DSD - Digital System Design 1

• Digital

• Concerned with the interconnection among digital components andmodules» Best Digital System example is General Purpose Computer

• Logic Design

• Deals with the basic concepts and tools used to design digital hardwareconsisting of logic circuits

» Circuits to perform arithmetic operations (+, -, x, ÷)

• Digital Signal : Decimal values are difficult torepresentinelectrical systems. It is easier to use two voltagevaluesthanten.

• Digital Signals have two basic states: **1 (logic "high", or H, or "on")**
0 (logic "low", or L, or "off")

• Digital values are in a *binary* format. Binarymeans2states.

• A good example of binary is a light (only onor off)*on of*

[redacted] 3

• Bits and Pieces of DLDHistory

• George Boole

  • Mathematical Analysis of Logic (1847)

  • An Investigation of Laws of Thoughts; Mathematical Theories of LogicandProbabilities (1854)

• Claude Shannon • Rediscovered the Boole

  • “ A Symbolic Analysis of Relay and Switching Circuits “ • Boolean Logic and

Boolean Algebra were Applied to Digital Circuitry

---------- Beginning of the Digital Age and/or Computer Age World War II Computers as Calculating Machines

Digital Systems andBinaryNumbers

 Digital age and information age Digital computers

– General purposes

– Many scientific, industrial and commercial applications • Digital systems

– Telephone switching exchanges

– Digital camera

– Electronic calculators, PDA's
– Digital TV

- Discrete information-processing systems – Manipulate discrete elements of information – For example, {1, 2, 3, …} and {A, B, C, …}…

Binary Digital Signal

- An information variable represented by physical quantity.• For digital systems, the variable takes ondiscretevalues.– Two level, or binary values are the most prevalent values. • Binary values are represented abstractly by:

– Digits 0 and 1

$V(t)$

– Words (symbols) False (F) and True (T) – Words (symbols) Low (L) and High (H)

Logic1

– And words On and Off

undefine

• Binary values are represented by values or ranges of values of physical quantities. t

