ATME College of Engineering

Digital Logic Circuits – BEE306A

Prepared By,

Ms. Swathi C A

Asst Professor

Dept of EEE

ATMECE, Mysuru

Course Outline

Course Code	Course Title	Core/Elective	Prerequisite	Contact Hours			Total Hrs/ Sessions
				L	T	P	
BEE306A	Digital Logic Circuits	Elective	Basic Electronics	3	-	-	40

Module – 1: Principles of Combinational Logic: Definition of combinational logic, canonical forms, Generation of switching equations from truth tables, Karnaugh maps-3,4,5 variables, Incompletely specified functions (Don‘t care terms) Simplifying Max term equations, Quine-McCluskey minimization technique, Quine-McCluskey using don‘t care terms, Reduced prime implicants Tables.	
Bloom’s Taxonomy Level	L1 – Remembering, L2 – Understanding, L3 – Applying, L4 – Analyzing,
Module – 2: Analysis and Design of Combinational logic: General approach to combinational logic design, Decoders, BCD decoders, Encoders, digital multiplexers, Using multiplexers as Boolean function generators, Adders and subtractors, Cascading full adders, Look ahead carry, Binary comparators	
Bloom’s Taxonomy Level	L1 – Remembering, L2 – Understanding, L3 – Applying, L4 – Analysing,
Module – 3: Flip-Flops: Basic Bistable elements, Latches, Timing considerations, The master-slave flip-flops (pulsetriggered flip-flops): SR flip-flops, JK flip-flops, Edge triggered flip- flops, Characteristic equations .	
Bloom’s Taxonomy Level	L1 – Remembering, L2 – Understanding
Module – 4: Flip-Flops Applications: Registers, binary ripple counters, synchronous binary counters, Counters based on shift registers, Design of a synchronous counter, Design of a synchronous mod-n counter using clocked T, JK, D and SR flip-flops.	
Bloom’s Taxonomy Level	L1 – Remembering, L2 – Understanding, L3 – Applying, L – 4 Analysing,
Module – 5: Sequential Circuit Design: Mealy and Moore models, State machine notation, Synchronous Sequential circuit analysis, Construction of state diagrams, counter design. Memories: Read only and Read/Write Memories, Programmable ROM, EPROM, Flash memory.	

Course Outcomes

1. Develop simplified switching equation using Karnaugh Maps and Quine McClusky techniques.[L3]
2. Apply the design procedures for Multiplexer, Encoder, Decoder, Adder, Subtractors and Comparator as digital combinational control circuits.[L3]
3. Illustrate the design of flip flops and development of its characteristic equation.[L2]
4. Apply the design procedures for counters and shift registers as sequential control circuits.[L3]
5. Develop Mealy/Moore Models and state diagrams for the given clocked sequential circuits and Interpret the functioning of Read only and Read/Write Memories, Programmable ROM, EPROM and Flash memory.[L3]

List of Reference Books

1. Digital Logic Applications and Design, John M Yarbrough, Thomson Learning 2001 ISBN 981- 240-062-1.
2. Digital Principles and Design Donald D. Givone McGraw Hill 2002 ISBN 978-0- 07-052906-9.

Fundamentals of Digital Systems

- Digital systems are designed to store, process, and communicate information in digital form.
- They are found in a wide range of applications, such as communication systems, digital instruments.

Importance of Digital Systems

1. Digital systems are generally easier to design. The circuits used in digital systems are switching circuits, where exact values of voltage or current are not important, only the range (HIGH or LOW) in which they fall.

2. Information storage is easy.

3. Accuracy and precision are easier to maintain throughout the system.

4. Operation can be programmed. It is fairly easy to design digital systems

5. Digital circuits are less affected by noise. Spurious fluctuations in voltage (noise) are not as critical in digital systems because the exact value of a voltage is not important, as long as the noise is not large enough to prevent us from distinguishing a HIGH from a LOW.

6. More digital circuitry can be fabricated on IC chips.

Fundamentals: Logic Gates

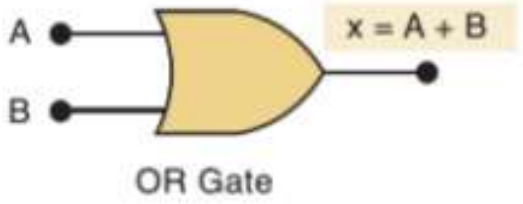
- Logic gates are the basic building blocks of any digital system.
- Implements Boolean functions that performs a logic operation on one or more bits of input and gives a bit as an output
- The relationship between the input and the output is based on a certain logic.

