
11
10

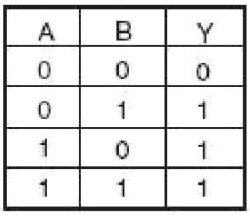
Different Styles of row and column identification

Having accounted for groups with all 1s, the minimum ‗sum-of-products‘ or ‗product-of-sums‘ expressions can be written directly from the Karnaugh map. Minterm Karnaugh map and Maxterm Karnaugh map of the Boolean function of a two-input OR gate. The Minterm and Maxterm Boolean expressions for the two-input OR gate are as follows:

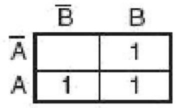
$$Y = A + B \text{ (maxterm or product-of-sums)}$$

$$Y = \overline{A}.B + A.\overline{B} + A.B \text{ (minterm or sum-of-products)}$$

Truth table

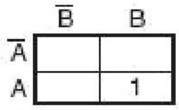


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



	$\overline{B}$	B
$\overline{A}$		1
A	1	1

Sum-of-products K-map



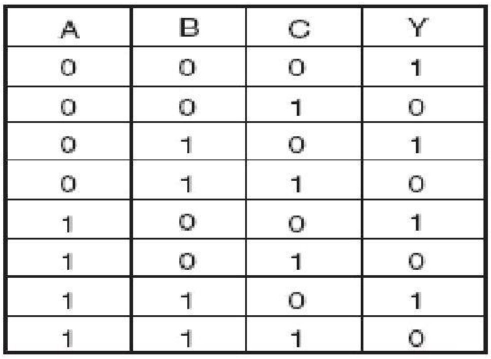
	$\overline{B}$	B
$\overline{A}$		
A		1

Product-of-sums K-map

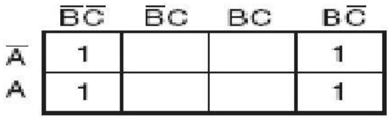
Minterm Karnaugh map and Maxterm Karnaugh map of the three variable Boolean function

$$Y = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.\overline{C}$$

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

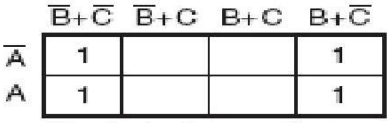


A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	1			1
A	1			1

Sum-of-products K-map



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

Product-of-sums K-map

The truth table, Minterm Karnaugh map and Maxterm Karnaugh map of the four

variable Boolean function

$$Y = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.D + A.B.\overline{C}.\overline{D} + A.B.\overline{C}.D$$

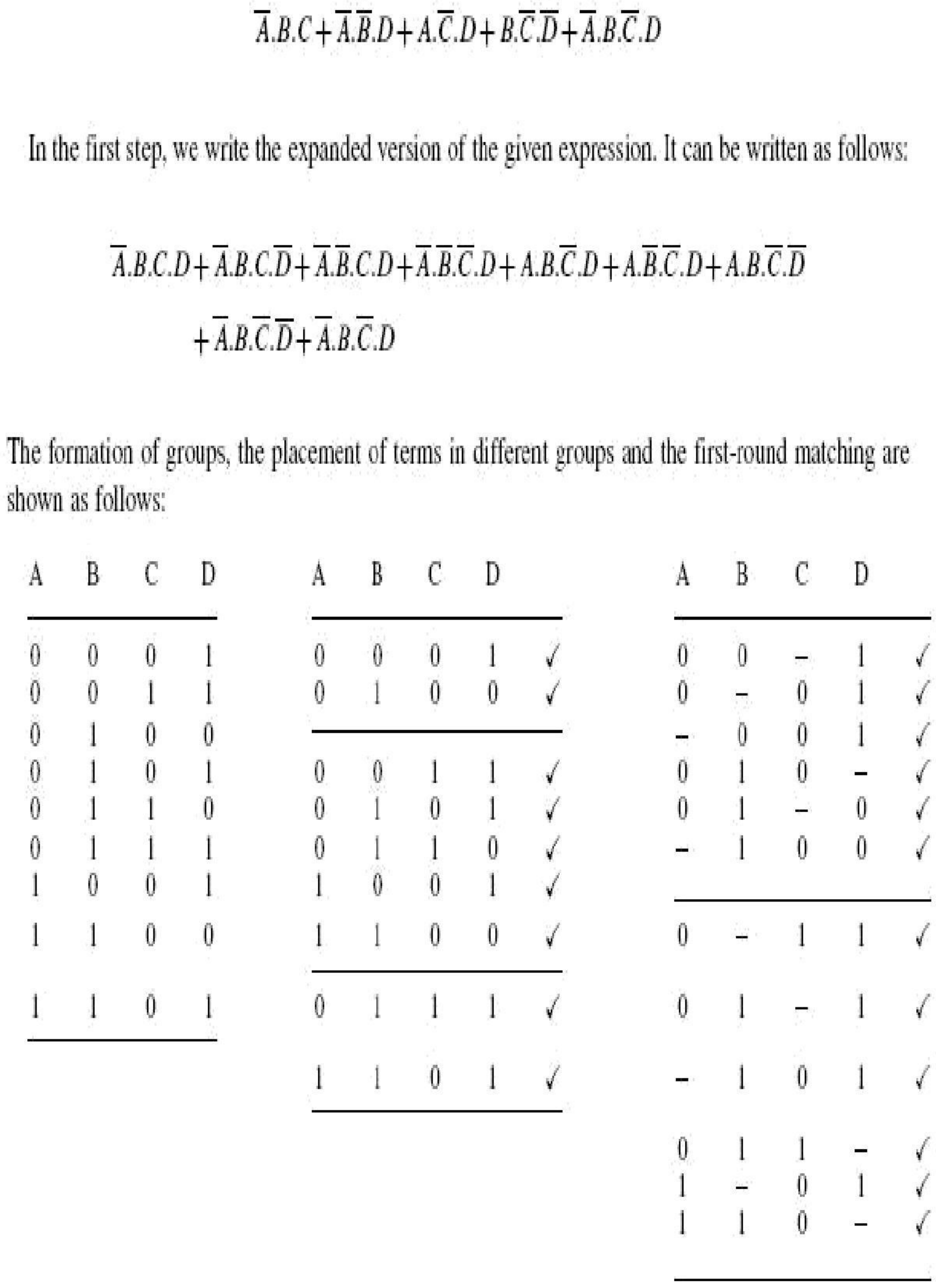
$$Y = (A + B + \overline{C} + D).(A + B + \overline{C} + \overline{D}).(A + \overline{B} + \overline{C} + D).(A + \overline{B} + \overline{C} + \overline{D})$$

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

To illustrate the process of forming groups and then writing the corresponding minimized Boolean expression, The below figures respectively show minterm and maxterm Karnaugh maps for the Boolean functions expressed by the below equations. The minimized expressions as deduced from Karnaugh maps in the two cases are given by Equation in the case of the minterm Karnaugh map and Equation in the case of the maxterm Karnaugh map:

1.6.2 Quine–McCluskey Tabular Method

The Quine–McCluskey tabular method of simplification is based on the complementation theorem, which says that XY +XY' =X. where X represents either a variable or a term or an expression and Y is a variable. This theorem implies that, if a Boolean expression contains two terms that differ only in one variable, then they can be combined together and replaced with a term that is smaller by one literal.Let us consider an example. Consider the following sum-of-products expression:



$$\overline{A}.B.C+\overline{A}.\overline{B}.D+A.\overline{C}.D+B.\overline{C}.\overline{D}+\overline{A}.B.\overline{C}.D$$

In the first step, we write the expanded version of the given expression. It can be written as follows:

$$\overline{A}.B.C.D+\overline{A}.B.C.\overline{D}+\overline{A}.\overline{B}.C.D+\overline{A}.\overline{B}.\overline{C}.D+A.B.\overline{C}.D+A.\overline{B}.\overline{C}.D+A.B.\overline{C}.\overline{D}$$
$$+\overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.\overline{C}.D$$

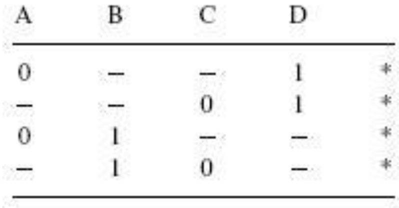
The formation of groups, the placement of terms in different groups and the first-round matching are shown as follows:

A	B	C	D
0	0	0	1
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	1
1	1	0	0
1	1	0	1

A	B	C	D	
0	0	0	1	✓
0	1	0	0	✓
0	0	1	1	✓
0	1	0	1	✓
0	1	1	0	✓
1	0	0	1	✓
1	1	0	0	✓
0	1	1	1	✓
1	1	0	1	✓

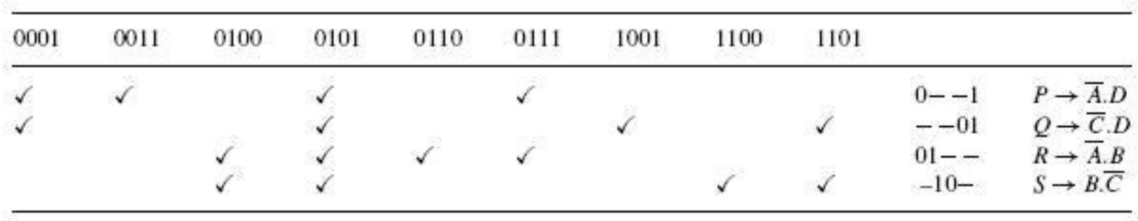
A	B	C	D	
0	0	–	1	✓
0	–	0	1	✓
–	0	0	1	✓
0	1	0	–	✓
0	1	–	0	✓
–	1	0	0	✓
0	–	1	1	✓
0	1	–	1	✓
–	1	0	1	✓
0	1	1	–	✓
1	–	0	1	✓
1	1	0	–	✓

The second round of matching begins with the table shown on the previous page. Each term in the first group is compared with every term in the second group. For instance, the first term in the first group 00−1 matches with the second term in the second group 01−1 to yield 0—1, which is recorded in the table shown below. The process continues until all terms have been compared for a possible match. Since this new table has only one group, the terms contained therein are all prime implicants. In the present example, the terms in the first and second tables have all found a match. But that is not always the case.



A	B	C	D	
0	–	–	1	*
–	–	0	1	*
0	1	–	–	*
–	1	0	–	*

The next table is what is known as the prime implicant table. The prime implicant table contains all the original terms in different columns and all the prime implicants recorded in different rows as shown below:



0001	0011	0100	0101	0110	0111	1001	1100	1101		
✓	✓		✓		✓				0– –1	$P \rightarrow \overline{A}.D$
✓			✓			✓		✓	– –01	$Q \rightarrow \overline{C}.D$
		✓	✓	✓	✓				01– –	$R \rightarrow \overline{A}.B$
		✓	✓				✓	✓	–10–	$S \rightarrow B.\overline{C}$

Each prime implicant is identified by a letter. Each prime implicant is then examined one by one and the terms it can account for are ticked as shown. The next step is to write a product-of-sums expression using the prime implicants to account for all the terms. In the present illustration, it is given as follows.

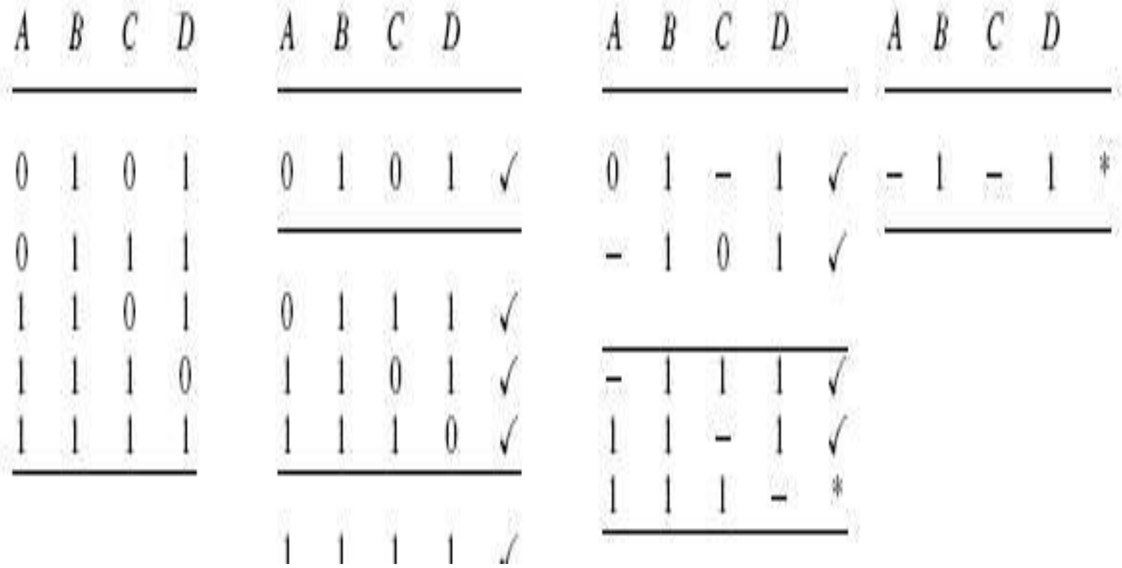
$$(P+Q).(P).(R+S).(P+Q+R+S).(R).(P+R).(Q).(S).(Q+S)$$

Obvious simplification reduces this expression to PQRS which can be interpreted to mean that all prime implicants, that is, P, Q, R and S, are needed to account for all the original terms. Therefore, the minimized expression = $\overline{A}.D+\overline{C}.D+\overline{A}.B+B.\overline{C}$.

What has been described above is the formal method of determining the optimum set of prime implicants. In most of the cases where the prime implicant table is not too complex, the exercise can be done even intuitively. The exercise begins with identification of those terms that can be accounted for by only a single prime implicant. In the present example, 0011, 0110, 1001 and 1100 are such terms. As a result, P, Q, R and S become the essential prime implicants. The next step is to find out if any terms have not been covered by the essential prime implicants. In the present case, all terms have been covered by essential prime implicants. In fact, all prime implicants are essential prime implicants in the present example. As another illustration, let us consider a product-of-sums expression given by

(A'+B'+C'+D') (A'+B'+C'+D) (A'+B'+C+D') (A+B'+C+D')

The procedure is similar to that described for the case of simplification of sum-of-products expressions. The resulting tables leading to identification of prime implicants are as follows:



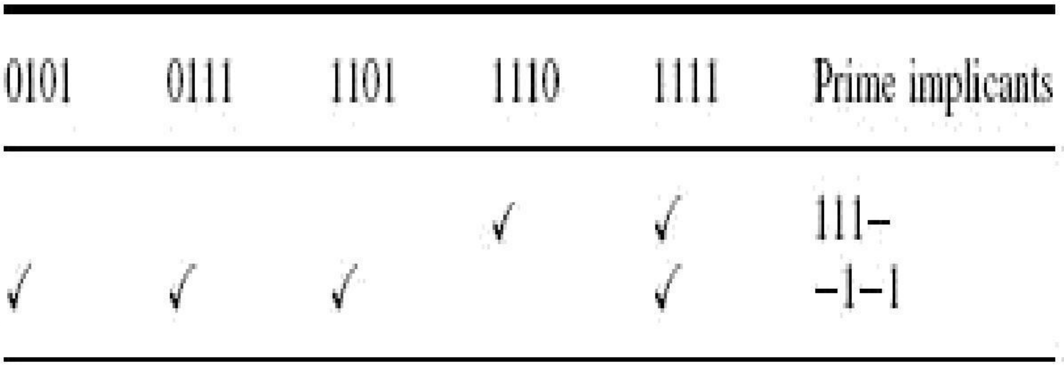
A	B	C	D
0	1	0	1
0	1	1	1
1	1	0	1
1	1	1	0
1	1	1	1

A	B	C	D	
0	1	0	1	√
0	1	1	1	√
1	1	0	1	√
1	1	1	0	√
1	1	1	1	√

A	B	C	D	
0	1	-	1	√
-	1	0	1	√
-	1	1	1	√
1	1	-	1	√
1	1	1	-	*

A	B	C	D	
-	1	-	1	*

The prime implicant table is constructed after all prime implicants have been identified to look for the optimum set of prime implicants needed to account for all the original terms. The prime implicant table shows that both the prime implicants are the essential ones:



0101	0111	1101	1110	1111	Prime implicants
			√	√	111–
√	√	√		√	–1–1

The minimized expression = $(\overline{A}+\overline{B}+\overline{C}).(\overline{B}+\overline{D})$.

Possible Part B Questions:

1. Convert $(237)_{10}$ to Hexadecimal $(?)_{16}$ and Octal $(?)_8$
2. Convert $(1011011)_{\text{binary}}$ to $(?)_{\text{gray}}$
3. State De-Morgans Law
4. Write Associative and Distributive Laws
5. List the binary codes.
6. Convert $(43)_{10}$ to $(?)_{16 \text{ and }} (?)_8$
7. Convert $(2567)_8$ to $(?)_{16 \text{ and }} (?)_2$
8. Convert $(110010)_2$ to $(?)_{16 \text{ and }} (?)_8$
9. What are canonical forms
10. Convert $(10010)_2$ to (?)gray
11. Differentiate between POS and SOP.
12. Convert $(10001)_{\text{gray}}$ to $(?)_2$
13. Give the applications of K map
14. Convert $(10010110)_{10}$ to $(?)_{16 \text{ and }} (?)_8$
15. What are don’t care terms?
16. What is commutative law?
17. Define absorption law
18. Write Consensus theorem
19. Define principle of duality
20. What are weighted codes? Give one example
21. Classify binary codes.
22. Define non-weighted codes
23. Give the advantages of numbering system
24. List out the commonly used numbering systems
25. Why hexadecimal is preferred over octal numbering system

Possible Part C Questions:

1. Explain in detail about numbering systems in detail
2. Write short notes on Binary, Octal, Decimal and Hexadecimal numbering systems
3. Describe in detail about binary codes
4. Write short notes on weighted and non-weighted codes
5. Explain numbering system conversions with an example
6. Describe Boolean postulates with examples
7. Write short notes on Alphanumeric codes and error detecting and correcting codes
8. Explain in detail about signed arithmetic with examples
9. State and prove Demorgans theorems
10. State and prove Boolean laws
11. Write short notes on Minterms, Maxterms, SOP and POS functions
12. Simplify the Boolean Function F(X,Y,Z)= XY+X’Z+YZ and implement using gates.
13. Simplify the Boolean Function F(X,Y)= XY+X’Y+X’Y’+Y’X and implement using universal gates
14. Using a K-map, find the POS form of F=∏M (0,4,8,12,3,7,11,15) and implement
15. Minimize the following Boolean function using K-Map and draw the combinational circuit F(A,B,C,D)= ∑m (0,1,4,7,11,13,14,15)+ ∑d(2,9)
16. Minimize the following Boolean function using K-Map and draw the combinational circuit F(A,B,C,D)= ∑m (0,1,2,4,6,8,10,13,14,15)

17. Minimize the following Boolean function using K-Map and draw the combinational circuit F(A,B,C,D)= ∏M (0,1,4,7,11,13,14,15)+ ∑d(2,9)
18. Find MSP for the function F(A,B,C,D)= ∑m (0,2,4,6,8,10,12,14)
19.

Karpagam Academy of Higher Education
(Established under Section 3 of UGC Act 1956)
Eachanari, Coimbatore-641 021. INDIA

Department of Electronics and Communication Engineering
Faculty of Engineering

DIGITAL ELECTRONICS

LECTURE NOTES-UNIT II

PREPARED BY
Dr.S.Bhavani,HOD/ECE

14BEEC403 DIGITAL ELECTRONICS 3 0 0 3 100

INTENDED OUTCOMES:

- To introduce number systems and codes
- To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
- To introduce the methods for simplifying Boolean expressions
- To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
- To introduce the concept of memories and programmable logic devices.

UNIT-I NUMBER SYSTEMS

Binary, Octal, Decimal, Hexadecimal-Number base conversions – complements – signed Binary numbers. Binary Arithmetic- Binary codes: Weighted –BCD-2421-Gray code-Excess 3 code-ASCII –Error detecting code – conversion from one code to another-Boolean postulates and laws –De-Morgan's Theorem- Principle of Duality- Boolean expression – Boolean function-Minimization of Boolean expressions – Sum of Products (SOP) –Product of Sums (POS)-Minterm- Maxterm- Canonical forms – Conversion between canonical forms –Karnaugh map Minimization – Don't care conditions.

UNIT-II LOGIC GATES AND COMBINATIONAL CIRCUITS LOGIC GATES

AND, OR, NOT, NAND, NOR, Exclusive – OR and Exclusive – NOR- Implementations of Logic Functions using gates, NAND –NOR implementations –Multi level gate implementations-Multi output gate implementations. TTL and CMOS Logic and their characteristics –Tristate gates.

COMBINATIONAL CIRCUITS: Design procedure – Adders-Subtractors – Serial adder/ Subtractor - Parallel adder/ Subtractor- Carry look ahead adder- BCD adder- Magnitude Comparator- Multiplexer/ Demultiplexer- encoder / decoder – parity checker – code converters. Implementation of combinational logic using MUX, ROM, PAL and PLA.

UNIT-III SEQUENTIAL CIRCUIT

Flip flops SR, JK, T, D and Master slave – Characteristic table and equation –Application table – Edge triggering –Level Triggering –Realization of one flip flop using other flip flops – Asynchronous / Ripple counters – Synchronous counters –Modulo – n counter –Classification of sequential circuits – Moore and Mealy -Design of Synchronous counters: state diagram- State table –State minimization –State assignment- ASM-Excitation table and maps-Circuit implementation - Register – shift registers- Universal shift register – Shift counters – Ring counters.

UNIT-IV ASYNCHRONOUS SEQUENTIAL CIRCUITS

Design of fundamental mode and pulse mode circuits – primitive state / flow table – Minimization of primitive state table –state assignment – Excitation table – Excitation map-cycles – Races –Hazards: Static –Dynamic –Essential –Hazards elimination.

UNIT-V MEMORY DEVICES

Classification of memories –RAM organization – Write operation –Read operation – Memory cycle - Timing wave forms – Memory decoding – memory expansion – Static RAM Cell-Bipolar RAM cell – MOSFET RAM cell –Dynamic RAM cell –ROM organization - PROM –EPROM – EEPROM –EAPROM –Programmable Logic Devices –Programmable Logic Array (PLA)-Programmable Array Logic (PAL)-Field Programmable Gate Arrays (FPGA).

TEXT BOOKS:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Morris Mano.M	Digital Design	Prentice Hall of India Pvt. Ltd., New Delhi	2003
2	John M .Yarbrough	Digital Logic Applications and Design	Thomson- Vikas publishing house, New Delhi	2002

REFERENCES:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Salivahanan.S and Arivazhagan.S	Digital Circuits and Design	Vikas Publishing House Pvt. Ltd, New Delhi	2004
2	Charles H.Roth	Fundamentals of Logic Design	Thomson Publication Company, New Delhi.	2003
3	Donald P.Leach and Albert Paul Malvino	Digital Principles and Applications	Tata McGraw Hill Publishing Company Limited, New Delhi	2003
4	Jain.R.P	Modern Digital Electronics	Tata McGraw–Hill publishing company limited, New Delhi	2003
5	Thomas L. Floyd	Digital Fundamentals	Pearson Education, New Delhi	2003

WEBSITES:

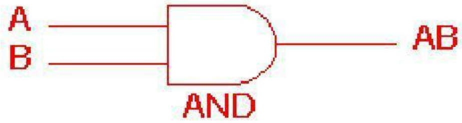
http://www.allaboutcircuits.com/vol_2/chpt_9/2.html
http://www.educypedia.be/electronics/digital.html

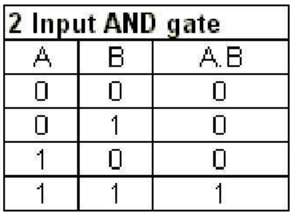
UNIT-II LOGIC GATES AND COMBINATIONAL CIRCUITS LOGIC GATES

2.Logic gates

Digital systems are constructed using logic gates. Logic gates are digital circuits which has one or more inputs and a single output and which performs arithmetic and logical operations like addition, multiplication and inversion. There are three basic gates, which are called as AND, OR and NOT gates. The derived gates are gates which are derived using basic gates. Examples of derived gates are NAND, NOR, EXOR and EXNOR gates. The basic operations of all the gates are described below along with truth tables.

AND gate

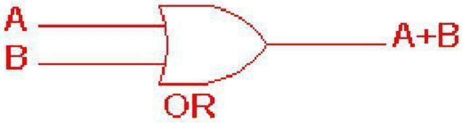


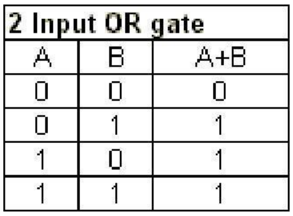


2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

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

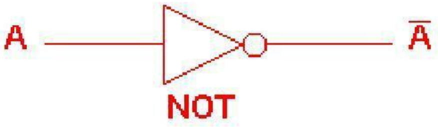


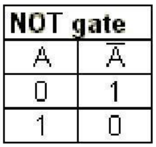


2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

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.

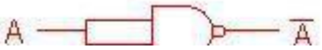
NOT gate

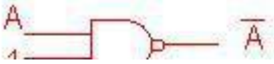




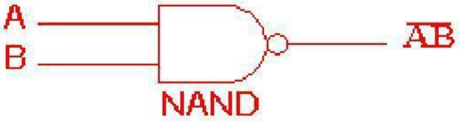
NOT gate	
A	$\overline{A}$
0	1
1	0

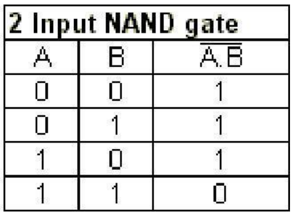
The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, then the inverted output is complement or inversion of A. This is also shown as A', or A with a bar over the top, as shown at the outputs. 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

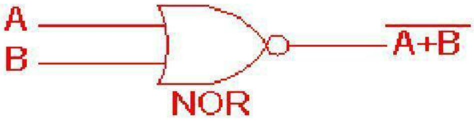




2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

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.

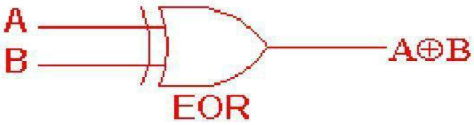
NOR gate



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.

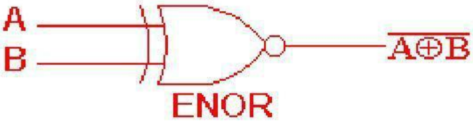
EXOR gate



2 Input EXOR gate		
A	B	$A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

The Excusive OR gate is a circuit which will give high output if both the inputs are high or low. An encircled plus sign(⊕), is used to show the EX-OR 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

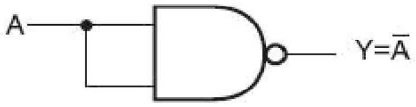
The '**Exclusive-NOR'** gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion. The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.

Note:

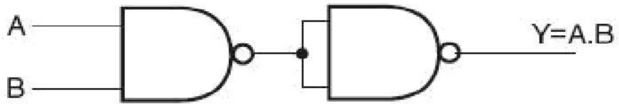
A function in *sum of products* form can be implemented using NAND gates by replacing all AND and OR gates by NAND gates.A function in *product of sums* form can be implemented using NOR gates by replacing all AND and OR gates by NOR gates.

2.1 Universal Gates

NAND, & NOR are called as Universal gates as all the other gates can be constructed using NAND and NOR

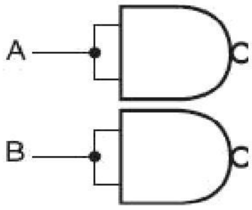


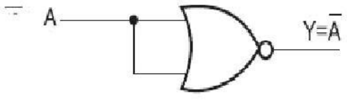
(a)



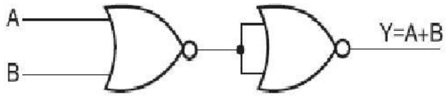
(b)

Implementation of basic logic gates using only NAND gates. a connection for open collector NAND gates. The output in this case would be



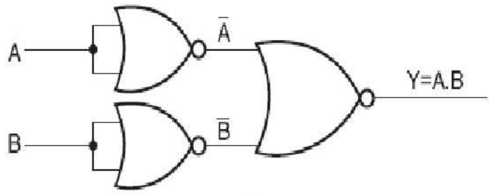


(a)



(b)

$$Y = \overline{AB}.\overline{CD}.\overline{EF}$$



(c)

2.2 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 the following steps:

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

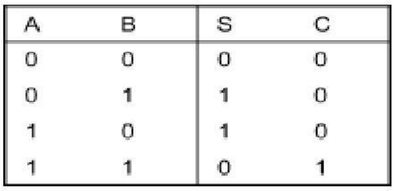
2.3 ADDERS

2.3.1 Half-Adder

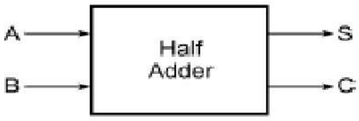
A half-adder is an arithmetic circuit block that can be used to add two bits. Such a circuit thus has two inputs that represent the two bits to be added and two outputs, with one producing the SUM output and the other producing the CARRY. Figure 3.2 shows the truth table of a half-adder, showing all possible input combinations and the corresponding outputs.

The Boolean expressions for the SUM and CARRY outputs are given by the equations below

$$\text{SUM } S = A.\overline{B}+\overline{A}.B$$
$$\text{CARRY } C = A.B$$

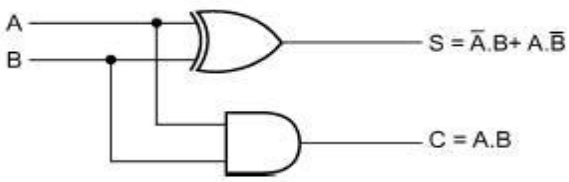


A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Truth Table of Half Adder

An examination of the two expressions tells that there is no scope for further simplification. While the first one representing the SUM output is that of an EX-OR gate, the second one representing the CARRY output is that of an AND gate. However, these two expressions can certainly be represented in different forms using various laws and theorems of Boolean algebra to illustrate the flexibility that the designer has in hardware-implementing as simple a combinational function as that of a half-adder.