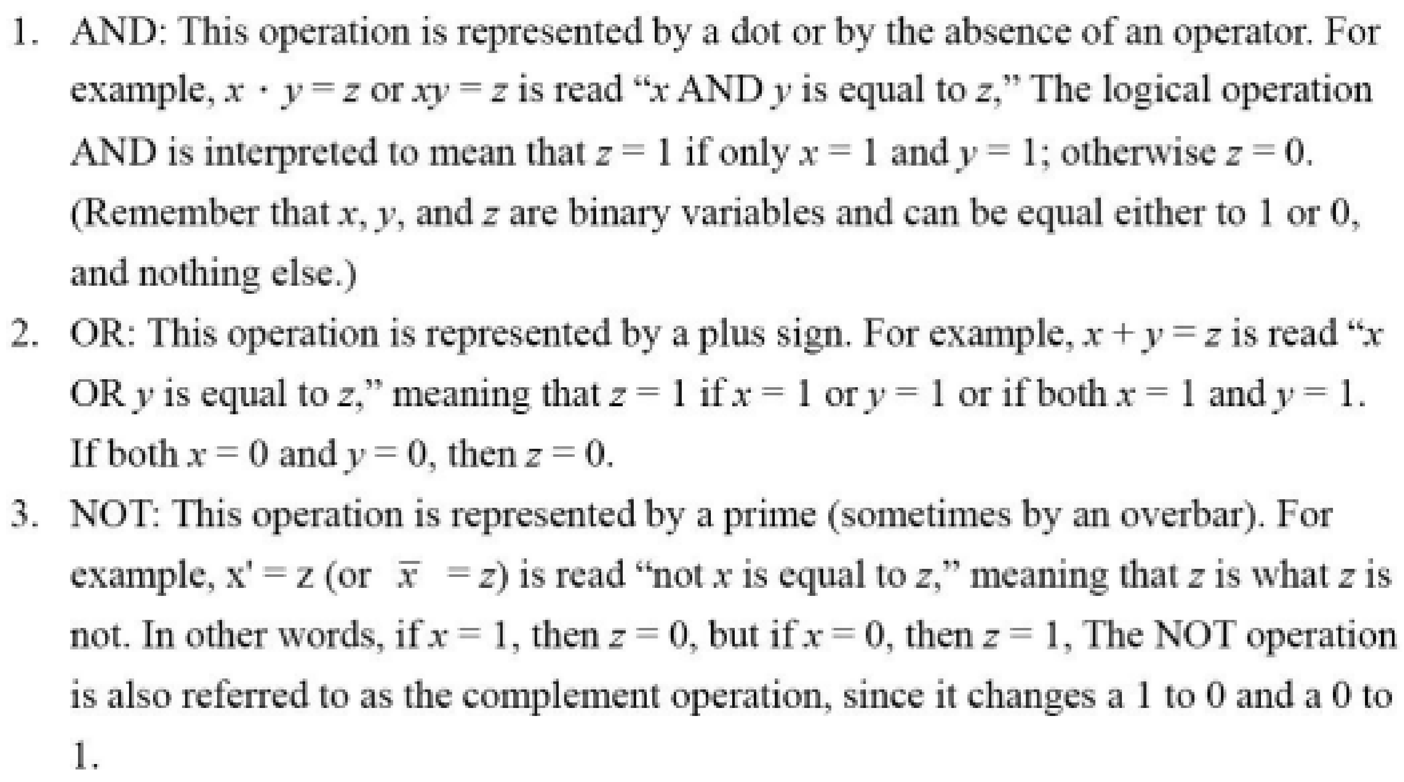
Logic0

Binary digital signal

Binary Logic

• Definition of Binary Logic

– Binary logic consists of binary variables and a set of logical operations. – The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc, witheach variable having two and only two distinct possible values: 1 and 0, – Three basic logical operations: AND, OR, and NOT.



1. AND: This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read "x AND y is equal to z," The logical operation AND is interpreted to mean that $z = 1$ if only $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y, and z are binary variables and can be equal either to 1 or 0, and nothing else.)
2. OR: This operation is represented by a plus sign. For example, $x + y = z$ is read "x OR y is equal to z," meaning that $z = 1$ if $x = 1$ or $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
3. NOT: This operation is represented by a prime (sometimes by an overbar). For example, $x' = z$ (or $\bar{x} = z$) is read "not x is equal to z," meaning that z is what z is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$, The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.

