# Govt Polytechnic Darbhanga

# Semester IV

# Dig Elect µP

# Chapter 1 Notes

# sub Code (161802)

*By Prof. D N Singh*

*CSE Department*

*DR. B K Roy*

*Principal*

*Govt. Polytechnic Darbhanga*

# MODULE-I

## NUMBER SYSTEMS

**Many number systems are in use in digital technology. The most common are the decimal,binary, octal, and hexadecimal systems. The decimal system is clearly the most familiar to usbecause it is a tool that we use every day. Examining some of its characteristics will help us tobetter understand the other systems. In the next few pages we shall introduce four numericalrepresentation systems that are used in the digital system. There are other systems, which we**

**will look at briefly.**

- **Decimal**
- **Binary**
- **Octal**
- **Hexadecimal**

## Decimal System

**The decimal system is composed of 10 numerals or symbols. These 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Using these symbols as digits of a number, we can express any quantity. The decimal system is also called the base-10 system because it has 10 digits.**

**Decimal Examples**

$3.14_{10}$
$52_{10}$
$1024_{10}$
$64000_{10}$

GOVT POLYTECHNIC DARBHANGA

## Binary System

**In the binary system, there are only two symbols or possible digit values, 0 and 1. This base-2 system canbe used to represent any quantity that can be represented in decimal or other base system.In digital systems the information that is being processed is usually presented in binary form. Binaryquantities can be represented by any device that has only two operating states or possible conditions.**

**E.g..a switch is only open or closed. We arbitrarily (as we define them) let an open switch represent binary 0 and a closed switch represent binary 1. Thus we can represent any binary number by using series of switches.**

## Octal System

**The octal number system has a base of eight, meaning that it has eight possible digits: 0,1,2,3,4,5,6,7.**

**octal to Decimal Conversion**

$237_8 = 2 \times (8_2) + 3 \times (8_1) + 7 \times (8_0) = 159_{10}$
$24.6_8 = 2 \times (8_1) + 4 \times (8_0) + 6 \times (8_{-1}) = 20.75_{10}$
$11.1_8 = 1 \times (8_1) + 1 \times (8_0) + 1 \times (8_{-1}) = 9.125_{10}$
$12.3_8 = 1 \times (8_1) + 2 \times (8_0) + 3 \times (8_{-1}) = 10.375_{10}$

## Hexadecimal System

**The hexadecimal system uses base 16. Thus, it has 16 possible digit symbols. It uses the digits 0 through 9 plus the letters A, B, C, D, E, and F as the 16 digit symbols.**

**Hexadecimal to Decimal Conversion**

$24.6_{16} = 2 \times (16_1) + 4 \times (16_0) + 6 \times (16_{-1}) = 36.375_{10}$
$11.1_{16} = 1 \times (16_1) + 1 \times (16_0) + 1 \times (16_{-1}) = 17.0625_{10}$
$12.3_{16} = 1 \times (16_1) + 2 \times (16_0) + 3 \times (16_{-1}) = 18.1875_{10}$

GOVT POLYTECHNIC DARBHANGA

## Code Conversion

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

## Binary-To-Decimal Conversion

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

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

## Octal-To-Binary Conversion

**Each Octal digit is represented by three binary digits.**

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

## Octal-To-Hexadecimal Hexadecimal-To-Octal Conversion

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

GOVT POLYTECHNIC DARBHANGA

## Binary Codes

**Binary codes are codes which are represented in binary system with modification from the original ones. Below we will be seeingthe following:**

- **Weighted Binary Systems**
- **Non Weighted Codes**

## Weighted Binary Systems

**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.**

## 8421 Code/BCD Code

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

**Example: The bit assignment 1001, can be seen by its weights to represent the decimal 9 because:**

**1x8+0x4+0x2+1x1 = 9**

## 2421 Code

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

## 5211 Code

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

## Reflective Code

GOVT POLYTECHNIC DARBHANGA

**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.**

## 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).**

## 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 are 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 Graycode has got a special place.**

## Error Detecting and Correction 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

**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.**

GOVT POLYTECHNIC DARBHANGA

**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.**

**There are 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.**

**ECC (Error correcting codes) are used also in memories, networking, Hard disk, CDROM, DVD etc. Normally in networking chips (ASIC), we have 2 Error detection bits and 1 Error correction bit.**