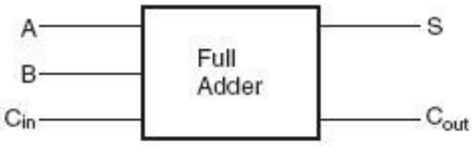


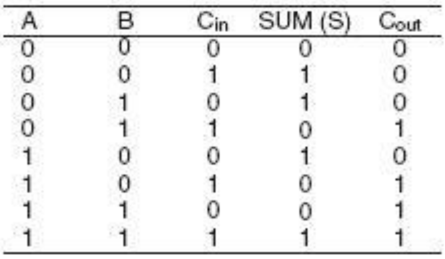
Logic Implementation of Half Adder

Although the simplest way to hardware-implement a half-adder would be to use a two-input EX-OR gate for the SUM output and a two-input AND gate for the CARRY output, as shown in Fig. 3.3, it could also be implemented by using an appropriate arrangement of either NAND or NOR gates.

2.3.2 Full Adder

A full adder circuit is an arithmetic circuit block that can be used to add three bits to produce a SUM and a CARRY output. Such a building block becomes a necessity when it comes to adding binary numbers with a large number of bits. The full adder circuit overcomes the limitation of the half-adder, which can be used to add two bits only. Let us recall the procedure for adding larger binary numbers. We begin with the addition of LSBs of the two numbers. We record the sum under the LSB column and take the carry, if any, forward to the next higher column bits. As a result, when we add the next adjacent higher column bits, we would be required to add three bits if there were a carry from the previous addition. We have a similar situation for the other higher column bits. Also until we reach the MSB. A full adder is therefore essential for the hardware implementation of an adder circuit capable of adding larger binary numbers. A half-adder can be used for addition of LSBs only.





A	B	C_{in}	SUM (S)	C_{out}
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

Truth Table of Full Adder

Figure shows the truth table of a full adder circuit showing all possible input combinations and corresponding outputs. In order to arrive at the logic circuit for hardware implementation of a full adder, we will firstly write the Boolean expressions for the two output variables, that is, the SUM and CARRY outputs, in terms of input variables. The Boolean expressions for the two output variables are given in Equation below for the SUM output (S) and in above Equation for the CARRY output (Cout):

$$S = \overline{A}.\overline{B}.C_{in} + \overline{A}.B.\overline{C}_{in} + A.\overline{B}.\overline{C}_{in} + A.B.C_{in}$$

$$C_{out} = \overline{A}.B.C_{in} + A.\overline{B}.C_{in} + A.B.\overline{C}_{in} + A.B.C_{in}$$

The next step is to simplify the two expressions. We will do so with the help of the Karnaugh mapping technique. Karnaugh maps for the two expressions are given in Fig. 3.5(a) for the SUM output and Fig. 3.5(b) for the CARRY output.

$$C_{out} = B.C_{in} + A.B + A.C_{in}$$

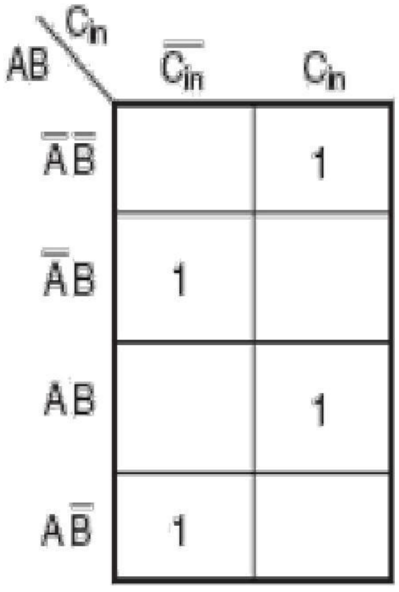
Figure shows the logic circuit diagram of the full adder. A full adder can also be seen to comprise two half-adders and an OR gate. The expressions for SUM and CARRY outputs can be rewritten as follows:

$$S = \overline{C}_{in}.(\overline{A}.B + A.\overline{B}) + C_{in}.(A.B + \overline{A}.\overline{B})$$

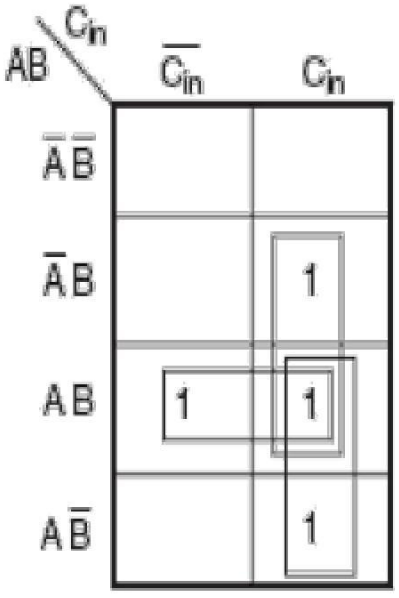
$$S = \overline{C}_{in}.(\overline{A}.B + A.\overline{B}) + C_{in}.(\overline{\overline{A}.B + A.\overline{B}})$$

Similarly, the expression for CARRY output can be rewritten as follows:

$$
\begin{aligned}
C_{out} &= B.C_{in}.(A+\overline{A})+A.B+A.C_{in}.(B+\overline{B})\\
&= A.B+A.B.C_{in}+\overline{A}.B.C_{in}+A.B.C_{in}+A.\overline{B}.C_{in}=A.B+A.B.C_{in}+\overline{A}.B.C_{in}+A.\overline{B}.C_{in}\\
&= A.B.(1+C_{in})+C_{in}.(\overline{A}.B+A.\overline{B})
\end{aligned}
$$



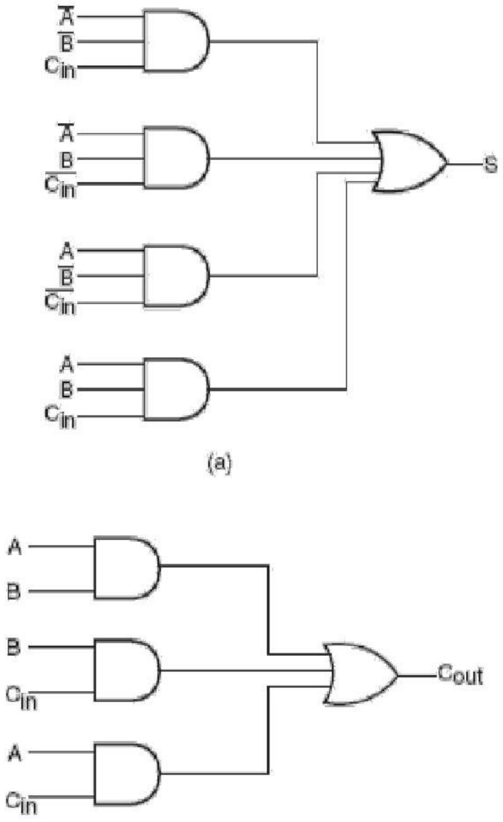
(a)



(b)

Karnaugh Map for the sum and carry out of a full adder

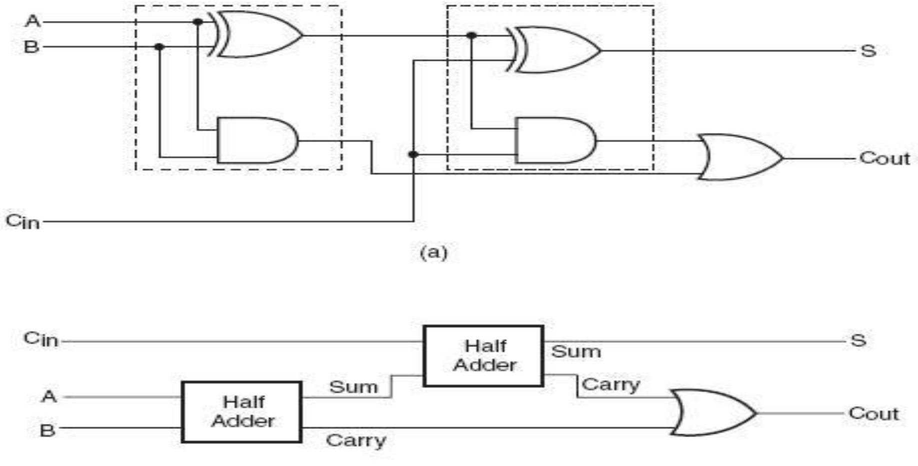
$$C_{out} = A.B + C_{in}.(\overline{A}.B + A.\overline{B})$$



(a)

(b)

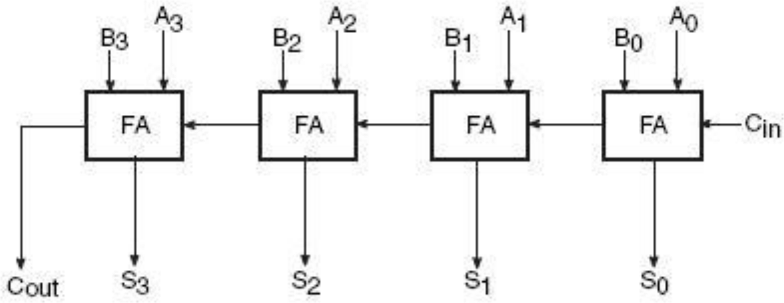
Logic circuit diagram of full adder



(a)

(b)

Logic Implementation of a full adder with Half Adders

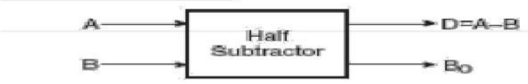


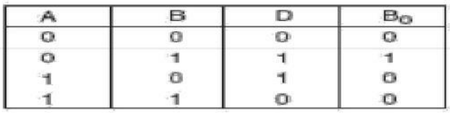
2.3.3 Half-Subtractor

We will study the use of adder circuits for subtraction operations in the following pages. Before we do that, we will briefly look at the counterparts of half-adder and full adder circuits in the half-subtractor and full subtractor for direct implementation of subtraction operations using logic gates.A half-subtractor is a combinational circuit that can be used to subtract one binary digit from another to produce a DIFFERENCE output and a BORROW output. The BORROW output here specifies whether a _1‘ has been borrowed to perform the subtraction. The truth table of a half-subtractor, as shown in Fig. 3.9, explains this further. The Boolean expressions for the two outputs are given by the equations

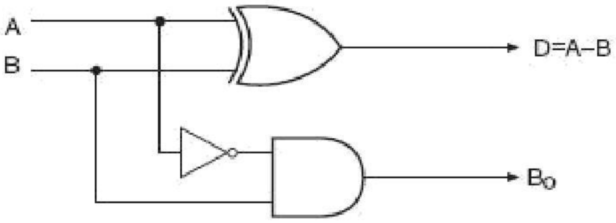
$$D = \overline{A}.B + A.\overline{B}$$

$$B_o = \overline{A}.B$$





A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



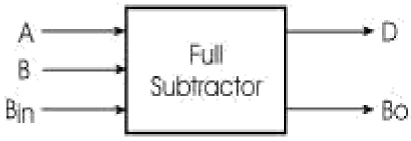
Logic Diagram of a Half Subtractor

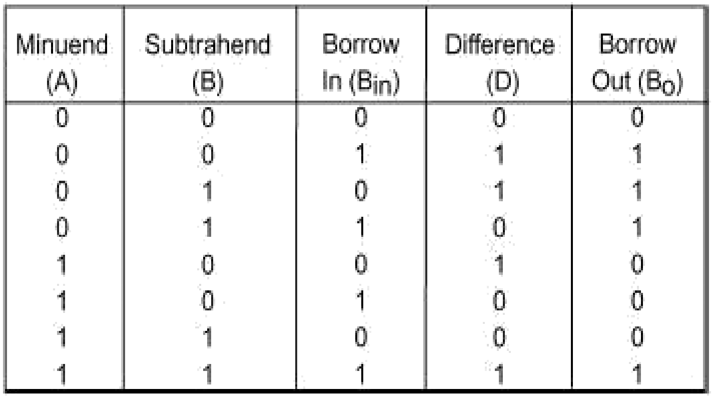
2.3.4 Full Subtractor

A full subtractor performs subtraction operation on two bits, a minuend and a subtrahend, and also takes into consideration whether a _1‘ has already been borrowed by the previous adjacent lower minuend bit or not. As a result, there are three bits to be handled at the input of a full subtractor, namely the two bits to be subtracted and a borrow bit designated as Bin . There are two outputs, namely the DIFFERENCE output D and the BORROW output Bo The Boolean expressions for the two output variables are given by the equations

$$D = \overline{A}.\overline{B}.B_{in} + \overline{A}.B.\overline{B}_{in} + A.\overline{B}.\overline{B}_{in} + A.B.B_{in}$$

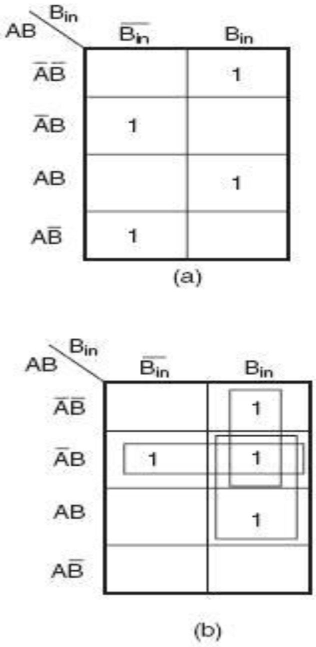
$$B_o = \overline{A}.\overline{B}.B_{in} + \overline{A}.B.\overline{B}_{in} + \overline{A}.B.B_{in} + A.B.B_{in}$$





Minuend (A)	Subtrahend (B)	Borrow In (B_{in})	Difference (D)	Borrow Out (B_o)
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

Truth Table of Full Subtractor



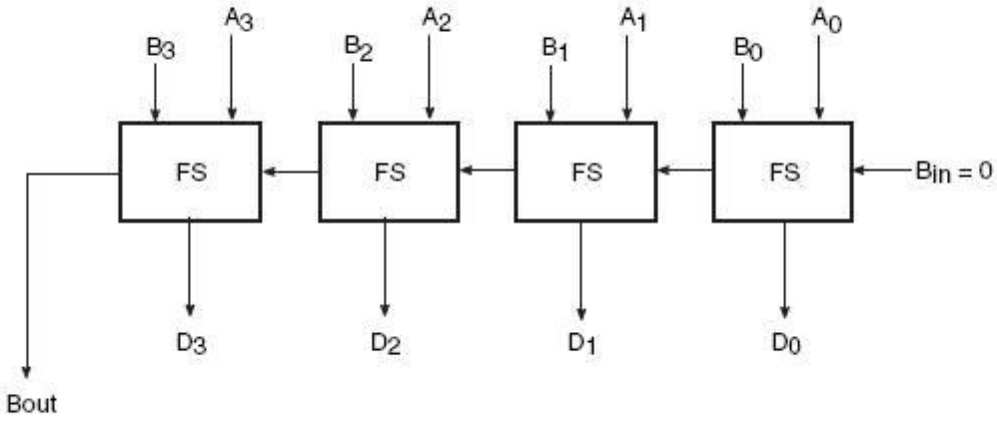
AB \ B_{in}	$\overline{B_{in}}$	B_{in}
$\overline{A}\,\overline{B}$		1
$\overline{A}B$	1	
AB		1
$A\overline{B}$	1	

(a)

AB \ B_{in}	$\overline{B_{in}}$	B_{in}
$\overline{A}\,\overline{B}$		1
$\overline{A}B$	1	1
AB		1
$A\overline{B}$		

(b)

K Maps for Difference and Borrow outputs.



2.3.5 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 ⊕ x = x" and 0 ⊕ x = x

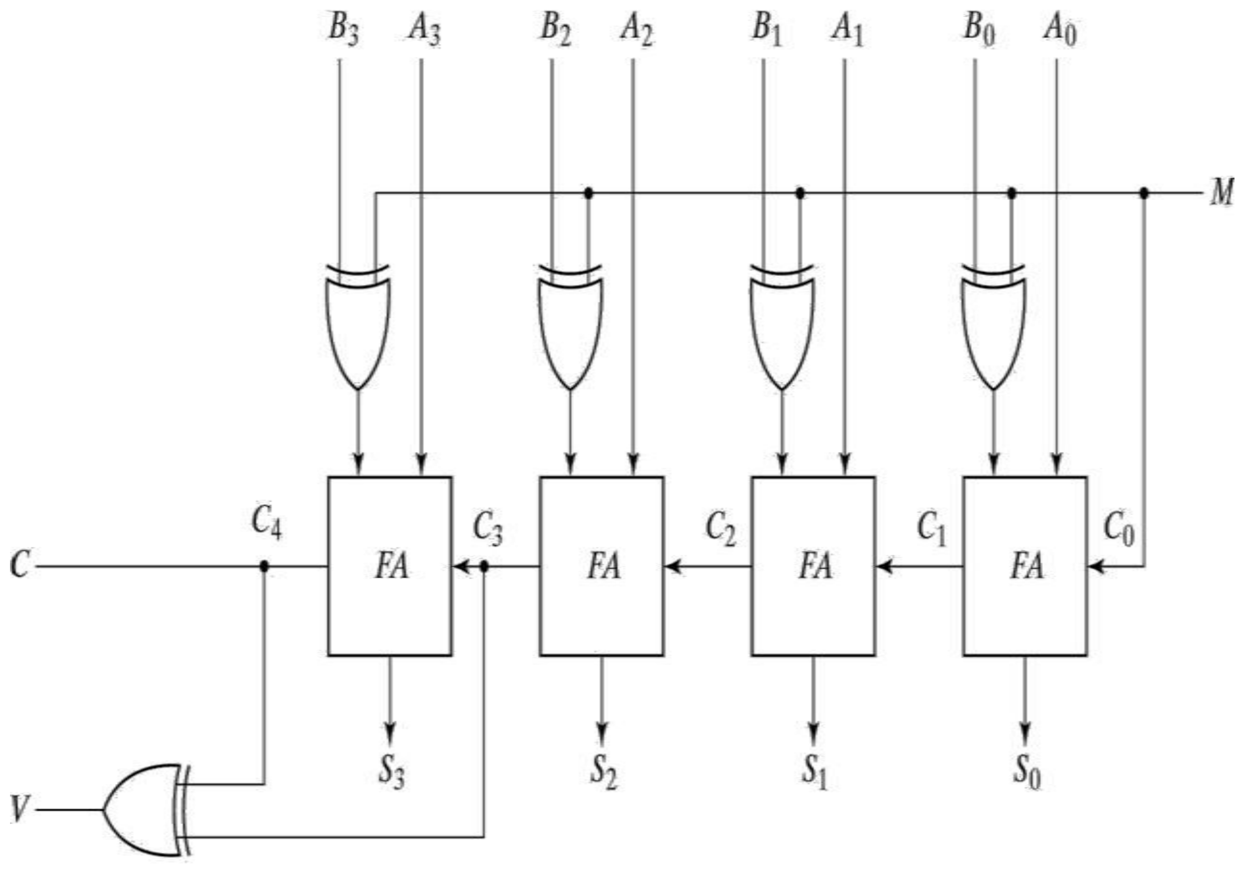


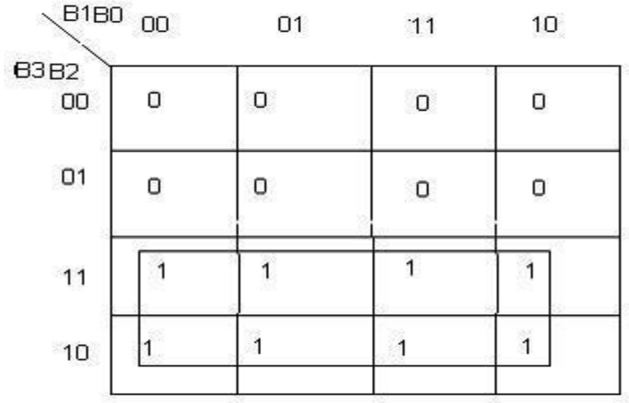
Fig. 4-13 4-Bit Adder Subtractor

2.4 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

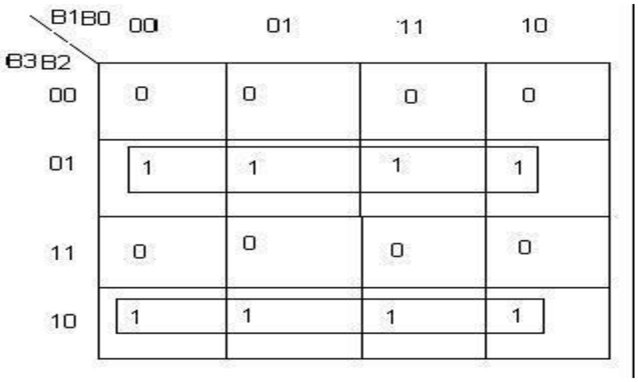
K-MAP FOR G3:



B3B2 \ B1B0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Equation for G3= B3

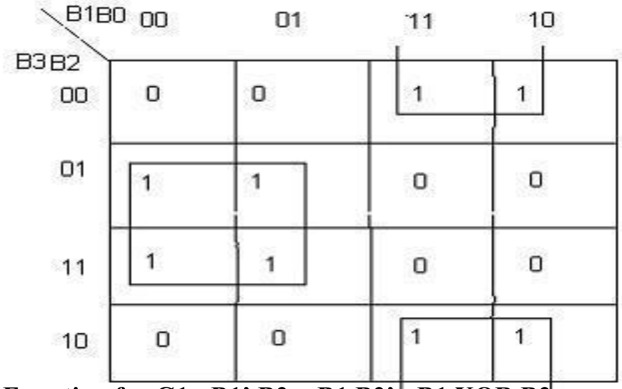
K-MAP FOR G2:



B3B2 \ B1B0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

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

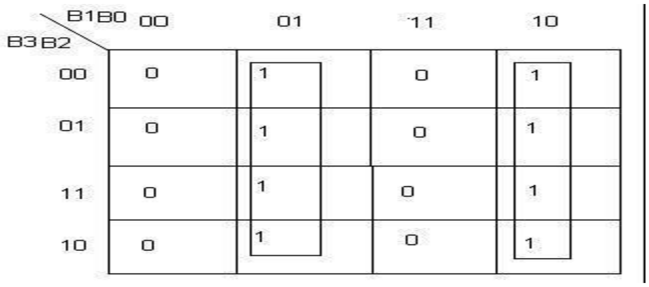
K-MAP FOR G1:



B3B2 \ B1B0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

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

K-MAP FOR G0:



B3B2 \ B1B0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

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

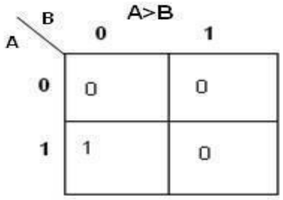
2.5 COMPARATORS

- It is a combinational circuit that compares to numbers and determines their relative magnitude .
- 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.f 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

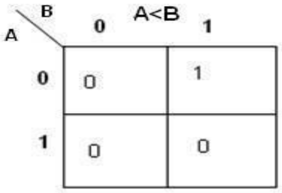
And now we find the equations using K-maps each for f (A>B), f (A=B) and f (A<B) as follow:



A>B

A \ B	0	1
0	0	0
1	1	0

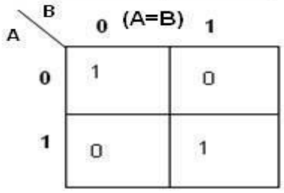
Equation is A>B = A.$\overline{B}$



A<B

A \ B	0	1
0	0	1
1	0	0

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



(A=B)

A \ B	0	1
0	1	0
1	0	1

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

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

2.6 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

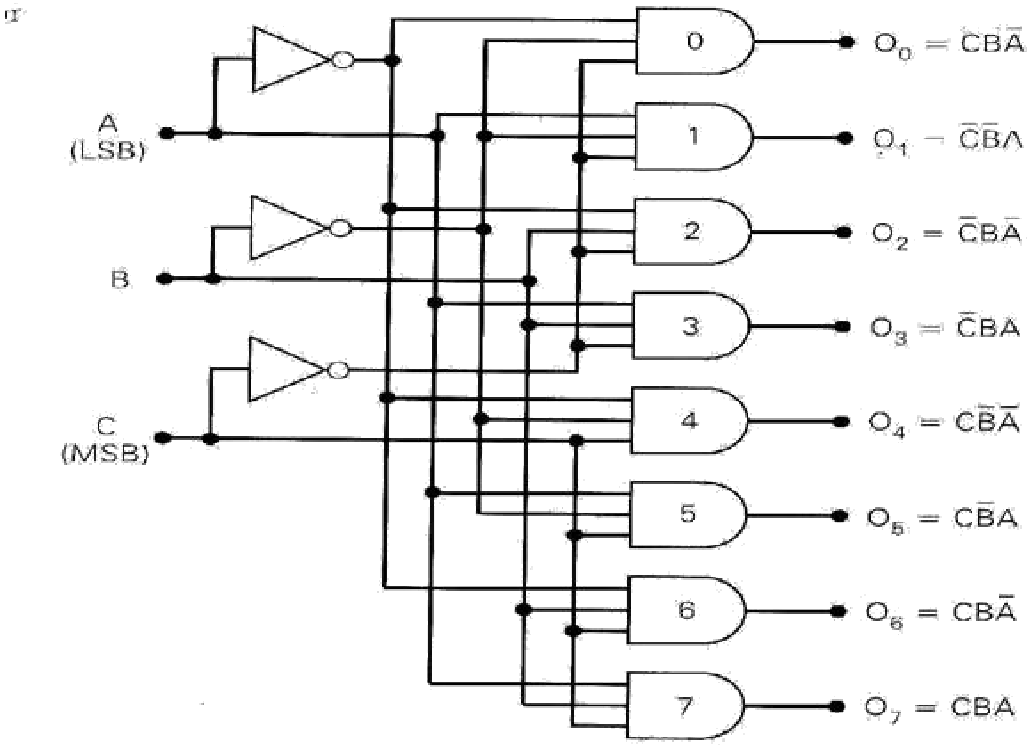
2.7 DECODERS

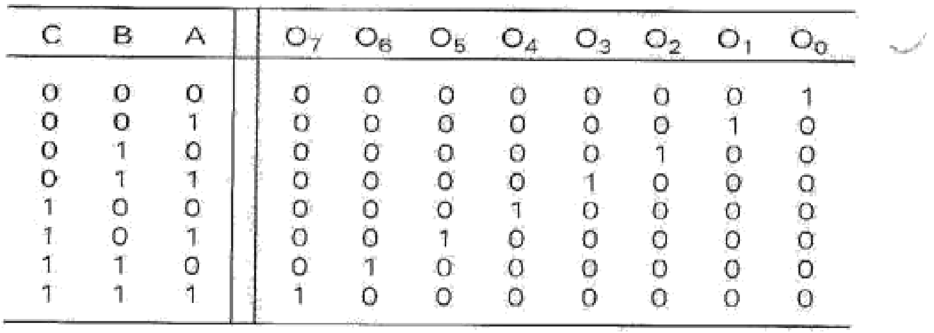
Accepts a value and decodes it

Output corresponds to value of *n* inputs

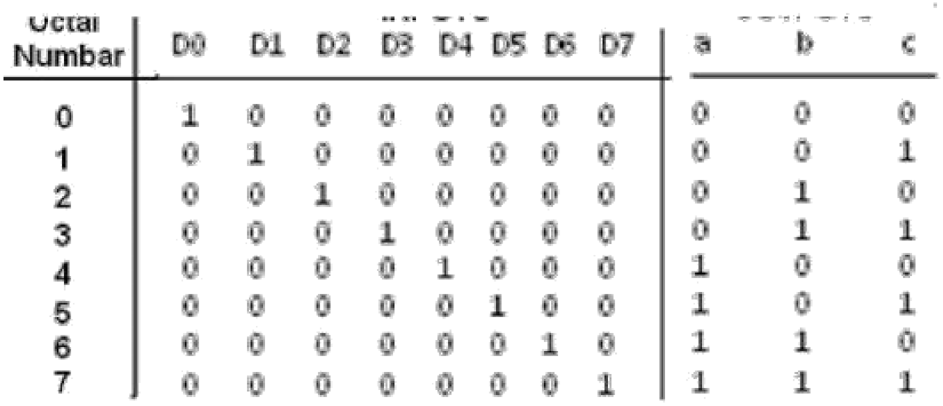
It Consists of:

Inputs (n)
Outputs (2n , numbered from 0 to 2n - 1)
Selectors / Enable (active high or active low)

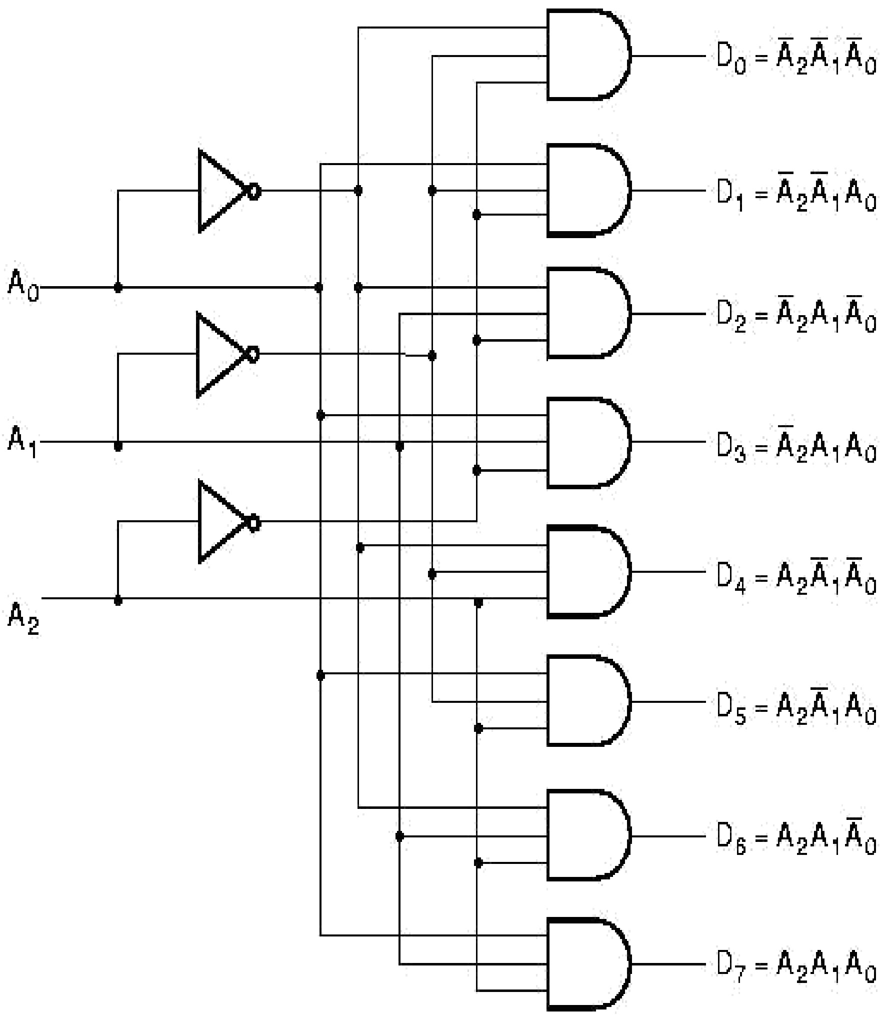




C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

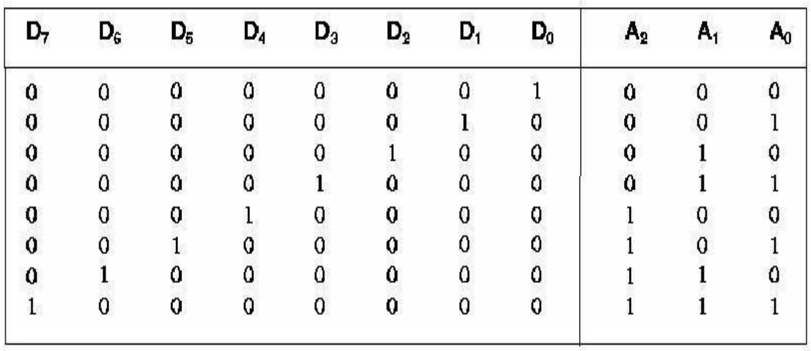


Octal Numbar	D0	D1	D2	D3	D4	D5	D6	D7	a	b	c
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1


A_0
A_1
A_2
$D_0 = \overline{A}_2\overline{A}_1\overline{A}_0$
$D_1 = \overline{A}_2\overline{A}_1A_0$
$D_2 = \overline{A}_2A_1\overline{A}_0$
$D_3 = \overline{A}_2A_1A_0$
$D_4 = A_2\overline{A}_1\overline{A}_0$
$D_5 = A_2\overline{A}_1A_0$
$D_6 = A_2A_1\overline{A}_0$
$D_7 = A_2A_1A_0$

2.8 ENCODER

Perform the inverse operation of a decoder.
It has 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

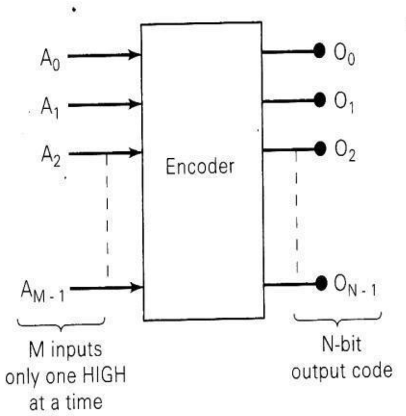
Can be implemented with 3 OR gates A0 =
D1 + D3 + D5 + D7;

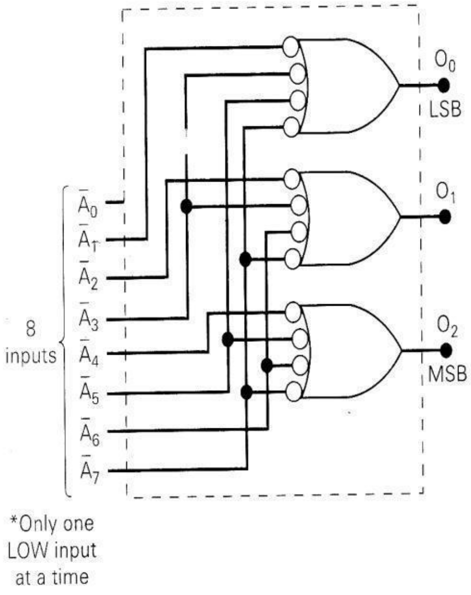
A1 = D2 + D3 + D6 + D7;

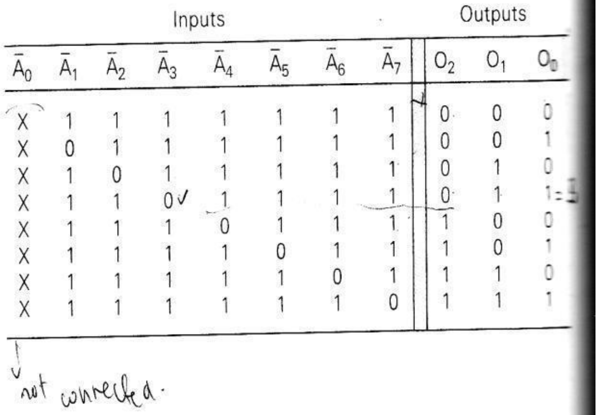
A2 = D4 + D5 + D6 + D7;

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

FIGURE 9-12 General encoder diagram.