Binary Logic gates

- Truth Tables, Boolean Expressions, and Logic Gates **AND ORNOT** *x y z*

x y z

xz

0 0 0

0 0 0

01

0 1 0

0 1 1

10

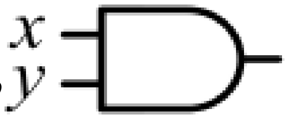
1 0 0

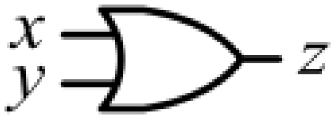
1 0 1

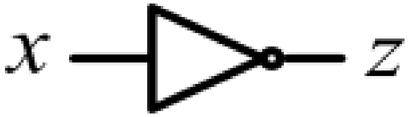
1 1 1

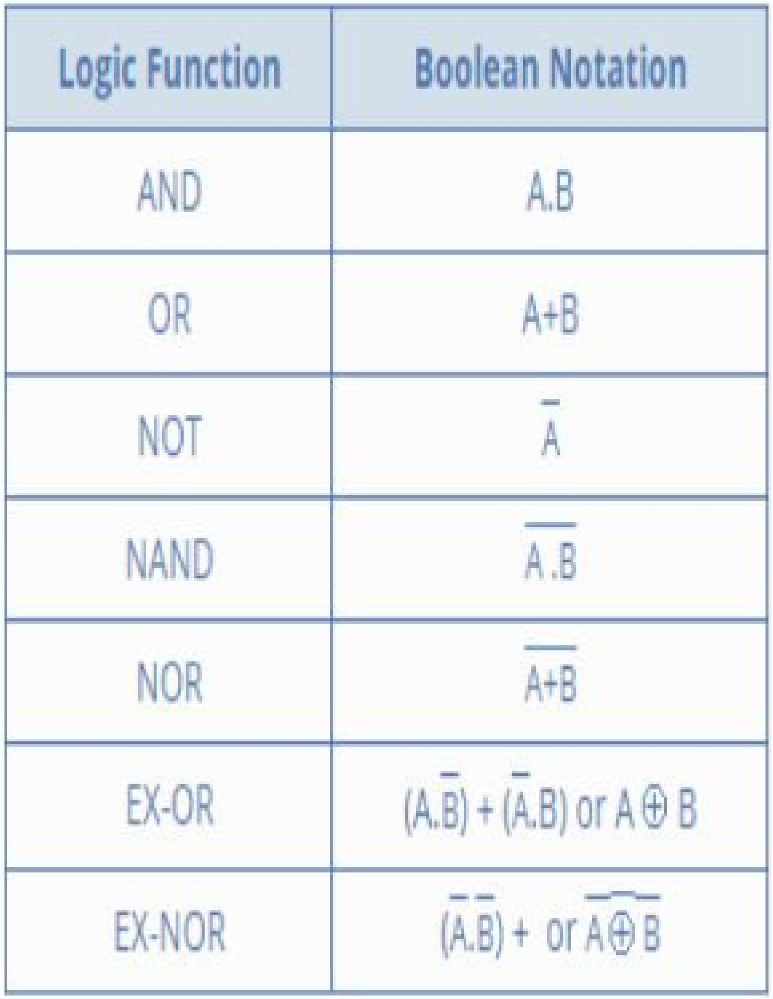
1 1 1

$z = x \bullet y = x\,y$ $z = x + y$ $z = x$ $= x'$



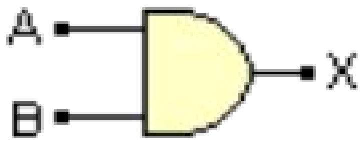






Logic Function	Boolean Notation
AND	$A.B$
OR	$A+B$
NOT	$\overline{A}$
NAND	$\overline{A.B}$
NOR	$\overline{A+B}$
EX-OR	$(A.\overline{B}) + (\overline{A}.B)$ or $A \oplus B$
EX-NOR	$(\overline{A}.\overline{B}) +$ or $\overline{A \oplus B}$

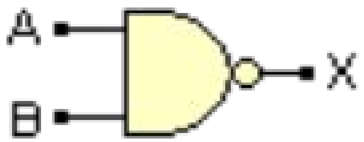
AND



$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

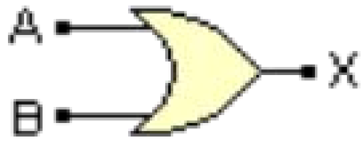
NAND



$X = \overline{A \cdot B}$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

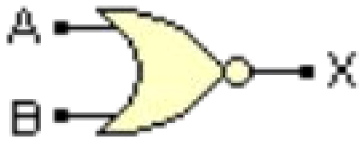
OR



$X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

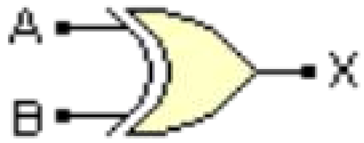
NOR



$X = \overline{A + B}$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

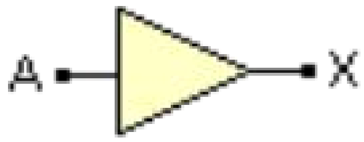
XOR



$X = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

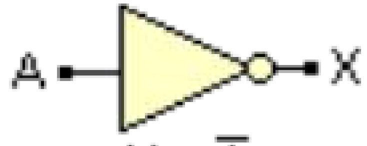
Buffer



$X = A$

A	X
0	0
1	1

Inverter (NOT)