## Hamming Code

**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 bits errors and cannot distinguish between 1-bit and 2-bits inconsistencies. It can't – in general – detect 3(or more)-bits errors The idea is that the failed bit position in an n-bit string (which we'll call X) can be represented in binary with log2(n) bits, hence we'll try to get it adding just log2(n) bits.**

GOVT POLYTECHNIC DARBHANGA

## ASCII Code

**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 27 = 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.**

## BOOLEAN ALGEBRA AND LOGIC GATES

**The English mathematician George Boole (1815-1864) sought to give symbolic formto Aristotle‗s system of logic. Boole wrote a treatise on the subject in 1854, titled AnInvestigation of the Laws of Thought, on Which Are Founded the Mathematical Theories ofLogic and Probabilities, which codified several rules of relationship between mathematicalquantities limited to one of two possible values: true or false, 1 or 0. His mathematicalsystembecame known as Boolean algebra. All arithmetic operations performed with Booleanquantities have but one of two possibleOutcomes: either 1 or 0. There is no such thing as ‖2‖ or ‖-1‖ or ‖1/2‖ in the Boolean world.**

**It is a world in which all other possibilities are invalid by fiat. As one might guess, this is notthe kind of math you want to use when balancing a check book or calculating current througha resistor.**

**However, Claude Shannon of MIT fame recognized how Boolean algebra could be applied toon-and-off circuits, where all signals are characterized as either ‖high‖ (1) or ‖low‖ (0).His1938 thesis, titled A Symbolic Analysis of Relay and Switching Circuits, put Boole‗stheoretical work to use in a way Boole never could have imagined, giving us a powerfulmathematical tool for designing and analyzing digital circuits.**

**Like ‖normal‖ algebra, Boolean algebra uses alphabetical letters to denote variables.Unlike ‖normal‖ algebra, though, Boolean variables are always CAPITAL letters, neverlowercase.**

**Because they are allowed to possess only one of two possible values, either 1 or 0, each andevery variable has a complement: the opposite of its value. For example, if variable ‖A‖ has avalue of 0, then the complement of A has a value of 1. Boolean notation uses a bar above thevariable character to denote complementation, like this:**

GOVT POLYTECHNIC DARBHANGA

If: $A=0$

Then: $\overline{A}=1$

If: $A=1$

Then: $\overline{A}=0$

**In written form, the complement of ‖A‖ denoted as ‖A-not‖ or ‖A-bar‖. Sometimes a ‖prime‖symbol is used to represent complementation. For example, A‘ would be the complement ofA, much the same as using a prime symbol to denote differentiation in calculus rather thanthe fractional notation dot. Usually, though, the ‖bar‖ symbol finds more wide spread use thanthe ‖prime‖ symbol, for reasons that will become more apparent laterin this chapter.**

## Boolean Arithmetic:

**Let us begin our exploration of Boolean algebra by adding numbers together:**

**0 + 0 = 0**

**0 + 1 = 1**

**1 + 0 = 1**

**1 + 1 = 1**

**The first three sums make perfect sense to anyone familiar with elementary addition. TheLast sum, though, is quite possibly responsible for more confusion than any other singlestatement in digital electronics, because it seems to run contrary to the basic principles ofmathematics.Well, it does contradict principles of addition for real numbers, but not for Boolean numbers.Remember that in the world of Boolean algebra, there are only two possible values for anyquantity and for any arithmetic operation: 1 or 0. There is no such thing as ‖2‖ within thescope of Boolean values. Since the sum ‖1 + 1‖ certainly isn‘t 0, it must be 1 by process ofelimination.**

## Principle of Duality:

GOVT POLYTECHNIC DARBHANGA

**It states that every algebraic expression is deducible from the postulates of Booleanalgebra,and it remains valid if the operators & identity elements are interchanged. If theinputs of a NOR gate are inverted we get a AND equivalent circuit. Similarly when the inputsof a NAND gate are inverted, we get a OR equivalent circuit.This property is called duality.**

## Theorems of Boolean algebra

**The theorems of Boolean algebra can be used to simplify many a complex Booleanexpression and also to transform the given expression into a more useful and meaningful**