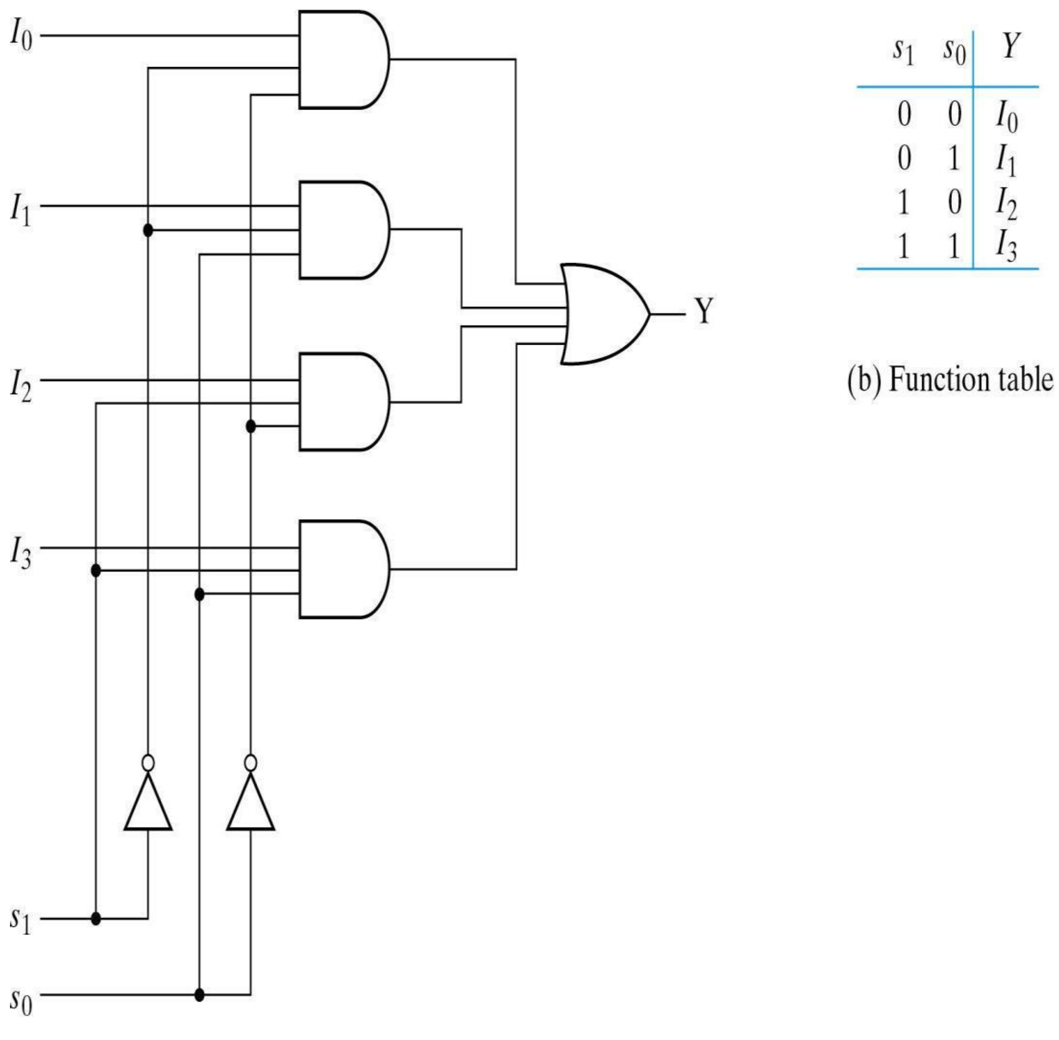


Inputs								Outputs		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0 ✓	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1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

not connected.

2.9 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



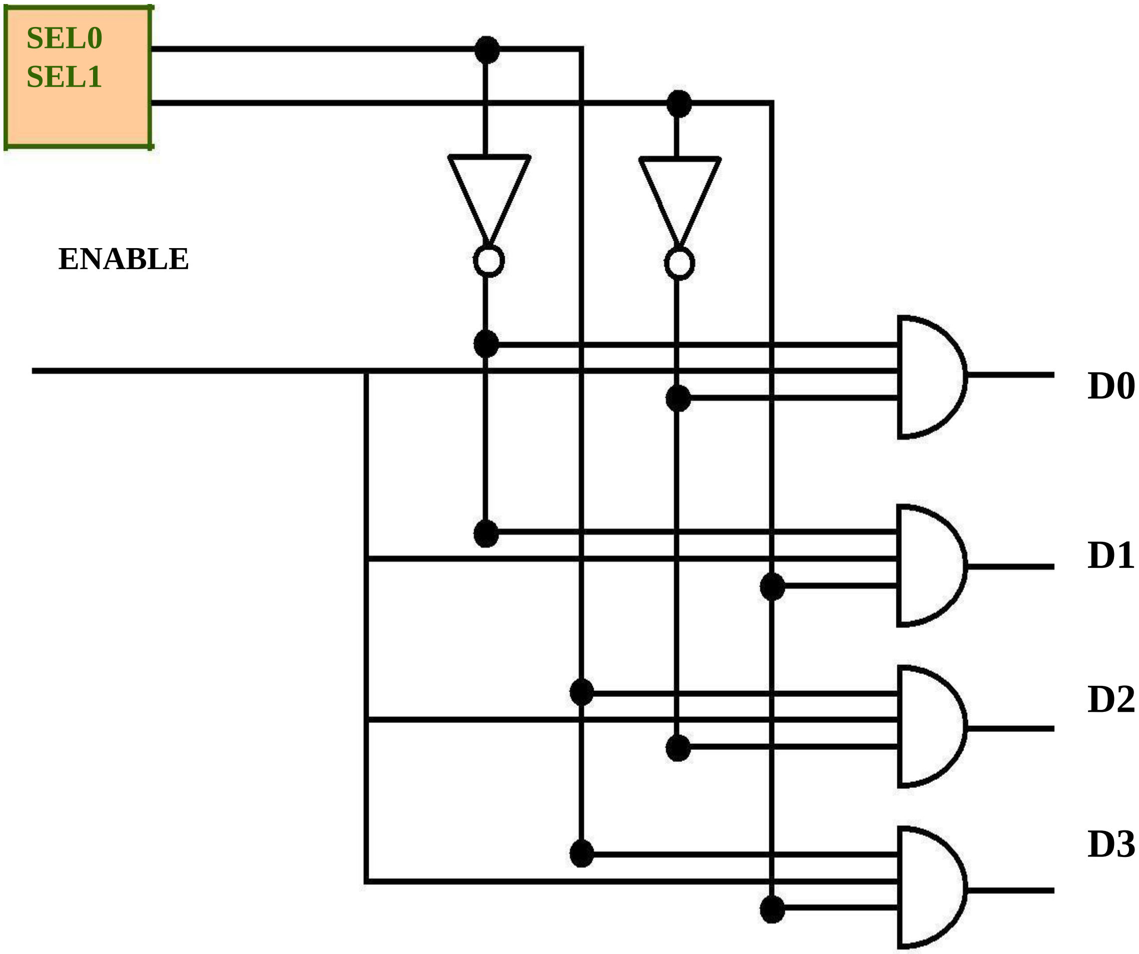
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

(a) Logic diagram

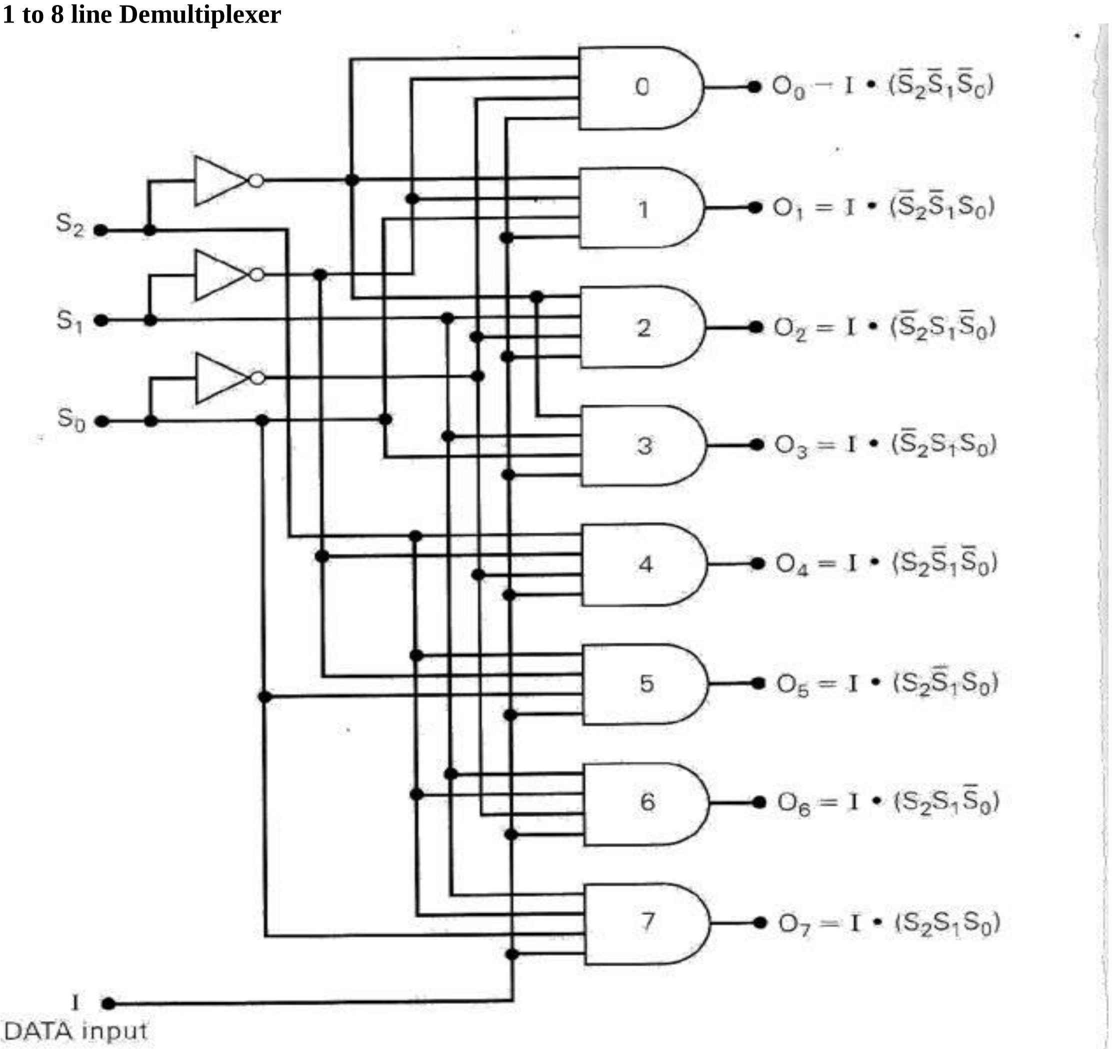
Fig. 4-25 4-to-1-Line Multiplexer

2.10 DEMULTIPLEXER



ENABLE	SEL1	SEL2	D0	D1	D2	D3
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


S_2
S_1
S_0
0
1
2
3
4
5
6
7
$O_0 = I \cdot (\bar{S}_2\bar{S}_1\bar{S}_0)$
$O_1 = I \cdot (\bar{S}_2\bar{S}_1S_0)$
$O_2 = I \cdot (\bar{S}_2S_1\bar{S}_0)$
$O_3 = I \cdot (\bar{S}_2S_1S_0)$
$O_4 = I \cdot (S_2\bar{S}_1\bar{S}_0)$
$O_5 = I \cdot (S_2\bar{S}_1S_0)$
$O_6 = I \cdot (S_2S_1\bar{S}_0)$
$O_7 = I \cdot (S_2S_1S_0)$
I
DATA input

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

Karpagam Academy of Higher Education

(Established under Section 3 of UGC Act 1956)

Eachanari, Coimbatore-641 021. INDIA

Department of Electronics and Communication Engineering
Faculty of Engineering

DIGITAL ELECTRONICS

LECTURE NOTES

PREPARED BY
Dr.S.Bhavani,HOD/ECE

14BEEC403 DIGITAL ELECTRONICS 3 0 0 3 100

INTENDED OUTCOMES:

- To introduce number systems and codes
- To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
- To introduce the methods for simplifying Boolean expressions
- To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
- To introduce the concept of memories and programmable logic devices.

UNIT-I NUMBER SYSTEMS

Binary, Octal, Decimal, Hexadecimal-Number base conversions – complements – signed Binary numbers. Binary Arithmetic- Binary codes: Weighted –BCD-2421-Gray code-Excess 3 code-ASCII –Error detecting code – conversion from one code to another-Boolean postulates and laws –De-Morgan's Theorem- Principle of Duality- Boolean expression – Boolean function-Minimization of Boolean expressions – Sum of Products (SOP) –Product of Sums (POS)-
Minterm- Maxterm- Canonical forms – Conversion between canonical forms –Karnaugh map Minimization – Don't care conditions.

UNIT-II LOGIC GATES AND COMBINATIONAL CIRCUITS LOGIC GATES

AND, OR, NOT, NAND, NOR, Exclusive – OR and Exclusive – NOR- Implementations of Logic Functions using gates, NAND –NOR implementations –Multi level gate implementations-Multi output gate implementations. TTL and CMOS Logic and their characteristics –Tristate gates.

COMBINATIONAL CIRCUITS: Design procedure – Adders-Subtractors – Serial adder/ Subtractor - Parallel adder/ Subtractor- Carry look ahead adder- BCD adder- Magnitude Comparator- Multiplexer/ Demultiplexer- encoder / decoder – parity checker – code converters. Implementation of combinational logic using MUX, ROM, PAL and PLA.

UNIT-III SEQUENTIAL CIRCUIT

Flip flops SR, JK, T, D and Master slave – Characteristic table and equation –Application table – Edge triggering –Level Triggering –Realization of one flip flop using other flip flops – Asynchronous / Ripple counters – Synchronous counters –Modulo – n counter –Classification of sequential circuits – Moore and Mealy -Design of Synchronous counters: state diagram- State table –State minimization –State assignment- ASM-Excitation table and maps-Circuit implementation - Register – shift registers- Universal shift register – Shift counters – Ring counters.

UNIT-IV ASYNCHRONOUS SEQUENTIAL CIRCUITS

Design of fundamental mode and pulse mode circuits – primitive state / flow table – Minimization of primitive state table –state assignment – Excitation table – Excitation map-cycles – Races –Hazards: Static –Dynamic –Essential –Hazards elimination.

UNIT-V MEMORY DEVICES

Classification of memories –RAM organization – Write operation –Read operation – Memory cycle - Timing wave forms – Memory decoding – memory expansion – Static RAM Cell-Bipolar RAM cell – MOSFET RAM cell –Dynamic RAM cell –ROM organization - PROM –EPROM – EEPROM –EAPROM –Programmable Logic Devices –Programmable Logic Array (PLA)-Programmable Array Logic (PAL)-Field Programmable Gate Arrays (FPGA).

TEXT BOOKS:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Morris Mano.M	Digital Design	Prentice Hall of India Pvt. Ltd., New Delhi	2003
2	John M .Yarbrough	Digital Logic Applications and Design	Thomson- Vikas publishing house, New Delhi	2002

REFERENCES:

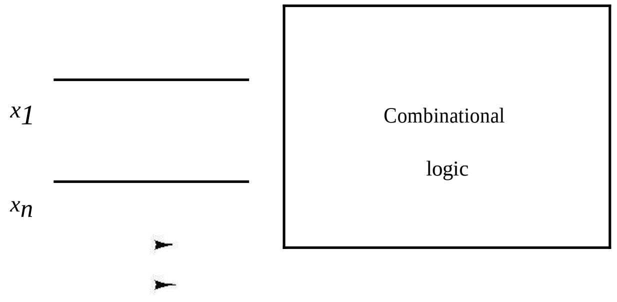
S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Salivahanan.S and Arivazhagan.S	Digital Circuits and Design	Vikas Publishing House Pvt. Ltd, New Delhi	2004
2	Charles H.Roth	Fundamentals of Logic Design	Thomson Publication Company, New Delhi.	2003
3	Donald P.Leach and Albert Paul Malvino	Digital Principles and Applications	Tata McGraw Hill Publishing Company Limited, New Delhi	2003
4	Jain.R.P	Modern Digital Electronics	Tata McGraw–Hill publishing company limited, New Delhi	2003
5	Thomas L. Floyd	Digital Fundamentals	Pearson Education, New Delhi	2003

WEBSITES:

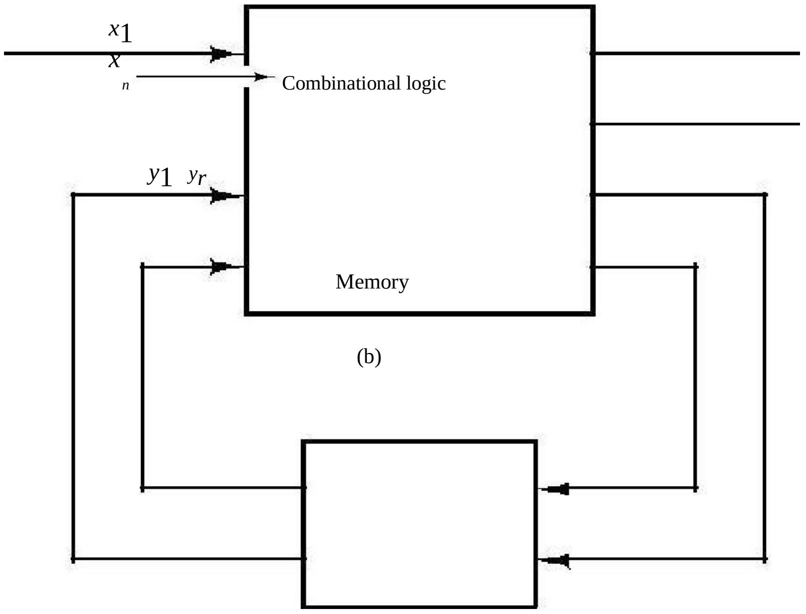
http://www.allaboutcircuits.com/vol_2/chpt_9/2.html
http://www.educypedia.be/electronics/digital.html

UNIT-III SEQUENTIAL CIRCUIT

SEQUENTIAL CIRCUITS



(a)



(b)

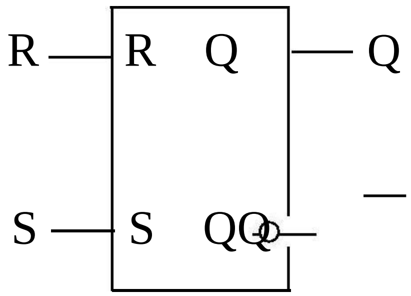
A sequential circuit is one in which the output at any instant of time depends on present input Present Input and past output. Hence it requires a memory. Flipflop is a latch with clock pulse. Clock pulse acts as a control signal. If clock is present circuit will work else it is disabled.

3.1 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

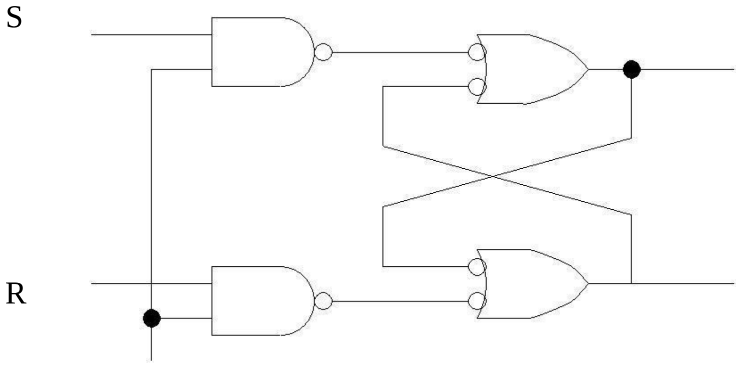


Truth Table:

S	R	Q(t)
0	0	No change
0	1	0
1	0	1
1	1	Invalid

Characteristic table

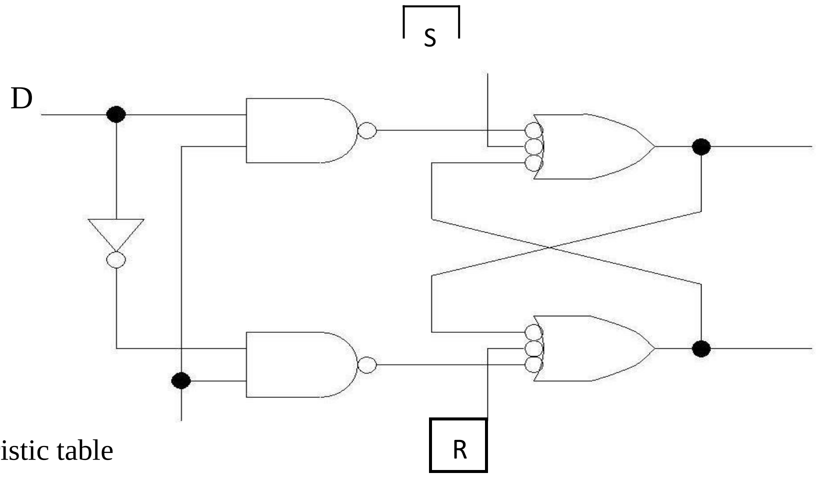
PS(Qt)	S	R	NS(Qt+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Invalid
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Invalid



Characteristic Equation

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

D Flip-Flop:



Characteristic table

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

Characteristic Equation

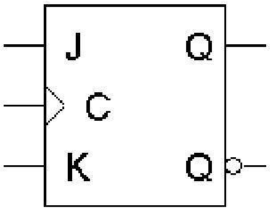
$Q(t+1) = D$

EXCITATION TABLE

Q	$\mathbf{Q_{(next)}}$	**D**
0	0	0
0	1	1
1	0	0
1	1	1

JK Flip-Flop

Graphical Symbol

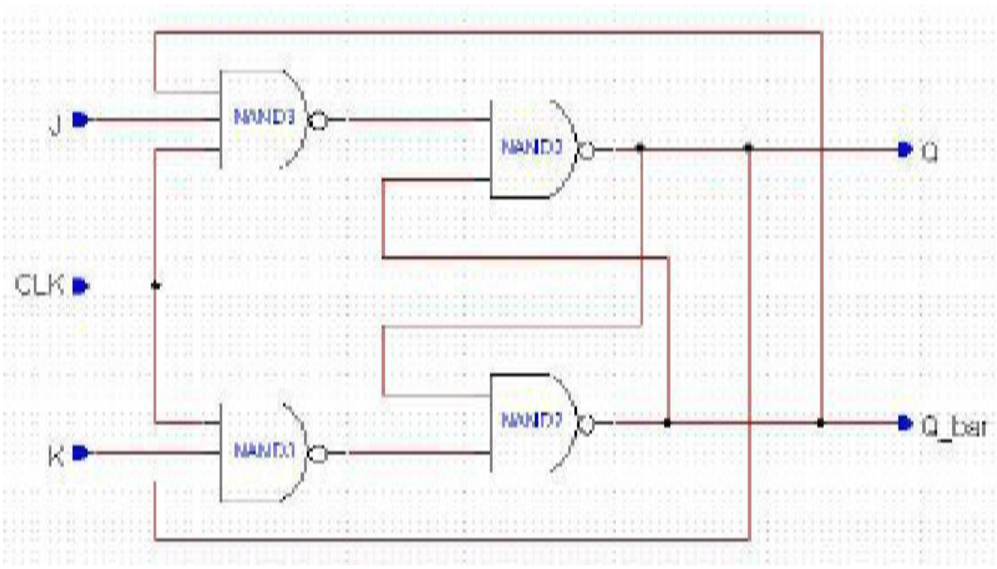


J K	Q(t+1)	Operation
0 0	Q(t)	No change
0 1	0	Reset
1 0	1	Set
1 1	Q'(t)	Complement

Characteristic Equation

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

Logic Diagram

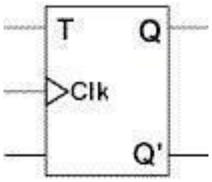


EXCITATION TABLE

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

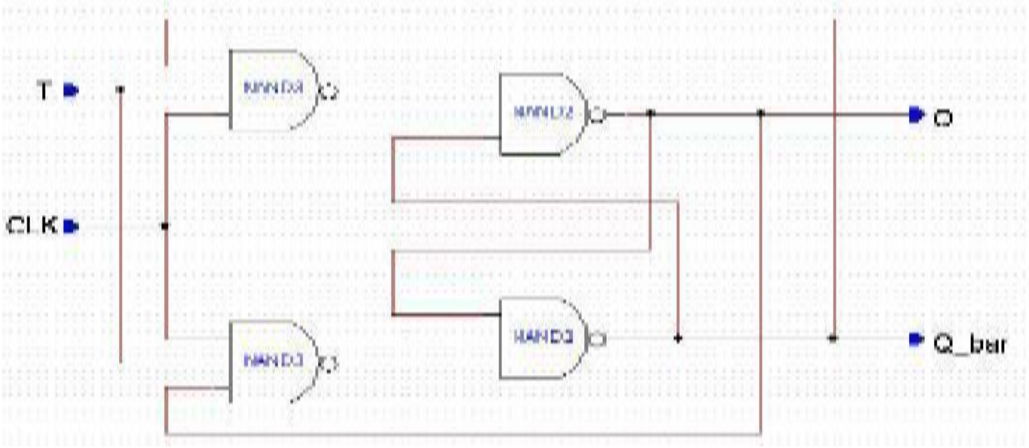
T	Q(t+1)	Operation
0	Q(t)	No change
1	Q’(t)	Complement

Characteristic Equation

Q(t+1) = T’Q(t) + TQ’(t)

= TQ(t)

Logic Diagram

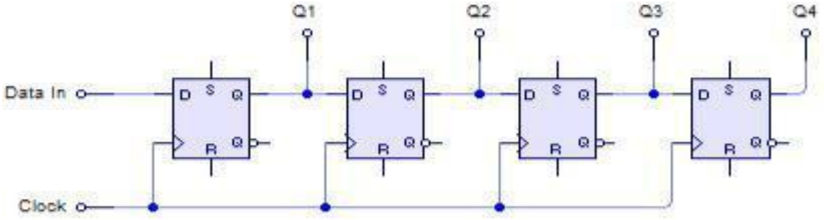


Q	$Q_{(next)}$	T
0	0	0
0	1	1
1	0	1
1	1	0

EXCITATION TABLE

3.2 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.