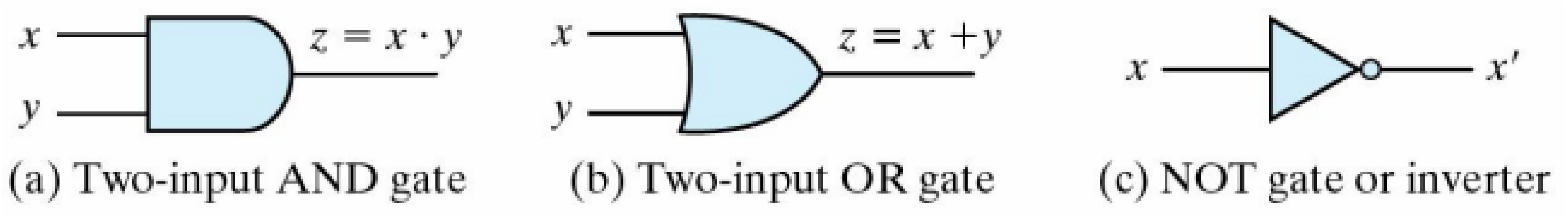


$X = \overline{A}$

A	X
0	1
1	0

Binary Logic

- Logic gates



(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

–

Graphic Symbols and Input-Output SignalsforLogicgates:

Fig. 1.4 Symbols for digital logic circuits

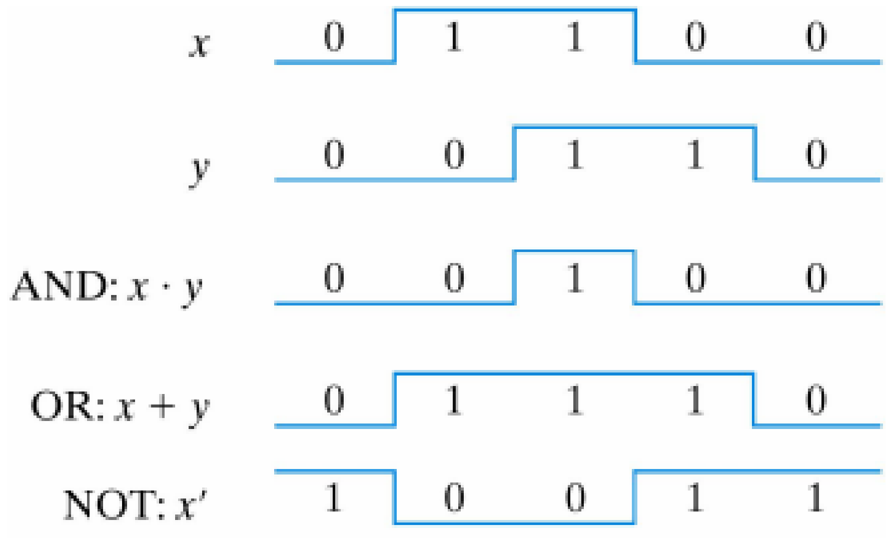
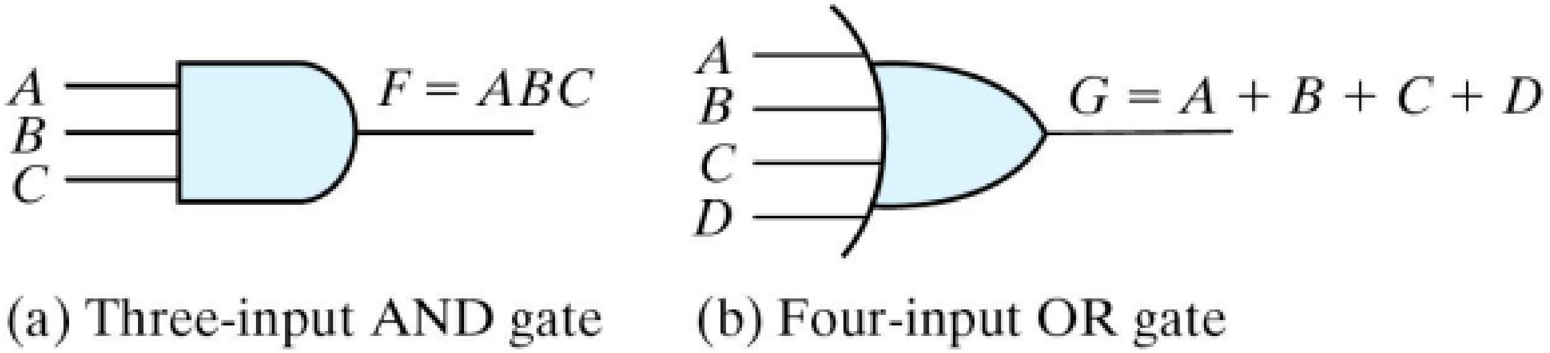