**equivalent expression. The theorems are presented as pairs, with the two theorems in a given pair being the dual of each other. These theorems can be very easily verified by the method of 'perfect induction'. According to this method, the validity of the expression is tested for all possible combinations of values of the variables involved. Also, since the validity of thetheorem is based on its being true for all possible combinations of values of variables, there is no reason why a variable cannot be replaced with its complement, or vice versa, without disturbing the validity. Another important point is that, if a given expression is valid, its dual will also be valid.**

**Theorem 1 (Operations with '0' and '1')**

**(a) 0.X = 0 and (b) 1+X= 1**

**Where X is not necessarily a single variable – it could be a term or even a large expression.**

**Theorem 1(a) can be proved by substituting all possible values of X, that is, 0 and 1, into the given expression and checking whether the LHS equals the RHS:**

**• For X = 0, LHS = 0.X = 0.0 = 0 = RHS.**

**• For X= 1, LHS = 0.1 = 0 = RHS.**

**Thus, 0.X =0 irrespective of the value of X, and hence the proof.**

**Theorem 1(b) can be proved in a similar manner. In general, according to theorem 1,**

0. **(Boolean expression) = 0 and 1+ (Boolean expression) =1.**
1. **For example: 0. (A.B+B.C +C.D) = 0 and 1+ (A.B+B.C +C.D) = 1, where A, B and C are**

**Boolean variables.**

GOVT POLYTECHNIC DARBHANGA

**Theorem 2 (Operations with '0' and '1')**

**(a) 1.X = X and (b) 0+X = X**

**where X could be a variable, a term or even a large expression. According to this theorem,ANDing a Boolean expression to '1' orORing '0' to it makes no difference to the expression:**

**• For X = 0, LHS = 1.0 = 0 = RHS.**

**• For X = 1, LHS = 1.1 = 1 = RHS.**

**Also,**

**1. (Boolean expression) = Boolean expression and 0 + (Boolean expression) = Boolean expression.**

**For example,**

**1.(A+B.C +C.D) = 0+(A+B.C +C.D) = A+B.C +C.D**

**Theorem 3 (Idempotent or Identity Laws)**

**(a) X.X.X……X = X and (b) X+X+X +⋯+X = X**

**Theorems 3(a) and (b) are known by the name of idempotent laws, also known as identity laws.**

**Theorem 3(a) is a direct outcome of an AND gate operation, whereas theorem 3(b) representsan OR gate operation when all the inputs of the gate have been tied together. The scope ofidempotent laws can be expanded further by considering X to be a term or an expression. Forexample, let us apply idempotent laws to simplify the following Boolean expression:**

GOVT POLYTECHNIC DARBHANGA

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

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

**Theorem 4 (Complementation Law)**

**(a) X_X = 0 and (b) X+X = 1**

**According to this theorem, in general, any Boolean expression when ANDed to its complement yields a '0' and when ORed to its complement yields a '1', irrespective of the complexity of the expression:**

- For $X=0$, $\overline{X}=1$. Therefore, $X.\overline{X}=0.1=0$.
- For $X=1$, $\overline{X}=0$. Therefore, $X.\overline{X}=1.0=0$.

**Hence, theorem 4(a) is proved. Since theorem 4(b) is the dual of theorem 4(a), its proof is implied.**

**The example below further illustrates the application of complementation laws:**

$$(A+B.C)(\overline{A+B.C})=0 \quad \text{and} \quad (A+B.C)+(\overline{A+B.C})=1$$

**Theorem 5 (Commutative property)**

**Mathematical identity, called a ‖property‖ or a ‖law,‖ describes how differingvariables relate to each other in a system of numbers. One of these properties is known as the commutative property, and it applies equally to addition and multiplication.**

**In essence,the commutative property tells us we can reverse the order of variables that are either addedtogether or multiplied together without changing the truth of the expression:**