3.3 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

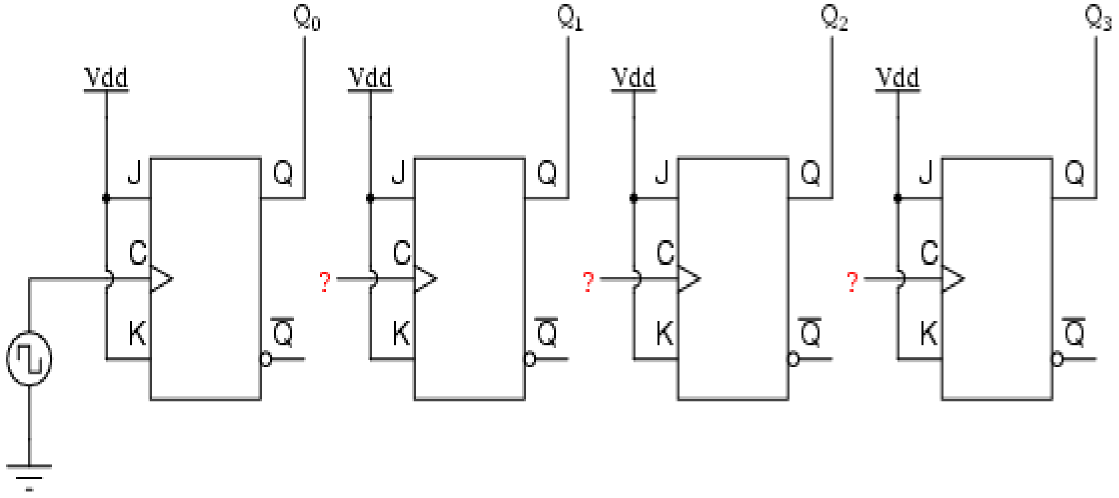
Up–down counter – counts both up and down, under command of a control input Ring counter – formed by a shift register with feedback connection in a ring

Johnson counter – a *twisted* ring counter casscaded counter

3.3.1Asynchronous Up-counter (Ripple Counter)

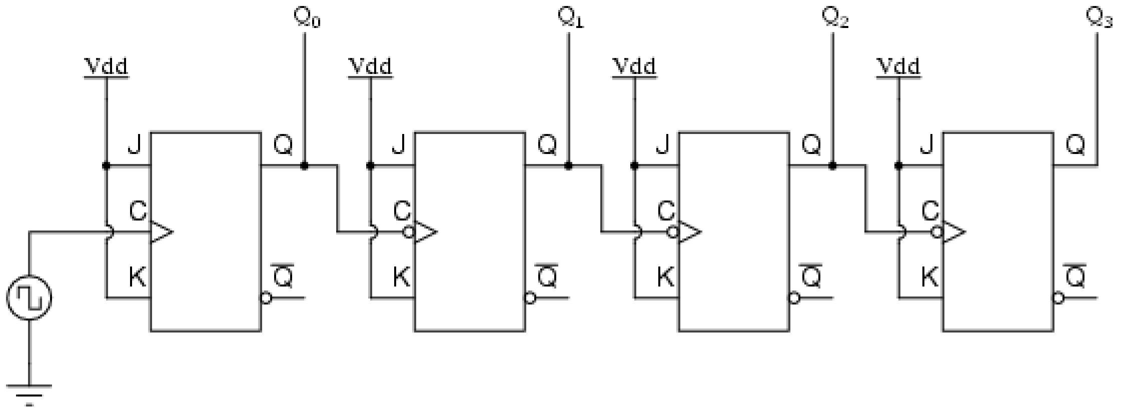
The binary count sequences follow a pattern of octave (factor of 2) frequency division, and that J-K flip-flop multivibrators set up for the "toggle" mode are capable of performing this type of frequency division, we can envision a circuit made up of several J-K flip-flops, cascaded to produce four bits of output. The main problem facing us is to determine *how* to connect these flip-flops together so that they toggle at the right times to produce the proper binary sequence. Examine the following binary count sequence, paying attention to patterns preceding the "toggling" of a bit between 0 and 1:

Starting with four J-K flip-flops connected in such a way to always be in the "toggle" mode, we need to determine how to connect the clock inputs in such a way so that each succeeding bit toggles when the bit before it transitions from 1 to 0. The Q outputs of each flip-flop will serve as the respective binary bits of the final, four-bit count:

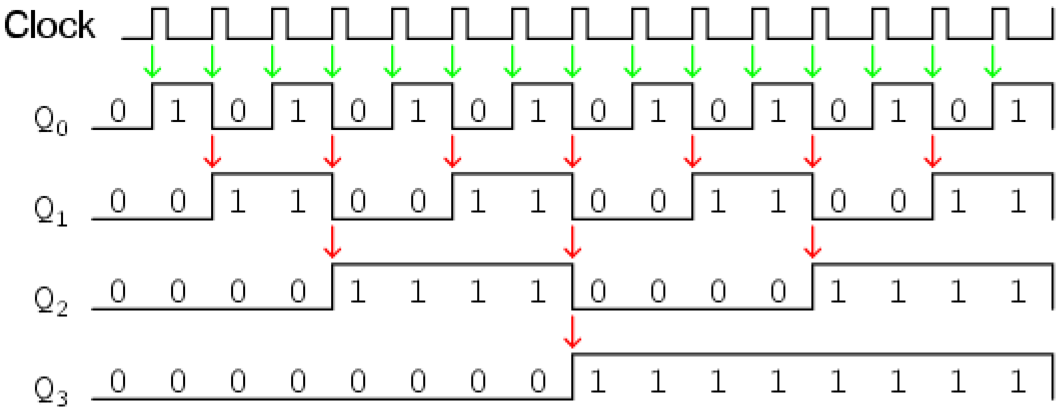


If we used flip-flops with negative-edge triggering (bubble symbols on the clock inputs), we could simply connect the clock input of each flip-flop to the Q output of the flip-flop before it, so that when the bit before it changes from a 1 to a 0, the "falling edge" of that signal would "clock" the next flip-flop to toggle the next bit:

A four-bit "up" counter



This circuit would yield the following output waveforms, when "clocked" by a repetitive source of pulses from an oscillator:



The first flip-flop (the one with the Q_0 output), has a positive-edge triggered clock input, so it toggles with each rising edge of the clock signal. Notice how the clock signal in this example has a duty cycle less than 50%. I've shown the signal in this manner for the purpose of demonstrating how the clock signal need not be symmetrical to obtain reliable, "clean" output bits in our four-bit binary sequence. In the very first flip-flop circuit shown in this chapter, I used the clock signal itself as one of the output bits. This is a bad practice in counter design, though, because it necessitates the use of a square wave signal with a 50% duty cycle ("high" time = "low" time) in order to obtain a count sequence where each and every step pauses for the same amount of time. Using one J-K flip-flop for each output bit, however, relieves us of the necessity of having a symmetrical clock signal, allowing the use of practically any variety of high/low waveform to increment the count sequence.

As indicated by all the other arrows in the pulse diagram, each succeeding output bit is toggled by the action of the preceding bit transitioning from "high" (1) to "low" (0). This is the pattern necessary to generate an "up" count sequence. less obvious solution for generating an "up" sequence using positive-edge triggered flip-flops is to "clock" each flip-flop using the Q' output of the preceding flip-flop rather than the Q output. Since the Q' output will always be the exact opposite state of the Q output on a J-K flip-flop (no invalid states with this type of flip-flop), a high-to-low transition on the Q output will be accompanied by a low-to-high transition on the Q' output. In other words, each time the Q output of a flip-flop transitions from 1 to 0, the Q' output of the same flip-flop will transition from 0 to 1, providing the positive-going clock pulse we would need to toggle a positive-edge triggered flip-flop at the right moment:

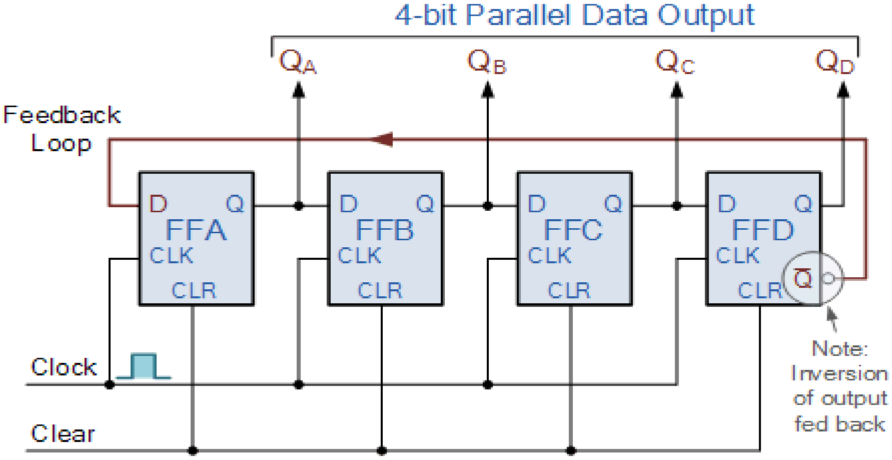
4 bit Johnsons counter:

Johnson Ring Counter

The **Johnson Ring Counter** or "Twisted Ring Counters", is another shift register with feedback exactly the same as the standard *Ring Counter* above, except that this time the inverted output Q of the last flip-flop is now connected back to the input D of the first flip-flop as shown below.The main advantage of this type of ring counter is that it only needs half the number of flip-flops compared to the standard ring counter then its modulo number is halved. So a "n-stage" Johnson counter will circulate a single data bit giving sequence of 2n different states and can therefore be considered as a "mod-2n counter".

This inversion of Q before it is fed back to input D causes the counter to "count" in a different way. Instead of counting through a fixed set of patterns like the normal ring counter such as for a 4-bit counter, "0001"(1), "0010"(2), "0100"(4), "1000"(8) and repeat, the Johnson counter counts up and then down as the initial logic "1" passes through it to the right replacing the preceding logic "0".

4-bit Johnson Ring Counter



A 4-bit Johnson ring counter passes blocks of four logic "0" and then four logic "1" thereby producing an 8-bit pattern. As the inverted output Q is connected to the input D this 8-bit pattern continually repeats. For example, "1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000" and this is demonstrated in the following table below.

Truth Table for a 4-bit Johnson Ring Counter

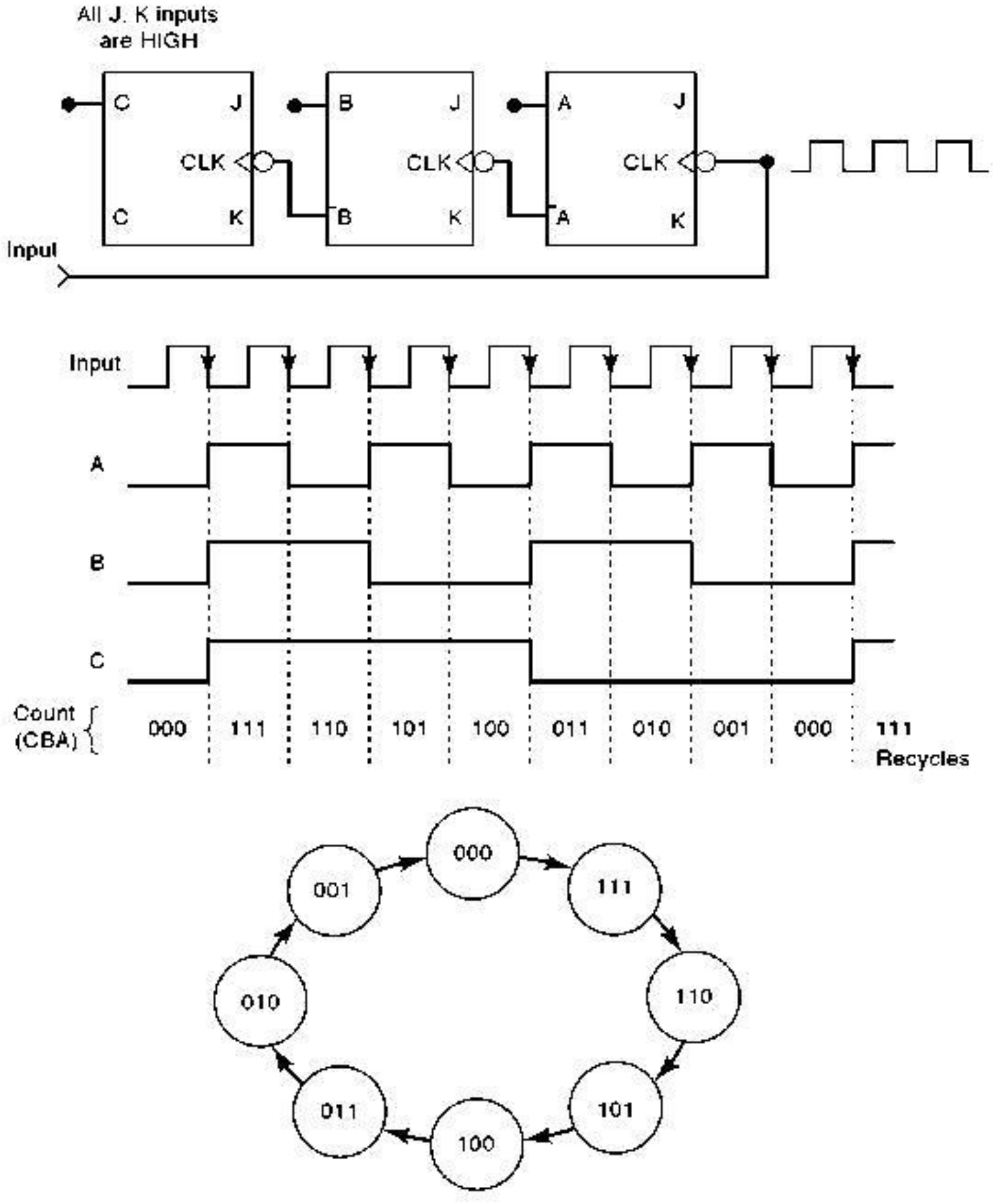
Clock Pulse No	FFA	FFB	FFC	FFD
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

As well as counting or rotating data around a continuous loop, ring counters can also be used to detect or recognise various patterns or number values within a set of data. By connecting simple logic gates such as the AND or the OR gates to the outputs of the flip-flops the circuit can be made to detect a set number or value.

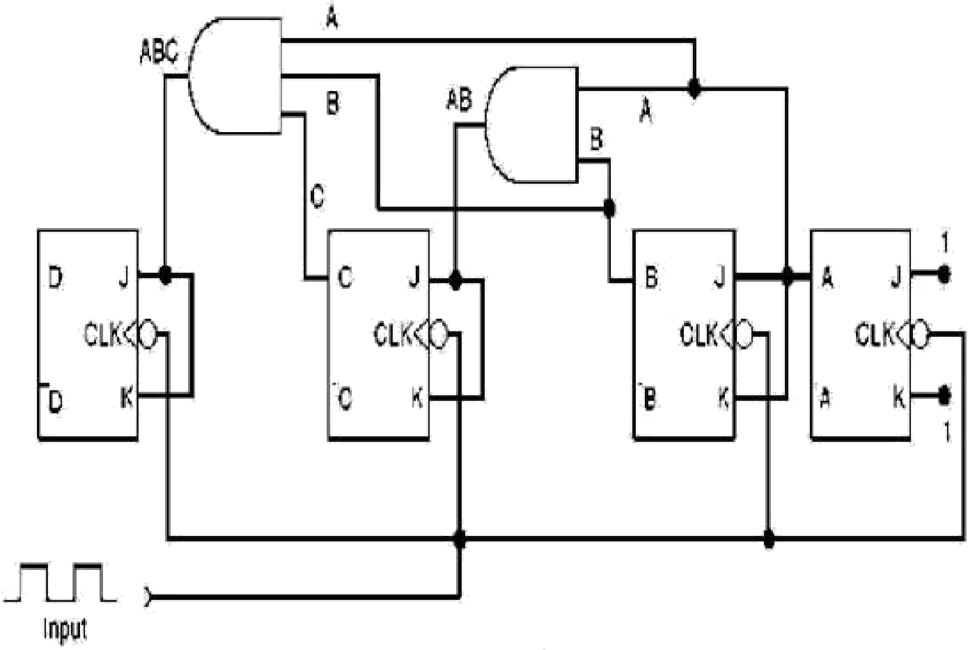
Standard 2, 3 or 4-stage **Johnson Ring Counters** can also be used to divide the frequency of the clock signal by varying their feedback connections and divide-by-3 or divide-by-5 outputs are also available. For example, a 3-stage Johnson Ring Counter could be used as a 3-phase, 120 degree phase shift square wave generator by connecting to the data outputs at A, B and NOT-B.

The standard 5-stage Johnson counter such as the commonly available CD4017 is generally used as a synchronous decade counter/divider circuit. Other combinations such as the smaller 2-stage circuit which is also called a “Quadrature” (sine/cosine) Oscillator or Generator can be used to produce four individual outputs that are each 90 degrees “out-of-phase” with respect to each other to produce a 4-phase timing signal as shown below.

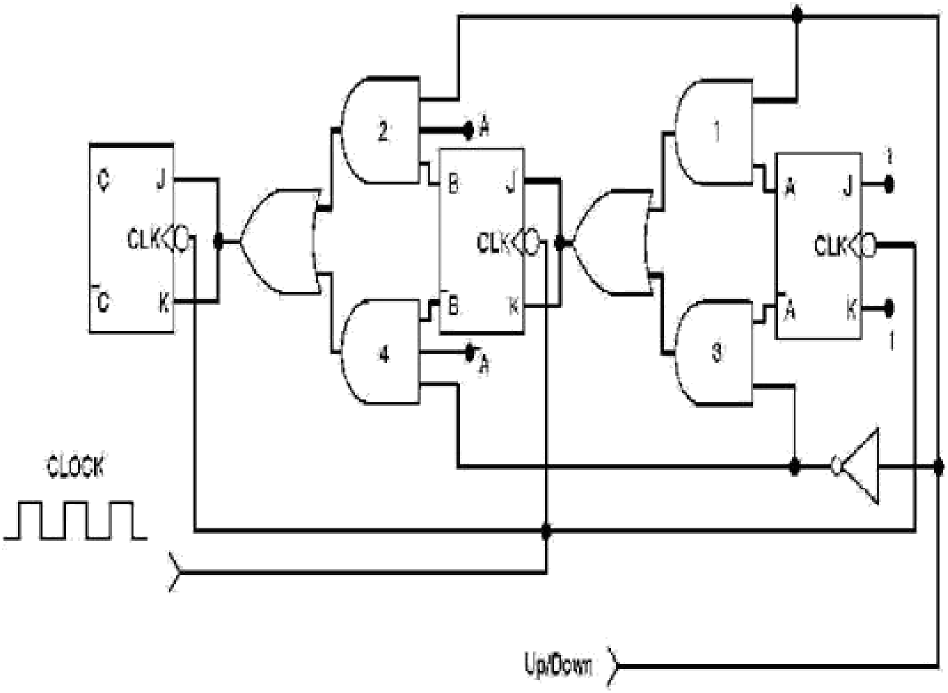
3.3.2 Synchronous Counter



Synchronous (Parallel) Counter

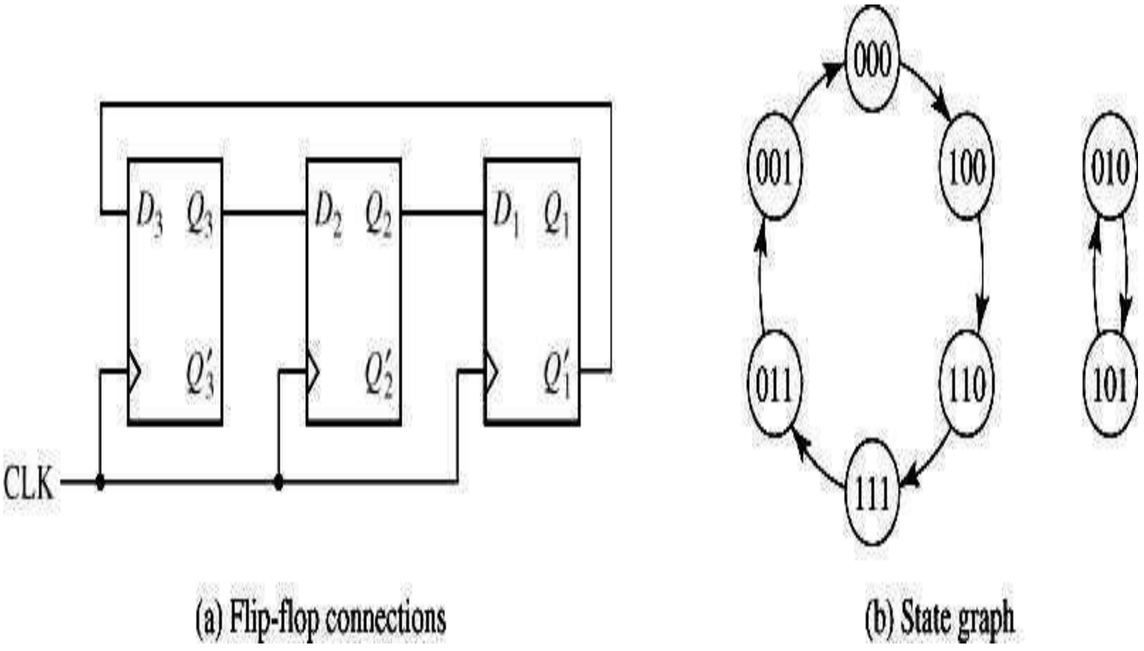

ABC
A
B
C
AB
A
B
D
J
CLK
D
K
C
J
CLK
C
K
B
J
CLK
B
K
A
J
CLK
A
K
1
1
Input

(a)


2
A
4
A
1
3
C
J
CLK
C
K
B
J
CLK
B
K
A
J
CLK
A
K
1
1
CLOCK
Up/Down

(a)

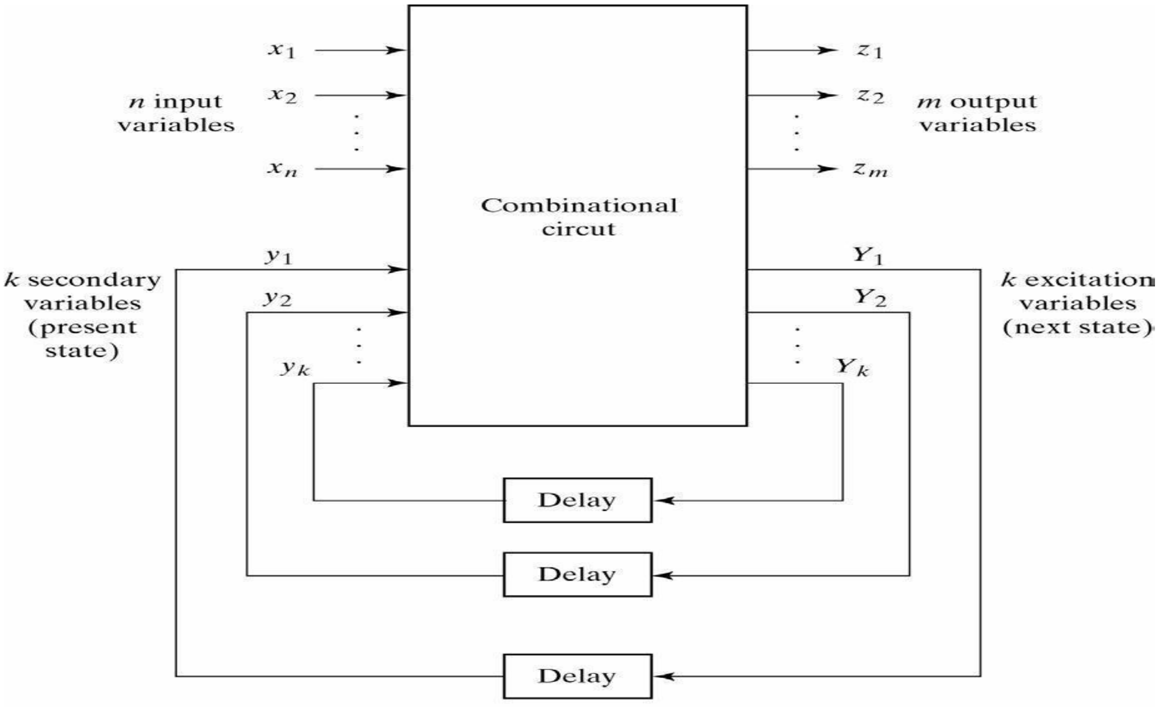
3.3.3 Johnson Counter



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

3.4 SYNCHRONOUS SEQUENTIAL CIRCUIT

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.

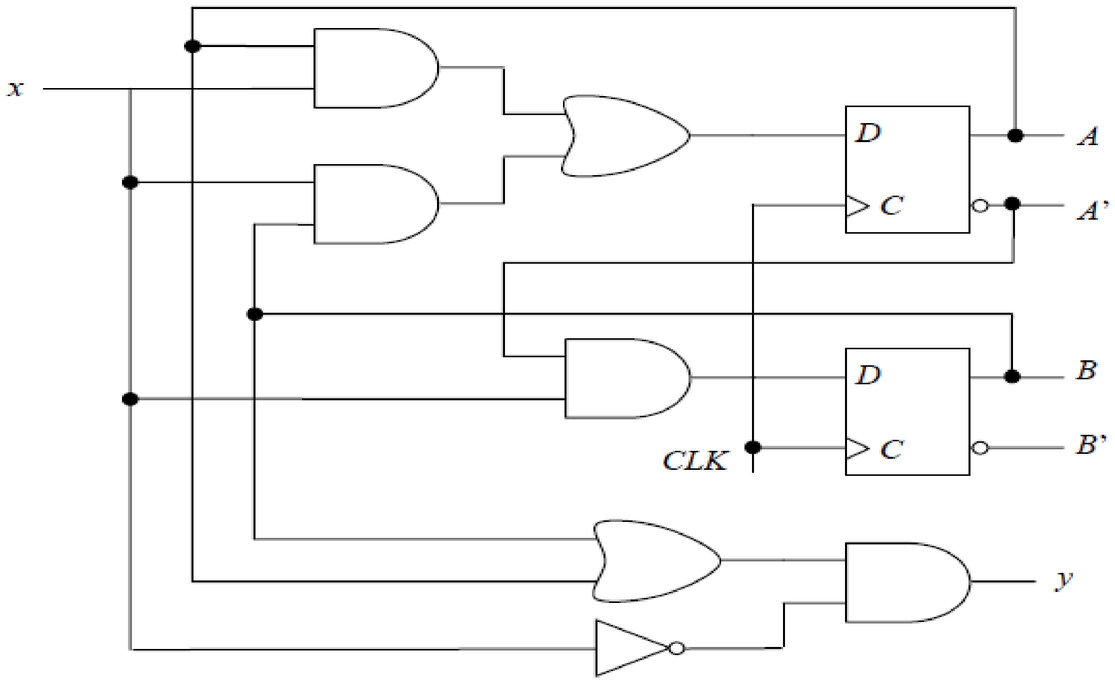
Depending on regulations of functioning, digital circuits are divided into synchronous and asynchronous. In accordance with this, behavior of devices obeys synchronous or asynchronous logic.

3.4.1 Analysis and synthesis of synchronous sequential circuit

Analysis of Clocked Sequential Circuits

A clocked sequential circuit can be analyzed by using: (i) The state equation obtained by replacing flip-flop input equations in its characteristic equation, or (ii) a state table. - *Analysis with D flip-flops*: Write the Boolean expression for each flip-flop input,which is, in fact, the flip-flop output right after the next active clock edge.