Fig. 1.5 Input-Output signals for gates[10]

Binary Logic

- Logic gates
  - Graphic Symbols and Input-Output SignalsforLogic



gates: Fig. 1.6 Gates with multiple inputs

MoreGates

NAND NOR XOR XNOR

x

x

F F

y

y

x

y

F

x

y

F

x

y

F

x

y

F

0

0

1

0

0

1

0

0

0

0

0

1

0

1

1

0

1

0

0

1

1

0

1

0

1

0

1

1

0

0

1

0

1

1

0

0

1

1

0

1

1

0

1

1

0

<u>1</u>

<u>1</u>

<u>1</u>

• NAND: Opposite of AND (“NOT AND”) • NOR: Opposite of OR (“NOT OR”)

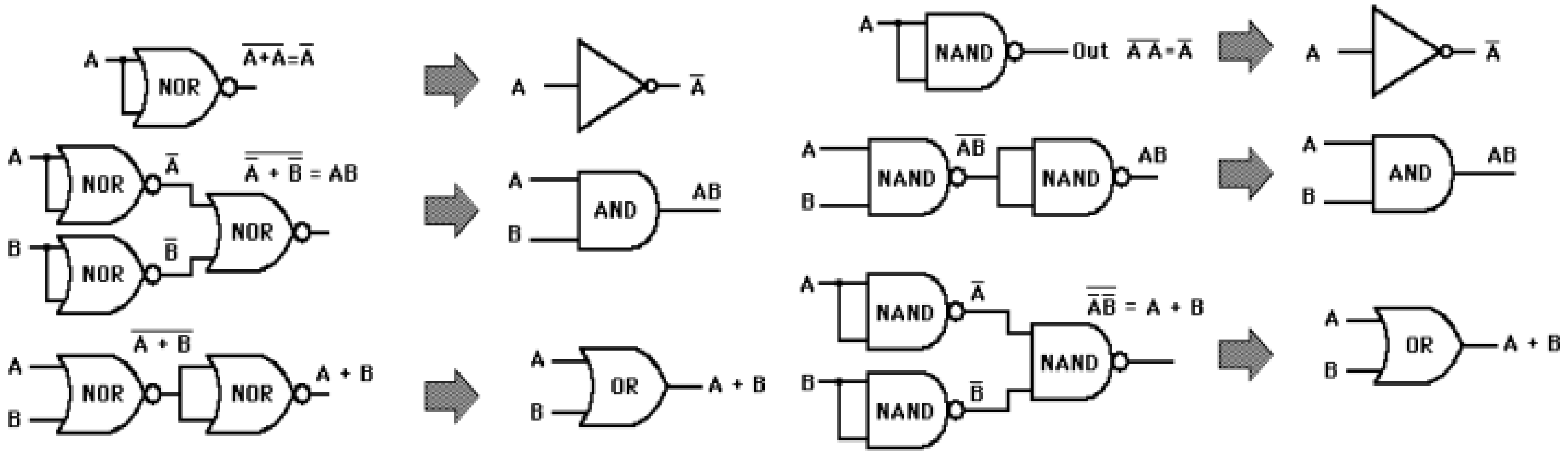
• XOR: Exactly 1 input is 1, for 2-input XOR. (For more inputs -- odd number of 1s)

• XNOR: Opposite of XOR (“NOT XOR”)

Universal Gate

• **NAND and NOR** Gates are called ***Universal Gates*** becauseAND, ORandNOT gates can be implemented &created by using thesegates.