Logic Gates [Contd. .]

- The basic logic gates are classified into seven types:
- AND gate, OR gate, NOT, XOR gate, NAND gate, NOR gate, XNOR gate.
- The truth table is used to show the logic gate function

1. OR Gate: Expression**: x = A + B**

Symbol:



Truth Table of OR Gate

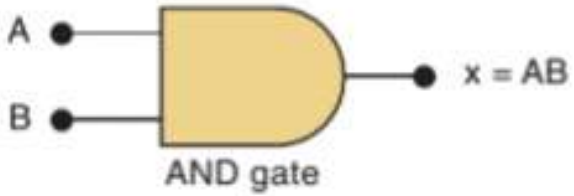
OR

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gates [Contd. .]

2. AND Gate: Expression**: x = A . B**

Symbol:



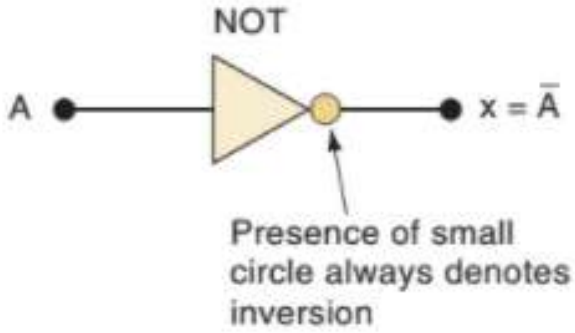
Truth Table of AND Gate

AND

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

3. NOT Gate: Expression: A = A’

Symbol:



Truth Table of NOT Gate

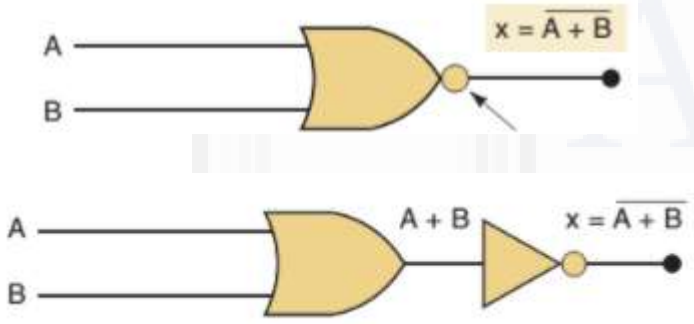
NOT

A	x = Ā
0	1
1	0

Logic Gates [Contd. .]

4. NOR Gate: Expression: $x = \overline{A + B}.$

Symbol:



Truth Table of NOR Gate

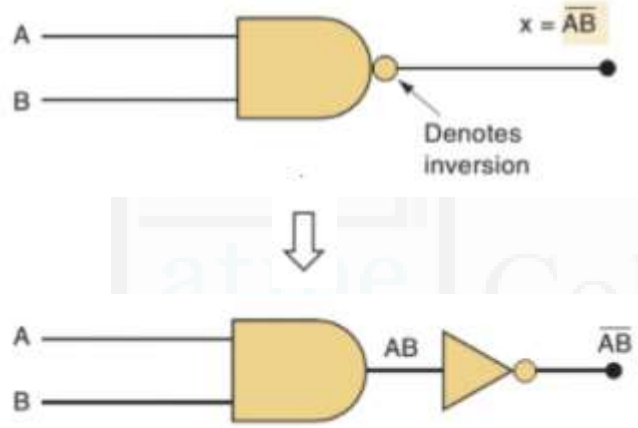
		OR	NOR
A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Logic Gates [Contd. .]

5. NAND Gate: Expression:

$$x = \overline{AB}.$$

Symbol:



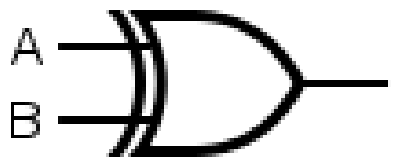
Truth Table of NOR Gate

		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Logic Gates [Contd. .]