- Example:



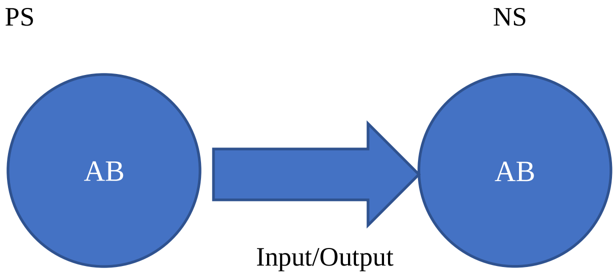
State equation can be written a and State table or transition table lists all possible binary combinations of present state and inputs.

	m flip-flops		n inputs	$A.x+B.x$	$A'.x$	$A.x'+B.x'$
	Present state		Input	Next state		Output
	A	*B*	*x*	*A*	*B*	*y*
2^{m+n} rows	0	0	0	0	0	0
	0	0	1	0	1	0
	0	1	0	0	0	1
	0	1	1	1	1	0
	1	0	0	0	0	1
	1	0	1	1	0	0
	1	1	0	0	0	1
	1	1	1	1	0	0

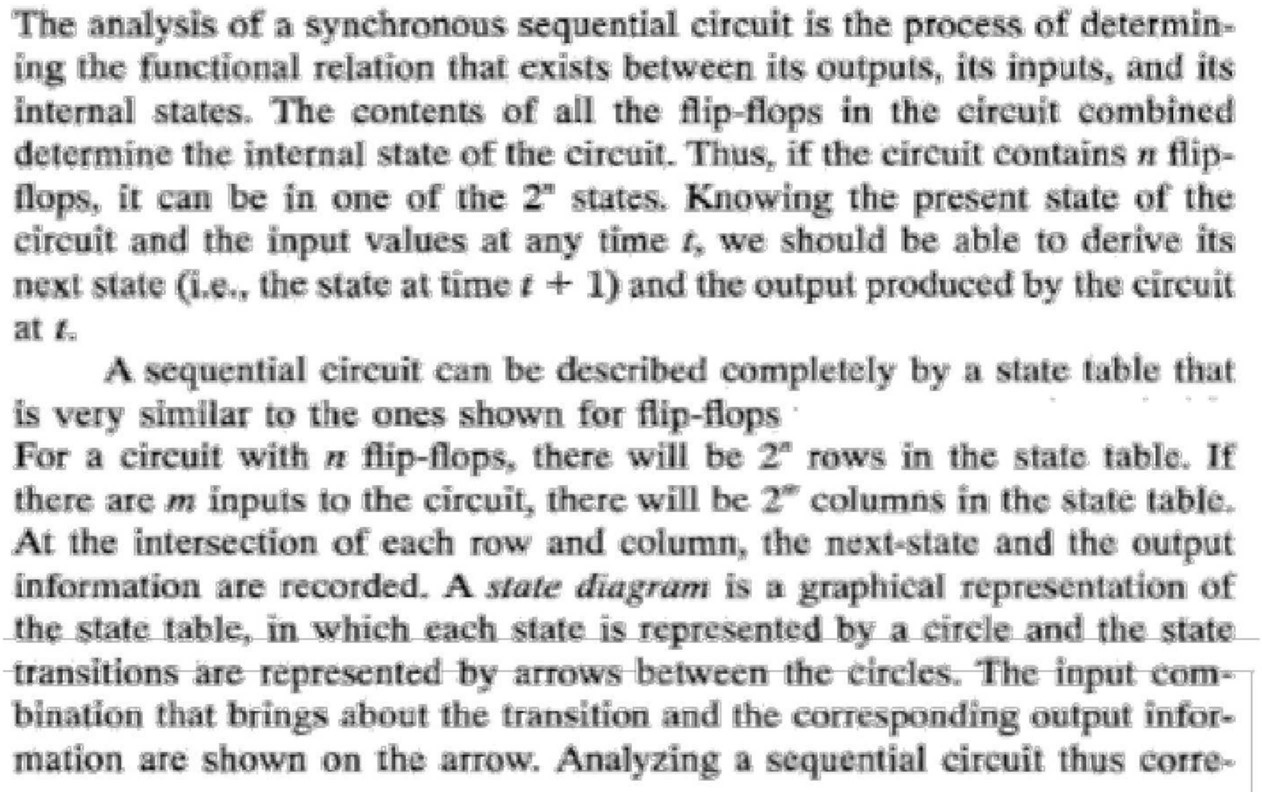
For some applications (like state reduction) we may use the following form of the state table

PS		NS				OUTPUT	
A	B	X=0	X=0	X=0	X=1	X=0	X=1
		A	B	A	B		
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

State diagram



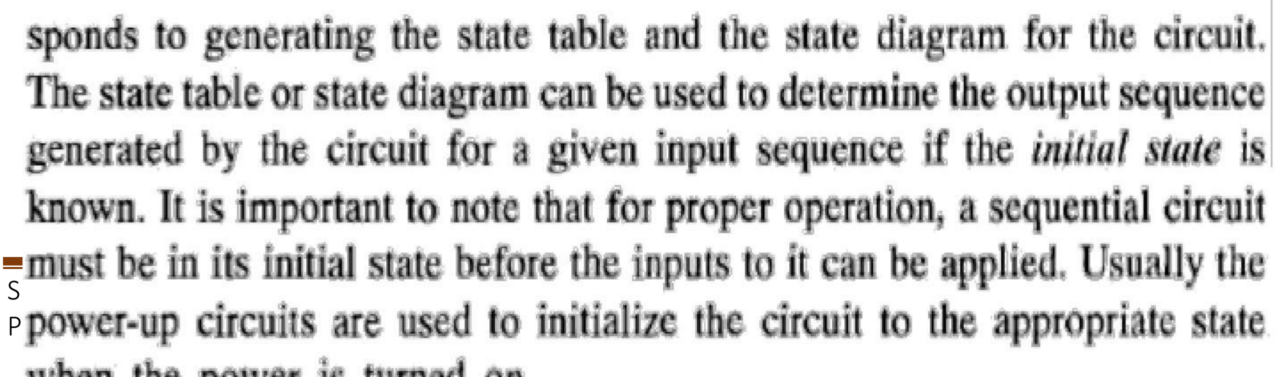
The state diagram follows directly from the state table.



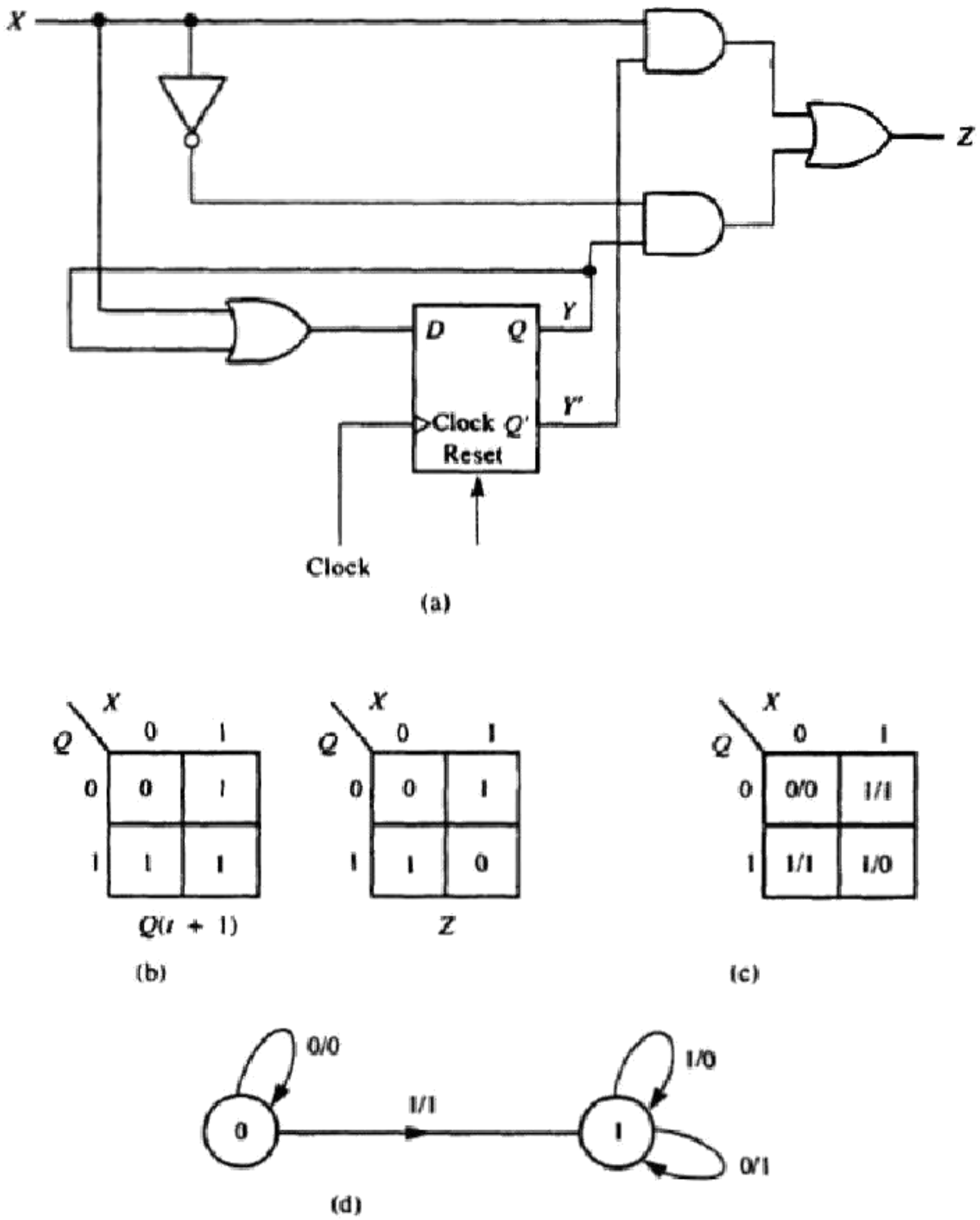
The analysis of a synchronous sequential circuit is the process of determining the functional relation that exists between its outputs, its inputs, and its internal states. The contents of all the flip-flops in the circuit combined determine the internal state of the circuit. Thus, if the circuit contains n flip-flops, it can be in one of the 2^n states. Knowing the present state of the circuit and the input values at any time t, we should be able to derive its next state (i.e., the state at time $t + 1$) and the output produced by the circuit at t.

A sequential circuit can be described completely by a state table that is very similar to the ones shown for flip-flops

For a circuit with n flip-flops, there will be 2^n rows in the state table. If there are m inputs to the circuit, there will be 2^m columns in the state table. At the intersection of each row and column, the next-state and the output information are recorded. A *state diagram* is a graphical representation of the state table, in which each state is represented by a circle and the state transitions are represented by arrows between the circles. The input combination that brings about the transition and the corresponding output information are shown on the arrow. Analyzing a sequential circuit thus corre-



sponds to generating the state table and the state diagram for the circuit. The state table or state diagram can be used to determine the output sequence generated by the circuit for a given input sequence if the *initial state* is known. It is important to note that for proper operation, a sequential circuit must be in its initial state before the inputs to it can be applied. Usually the power-up circuits are used to initialize the circuit to the appropriate state when the power is turned on.



(a)

Q \ X	0	1
0	0	1
1	1	1

Q(t + 1)

Q \ X	0	1
0	0	1
1	1	0

Z

(b)

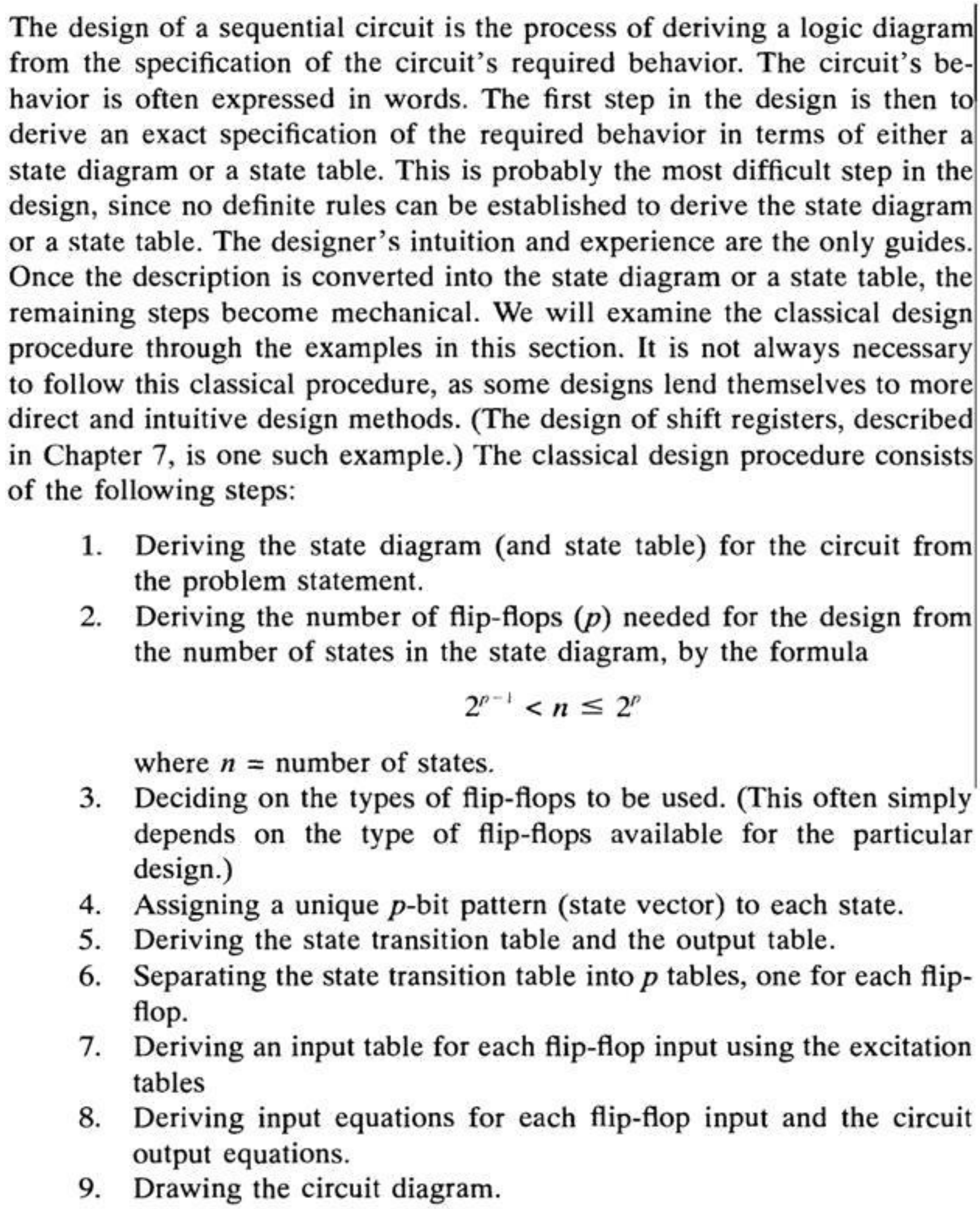
Q \ X	0	1
0	0/0	1/1
1	1/1	1/0

(c)

(d)

Sequential circuit analysis. (a) Circuit; (b) next-state and output tables; (c) transition table; (d) state diagram; (e) timing diagram for level input; (f) timing diagram for synchronous pulse input.

3.4.2 Design of synchronous sequential circuit



The design of a sequential circuit is the process of deriving a logic diagram from the specification of the circuit's required behavior. The circuit's behavior is often expressed in words. The first step in the design is then to derive an exact specification of the required behavior in terms of either a state diagram or a state table. This is probably the most difficult step in the design, since no definite rules can be established to derive the state diagram or a state table. The designer's intuition and experience are the only guides. Once the description is converted into the state diagram or a state table, the remaining steps become mechanical. We will examine the classical design procedure through the examples in this section. It is not always necessary to follow this classical procedure, as some designs lend themselves to more direct and intuitive design methods. (The design of shift registers, described in Chapter 7, is one such example.) The classical design procedure consists of the following steps:

1. Deriving the state diagram (and state table) for the circuit from the problem statement.
2. Deriving the number of flip-flops (p) needed for the design from the number of states in the state diagram, by the formula

   $$2^{p-1} < n \leq 2^{p}$$

   where n = number of states.
3. Deciding on the types of flip-flops to be used. (This often simply depends on the type of flip-flops available for the particular design.)
4. Assigning a unique p-bit pattern (state vector) to each state.
5. Deriving the state transition table and the output table.
6. Separating the state transition table into p tables, one for each flip-flop.
7. Deriving an input table for each flip-flop input using the excitation tables
8. Deriving input equations for each flip-flop input and the circuit output equations.
9. Drawing the circuit diagram.

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.

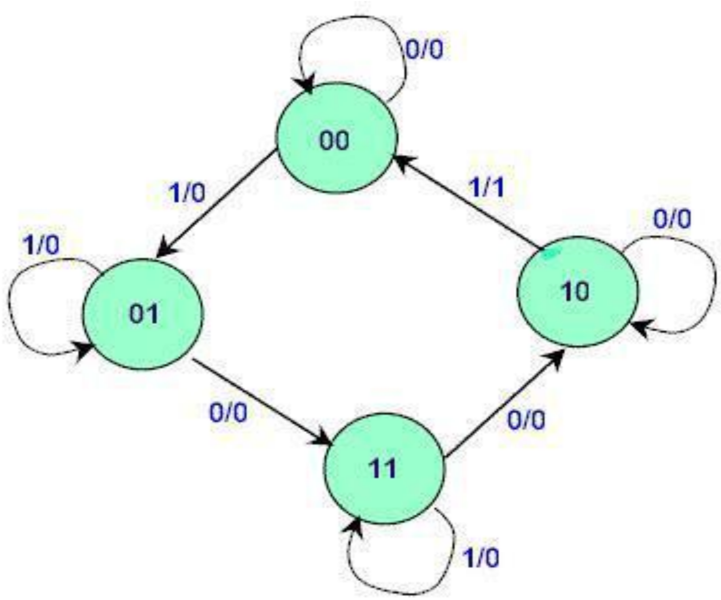


Figure . State Diagram

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

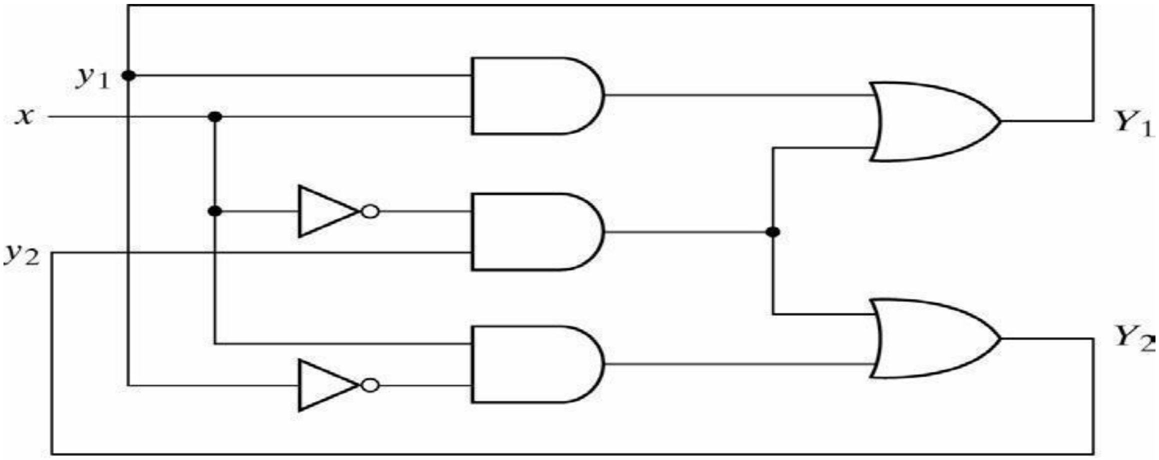
In the design of sequential circuits, we need to reduce the number of flip flops and the number of logic gates used in the combinational circuit part. Reduction of the number of flip-flops may result from the reduction of the number of states in the circuit. This is possible if we are interested in the 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.

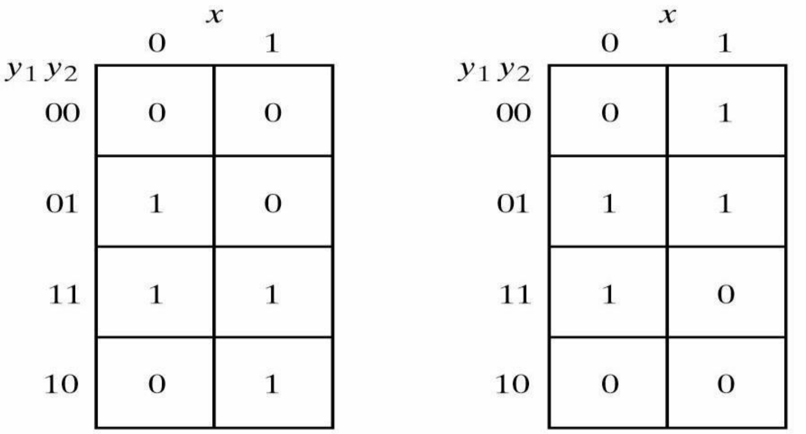
State	Assignment 1	Assignment 2
A	001	000
B	000	010
C	010	101
D	110	111

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:

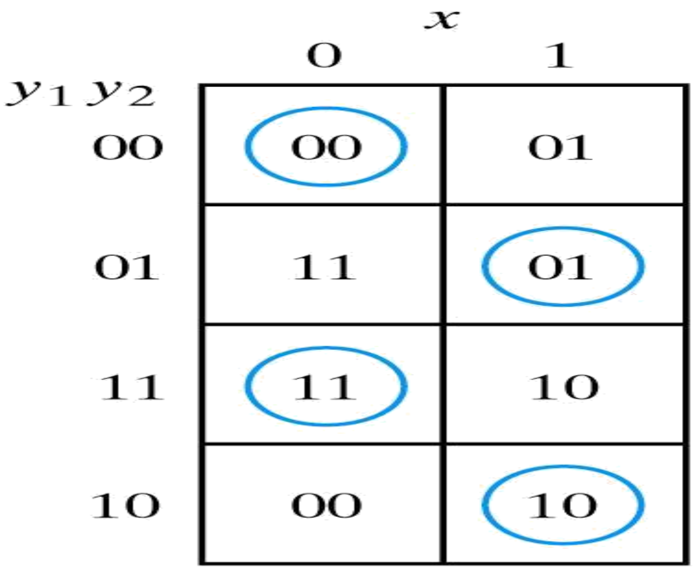


$y_1 y_2$ \ x	0	1
00	0	0
01	1	0
11	1	1
10	0	1

(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

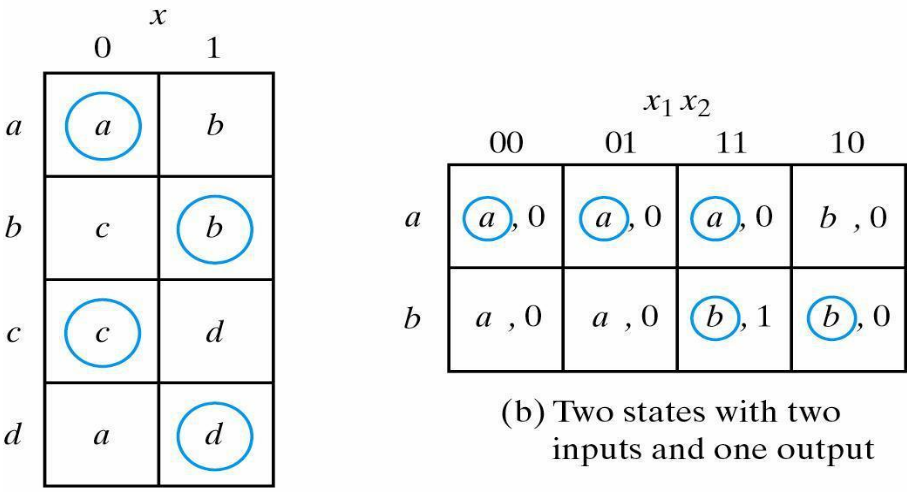
(b) Map for $Y_2 = xy'_1 + x'y_2$



y_1y_2 \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

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

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.



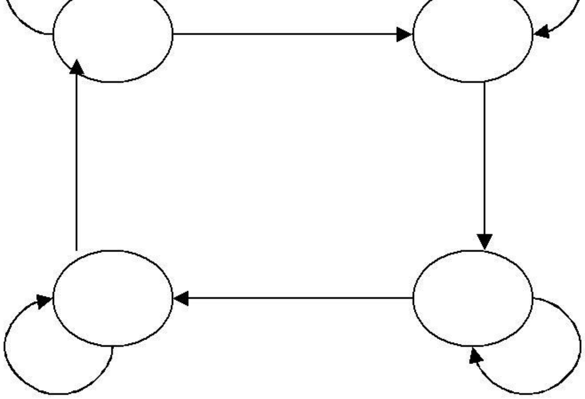
	x = 0	1
a	a	b
b	c	b
c	c	d
d	a	d

(a) Four states with one input

x_1x_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

Frequently asked Questions



7)

Karpagam Academy of Higher Education

(Established under Section 3 of UGC Act 1956)

Eachanari, Coimbatore-641 021. INDIA

Department of Electronics and Communication Engineering
Faculty of Engineering

DIGITAL ELECTRONICS

LECTURE NOTES

PREPARED BY
Dr.S.Bhavani,HOD/ECE

14BEEC403 DIGITAL ELECTRONICS 3 0 0 3 100

INTENDED OUTCOMES:

- To introduce number systems and codes
- To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
- To introduce the methods for simplifying Boolean expressions
- To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
- To introduce the concept of memories and programmable logic devices.

UNIT-I NUMBER SYSTEMS

Binary, Octal, Decimal, Hexadecimal-Number base conversions – complements – signed Binary numbers. Binary Arithmetic- Binary codes: Weighted –BCD-2421-Gray code-Excess 3 code-ASCII –Error detecting code – conversion from one code to another-Boolean postulates and laws –De-Morgan's Theorem- Principle of Duality- Boolean expression – Boolean function-Minimization of Boolean expressions – Sum of Products (SOP) –Product of Sums (POS)-Minterm- Maxterm- Canonical forms – Conversion between canonical forms –Karnaugh map Minimization – Don't care conditions.

UNIT-II LOGIC GATES AND COMBINATIONAL CIRCUITS LOGIC GATES

AND, OR, NOT, NAND, NOR, Exclusive – OR and Exclusive – NOR- Implementations of Logic Functions using gates, NAND –NOR implementations –Multi level gate implementations-Multi output gate implementations. TTL and CMOS Logic and their characteristics –Tristate gates.

COMBINATIONAL CIRCUITS: Design procedure – Adders-Subtractors – Serial adder/ Subtractor - Parallel adder/ Subtractor- Carry look ahead adder- BCD adder- Magnitude Comparator- Multiplexer/ Demultiplexer- encoder / decoder – parity checker – code converters. Implementation of combinational logic using MUX, ROM, PAL and PLA.

UNIT-III SEQUENTIAL CIRCUIT

Flip flops SR, JK, T, D and Master slave – Characteristic table and equation –Application table – Edge triggering –Level Triggering –Realization of one flip flop using other flip flops – Asynchronous / Ripple counters – Synchronous counters –Modulo – n counter –Classification of sequential circuits – Moore and Mealy -Design of Synchronous counters: state diagram- State table –State minimization –State assignment- ASM-Excitation table and maps-Circuit implementation - Register – shift registers- Universal shift register – Shift counters – Ring counters.

UNIT-IV ASYNCHRONOUS SEQUENTIAL CIRCUITS

Design of fundamental mode and pulse mode circuits – primitive state / flow table – Minimization of primitive state table –state assignment – Excitation table – Excitation map-cycles – Races –Hazards: Static –Dynamic –Essential –Hazards elimination.

UNIT-V MEMORY DEVICES

Classification of memories –RAM organization – Write operation –Read operation – Memory cycle - Timing wave forms – Memory decoding – memory expansion – Static RAM Cell-Bipolar RAM cell – MOSFET RAM cell –Dynamic RAM cell –ROM organization - PROM –EPROM – EEPROM –EAPROM –Programmable Logic Devices –Programmable Logic Array (PLA)-Programmable Array Logic (PAL)-Field Programmable Gate Arrays (FPGA).

TEXT BOOKS:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Morris Mano.M	Digital Design	Prentice Hall of India Pvt. Ltd., New Delhi	2003
2	John M .Yarbrough	Digital Logic Applications and Design	Thomson- Vikas publishing house, New Delhi	2002

REFERENCES:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Salivahanan.S and Arivazhagan.S	Digital Circuits and Design	Vikas Publishing House Pvt. Ltd, New Delhi	2004
2	Charles H.Roth	Fundamentals of Logic Design	Thomson Publication Company, New Delhi.	2003
3	Donald P.Leach and Albert Paul Malvino	Digital Principles and Applications	Tata McGraw Hill Publishing Company Limited, New Delhi	2003
4	Jain.R.P	Modern Digital Electronics	Tata McGraw–Hill publishing company limited, New Delhi	2003
5	Thomas L. Floyd	Digital Fundamentals	Pearson Education, New Delhi	2003

WEBSITES:

http://www.allaboutcircuits.com/vol_2/chpt_9/2.html
http://www.educypedia.be/electronics/digital.html

STAFF HOD

UNIT-IV ASYNCHRONOUS SEQUENTIAL CIRCUITS

4. 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).

4.1Analysis 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.

4.2 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

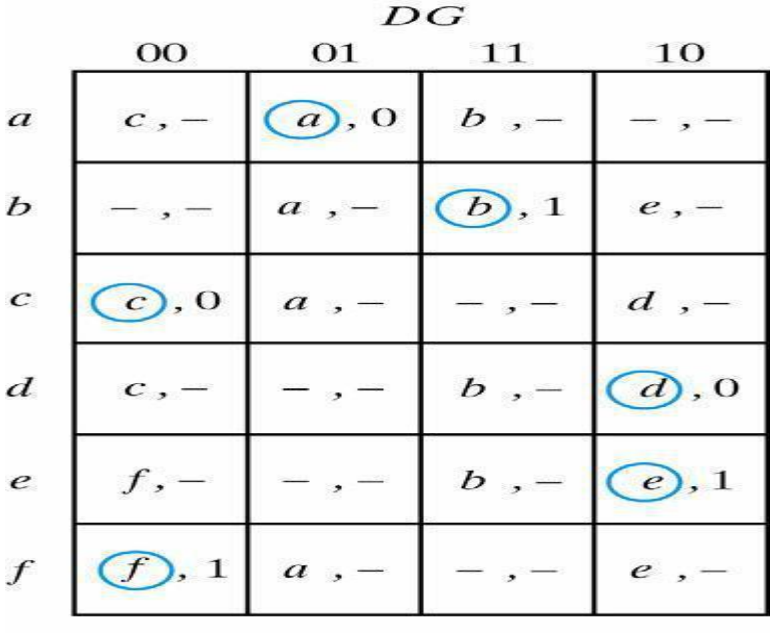
Design a gated latch circuit with two inputs, *G* (gate) and *D* (data), and one output *Q*. The gated latch is a memory element that accepts the value of *D* when *G* = 1 and retains this value after *G* goes to 0. Once *G* = 0, a change in *D* does not change the value of the output *Q*.