NAND Gate Implementations NOR Gate Implementations


NOR
$\overline{A+A}=\overline{A}$
A
$\overline{A}$
NOR
$\overline{A}$
$\overline{\overline{A}+\overline{B}}=AB$
NOR
B
$\overline{B}$
AND
AB
NOR
$\overline{A+B}$
A + B
OR
NAND
Out
$\overline{A\,A}=\overline{A}$
$\overline{AB}$
NAND
$\overline{\overline{A}\,\overline{B}}=A+B$

Boolean Algebra

Boolean Algebra : George Boole(English mathematician), 1854

Invented by George Boole in 1854

- **An algebraic structure defined by a set B = {0, 1}, together withtwobinaryoperators(+ and ·) and a unary operator ( )**

"An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of LogicandProbabilities"

Boolean Algebra

{(1,0), Var, (NOT, AND, OR), Thms}

❑Mathematical tool to express and analyze digital (logic) circuits❑Claude Shannon, the first to apply Boole's work, 1938 "A Symbolic Analysis of Relay and Switching Circuits" at MIT

❑This chapter covers Boolean algebra, Boolean expressionanditsevaluation and simplification, and VHDL program

Boolean AlgebraTerminology

- Boolean function: **F(a,b,c) = a'bc + abc' + ab + c** • ***Variable***
  - Represents a value (0 or 1)

– Three variables: a, b, and c

- ***Literal***

  – Appearance of a variable, in true or complemented form– Nine literals: a’

  , b, c, a, b, c’

  , a, b, and c

- Expression has five terms including four AND terms and the ORtermthatcombines the first-level ANDterms.

- ***Product term***

  – Product of literals

  – Four product terms: a’bc, abc’

  , ab, c

- ***Sum-of-products***

  – Equation written as OR of product terms only – Above equation is in sum-of-products form. “F = (a+b)c + d” is not.

Representations of BooleanFunctions

English 1: F outputs 1 when a is 0 and b is 0, or when a is 0 and b is 1.

English 2: F outputs 1 when a is 0, regardless of b's value

(a)

a

a

b

F

b 0 0 1

Equation 1: F(a,b) = a'b' + a'b

F

Equation 2: F(a,b) = a' 0

(c) 1

100

1

101

(b)

Circuit 1

Truthtable

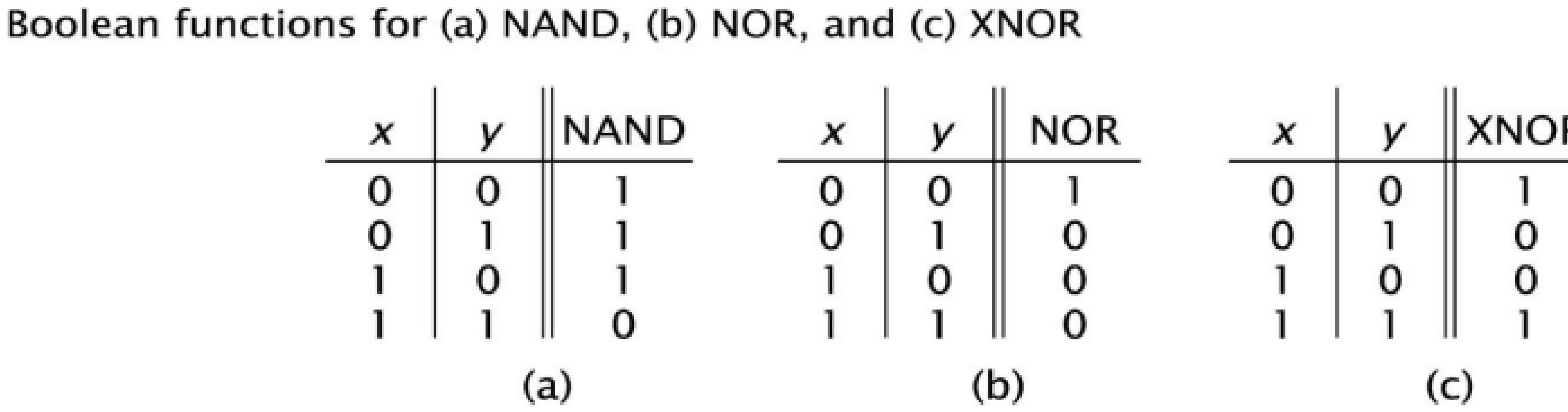
a

F

(d)