6. XOR Gate: Expression**: x =** $A \oplus B$

Symbol:



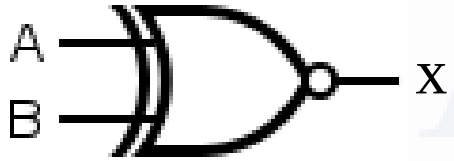
Truth Table of XOR Gate

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Logic Gates [Contd. .]

7. XNOR Gate: Expression**: x =** $: A \odot B$.

Symbol:



Truth Table of XOR Gate

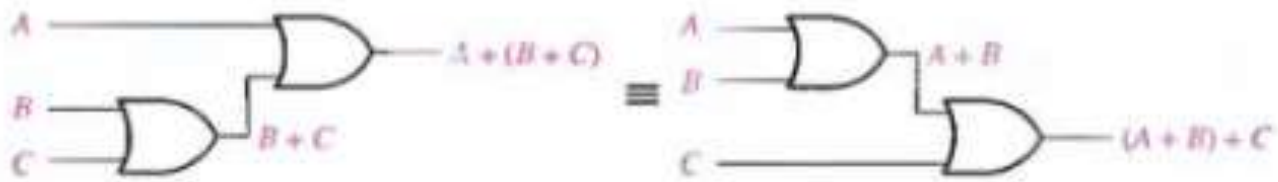
Input		Output
A	**B**	**A XNOR B**
0	0	1
0	1	0
1	0	0
1	1	1

Laws and Rules of Boolean Algebra

Basic Laws:

1. **Commutative Law : A+B = B+A**

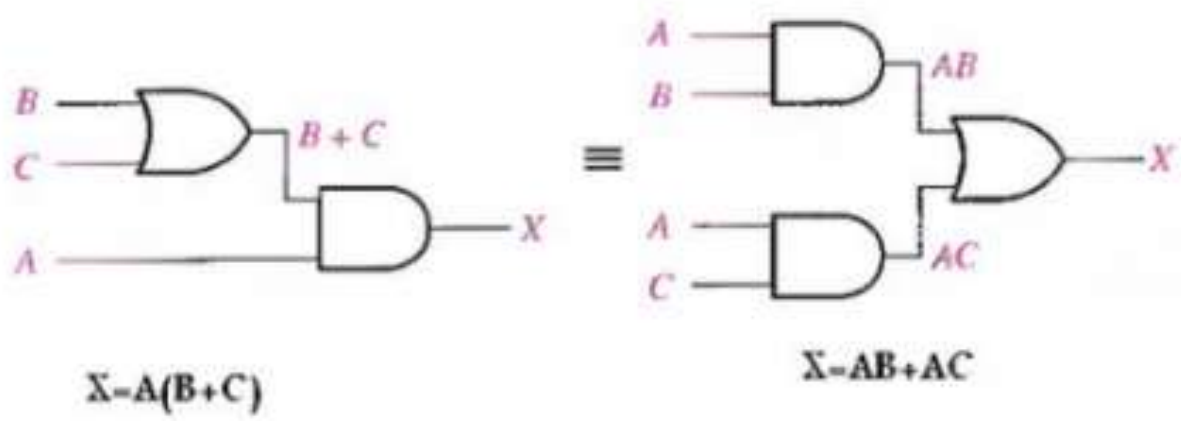
2. **Associative Law : A+ (B+C) = (A+B)+C**



Laws and Rules of Boolean Algebra

Basic Laws:

3. Distributive Law: A (B+C) = AB+ AC



Laws and Rules of Boolean Algebra

Basic Laws:

4. Duality Property

- Duals are just the opposites of original operators

- Switching Logic depends upon Duality property b/w AND and OR operators

Eg: Operator	Dual
AND	OR
OR	AND
0	1
1	0

Laws and Rules of Boolean Algebra

Basic Laws:

5. Idempotency Property

- Idempotency refers to "sameness"

Hence, $x + x = x$

$x.x = x$

Laws and Rules of Boolean Algebra

12 Basic Rules of Boolean Algebra

1. $A+0=A$	7. $A \cdot A = A$
2. $A+1=1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A+AB=A$
5. $A+A=A$	11. $A+\overline{A}B=A+B$
6. $A+\overline{A}=1$	12. $(A+B)(A+C)=A+BC$