**Commutative property of addition**

**A + B = B + A**

**Commutative property of multiplication**

**AB = BA**

GOVT POLYTECHNIC DARBHANGA

**Theorem 6 (Associative Property)**

**The Associative Property,again applying equally well to addition and multiplication.**

**This property tells us we can associate groups of added or multiplied variables together with**

**parentheses without altering the truth of the equations.**

**Associative property of addition**

**A + (B + C) = (A + B) + C**

**Associative property of multiplication**

**A (BC) = (AB) C**

**Theorem 7 (Distributive Property)**

**The Distributive Property, illustrating how to expand a Boolean expression formed by**

**the product of a sum, and in reverse shows us how terms may be factored out of Boolean**

**sums-of-products:**

**Distributive property**

**A (B + C) = AB + AC**

**Theorem 8 (Absorption Law or Redundancy Law)**

**(a) X+X.Y = X and (b) X.(X+Y) = X**

**The proof of absorption law is straightforward:**

**X+X.Y = X. (1+Y) = X.1 = X**

**Theorem 8(b) is the dual of theorem 8(a) and hence stands proved.**

**The crux of this simplification theorem is that, if a smaller term appears in a larger term, then**

**the larger term is redundant. The following examples further illustrate the underlying**

**concept:**

GOVT POLYTECHNIC DARBHANGA

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

and

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

## De-Morgan's First Theorem

**It States that —The complement of the sum of the variables is equal to the product of the complement of each variable . This theorem may be expressed by the following Boolean expression.**

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

## De-Morgan's Second Theorem

**It states that the —Complement of the product of variables is equal to the sum of complementsof each individual variables‖. Boolean expression for this theorem is**

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

GOVT POLYTECHNIC DARBHANGA

**Boolean Function**

Boolean functions are represented in various forms. The two popular forms are *truth tables* and *Venn diagrams*. Truth tables represent functions in a tabular form, while Venn diagrams provide a graphic representation. In addition, there are two algebraic representations known as the *standard* (or *normal*) *form* and the *canonical form*.

Draw the truth table for $Z = AB' + A'C + A'B'C$.

There are three component variables: $A$, $B$, and $C$. Hence, there will be $2^3 = 8$ combinations of values. These eight combinations are shown in the first three columns of Table 2.2. These combinations correspond to binary numbers 000 through 111, or $(0)_{10}$ through $(7)_{10}$.

To evaluate $Z$ in the example function, knowing the values for $A$, $B$, and $C$ at each row of the truth table (Table 2.2), we would first generate values for $A'$ and $B'$ and then generate values for $AB'$, $A'C$ and $A'B'C$ by ANDing the values in the appropriate columns for each row. Finally, we would derive the value of $Z$ by ORing the values in the last three columns for each row. Note that evaluating $A'B'C$ corresponds to ANDing $A'$ and $B'$ values, followed by ANDing the value of $C$. Similarly, if more than two values are to be ORed, they are ORed

Z=AB'+A'C+A'B'C'

| $A$ | $B$ | $C$ | $A'$ | $B'$ | $AB'$ | $A'C$ | $A'B'C$ | $Z$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

## 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.

## Minterms and Maxterms

**A minterm is the product of N distinct literals where each literal occurs exactly Anyboolean 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 :**

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

## Product-of-Sums Expressions

**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 truthtable by considering those input combinations that produce a logic '0' at the output. Eachsuch 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 uncomplemented and complemented variables, unlike sum-of products expressions where '0' and '1' respectively mean complemented and uncomplementedvariables.**

**Since each term in the case of the product-of-sums expression is going to be the sum ofliterals, this implies that it is going to be implemented using an OR operation. Now, an ORgate produces a logic '0' only when all its inputs are in the logic '0' state, which means thatthe first term corresponding to the second row of the truth table will be A+B+C. The productof-sums Boolean expression for this truth table is given by Transforming the given productof-sums expression into an equivalent sum-of-products expression is a straightforwardprocess. Multiplying out the given expression and carrying out the obvious simplificationprovides the equivalent sum-of-products expression:**

**A given sum-of-products expression can be transformed into an equivalent product-of sumsexpression by (a) taking the dual of the given expression, (b) multiplying out differenttermsto 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 productof sums expression of the sum-of products expression**

GOVT POLYTECHNIC DARBHANGA

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

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

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

GOVT POLYTECHNIC DARBHANGA

## Digital Logic Gates

**The basic logic gates are AND, OR, NAND, NOR, XOR, INV, and BUF. The last two are notstandard terms; they stand for ‘inverter’ and ‘buffer’, respectively. The symbols for thesegates and their corresponding Boolean expressions are given in Fig. 2.**

| Name | Expression | Symbol | Negative true symbol |
|---|---|---|---|
| AND | $AB$ | | |
| NAND | $\overline{AB}$ | | |
| OR | $A + B$ | | |
| NOR | $\overline{A + B}$ | | |
| INVERT | $\bar{A}$ | | |
| BUFFER | $A$ | | |
| XOR | $A \oplus B$ | | |
| XNOR | $\overline{A \oplus B}$ | | |