Circuit 2

- A function can be represented in different ways – Above shows seven representations of the same functions F(a,b), using four differentmethods: English, Equation, Circuit, and Truth Table



Boolean functions for (a) NAND, (b) NOR, and (c) XNOR

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

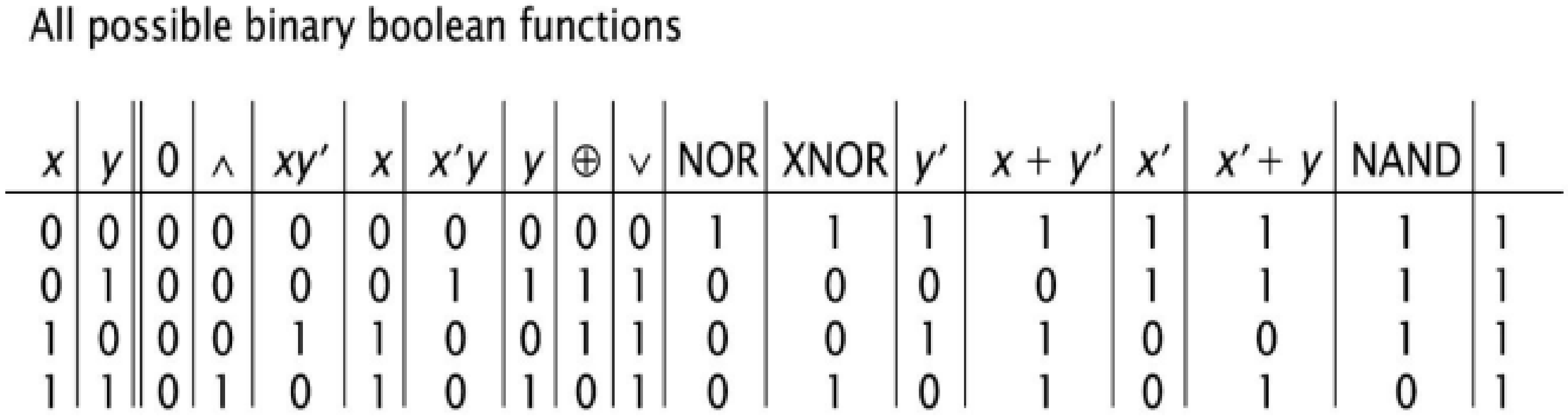
(a)

x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

(b)

x	y	XNO
0	0	1
0	1	0
1	0	0
1	1	1

(c)



All possible binary boolean functions

x	y	0	$\wedge$	xy'	x	$x'y$	y	$\oplus$	$\vee$	NOR	XNOR	y'	$x + y'$	x'	$x' + y$	NAND	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Operations andExpressions

- Boolean Addition

  –Logical OR operation

Ex 4-1) Determine the values of A, B, C, and D that make the sumtermA+B’+C+D’

Sol) all literals must be ‘0’ for the sum term to be ‘0’ A+B’+C+D’=0+1’+0+1’=0 →

A=0, B=1, C=0, and D=1 • Boolean Multiplication –Logical AND

operation

Ex 4-2) Determine the values of A, B, C, and D for AB’CD’ =1

Sol) all literals must be ‘1’ for the product term to be ‘1’ AB’CD’=10’10’=1 → A=1,

B=0, C=1, and D=0

tities of BooleanAlgebra

The relationship betweenasingle variable X, its complement X', and the binary constants 0and1

Laws of BooleanAlgebra

- Commutative Law: the order of literals does not matterA + B = B + A A B = BA

• Associative Law: the grouping of literals does not matter A + (B + C) = (A + B) + C (=A+B+C) A(BC) = (AB)C (=ABC)

Distributive Law : A(B + C) = AB + AC (A+B)(C+D) = AC + AD + BC + BD

Rules of Boolean Algebra

✔ **A+0=A** In math if you add 0 you have changednothinginBoolean Algebra ORing with 0 changes nothing

✓ **A•0=0** In math if 0 is multiplied withanythingyouget0.If you AND anything with 0 youget 0

✓ **A•1 =A** ANDing anything with 1 will yield theanything

✓ **A+A = A** ORing with itself will give thesameresult✓

A+A’=1 Either A or A’ must be 1 so A+A’ =1

✓ **A•A = A** ANDing with itself will give the same result ✓

A•A’=0 In digital Logic 1’ =0 and 0’ =1, so AA’

=0sinceoneofthe
inputs must be 0.