DeMorgan's Theorems

- DeMorgan's Theorems uses the principle of Duality

There are two statements under DeMorgan's Theorem

1. The complement of an AND function is the OR of the complemented input variables

Eg: $z = \overline{x.y}$ then $z = \bar{x} + \bar{y}$

2. The complement of an OR function is the AND of the complemented input variables

$z = \overline{x + y}$ then $z = \bar{x} \,.\, \bar{y}$

Reduction of Switching Equations using Boolean Algebra

Ex 1.

$$F = BC + B\bar{C} + BA$$

$$F = B(C + \bar{C}) + BA$$

$$F = B \cdot 1 + BA$$

$$F = B(1 + A)$$

$$F = B$$

Realization of Switching Functions

Eg: AB + A(B + C) + B(B + C)

AB + AB + AC + BB + BC

Since, (BB = B) to the fourth term

AB + AB + AC + B + BC

Since (AB + AB = AB) to the first two terms.

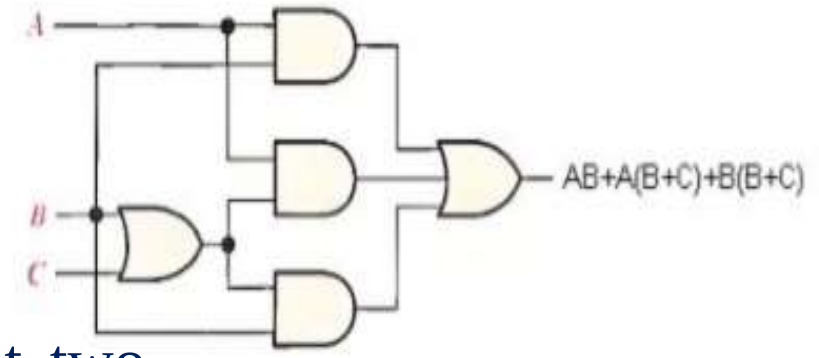
AB + AC + B + BC

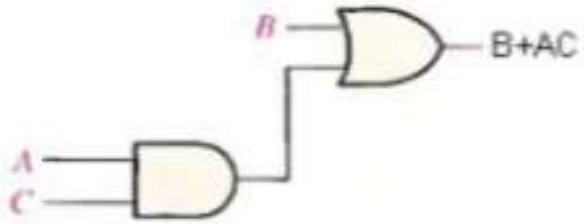
Since (B + BC = B) to the last two terms

AB + AC + B

(AB + B = B) to the first and third terms

→ B+AC





Definition of combinational Logic

There are two types of logic networks

Combinational network

Sequential networks

Combinational Logic Network

- It is a technique of combining the basic gates into circuits that perform some desired function.
- These circuits will not exhibit “Memory”
- Output of the system depends on the present input

Eg: Adders, Subtractors, Decoders, Encoders, Multipliers

Sequential Networks

- These networks exhibit a “Memory”
- i.e. The Output is characterized by the combination of present input and also depends upon on the past history of the input

Eg: Flip flops

Combinational Logic Diagram:

Important Terminologies

1. Literal: A Boolean variable or its complement is known as Literal. Ex: A, $\bar{A}$, X, $\bar{B}$ etc
2. Product Term: Logical AND (product) of multiple literals

Ex: AB, A$\bar{B}$, XY$\bar{Y}$ etc

3. Sum Term: Logical OR (Sum) of multiple literals

Ex: A+B, A+$\bar{B}$, X+Y+$\bar{Y}$ etc

4. Sum of Product (SOP):

When two or more product terms are summed by Boolean addition, the resulting expression is a sum-of-products (SOP). Ex: AB + ABC , ABC + CDE + BCD , AB + BCD + AC

Important Terminologies (Contd..)

5. Product-of-sums (POS) When two or more sum terms are multiplied, the resulting expression is a product-of-sums

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

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

6. Minterms: It is a product of 'n' variables in which each appears exactly once in true or complemented form.

7. Maxterms: It is a sum of the 'n' variables in which each appears exactly once in true or complemented form.

Important Terminologies (Contd..)

Minterms	Maxterms
Input variable is complemented when it has a value “0” in the Truth Table	Input variable is complemented when it has a value “1” in the Truth Table
Denoted by ‘m’	Denoted by ‘M’
Reprsented as SOP form	Represented as POS form

Canonical Form

- Canonical expressions are not simplified form
- These contain redundancies

Canonical sum of Products: It is a complete set of Minterms that define when an output variable is logical 1.