Solution

State table

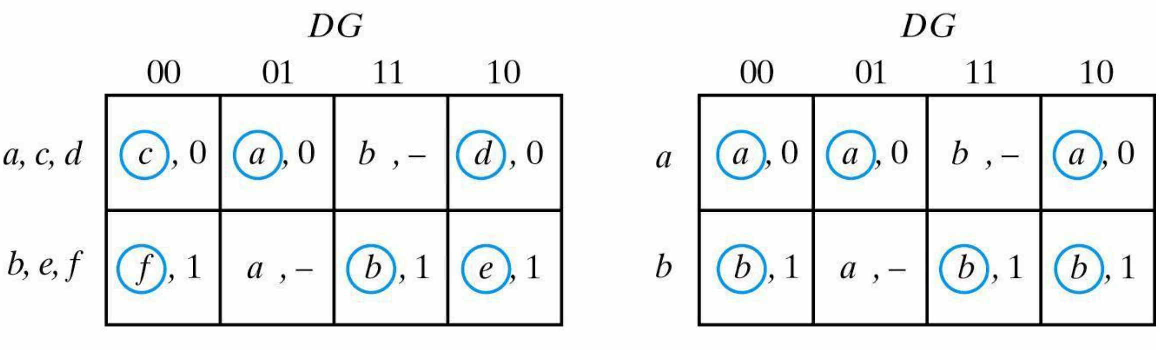
State	Inputs		Output
	D	G	Q
a	0	1	0
b	1	1	1
c	0	0	0
d	1	0	0
e	1	0	1
f	0	0	1

Primitive Flow table



	DG			
	00	01	11	10
a	*c* , –	$\textcircled{a}$, 0	*b* , –	– , –
b	– , –	*a* , –	$\textcircled{b}$, 1	*e* , –
c	$\textcircled{c}$, 0	*a* , –	– , –	*d* , –
d	*c* , –	– , –	*b* , –	$\textcircled{d}$, 0
e	*f* , –	– , –	*b* , –	$\textcircled{e}$, 1
f	$\textcircled{f}$, 1	*a* , –	– , –	*e* , –

Informal Merging



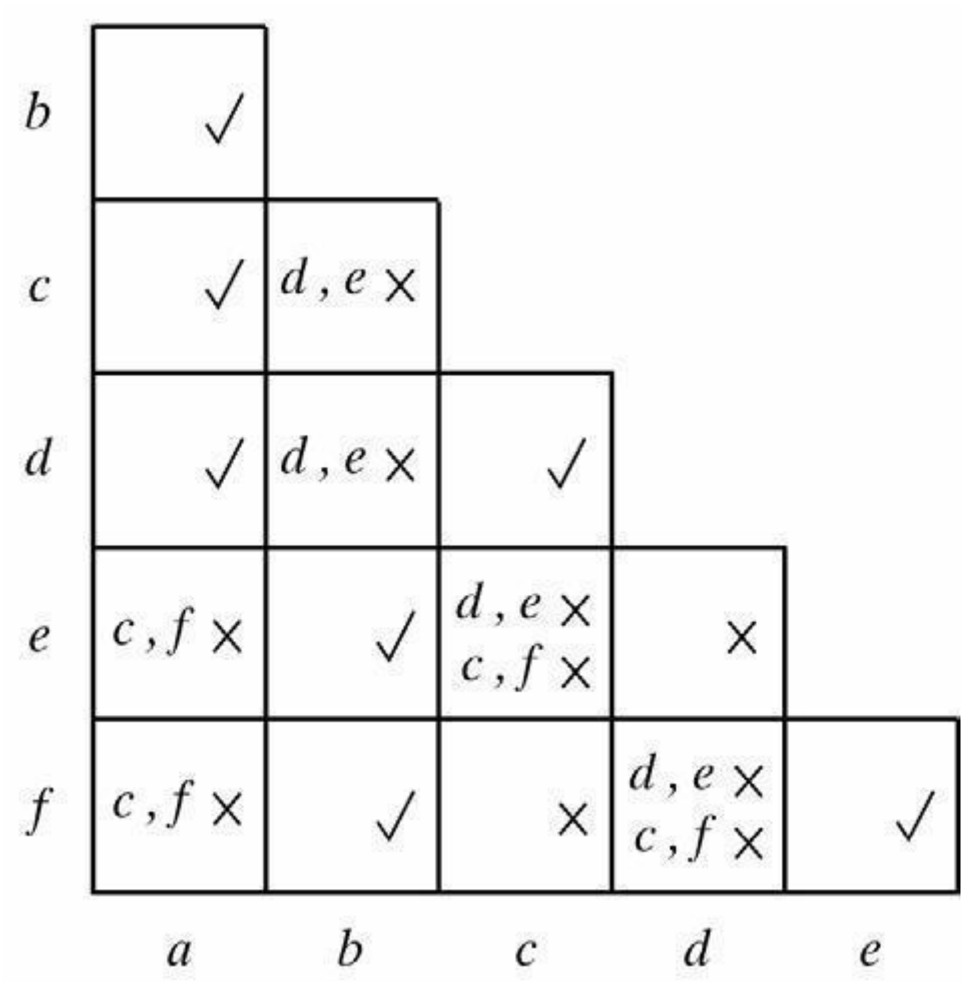
	DG 00	01	11	10
a, c, d	*c*, 0	*a*, 0	*b* , –	*d*, 0
b, e, f	*f*, 1	*a* , –	*b*, 1	*e*, 1

	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

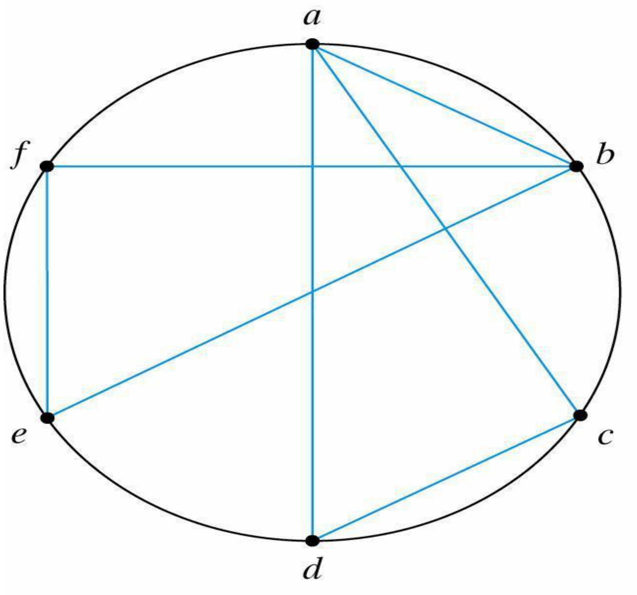
Compatible Pairs



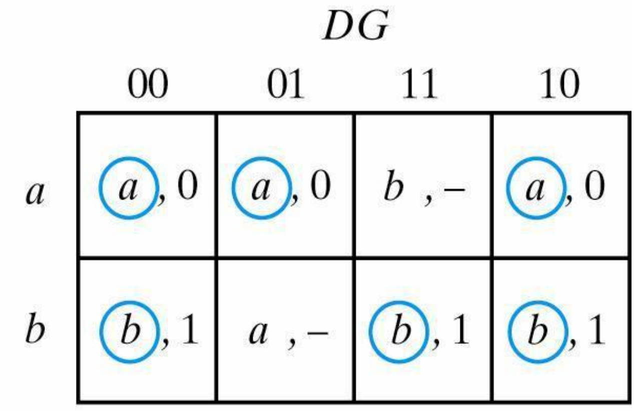
b	✓				
c	✓	*d* , *e* ×			
d	✓	*d* , *e* ×	✓		
e	*c* , *f* ×	✓	*d* , *e* × *c* , *f* ×	×	
f	*c* , *f* ×	✓	×	*d* , *e* × *c* , *f* ×	✓
	a	*b*	*c*	*d*	*e*

57

Maximal Compatibles

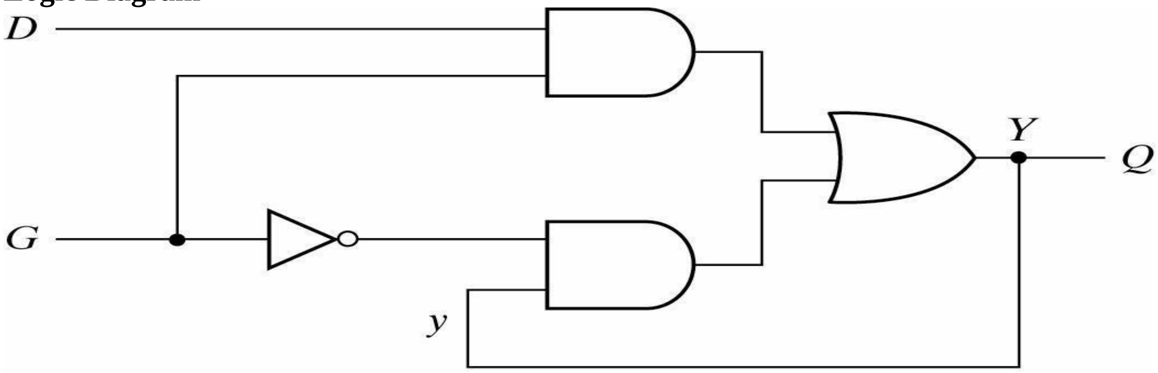


Reduced Table



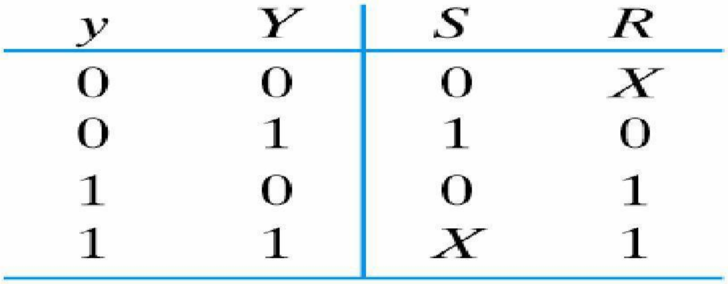
	DG 00	01	11	10
a	ⓐ, 0	ⓐ, 0	b , –	ⓐ, 0
b	ⓑ, 1	a , –	ⓑ, 1	ⓑ, 1

Logic Diagram

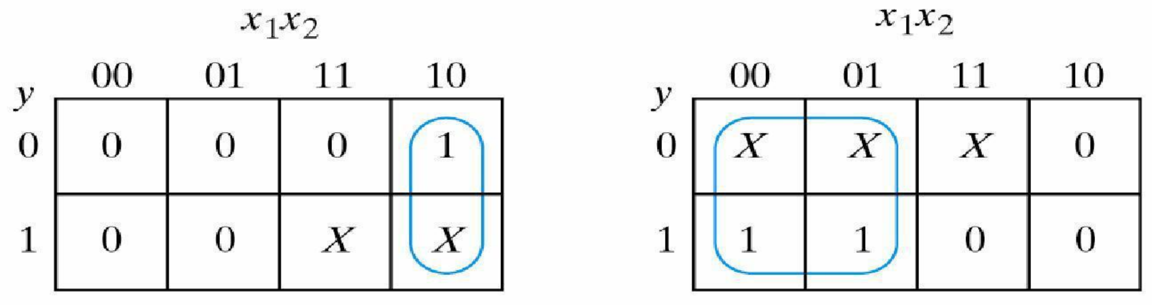


The same problem using SR Latch

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



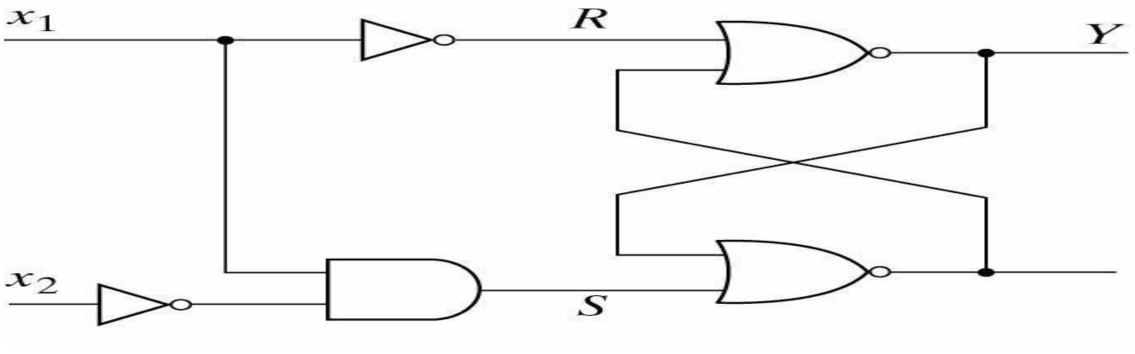
(c) Map for $S = x_1x'_2$

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

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

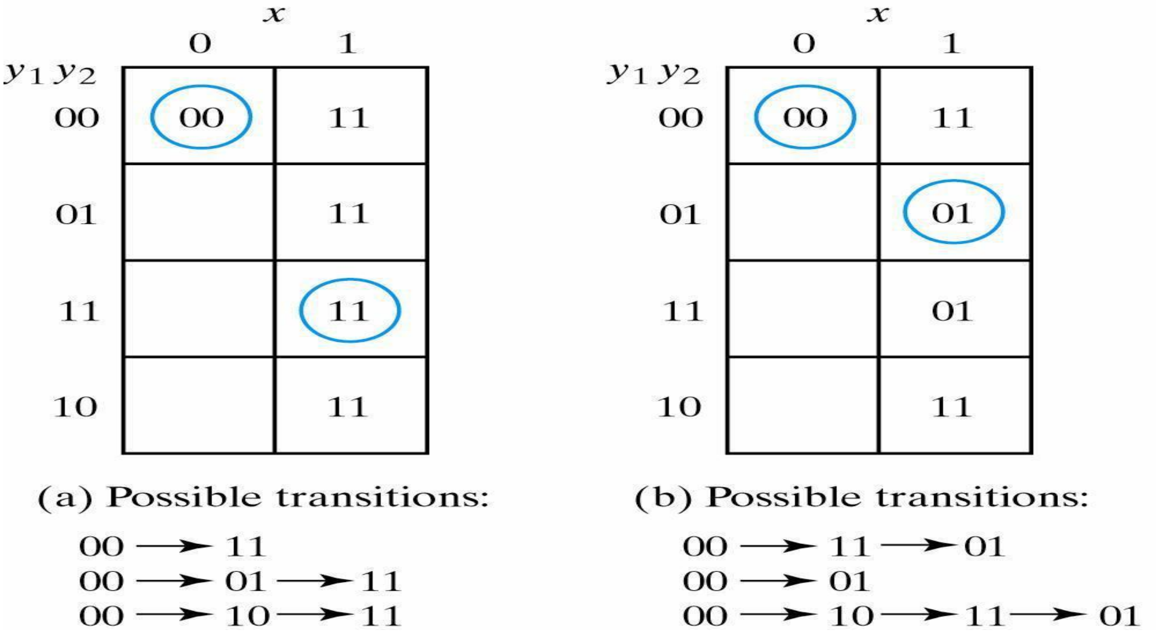
$y \backslash x_1x_2$	00	01	11	10
0	*X*	*X*	*X*	0
1	1	1	0	0

SR Latch Logic Diagram



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:



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

(a) Possible transitions:

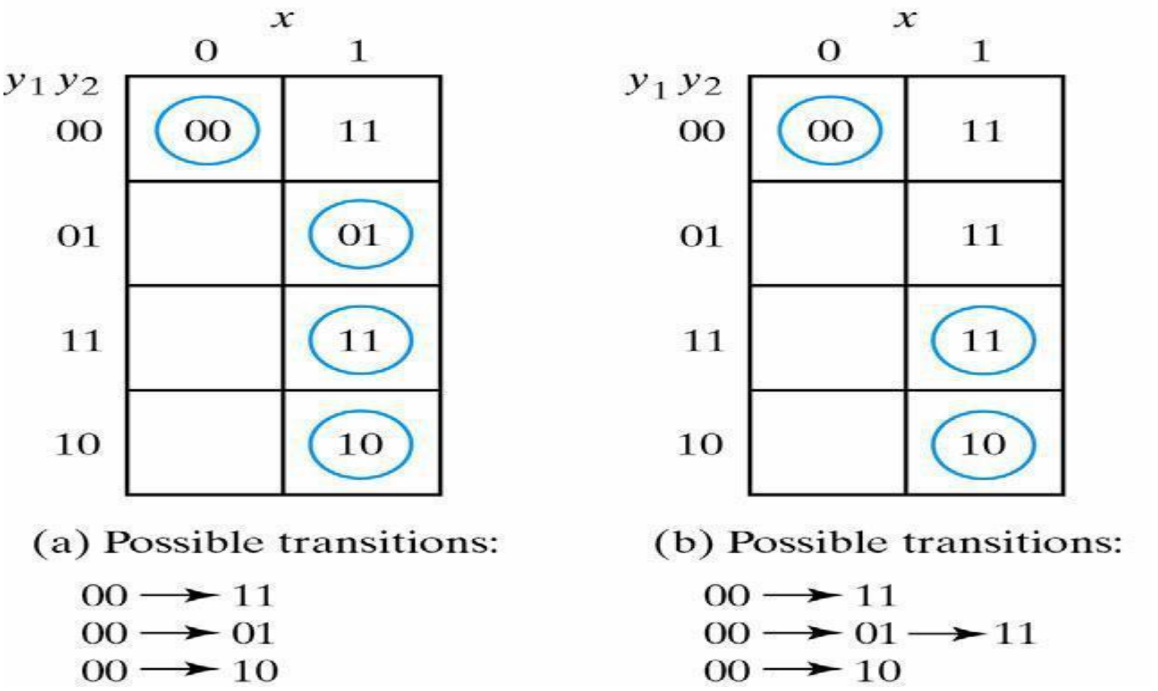
00 → 11
00 → 01 → 11
00 → 10 → 11

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

(b) Possible transitions:

00 → 11 → 01
00 → 01
00 → 10 → 11 → 01

The transition tables below illustrate critical races:



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

(a) Possible transitions:

00 → 11
00 → 01
00 → 10

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

(b) Possible transitions:

00 → 11
00 → 01 → 11
00 → 10

60

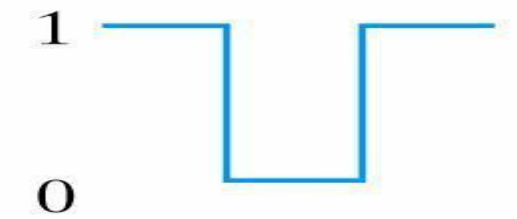
4.4 Hazards

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**.

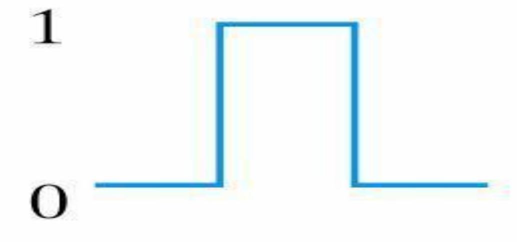
Hazards are a temporary problem, as the logic circuit will eventually settle to the desired function. However, despite the logic arriving at the correct output, it is imperative that hazards be eliminated as they can have an effect on other connected systems.

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:

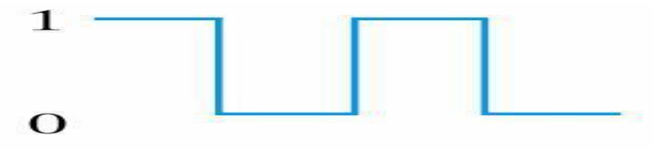
Static-1 Hazard: the output is currently 1 and after the inputs change, the output momentarily changes to 0 before settling on 1



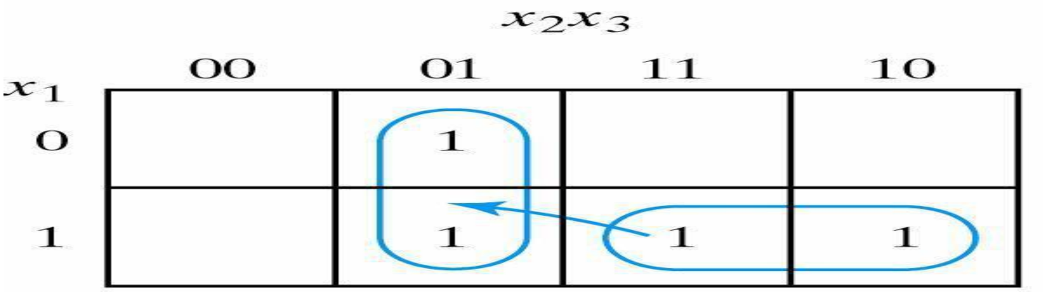
Static-0 Hazard: the output is currently 0 and after the inputs change, the output momentarily changes to 1 before settling on 0



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,

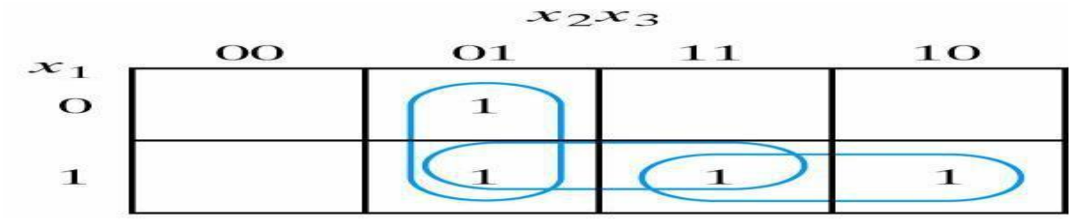


x_1 \ x_2x_3	00	01	11	10
0		1		
1		1	1	1

Normal K‟Map answer is

$Y = x1x2 + x2' x3$

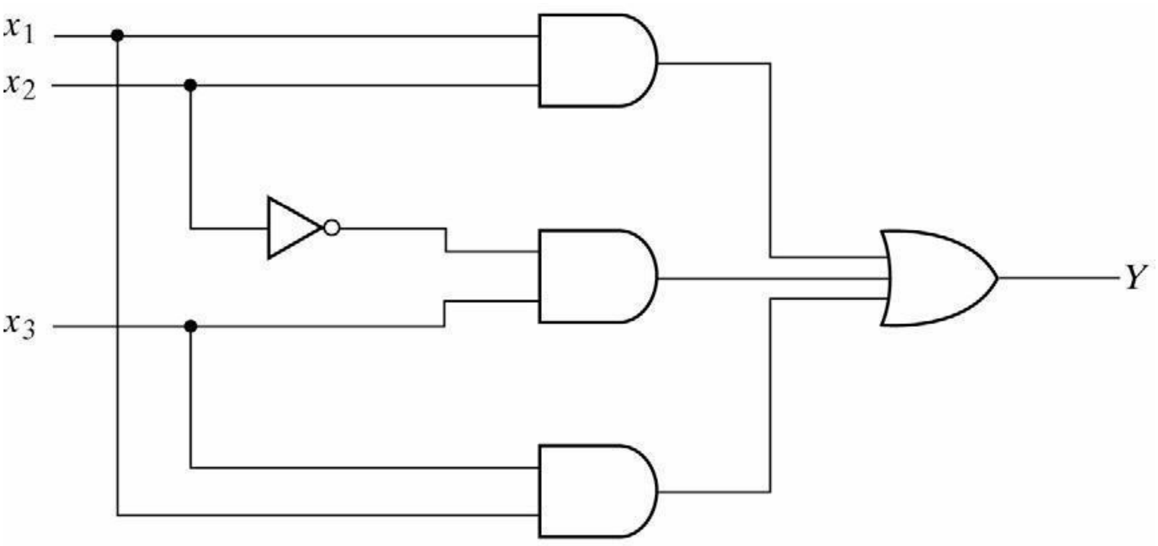
Hazard free map is



x_1 \ x_2x_3	00	01	11	10
0		1		
1		1	1	1

$Y = x1x2 + x2' x3 + x1x3$

Hazard Free Logic diagram is



End Semester Quetion paper

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – Regular

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours Maximum: 100 marks

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Convert $(237)_{10}$ to Hexadecimal $(?)_{16}$ and Octal $(?)_{8}$
2. Convert $(1011011)_{binary}$ to $(?)_{gray}$
3. State De-Morgans Law
4. Write Associative and Distributive Laws
5. Draw OR gate and derive the truth table.
6. What is Propagation Delay?
7. What are universal gates? Why it is called so.
8. Define Half Adder and Draw the circuit
9. Differentiate Combinational and Sequential logic circuits.
10. Draw 1x4 De-MUX block diagram
11. Draw T - flip flop & explain truth table
12. Draw RS - flip flop & explain truth table
13. What are the types of Shift Registers?
14. Define Counters.
15. State Dynamic Hazards
16. Define Races
17. Define Synchronous & Asynchronous Sequential logic circuits
18. Define PAL
19. What are the types of ROM?
20. Differentiate SRAM & DRAM

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

21..a. Explain Minterms and Maxterms in detail. (7)

b. State and Prove De Morgan's Theorem . (7)

(Or)

B. a. Using a K-map,Find the SOP form of $F=\sum m(0,4,8,12,3,7,11,15)+d(5)$ (7)

b. Using a K-map, Find the POS form of $F=\prod M\ (0,4,8,12,3,7,11,15)$ (7)

22.A.i. Minimize the following Boolean function using K-Map and draw the combinational Circuit F(A,B,C,D)= $\sum m(0,1,3,6,10)+\sum d(2,5,12)$ (7)

ii. Minimize the following Boolean function using K-Map and draw the combinational circuit F(A,B,C,D)= $\sum m\ (0,1,4,7,11,13,14,15)+\sum d(2,9)$ (7)

(Or)

B. i. Develop a circuit for each of the following Boolean Expressions using only NAND gates Y=AB(C+D) (7)

ii. Develop a circuit for each of the following Boolean Expressions using only NOR gates Y=AB+A'B+AC' (7)

23. A. i. Design and explain the working of mod-7 counter. (7)

ii Describe in detail about Synchronous sequential Circuit with examples (7)

(Or)

B. i. Describe in detail about Asynchronous sequential Circuit with examples (7)

ii. Draw Truth Table for RS,JK, T, D Flip flops (7)

24. A. i.An asynchronous sequential circuit is described by the following excitation and output function. $B=(A_1'B_2)B+(A_1+B_2)$ C=B

Draw the transition table and output map (7)

ii. An asynchronous sequential circuit is described by the following excitation and output function.

$Y=X_1X_2+(X_1+X_2)Y$

Z=Y

Draw the transition table and output map (7)

(Or)

B. i. Summarize the design procedure for asynchronous sequential circuit (7)

ii. Explain in detail about Fundamental mode sequential circuit (7)

25 . A. Define RAM and explain in detail about Static RAM and Dynamic RAM (14)

(Or)

B. Explain clearly the PAL,PLA and PLD (14)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014 DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING DIGITAL ELECTRONICS

Time: 3 hours **Maximum: 100 marks**

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. (a) 11101*101 (b)1111*11
2. Convert the following binary numbers into Excess 3 code: 1010 and 0110
3. Convert the following numbers into Decimal numbers: $(A28)_{16}$ and $(752)_8$
4. Convert (1011011)binary to (?)gray
5. State De-Morgans Law
6. Sketch the neat block diagram of 1 X 8 De-multiplexer.
7. Sketch the neat block diagram of 3 X 8 Decoder
8. Draw the EX-OR gate and derive the truthtable.
9. What are universal gates? Why it is called so.
10. Draw JK - Flip flop & explain truth table
11. Draw D - Flip flop & explain truth table
12. Explain master slave flip flop
13. What are the types of Shift Registers?
14. Define Counters.
15. Define cycles
16. Explain Primitive state flow table
17. State static hazards
18. What is PLA?
19. Classify the types of Memory
20. Define EAPROM

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16.a. i. Explain Minterms and Maxterms in detail (7)

ii. Perform BCD addition on the following terms (4+3)

```
3 5 4
5 6 3  +
-----------
-----------
```

Perform BCD Subtraction on the following terms 7 5 8

5 6 3 __

(Or)

B.i. Obtain the canonical product of sum form of the function (A,B,C,D)=(A+B)C(A+D) (7)

ii. Obtain the canonical product of sum form of the function (W,X,Y,Z)=(W+X)Y(W+Z’) (7)

17. A. i. Design BCD adder and Explain (7)

ii. Explain multiplexers and de multiplexers with truth table. (7)

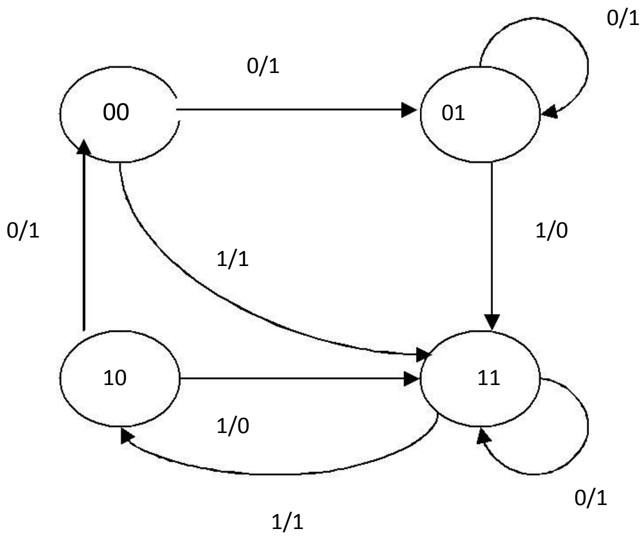
(Or)

B. i. Design 4 bit Adder using Full adder circuit (7)

ii. Design a 2 bit Comparator (7)

18.A.i.Explain Up Down counter in detail (7)

ii. Design the sequential circuit using suitable flip flops for the following state diagram (7)



(Or)

B. i. Design a 4 bit synchronous counter with ripple carry and loop a head carry (14)

19.A. i.Explain in detail about Pulse Mode Sequential circuit (7)

ii.Explain Races and Cycles in detail (7)

(Or)

B. i. Explain in detail about asynchronous sequential circuit. (7)

ii. Explain One hot state assignment in detail (7)

20.A. i. Implement the Boolean functions using PLA

F_1 (A,B,C)= $\sum$m(0,1,3,5); F_2 (A,B,C)=$\sum$m(0,3,5,7) (7)

ii. Implement the Boolean functions using PLA

F_1 (A,B,C)= $\sum$m(0,1,3,4,5) ; F_2 (A,B,C)=$\sum$m(0,3,5) (7)

(Or)

B. Explain the different types of ROM in detail. (14)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours **Maximum: 100 marks**

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Convert $(111101)_2$ to Gray and $(111111)_{GRAY}$ to binary
2. Perform 2's complement 10110101
3. Convert $(1011011)_{binary}$ to $(?)_{gray}$
4. Write example expression for PoS & SoP.
5. Draw the block diagram of 8 X 3 encoder
6. What is combinational logic circuit?
7. What are universal gates? Why it is called so.
8. What is full Adder? Define it
9. Explain Sequential circuit with block diagram
10. Define Counters
11. Draw 1x4 De-MUX block diagram
12. Draw MOD-6 counter diagram
13. What are the types of Shift Registers?
14. Draw RS - flip flop & explain truth table
15. List out the steps for Fundamental mode sequential circuit
16. Explain output mapping
17. What is synchronous sequential logic circuits
18. Write Short note on FPGA
19. Define PLA
20. List out the applications of memories

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16. a.i) Convert the following binary numbers into Gray code (4+3)

a.11110001110
b.01110011011

ii) Convert the following Gray codes into Binary code a.1001011

b.1001110

b.i) Convert the following binary numbers into Excess 3 code (4+3)

a. 1010

b. 0110

ii) Perform BCD addition on the following terms

4 5 2
2 4 1 +

(Or)

B. a. Using a K-map,Find the SOP form of F=∑m(0,4,8,12,3,7,11,15)+d(5) (7)

b. Using a K-map, Find the POS form of F=∏M (0,4,8,12,3,7,11,15) (7)

17. A. i. Design Excess 3 code to BCD converter (7)

ii. Design BCD to Excess 3 code converter (7)

(Or)

B. i. Design a full Adder circuit (7)

ii. Draw the circuit for 3 to 8 decoder and explain (7)

18.A.Explain in detail about Moore model (14)

(Or)

B. Write a short note on (1) Ripple counter (5)

(2)Ring counter (5)

(3)Johnson counter (4)

19.A. i. An asynchronous sequential circuit is described by the excitation and output functions $Y= X_1 X_2$ '+ $(X_1 + X_2$ ') Y Z = Y

Draw the logic diagram of the circuit and derive the transition table (7)

ii.An asynchronous sequential circuit is described by the excitation and output functions $Y= X_1'X_2+(X_1'+X_2) Y$ Z = Y

Draw the logic diagram of the circuit and derive the transition table (7)

(Or)

B.. i. An asynchronous sequential circuit has two internal states and one output. The excitation and output function describing the circuit are as follows. (14)

$Y= X_1 X_2+ X_1Y_2+X_2Y_1$: $Y_2=X_2+X_1Y_1Y_2+X_1Y_1$: $Z=X_2+Y_1$

a) Draw the logic diagram of the circuit b) Derive the transition table

20.A.i. A seven bit Hamming code is received as 1111101.What is the correct code? (7)

ii. List the types of programmable logic devices (7)

(Or)

B. i. Explain TTL cell with diagram (7)

ii. Explain MOS with diagram (7)

Prepared by: Bhavani.S
Designation: Professor

Department: ECE

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours **Maximum: 100 marks**

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Define Don’t care condition
2. Define K-map?
3. Explain ASCII
4. Sketch the neat block diagram. 4 – bit binary parallel adder.
5. Draw logic diagram for Y= (A-B)+(C-D)
6. Write the step by step design procedure
7. What is Propagation Delay?
8. What are universal gates? Why it is called so.
9. Explain state diagram
10. Write the classifications of shift registers
11. Define Counters
12. Draw 1x4 De-MUX block diagram
13. Draw MOD-6 counter diagram
14. What is PLA?
15. What is state reduction and assignment?
16. Draw Asynchronous sequential circuit block diagram?
17. Give example for Transition table
18. Draw the PLA structure
19. What is ROM?
20. Write Associative and Distributive Laws

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16.A. i. Perform 1’s complement Subtraction on the following terms (4+3)

```
1 1 1 0 1 1
1 0 0 0 0 1     __
-----------
-----------
```

72

ii. Perform 2’s compliment Subtraction on the following terms

1 0 0 1 1 0
1 0 0 0 0 0 __

----

iii.State and Prove De Morgan’s Theorem (7)

(Or)

B. i.Convert the following numbers into decimal number (4+3)

a. $(1110001)_2$

b. $(5637)_8$

ii.Convert the following numbers into Binary number (4+3)

c. $(565)_8$

d. $(AB1)_{16}$

17.A.. i. Design half adder and Half Subtractor. (7)

ii. Draw block diagram for 4 bit parallel Adder/Subtractor using full adder (7)

(Or)

B. i. Design Excess 3 code to BCD converter (7)

ii. Draw the circuit for 3 to 8 decoder and explain (7)

18.A. Explain in detail about Mealy model (14)

(Or)

B. i. Draw Logic diagram and Truth Table for RS,JK, T, D Flip flops (14)

19.A. An asynchronous sequential circuit has two internal states and one output. The excitation and output function describing the circuit are as follows.