**Figure 2:**

**All of the logical gate functions, as well as the Boolean relations discussed in the next section, follow from the truth tables for the AND and OR gates. We reproduce these below. We also show the XOR truth table, because it comes up quite often, although, as we shall see,it is not elemental.**

GOVT POLYT
ANGA

## Construction of a Karnaugh Map

**An n-variable Karnaugh map has 2n squares, and each possible input is allotted asquare. In the case of a mintermKarnaugh 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 maxtermKarnaugh 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 isnot unique for a given number of variables. The only condition to be satisfied is that thedesignation of adjacent rows and adjacent columns should be the same except for one of theliterals beingcomplemented. Also, the extreme rows and extreme columns are consideredadjacent.Some of the possible designation styles for two-, three- and four-variable mintermKarnaughmaps are shown in the figure below.**

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

**1. Each square containing a '1' must be considered at least once, although it can beconsidered as often as desired.**

**2. The objective should be to account for all the marked squares in the minimum numberof groups.**

**3. 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.**

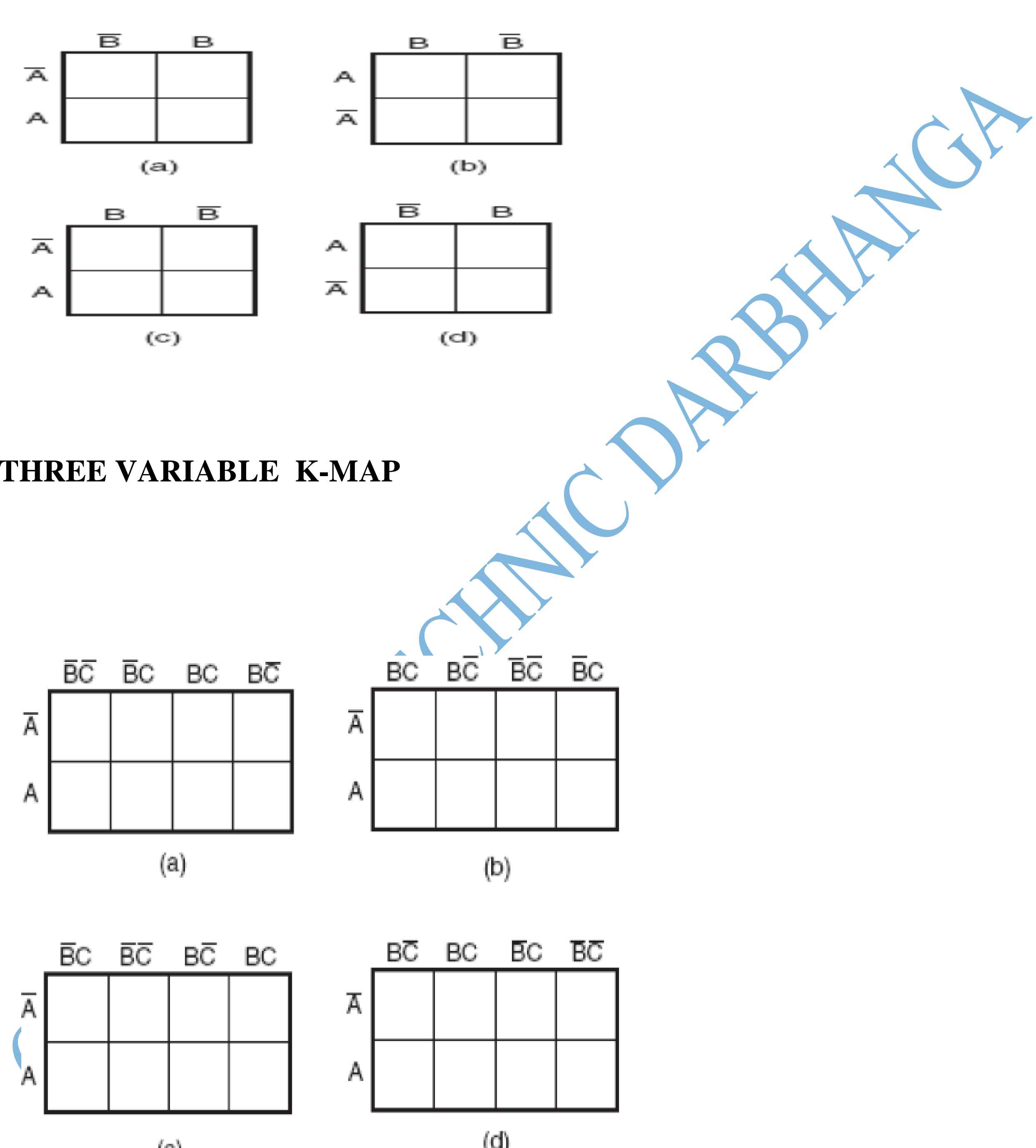
**4. Each group should be as large as possible, which means that a square should not beaccounted for by itself if it can be accounted for by a group of two squares; a group oftwo squares should not be made if the involved squares can be included in a group offour squares and so on.**