Ex: a'bc+ab'c+abc'

Canonical Product-of-sums: : It is a complete set of Maxterms that define when an output variable is logical 0.

Ex: (a+b+c) (a'+b+c) (a+b'+c)

Canonical Form

Input Variable			Minterm		Maxterm	
a	b	c	Term	Designation	Term	Designation
0	0	0	a’b’c’	m_0	a+b+c	M_0
0	0	1	a’b’c	m_1	a+b+c’	M_1
0	1	0	a’bc’	m_2	a+b’+c	M_2
1	0	0	ab’c’	m_3	a’+b+c	M_3
0	1	1	a’bc	m_4	a+b’+c’	M_4
1	0	1	ab’c	m_5	a’+b+c’	M_5
1	1	0	abc’	m_6	a’+b’+c	M_6
1	1	1	abc	m_7	a’+b’c’	M_7

Steps to convert SOP to Canonical Form

1. Identifying the missing variable(s) in each AND term
2. AND the missing term and its complement with the original AND term, xy(z+z') since (z+z') = 1
3. Apply the distribution property and expand the term

   i.e. xyz + xyz'

Steps to convert POS to Canonical Form

1. Identifying the missing variable(s) in each OR term
2. OR the missing term and its complement with the original OR term, x+y'+zz' since (zz' = 0)
3. Apply the distribution property and expand the term

   i.e. (x+y'+z) (x+y'+z')

Numerical on Canonical Form

Place the following equations into the proper canonical form

1. P = f(a,b,c) = ab'+ac'+bc

Soln: First term, AND (c+c') with ab': ab'(c+c') = ab'c+ab'c'

Second Term, AND (b+b') with ac' : ac'(b+b') = abc'+ab'c'

Third Term, AND (a+a') with bc : abc + a'bc

→ P = ab'c+ab'c' + abc'+ab'c' + abc + a'bc

P = ab'c+ ab'c' + abc'+ abc + a'bc

2. G = f(w,x,y,z) = w'x+yz' ----- Try it !!

Numerical on Canonical Form

3. T = f(a,b,c) = (a+b')(b'+c) ----- (POS Equn)

Soln:

First term, OR cc' with (a+b') = (a+b'+c) (a+b'+c')

Second Term, OR aa' with (b'+c) = (a+b'+c) (a'+b'+c)

→ T = f(a,b,c) = (a+b'+c) (a+b'+c') (a'+b'+c)

4. J = f(A,B,C,D) = (A+B'+C+D)(A+B'+C+D') ---- Try it !!

Generation of Switching Equations from Truth Table

Input Variable			Minterm		Maxterm	
a	b	c	Term	Designation	Term	Designation
0	0	0	a'b'c'	m_0	a+b+c	M_0
0	0	1	a'b'c	m_1	a+b+c'	M_1
0	1	0	a'bc'	m_2	a+b'+c	M_2
1	0	0	ab'c'	m_3	a'+b+c	M_3
0	1	1	a'bc	m_4	a+b'+c'	M_4
1	0	1	ab'c	m_5	a'+b+c'	M_5
1	1	0	abc'	m_6	a'+b'+c	M_6
1	1	1	abc	m_7	a'+b'c'	M_7