$Y=X_1X_2'+X_1Y_2+X_2Y_1$

$Y_2=X_2+X_1Y_1Y_2+X_1Y_1$

$Z=X_2+Y_1$

Draw the logic diagram of the circuit (14)

(Or)

B. i. Explain Races and Cycles in detail (7)

ii. Explain in detail about Fundamental mode sequential circuit (7)

20.A. i. Using PROM realize the following expressions (7)

$F_1\,(A,B,C)= \sum m(0,2,5,7)$

$F_2(A,B,C)= \sum m(1,4,6)$

ii. Implement the Following function using PLA (7)

$A(X,Y,Z)= \sum m(1,2,4,5,6)$

$B(X,Y,Z)= \sum m(0,2,4)$

$C(X,Y,Z)= \sum m(1,7)$

(Or)

B. i. Design a BCD to Excess 3 code converter and implement using PLA (7)

ii. Design a Excess 3 code to BCD code converter and implement using PLA (7)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours Maximum: 100 marks

PART – A (15 x 2 = 30 Marks)
Answer any FIFTEEN Question only

1. Convert $(185)_{10}$ to BCD & binary code
2. Minimize x (x’+y)
3. Define Canonical form
5. Convert (185) to BCD & binary code
4. Convert 011011)$_{binary}$ to (?)$_{gray}$
6. Minimize x (x’+y)
7. What is Propagation Delay?
8. What are universal gates? Why it is called so.
9. Define Canonical form
10. Define SIPO
11. What is HDL?
12. Define Moore & Mealy model?
13. Write short note on pulse mode circuits
14. What are the types of Shift Registers?
15. Define Counters.
16. How Hazards eliminated?
17. Define excitation table
18. Define PAL
19. What are the types of ROM?
20. Differentiate SRAM & DRAM

PART - B (5 x 14 = 70 Marks)
Answer ALL the questions

16. A. i.Obtain the canonical sum of product form of the function

74

(A,B,C,D)=A+BC+D (7)

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

(A,B,C,D)=(A+B)C(A+D) (7)

(Or)

B. i. Convert the hexadecimal number 4BC into decimal number. (4)

ii. Simplify the Boolean Function F(X,Y,Z)= XY+X'Z+YZ. (5)

iii. Simplify the Boolean Function F(X,Y)= XY+X'Y+X'Y'+Y'X. (5)

17. A. i. Design a 2 bit Comparator Design (7)

ii. Explain multiplexers and de multiplexers with truth table. (7)

(Or)

B. i. Design 4 bit Adder using Full adder circuit (7)

ii. Design BCD adder and Explain (7)

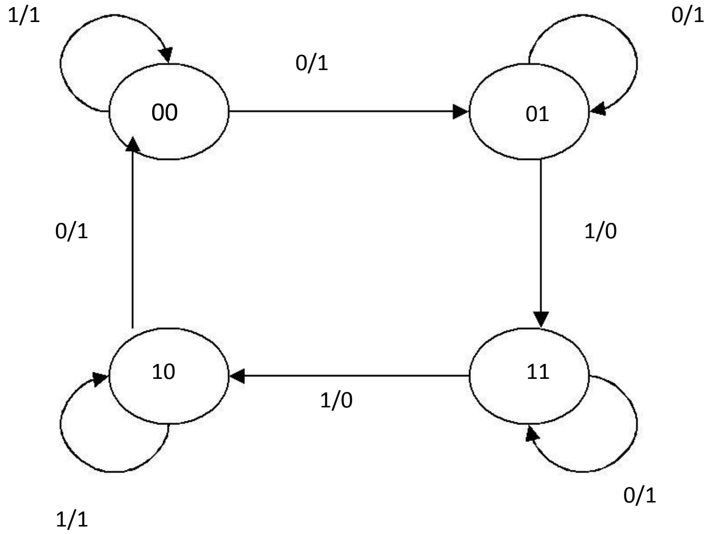
18.A. i. Design and explain the working of mod-7 counter. (7)

ii. Design and explain the working of mod-6 counter. (7)

(Or)

B. i. Describe in detail about Asynchronous sequential Circuit with examples (7)

ii.Draw the state table for the following sequential circuit and derive the expression.(7)



19. A. i. . i. Explain in detail about Pulse Mode Sequential circuit (7)

ii.An asynchronous sequential circuit is described by the excitation and output (7)

functions $Y= X_1'X_2+(X_1'+X_2) Y$ $Z = Y$

Draw the logic diagram of the circuit and derive the transition table

(Or)

B. i. Explain in detail about asynchronous sequential circuit. (7)

ii. Explain Hazards in detail (7)

20.A. Define RAM and explain in detail about Static RAM and Dynamic RAM (14)

(Or)

B.i. Explain the different types of ROM in detail (14)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Karpagam Academy of Higher Education

(Established under Section 3 of UGC Act 1956)

Eachanari, Coimbatore-641 021. INDIA

Department of Electronics and Communication Engineering
Faculty of Engineering

DIGITAL ELECTRONICS

LECTURE NOTES

PREPARED BY
Dr.S.Bhavani,HOD/ECE

14BEEC403 DIGITAL ELECTRONICS 3 0 0 3 100

INTENDED OUTCOMES:

- To introduce number systems and codes
- To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
- To introduce the methods for simplifying Boolean expressions
- To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
- To introduce the concept of memories and programmable logic devices.

UNIT-I NUMBER SYSTEMS

Binary, Octal, Decimal, Hexadecimal-Number base conversions – complements – signed Binary numbers. Binary Arithmetic- Binary codes: Weighted –BCD-2421-Gray code-Excess 3 code-ASCII –Error detecting code – conversion from one code to another-Boolean postulates and laws –De-Morgan's Theorem- Principle of Duality- Boolean expression – Boolean function-Minimization of Boolean expressions – Sum of Products (SOP) –Product of Sums (POS)-
Minterm- Maxterm- Canonical forms – Conversion between canonical forms –Karnaugh map Minimization – Don't care conditions.

UNIT-II LOGIC GATES AND COMBINATIONAL CIRCUITS LOGIC GATES

AND, OR, NOT, NAND, NOR, Exclusive – OR and Exclusive – NOR- Implementations of Logic Functions using gates, NAND –NOR implementations –Multi level gate implementations-Multi output gate implementations. TTL and CMOS Logic and their characteristics –Tristate gates.

COMBINATIONAL CIRCUITS: Design procedure – Adders-Subtractors – Serial adder/ Subtractor - Parallel adder/ Subtractor- Carry look ahead adder- BCD adder- Magnitude Comparator- Multiplexer/ Demultiplexer- encoder / decoder – parity checker – code converters. Implementation of combinational logic using MUX, ROM, PAL and PLA.

UNIT-III SEQUENTIAL CIRCUIT

Flip flops SR, JK, T, D and Master slave – Characteristic table and equation –Application table – Edge triggering –Level Triggering –Realization of one flip flop using other flip flops – Asynchronous / Ripple counters – Synchronous counters –Modulo – n counter –Classification of sequential circuits – Moore and Mealy -Design of Synchronous counters: state diagram- State table –State minimization –State assignment- ASM-Excitation table and maps-Circuit implementation - Register – shift registers- Universal shift register – Shift counters – Ring counters.

UNIT-IV ASYNCHRONOUS SEQUENTIAL CIRCUITS

Design of fundamental mode and pulse mode circuits – primitive state / flow table – Minimization of primitive state table –state assignment – Excitation table – Excitation map-cycles – Races –Hazards: Static –Dynamic –Essential –Hazards elimination.

UNIT-V MEMORY DEVICES

Classification of memories –RAM organization – Write operation –Read operation – Memory cycle - Timing wave forms – Memory decoding – memory expansion – Static RAM Cell-Bipolar RAM cell – MOSFET RAM cell –Dynamic RAM cell –ROM organization - PROM –EPROM – EEPROM –EAPROM –Programmable Logic Devices –Programmable Logic Array (PLA)-Programmable Array Logic (PAL)-Field Programmable Gate Arrays (FPGA).

TEXT BOOKS:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Morris Mano.M	Digital Design	Prentice Hall of India Pvt. Ltd., New Delhi	2003
2	John M .Yarbrough	Digital Logic Applications and Design	Thomson- Vikas publishing house, New Delhi	2002

REFERENCES:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Salivahanan.S and Arivazhagan.S	Digital Circuits and Design	Vikas Publishing House Pvt. Ltd, New Delhi	2004
2	Charles H.Roth	Fundamentals of Logic Design	Thomson Publication Company, New Delhi.	2003
3	Donald P.Leach and Albert Paul Malvino	Digital Principles and Applications	Tata McGraw Hill Publishing Company Limited, New Delhi	2003
4	Jain.R.P	Modern Digital Electronics	Tata McGraw–Hill publishing company limited, New Delhi	2003
5	Thomas L. Floyd	Digital Fundamentals	Pearson Education, New Delhi	2003

WEBSITES:

http://www.allaboutcircuits.com/vol_2/chpt_9/2.html
http://www.educypedia.be/electronics/digital.html

UNIT-V MEMORY DEVICES

5. MEMORY

Memory is a storage A memory for which bits can both be easily stored or retrieved (\written to" or \read from"). Here is a rundown on some terms:

RAM. In general, refers to random access memory. All of the devices we are considering to be \memories" (RAM, ROM, etc.) are random access. The term RAM has also come to mean memory which can be both easily written to and read from. There are two main technologies used for RAM:

1.) Static RAM. These essentially are arrays of flip-flops. They can be fabricated in ICs as large arrays of tint flip-flops.) \SRAM" is intrisically somewhat faster than dynamic RAM.

2.) Dynamic RAM. Uses capacitor arrays. Charge put on a capacitor will produce a HIGH bit if its voltage $V = Q=C$ exceeds the threshold for the logic standard in use. Since the charge will \leak" o through the resistance of the connections in times of order 1 msec, the stored information must be continuously refreshed (hence the term \dynamic"). Dynamic RAM can be fabricated with more bits per unit area in an IC than static RAM. Hence, it is usually the technology of choice for most large-scale IC memories.

ROM. *Read-only memory*. Information cannot be easily stored. The idea is that bits are initially defined and are never changed thereafter. As an example, it is generally prudent for the instructions used to initialize a computer upon initial power-up to be stored in ROM. The following terms refer to versions of ROM for which the stored bits *can* be over-written, but not easily.

PROM. *Programmable* ROM. Bits can be set on a programming bench by burning \fusible links," or equivalent. This technology is also used for programmable array logic (PALs), which we will briefly discuss in class.

EPROM. ROM which can be erased using ultraviolet light.

EEPROM. ROM which can be erased electronically.

A few other points of terminology:

As you know, a *bit* is a binary digit. It represents the smallest element of information.

A *byte* is 8 bits.

A \K" of memory is 2^{10} = 1024 bits (sometimes written KB). And a megabit (MB) is 1K 1K bits.

RAM is organized into many data \words" of some prescribed length. For example, a RAM which has 8*K* = 8192 memory locations, with each location storing a data word of \width" 16 bits, would be referred to as a RAM of size 8*K* 16. The total storage capacity of this memory would therefore be 128KB, or simply a \128K" memory. (With modern *very large scale integration* (VLSI) technology, a typical RAM IC might be 16 MB. Besides the memory \size," the other important specification for memory is the *access time*. This is the time delay between when a valid request for stored data is sent to a memory and when the corresponding bit of data appears at the output. A typical access time, depending upon the technology of the memory, might be 10 ns.

5.**1Memory Configuration**

As stated above, the term \memory" refers to a particular way of organizing information | by random access | which is distinct from the less specific term \data storage." Figure 36 shows how an 8-bit RAM (8 1) is organized. (This is a very small memory, but illustrates the concepts.) Our RAM consists of three main components: an 8-bit multiplexer, an 8-bit demultiplexer, and 8 bits of storage. The storage shown consists of edge-triggered D-type flip-flops. Hence, this is evidently a \static RAM." (There is no fundamental reason for using edge-triggered flip-flops. They could just as easily be level-triggered, like the simple \clocked" S-R flip-flop of Fig. 14.)

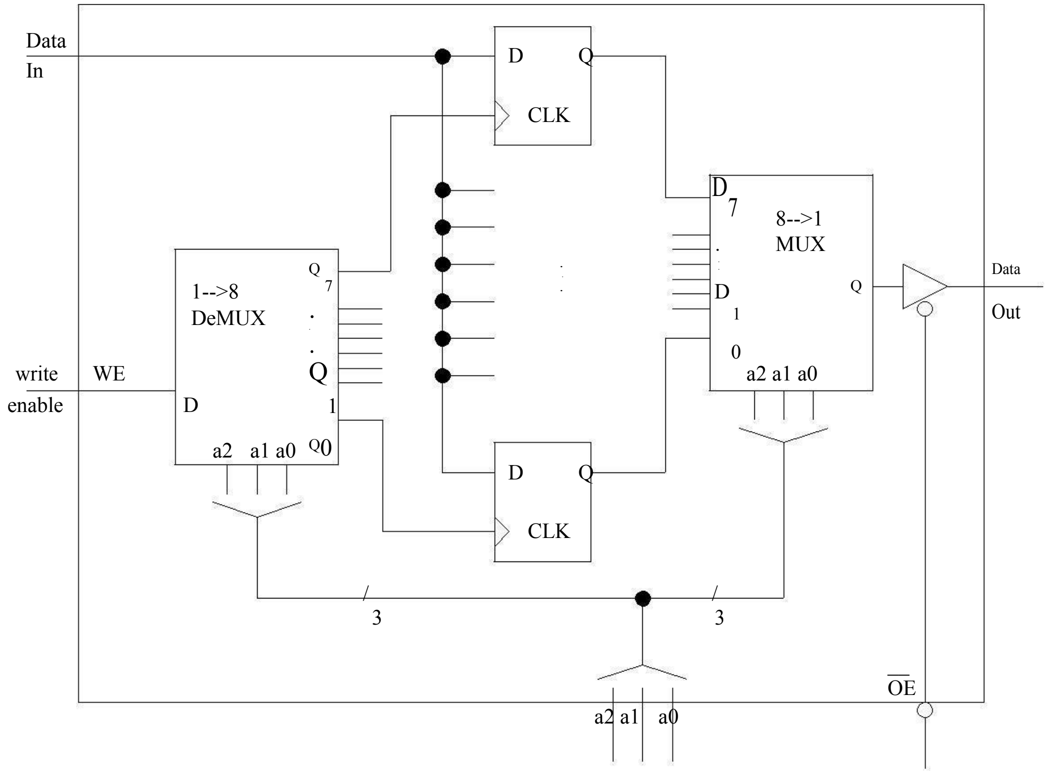


Figure 36: An 8   1 bit RAM.

Our example RAM has 6 external connections which are inputs (data in, write enable (WE), 3-state enable (OE), and 3 address bits ($A = a_2a_1a_0$), and has one output connection (data out), giving 7 external connections total, plus 2 for power/ground. To write information to the RAM, one would supply a valid address, for example A = 101. The data bit to be written to location 101 is to appear at the data input as either a logic HIGH or LOW signal. And to enable the writing into this bit, the WE signal must be asserted. This then appears at the Q_5 output of the demultiplexer, and is passed on to the appropriate flip-flop, which stores the input data bit and passes it on to the Q_5 multiplexer input.

To read data from our RAM, one asserts an address, so that the selected bit is sent to the MUX output and then the 3-state buffer. The purpose of the 3-state buffer is to ensure that no digital outputs are directly connected together, for example if our RAM output were connected to a data \bus," which in turn was connected to several other devices. Recall that the 3-state devices have outputs which are effectively disconnected if there is no enable signal. So if the output data connection of our RAM is connected to a data bus, then the OE signal must be coordinated with any other outputs also connected to the data bus. When it is OK to read data from the RAM (all other output devices are disconnected from the bus), the OE signal is asserted and the MUX output will appear at the RAM output.

One could of course also store the 8 bits of data directly to an 8-bit data register, rather than using the RAM configuration outlined above. In this case, the number of external connections is 17 (8 data in, 8 data out, and 1 clock), compared with the 7 of our RAM. For a more realistic case where the number of bits of memory n is much larger than our example, we generalize the above to arrive at 4 + $\log_2(n)$ external connections for the RAM, compared with 1 + 2n for the standalone register. Obviously for large n, the register is impractical, whereas the RAM remains reasonable. Actually, it is even somewhat better than this for the RAM case, since the number of external connections does not grow with the width of the stored data words. Hence, a RAM of size 1K 16 = 16 KB requires only 14 connections. This is to be compared with 32,001 connections for the register. Note that the RAM can only supply one bit at a time to the output. This may seem like a handicap, but is actually well matched to standard microprocessors.

5.2 A State Machine with Memory

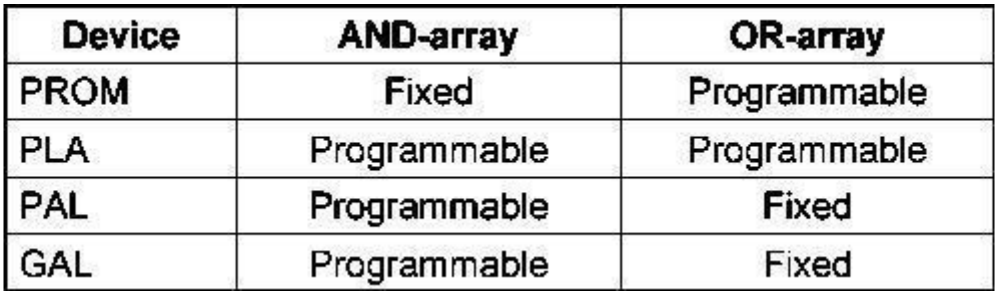
For reference, our usual state machine configuration is shown again in Fig. 37. Now we consider the use of a memory with a state machine, as depicted in Fig. 38. A random access memory is used in place of the usual combinational logic. (A ROM has been specified, to emphasize that we are not changing the memory | once it is defined initially, it is only read from. The memory is used to conveniently encode the connection between present and next states.

To start with, let's assume a state machine with no external inputs or outputs. Then the state machine's present state (PS) becomes an *address* which is input to the ROM. The *data word* stored in the ROM at that address then corresponds to the next state (NS). This correspondence had been initially programmed into the ROM, just as the specific combina-tional logic in our old state machine had to be pre-determined. So if the PS as defined by the *Q* bits at the data register are, for example, 1001, then the ROM data word at address 1001 will be the NS which is then passed back to the register. When there are also external inputs, as there will be for most anything of interest, these are combined with the PS bits to form a longer address for the ROM. Similarly, any external outputs are combined with the NS bits in the data word.

5.3 PROGRAMMABLE LOGIC DEVICES (PLD'S)

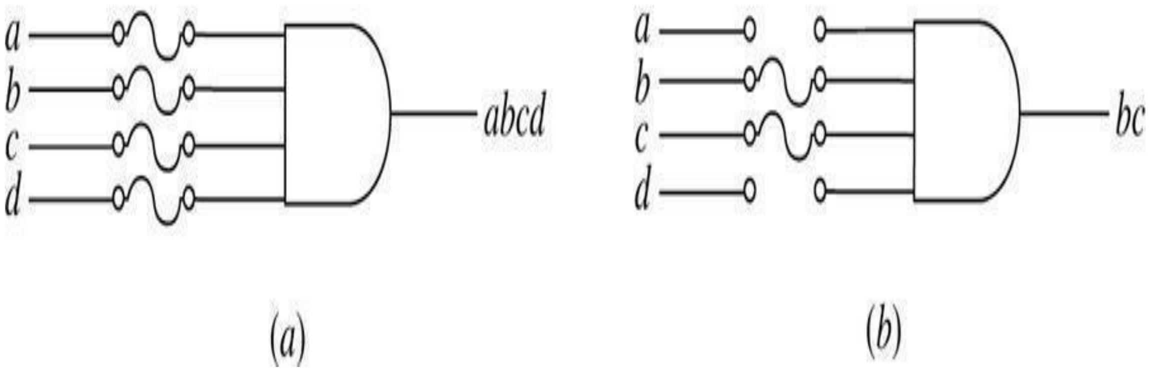
Programmable devices have their functionality programmed before they are first used.
Range in complexity from 100's to 10,000's of logic gates.

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



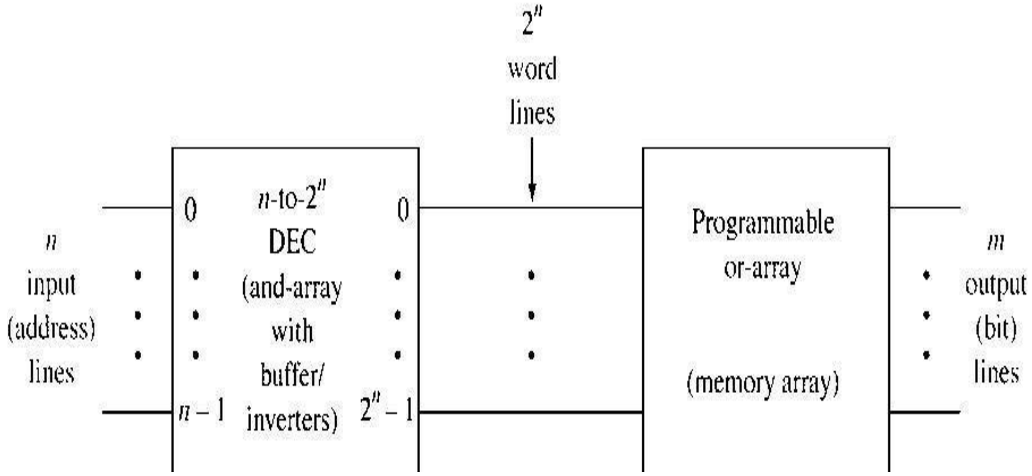
Device	AND-array	OR-array
PROM	Fixed	Programmable
PLA	Programmable	Programmable
PAL	Programmable	Fixed
GAL	Programmable	Fixed

Programming by blowing fuses.