**5. '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, notnecessary to**

GOVT POLYTECHNIC DARBHANGA

**account for all 'don't care' entries. Only such entries that can be used toadvantage should be used.**

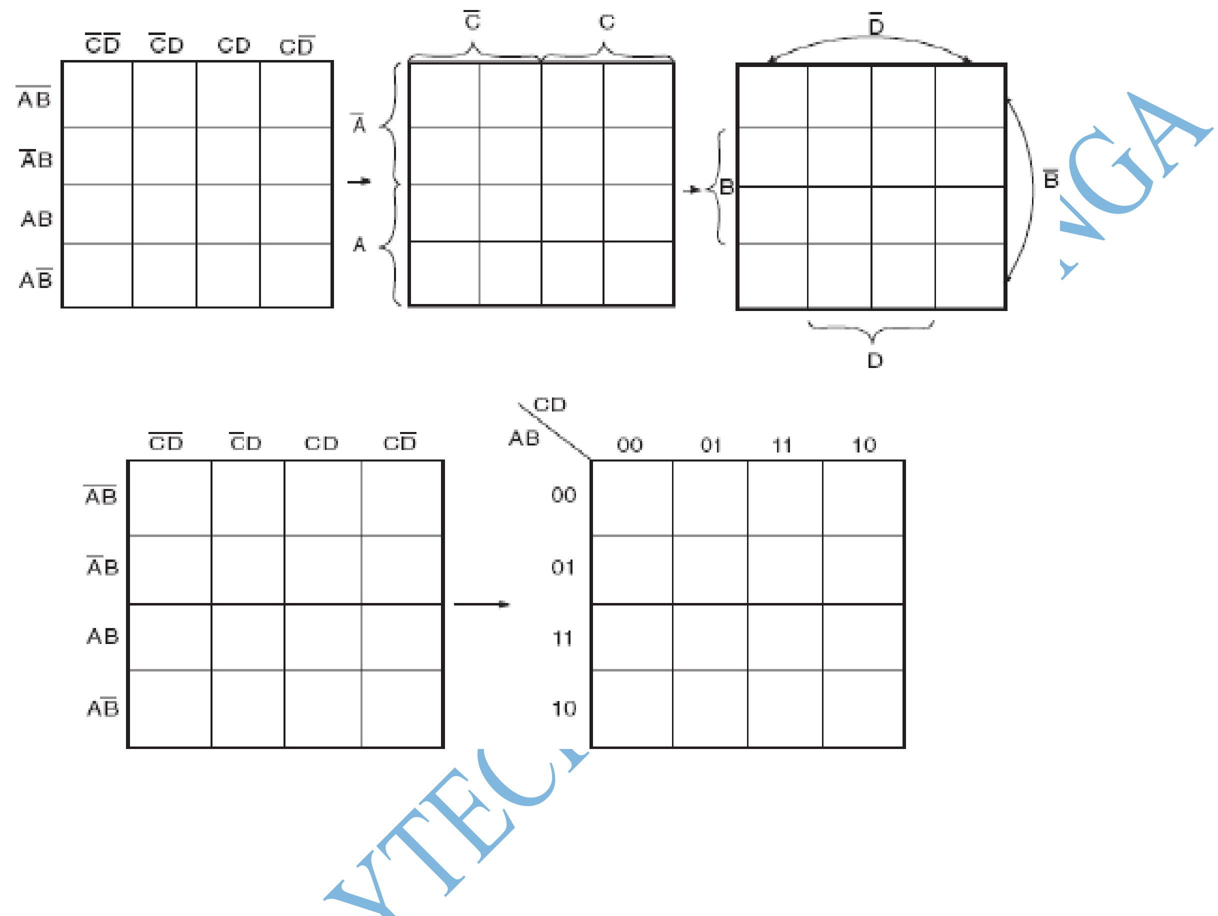
## TWO VARIABLE K-MAP




## THREE VARIABLE K-MAP

CHNIC DARBHANGA

# Four variable K Map




## Different Styles of row and column identification

**Having accounted for groups with all 1s, the minimum 'sum-of-products' or 'product-ofsums'expressions can be written directly from the Karnaugh map. MintermKarnaugh mapand MaxtermKarnaugh map of the Boolean function of a two-input OR gate. The Mintermand Maxterm Boolean expressions for the two-input OR gate are as follows:**

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

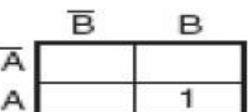
$$Y = \overline{A}.B + A.\overline{B} + A.B \text{ (minterm or sum-of-products)}$$

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




Product-of-sums K-map

**MintermKarnaugh map and MaxtermKarnaugh 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, MintermKarnaugh map and MaxtermKarnaugh 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 maxtermKarnaugh maps for the Boolean functions expressed