P (ab'c+ab'c'+abc'+abc+a'bc) => P = f(a,b,c) = $\sum(5,4,6,7,3)$

Numerical examples on Switching Equations from Truth Table

1. The Minterm expression for the output variable M is

$M = f(a,b,m,s) = a'bms+ab'ms+abms$

$\Rightarrow M = f(a,b,m,s) = \sum(7,11,15)$

2. The Maxterm expression for the output variable M is

$M = f(a,b,m,s) = (a'+b+m+s)\ (a+b'+m+s)\ (a+b+m+s)\ (a+b'+m'+s)$

$M = f(a,b,m,s) = \Pi\ (8,11,0,6)$

Numerical examples on Switching Equations from Truth Table

3. H = f(A,B,C) = A'BC+A'B'C+ABC Express the SOP Equation in a minterm list (shorthand decimal notation)

Soln: 1. $A'BC = 011_2 = 3$

2. $A'B'C = 001_2 = 1$

3. $ABC = 111_2 = 7$

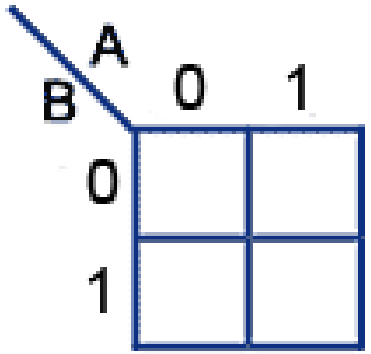
→ $H = f(A,B,C) = \sum(1,3,7)$

Introduction to Karnaugh-Maps

- Karnaugh Maps offer a graphical method of reducing a digital circuit to its minimum number of gates
- It is a Matrix of Squares
- Each square represents a "cell" containing Minterms or Maxterm from a Boolean equation
- K-Maps allows Identification of input variables and helps to reduce the output equation
- The Karnaugh map can be populated with data from either a truth table or a Boolean equation.

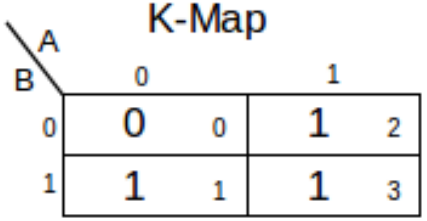
Types of K-Maps

- Circuits with 2 inputs A and B require maps with $2^2 = 4$ cells
- Circuits with 3 inputs A B and C require maps with $2^3 = 8$ cells
- Circuits with 4 inputs A B C and D require maps with $2^4 = 16$ cells
  - 2 Variable K-Map



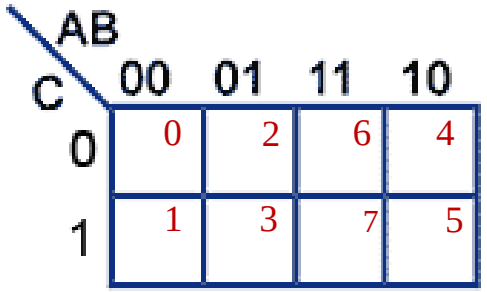
2 Variable Truth Table

	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1



Types of K-Maps

- 3 Variable K-Map



Input Variable binary weighting

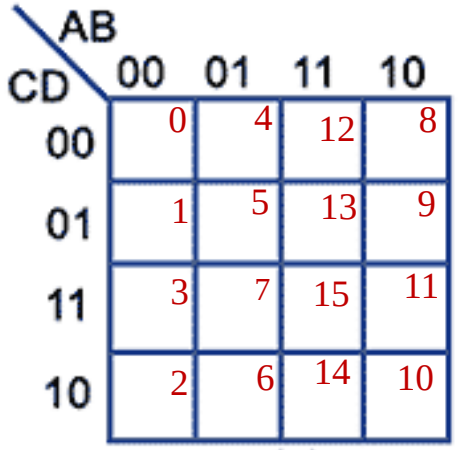
MSB LSB

A B C

- Note:
- Edge numbering does not follow the normal binary counting sequence.
- It uses a Gray Code sequence where only one bit changes from one cell to the next.

Types of K-Maps

- 4 Variable K-Map



Input Variable binary weighting

MSB LSB

A B C D

- Note:
- Edge numbering does not follow the normal binary counting sequence.
- It uses a Gray Code sequence where only one bit changes from one cell to the next.

Rules of Cell Grouping in K-Maps

1. Groups should contain as many ‘1’ cells (i.e. cells containing a logic 1) as possible and no blank cells.
2. Groups can only contain 1, 2, 4, 8, 16 or 32... etc. cells (powers of 2).
3. A ‘1’ cell can only be grouped with adjacent ‘1’ cells that are immediately above, below, left or right of that cell; no diagonal grouping.
4. Groups of ‘1’ cells can overlap. This helps make smaller groups as large as possible, which is an advantage in finding the simplest solution.
5. The top/bottom and left/right edges of the map are considered to be continuous, so larger groups can be made by grouping cells across the top and bottom or left and right edges of the map
6. There should be as few groups as possible.

PI and EPI

1. Prime Implicants (PI): Group of Minterms that can be combined with any other Minterms or groups
2. Essential Prime Implicants (EPI): It is a PI in which one or more Minterms are unique.

THANK YOU