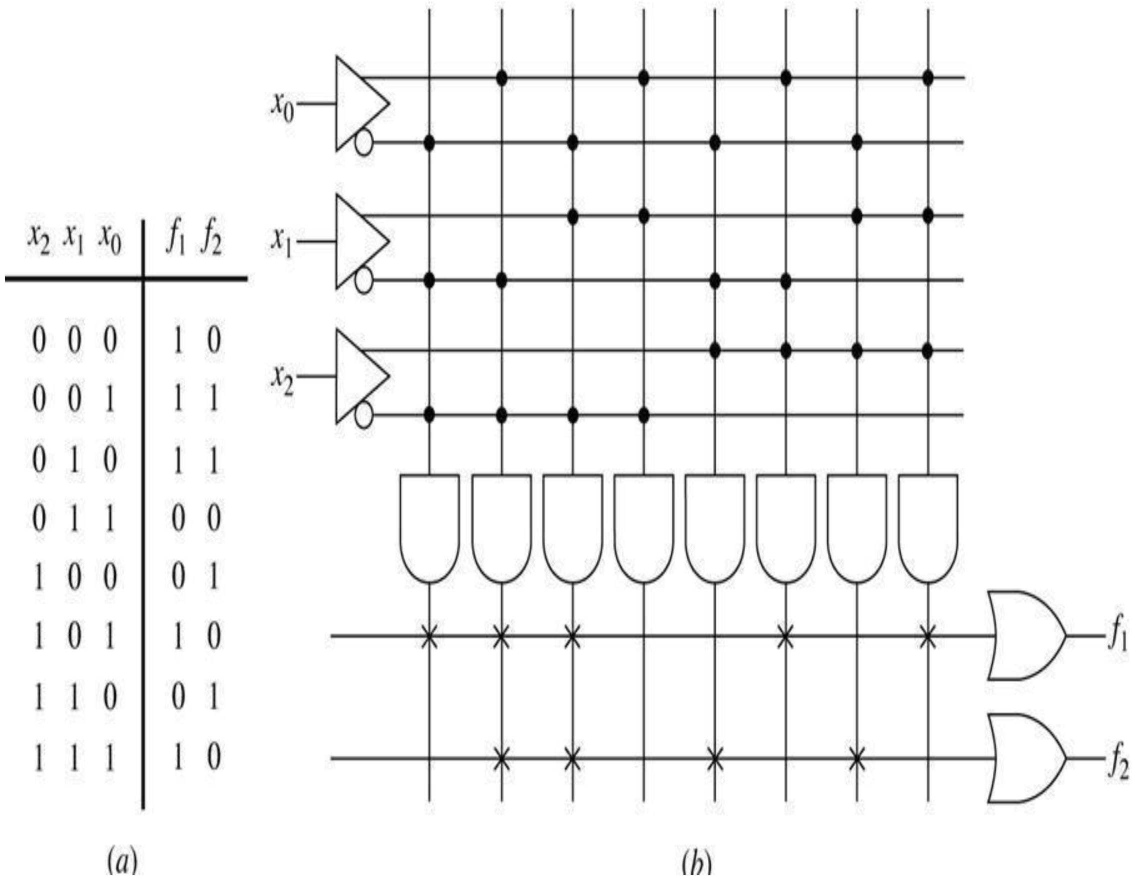


(*a*) Before programming. **(*b*) After Programming.**

PROM Notation

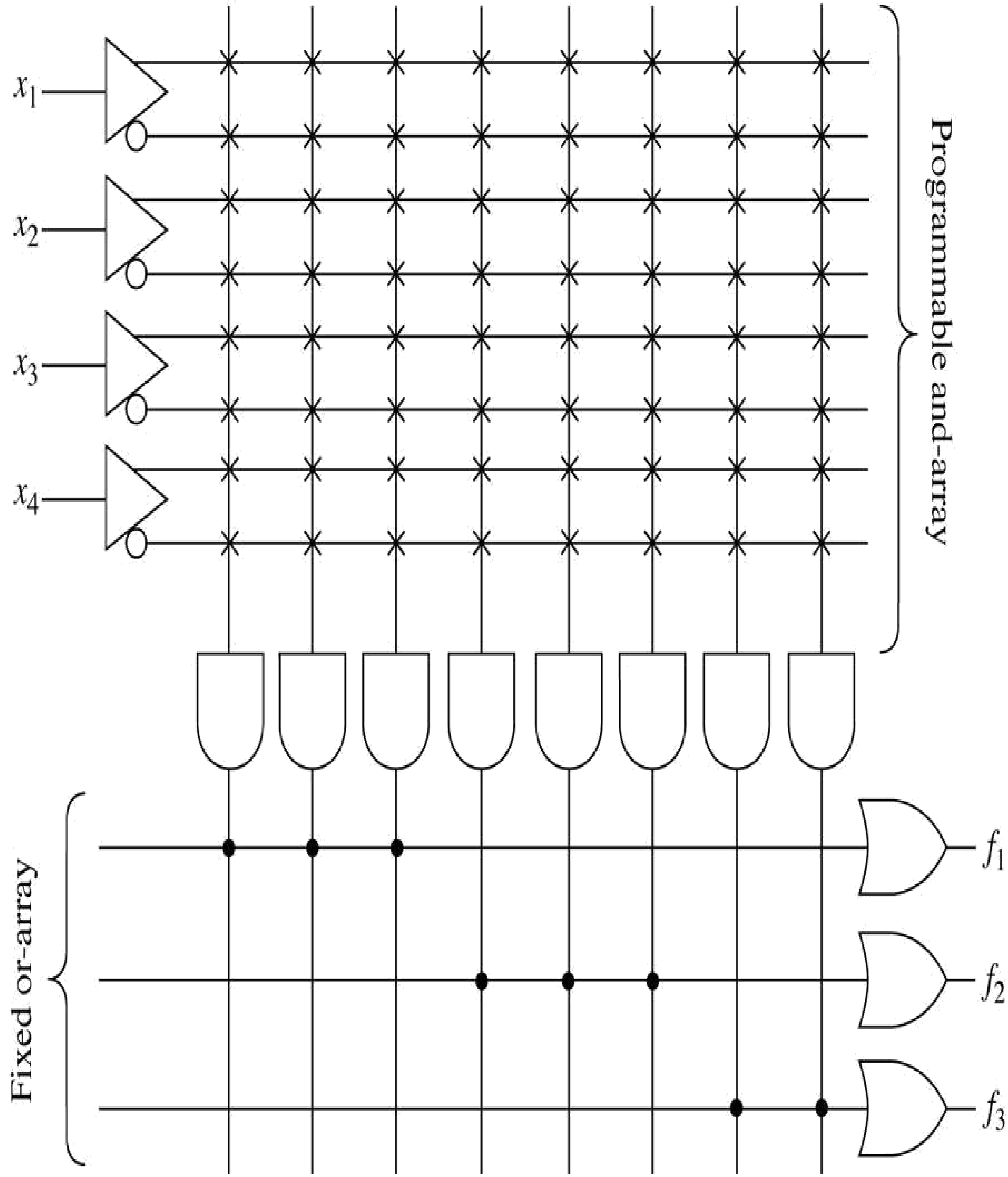


Using a PROM for logic design

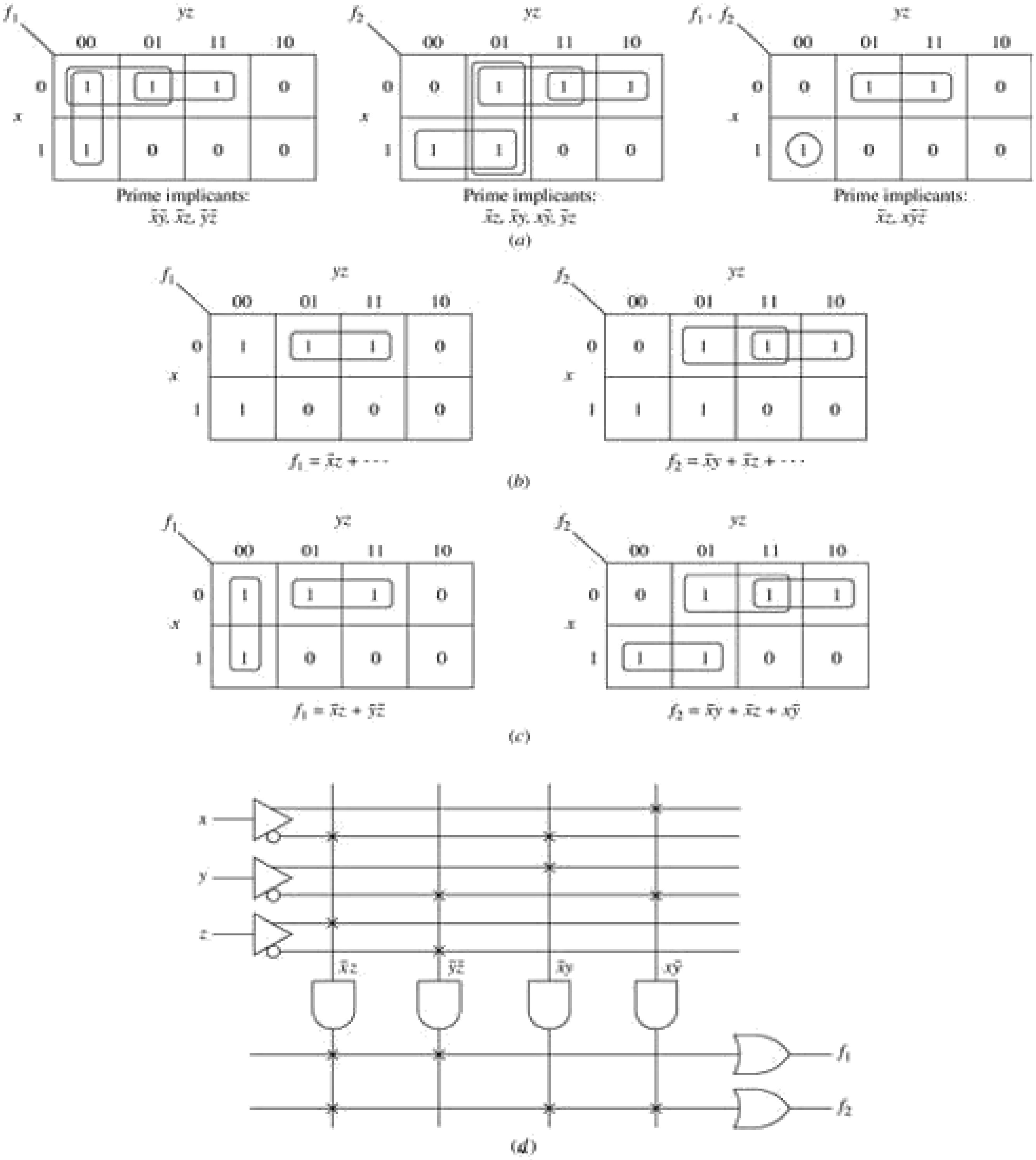


x_2	x_1	x_0	f_1	f_2
0	0	0	1	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

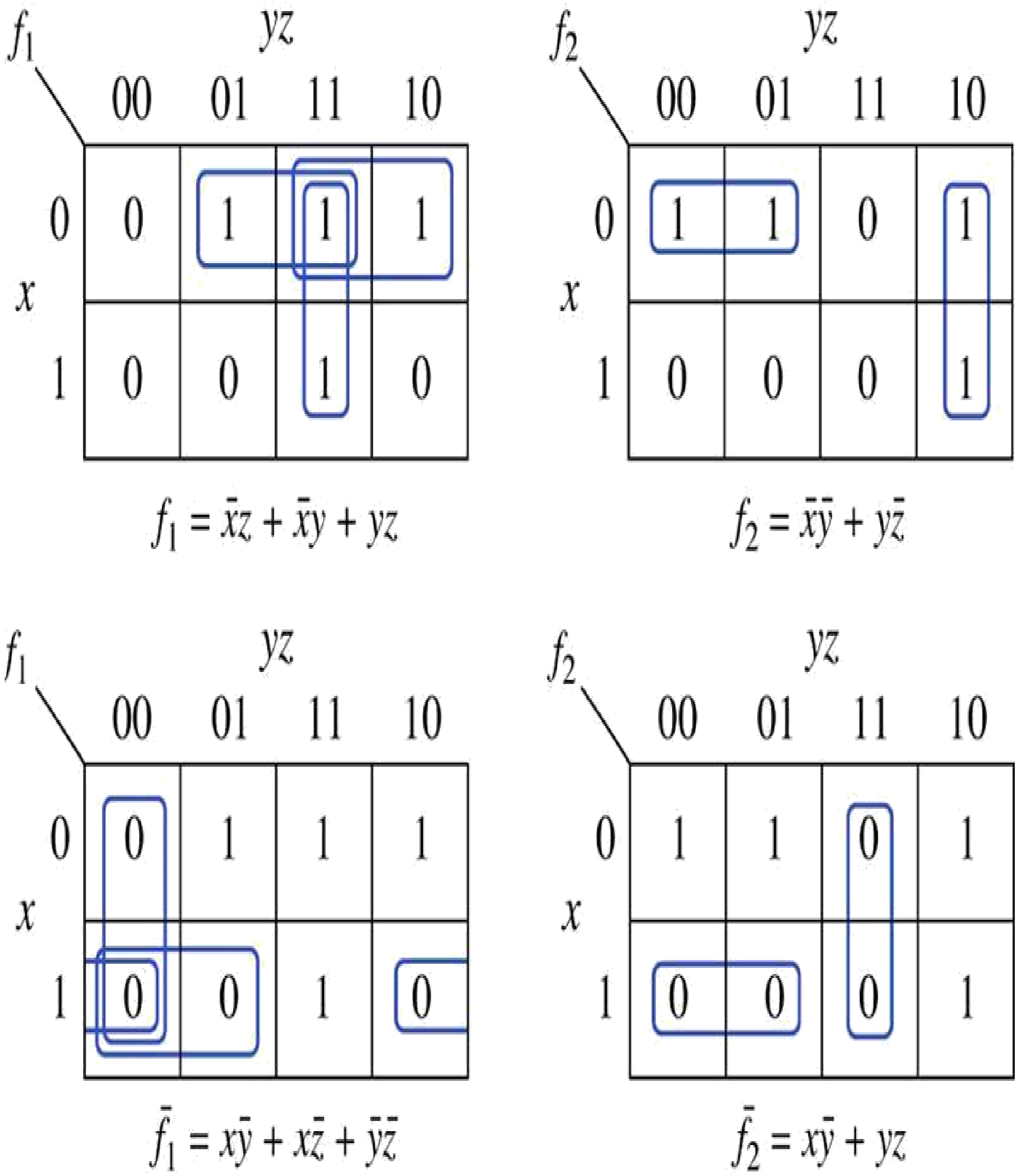
A simple four-input, three-output PAL device.



Example of combinational logic design using a PLA.

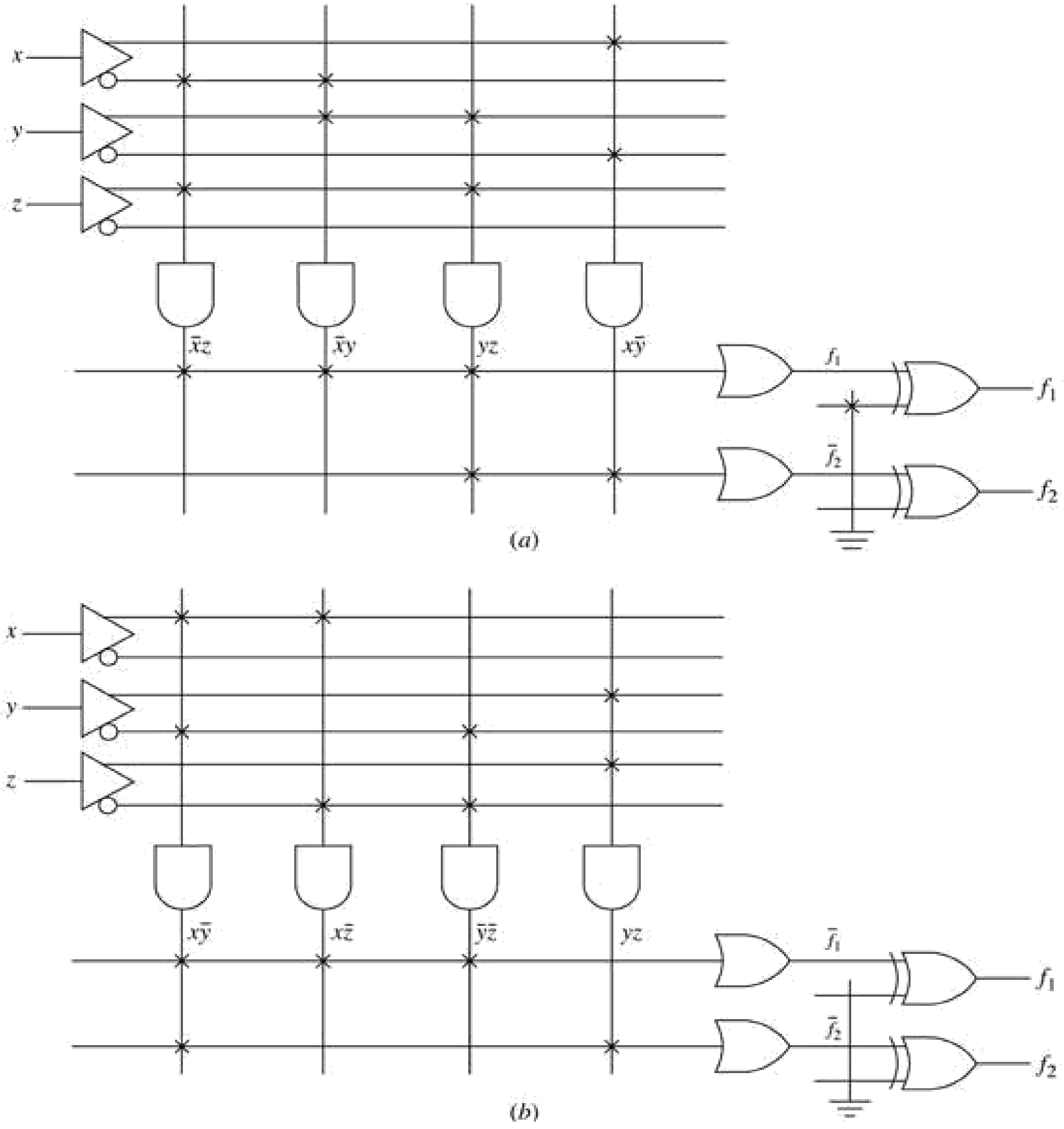

f_1
yz
00 01 11 10
x
0
1
f_2
$f_1 \cdot f_2$
Prime implicants:
$\bar{x}\bar{y}$, $\bar{x}z$, $\bar{y}\bar{z}$
Prime implicants:
$\bar{x}z$, $\bar{x}y$, $x\bar{y}$, $\bar{y}z$
Prime implicants:
$\bar{x}z$, $x\bar{y}\bar{z}$
(a)
$f_1 = \bar{x}z + \cdots$
$f_2 = \bar{x}y + \bar{x}z + \cdots$
(b)
$f_1 = \bar{x}z + \bar{y}\bar{z}$
$f_2 = \bar{x}y + \bar{x}z + x\bar{y}$
(c)
x
y
z
$\bar{x}z$
$\bar{y}\bar{z}$
$\bar{x}y$
$x\bar{y}$
f_1
f_2
(d)

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



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

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


x
y
z
$\bar{x}z$
$\bar{x}y$
yz
$x\bar{y}$
f_1
$\bar{f}_2$
f_1
f_2
x
y
z
$x\bar{y}$
$x\bar{z}$
$\bar{y}\bar{z}$
yz
$\bar{f}_1$
$\bar{f}_2$
f_1
f_2

(a)

(b)

End Semester Quetion paper

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – Regular

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours Maximum: 100 marks

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Convert $(237)_{10}$ to Hexadecimal $(?)_{16}$ and Octal $(?)_8$
2. Convert $(1011011)_{binary}$ to $(?)_{gray}$
3. State De-Morgans Law
4. Write Associative and Distributive Laws
5. Draw OR gate and derive the truth table.
6. What is Propagation Delay?
7. What are universal gates? Why it is called so.
8. Define Half Adder and Draw the circuit
9. Differentiate Combinational and Sequential logic circuits.
10. Draw 1x4 De-MUX block diagram
11. Draw T - flip flop & explain truth table
12. Draw RS - flip flop & explain truth table
13. What are the types of Shift Registers?
14. Define Counters.
15. State Dynamic Hazards
16. Define Races
17. Define Synchronous & Asynchronous Sequential logic circuits
18. Define PAL
19. What are the types of ROM?
20. Differentiate SRAM & DRAM

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

21..a. Explain Minterms and Maxterms in detail. (7)

b. State and Prove De Morgan's Theorem . (7)

(Or)

B. a. Using a K-map,Find the SOP form of $F=\sum m(0,4,8,12,3,7,11,15)+d(5)$ (7)

b. Using a K-map, Find the POS form of $F=\prod M\ (0,4,8,12,3,7,11,15)$ (7)

22.A.i. Minimize the following Boolean function using K-Map and draw the combinational Circuit F(A,B,C,D)= ∑m(0,1,3,6,10)+ ∑d(2,5,12) (7)

ii. Minimize the following Boolean function using K-Map and draw the combinational circuit F(A,B,C,D)= ∑m (0,1,4,7,11,13,14,15)+ ∑d(2,9) (7)

(Or)

B. i. Develop a circuit for each of the following Boolean Expressions using only NAND gates Y=AB(C+D) (7)

ii. Develop a circuit for each of the following Boolean Expressions using only NOR gates Y=AB+A'B+AC' (7)

23. A. i. Design and explain the working of mod-7 counter. (7)

ii Describe in detail about Synchronous sequential Circuit with examples (7)

(Or)

B. i. Describe in detail about Asynchronous sequential Circuit with examples (7)

ii. Draw Truth Table for RS,JK, T, D Flip flops (7)

24. A. i.An asynchronous sequential circuit is described by the following excitation and output function. $B=(A_1'B_2)B+(A_1+B_2)$ C=B

Draw the transition table and output map (7)

ii. An asynchronous sequential circuit is described by the following excitation and output function.

$Y=X_1X_2+(X_1+X_2)Y$

Z=Y

Draw the transition table and output map (7)

(Or)

B. i. Summarize the design procedure for asynchronous sequential circuit (7)

ii. Explain in detail about Fundamental mode sequential circuit (7)

25 . A. Define RAM and explain in detail about Static RAM and Dynamic RAM (14)

(Or)

B. Explain clearly the PAL,PLA and PLD (14)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014 DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING DIGITAL ELECTRONICS

Time: 3 hours **Maximum: 100 marks**

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. (a) 11101*101 (b)1111*11
2. Convert the following binary numbers into Excess 3 code: 1010 and 0110
3. Convert the following numbers into Decimal numbers: $(A28)_{16}$ and $(752)_8$
4. Convert (1011011)binary to (?)gray
5. State De-Morgans Law
6. Sketch the neat block diagram of 1 X 8 De-multiplexer.
7. Sketch the neat block diagram of 3 X 8 Decoder
8. Draw the EX-OR gate and derive the truthtable.
9. What are universal gates? Why it is called so.
10. Draw JK - Flip flop & explain truth table
11. Draw D - Flip flop & explain truth table
12. Explain master slave flip flop
13. What are the types of Shift Registers?
14. Define Counters.
15. Define cycles
16. Explain Primitive state flow table
17. State static hazards
18. What is PLA?
19. Classify the types of Memory
20. Define EAPROM

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16.a. i. Explain Minterms and Maxterms in detail (7)

ii. Perform BCD addition on the following terms (4+3)

```
3 5 4
5 6 3 +
-----------
-----------
```

Perform BCD Subtraction on the following terms 7 5 8

5 6 3 __

(Or)

B.i. Obtain the canonical product of sum form of the function (A,B,C,D)=(A+B)C(A+D) (7)

ii. Obtain the canonical product of sum form of the function (W,X,Y,Z)=(W+X)Y(W+Z’) (7)

17. A. i. Design BCD adder and Explain (7)

ii. Explain multiplexers and de multiplexers with truth table. (7)

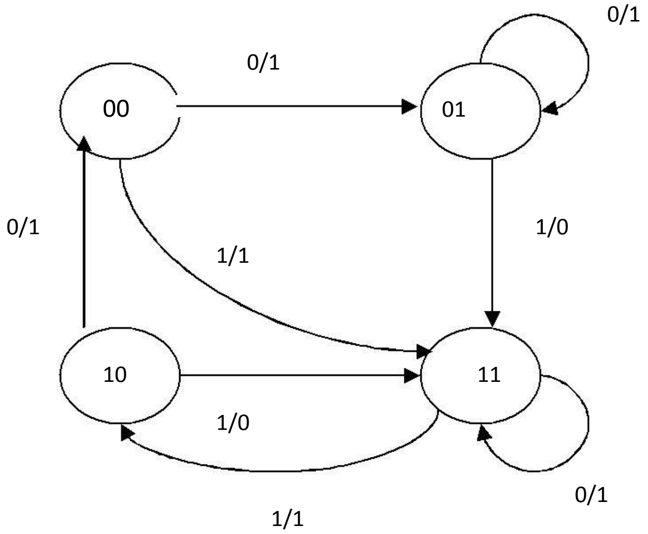
(Or)

B. i. Design 4 bit Adder using Full adder circuit (7)

ii. Design a 2 bit Comparator (7)

18.A.i.Explain Up Down counter in detail (7)

ii. Design the sequential circuit using suitable flip flops for the following state diagram (7)



(Or)

B. i. Design a 4 bit synchronous counter with ripple carry and loop a head carry (14)

19.A. i.Explain in detail about Pulse Mode Sequential circuit (7)

ii.Explain Races and Cycles in detail (7)

(Or)

B. i. Explain in detail about asynchronous sequential circuit. (7)

ii. Explain One hot state assignment in detail (7)

20.A. i. Implement the Boolean functions using PLA

F_1 (A,B,C)= ∑m(0,1,3,5); F_2 (A,B,C)=∑m(0,3,5,7) (7)

ii. Implement the Boolean functions using PLA

F_1 (A,B,C)= ∑m(0,1,3,4,5) ; F_2 (A,B,C)=∑m(0,3,5) (7)

(Or)

B. Explain the different types of ROM in detail. (14)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Reg. No…………………….

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours **Maximum: 100 marks**

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Convert $(111101)_2$ to Gray and $(111111)_{GRAY}$ to binary
2. Perform 2's complement 10110101
3. Convert $(1011011)_{binary}$ to $(?)_{gray}$
4. Write example expression for PoS & SoP.
5. Draw the block diagram of 8 X 3 encoder
6. What is combinational logic circuit?
7. What are universal gates? Why it is called so.
8. What is full Adder? Define it
9. Explain Sequential circuit with block diagram
10. Define Counters
11. Draw 1x4 De-MUX block diagram
12. Draw MOD-6 counter diagram
13. What are the types of Shift Registers?
14. Draw RS - flip flop & explain truth table
15. List out the steps for Fundamental mode sequential circuit
16. Explain output mapping
17. What is synchronous sequential logic circuits
18. Write Short note on FPGA
19. Define PLA
20. List out the applications of memories

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16. a.i) Convert the following binary numbers into Gray code (4+3)

a.11110001110

b.01110011011

ii) Convert the following Gray codes into Binary code

a.1001011
b.1001110

b.i) Convert the following binary numbers into Excess 3 code (4+3)

a. 1010
b. 0110

ii) Perform BCD addition on the following terms

4 5 2
2 4 1 +

(Or)

B. a. Using a K-map,Find the SOP form of $F=\sum m(0,4,8,12,3,7,11,15)+d(5)$ (7)

b. Using a K-map, Find the POS form of $F=\prod M\ (0,4,8,12,3,7,11,15)$ (7)

17. A. i. Design Excess 3 code to BCD converter (7)

ii. Design BCD to Excess 3 code converter (7)

(Or)

B. i. Design a full Adder circuit (7)

ii. Draw the circuit for 3 to 8 decoder and explain (7)

18.A.Explain in detail about Moore model (14)

(Or)

B. Write a short note on (1) Ripple counter (5)
(2)Ring counter (5)
(3)Johnson counter (4)

19.A. i. An asynchronous sequential circuit is described by the excitation and output functions $Y= X_1 X_2 '+ (X_1 + X_2 ') Y$ $Z = Y$

Draw the logic diagram of the circuit and derive the transition table (7)

ii.An asynchronous sequential circuit is described by the excitation and output functions $Y= X_1'X_2+(X_1'+X_2) Y$ $Z = Y$

Draw the logic diagram of the circuit and derive the transition table (7)

(Or)

B.. i. An asynchronous sequential circuit has two internal states and one output. The excitation and output function describing the circuit are as follows. (14)

$Y= X_1 X_2+ X_1Y_2+X_2Y_1$; $Y_2=X_2+X_1Y_1Y_2+X_1Y_1$; $Z=X_2+Y_1$

a) Draw the logic diagram of the circuit b) Derive the transition table

20.A.i. A seven bit Hamming code is received as 1111101.What is the correct code? (7)

ii. List the types of programmable logic devices (7)

(Or)

B. i. Explain TTL cell with diagram (7)

ii. Explain MOS with diagram (7)

Prepared by: Bhavani.S
Designation: Professor

Department: ECE

Reg. No……………………

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours **Maximum: 100 marks**

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Define Don't care condition
2. Define K-map?
3. Explain ASCII
4. Sketch the neat block diagram. 4 – bit binary parallel adder.
5. Draw logic diagram for Y= (A-B)+(C-D)
6. Write the step by step design procedure
7. What is Propagation Delay?
8. What are universal gates? Why it is called so.
9. Explain state diagram
10. Write the classifications of shift registers
11. Define Counters
12. Draw 1x4 De-MUX block diagram
13. Draw MOD-6 counter diagram
14. What is PLA?
15. What is state reduction and assignment?
16. Draw Asynchronous sequential circuit block diagram?
17. Give example for Transition table
18. Draw the PLA structure
19. What is ROM?
20. Write Associative and Distributive Laws

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16.A. i. Perform 1's complement Subtraction on the following terms (4+3)

```
1 1 1 0 1 1
1 0 0 0 0 1    __
-----------
-----------
```

ii. Perform 2's compliment Subtraction on the following terms

1 0 0 1 1 0
1 0 0 0 0 0 __

----

iii.State and Prove De Morgan's Theorem (7)

(Or)

B. i.Convert the following numbers into decimal number (4+3)

a. $(1110001)_2$

b. $(5637)_8$

ii.Convert the following numbers into Binary number (4+3)

c. $(565)_8$

d. $(AB1)_{16}$

17.A.. i. Design half adder and Half Subtractor. (7)

ii. Draw block diagram for 4 bit parallel Adder/Subtractor using full adder (7)

(Or)

B. i. Design Excess 3 code to BCD converter (7)

ii. Draw the circuit for 3 to 8 decoder and explain (7)

18.A. Explain in detail about Mealy model (14)

(Or)

B. i. Draw Logic diagram and Truth Table for RS,JK, T, D Flip flops (14)

19.A. An asynchronous sequential circuit has two internal states and one output. The excitation and output function describing the circuit are as follows.

$Y=X_1X_2'+X_1Y_2+X_2Y_1$

$Y_2=X_2+X_1Y_1Y_2+X_1Y_1$

$Z=X_2+Y_1$

Draw the logic diagram of the circuit (14)

(Or)

B. i. Explain Races and Cycles in detail (7)

ii. Explain in detail about Fundamental mode sequential circuit (7)

20.A. i. Using PROM realize the following expressions (7)

F_1 (A,B,C)= ∑m(0,2,5,7)

F_2(A,B,C)= ∑m(1,4,6)

ii. Implement the Following function using PLA (7)

A(X,Y,Z)= ∑m(1,2,4,5,6)

B(X,Y,Z)= ∑m(0,2,4)

C(X,Y,Z)= ∑m(1,7)

(Or)

B. i. Design a BCD to Excess 3 code converter and implement using PLA (7)

ii. Design a Excess 3 code to BCD code converter and implement using PLA (7)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE

Reg. No……………………..

Subject Code: 13BEEC203

KARPAGAM UNIVERSITY

(Under section 3 of UGC Act 1956)

COIMBATORE - 641 021.

(For the candidate admitted from 2013 onwards) – PART TIME

BE DEGREE EXAMINATIONS, MAY 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DIGITAL ELECTRONICS

Time: 3 hours Maximum: 100 marks

PART – A (15 x 2 = 30 Marks)

Answer any FIFTEEN Question only

1. Convert $(185)_{10}$ to BCD & binary code
2. Minimize x (x’+y)
3. Define Canonical form
5. Convert (185) to BCD & binary code
4. Convert 011011)$_{binary}$ to (?)$_{gray}$
6. Minimize x (x’+y)
7. What is Propagation Delay?
8. What are universal gates? Why it is called so.
9. Define Canonical form
10. Define SIPO
11. What is HDL?
12. Define Moore & Mealy model?
13. Write short note on pulse mode circuits
14. What are the types of Shift Registers?
15. Define Counters.
16. How Hazards eliminated?
17. Define excitation table
18. Define PAL
19. What are the types of ROM?
20. Differentiate SRAM & DRAM

PART - B (5 x 14 = 70 Marks)

Answer ALL the questions

16. A. i.Obtain the canonical sum of product form of the function

(A,B,C,D)=A+BC+D (7)

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

(A,B,C,D)=(A+B)C(A+D) (7)

(Or)

B. i. Convert the hexadecimal number 4BC into decimal number. (4)

ii. Simplify the Boolean Function F(X,Y,Z)= XY+X’Z+YZ. (5)

iii. Simplify the Boolean Function F(X,Y)= XY+X’Y+X’Y’+Y’X. (5)

17. A. i. Design a 2 bit Comparator Design (7)

ii. Explain multiplexers and de multiplexers with truth table. (7)

(Or)

B. i. Design 4 bit Adder using Full adder circuit (7)

ii. Design BCD adder and Explain (7)

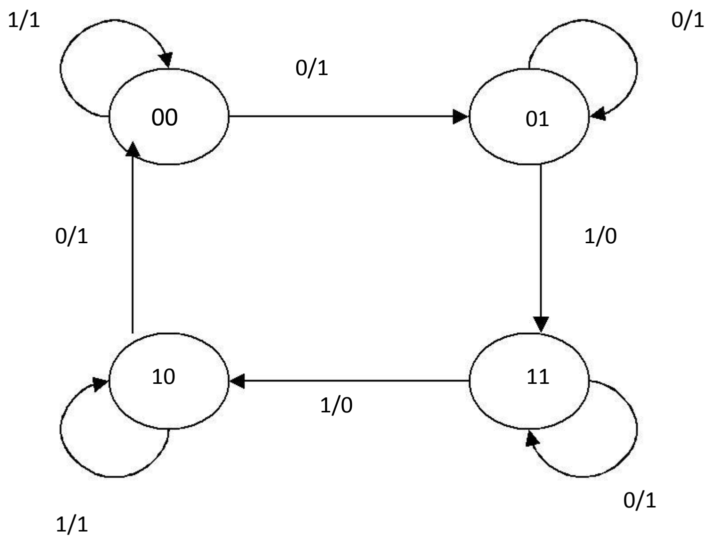
18.A. i. Design and explain the working of mod-7 counter. (7)

ii. Design and explain the working of mod-6 counter. (7)

(Or)

B. i. Describe in detail about Asynchronous sequential Circuit with examples (7)

ii.Draw the state table for the following sequential circuit and derive the expression.(7)



19. A. i. . i. Explain in detail about Pulse Mode Sequential circuit (7)

ii.An asynchronous sequential circuit is described by the excitation and output (7)

functions $Y= X_1'X_2+(X_1'+X_2) Y$ $Z = Y$

Draw the logic diagram of the circuit and derive the transition table

(Or)

B. i. Explain in detail about asynchronous sequential circuit. (7)

ii. Explain Hazards in detail (7)

20.A. Define RAM and explain in detail about Static RAM and Dynamic RAM (14)

(Or)

B.i. Explain the different types of ROM in detail (14)

Prepared by: Bhavani.S
Designation: Professor
Department: ECE