by the below equations. The minimizedexpressions as deduced from Karnaugh maps in the two cases are given by Equation in thecase of the mintermKarnaugh map and Equation in the case of the maxtermKarnaugh map:**

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

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

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

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

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

Truth table

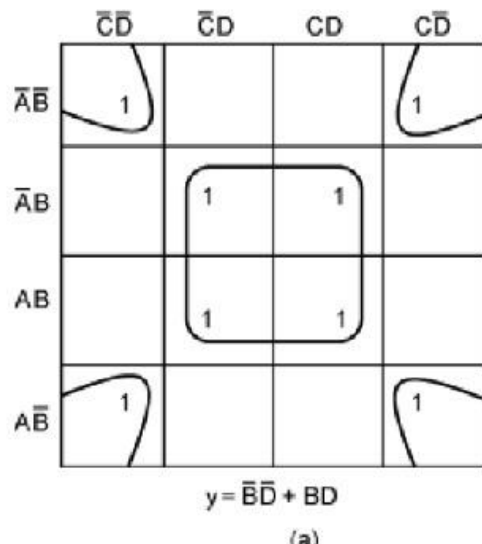
| A | B | C | D | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

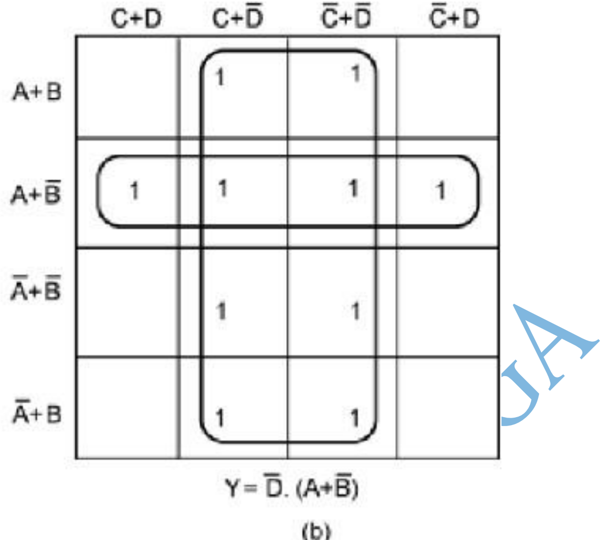
| | $\overline{C}\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\overline{B}$ | 1 | 1 | | |
| $\overline{A}B$ | 1 | 1 | | |
| $AB$ | 1 | 1 | | |
| $A\overline{B}$ | 1 | 1 | | |

Sum-of-products K-map

| | $\overline{C}+\overline{D}$ | $\overline{C}+D$ | $C+D$ | $C+\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}+\overline{B}$ | 1 | 1 | | |
| $\overline{A}+B$ | 1 | 1 | | |
| $A+B$ | 1 | 1 | | |
| $A+\overline{B}$ | 1 | 1 | | |

Product-of-sums K-map




Group formation in minterm and maxterm Karnaugh maps.

GOVT POLYTECHNIC DARINGBADI

GOVT POLYTECHNIC DARBHANGA