✔ **A = (A')'** If you not something twice you are backtothebeginning

✔ **A + A'B = A + B** If A is 1 the output is 1 If A is 0 the output is B✔ **A + AB = A**

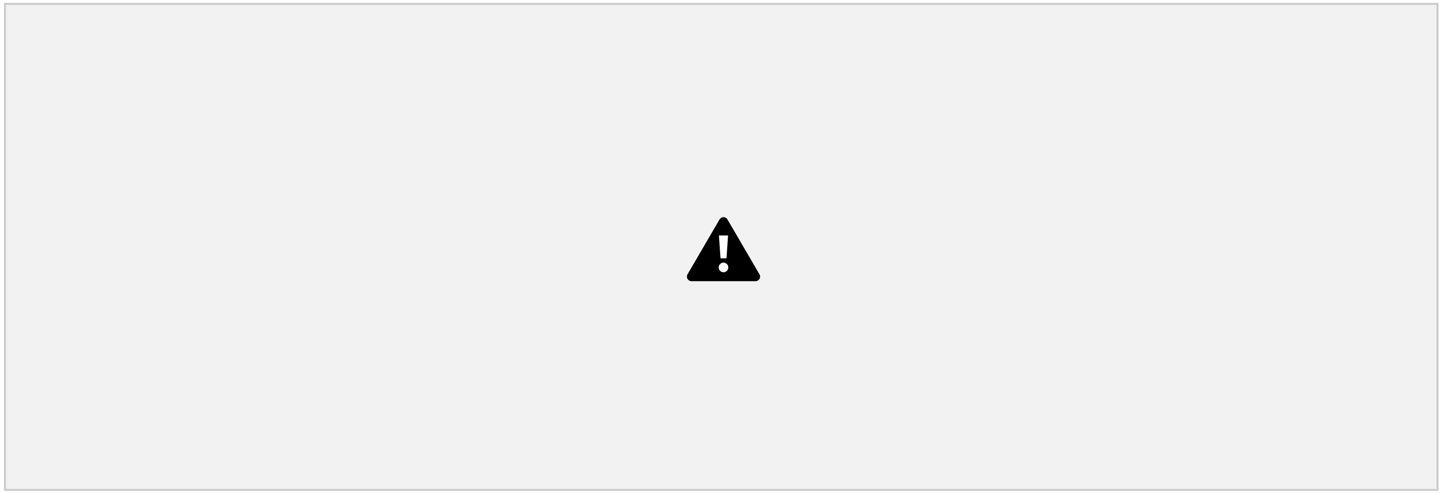
✔ **(A + B)(A + C) = A + BC** • DeMorgan's Theorem

– (A • B)'

= A' + B' and (A + B)'

= A' • B'

– DeMorgan's theorem will help to simplify digital circuitsusingNORsand NANDs his theorem states



Standard Forms of BooleanExpressions

❑The Sum-of-Products(SOP) Form Ex) AB+ABC, ABC+CDE+B’CD’❑The Product-of-Sums(POS) Form Ex) (A+B)(A+B+C), (A+B+C)(C+D+E)(B’+C+D’)❑Principle of Duality : SOP ⇔ POS ❑Domain of a Boolean Expression : The set of variables contained in the expressionEx) A’B+AB’C : the domain is {A, B, C}

✔Standard SOP Form (Canonical SOP Form)

– For all the missing variables, apply (x+x')=1 to the AND terms of theexpression– List all the min-terms in forms of the complete set of variables inascendingorder

Ex : Convert the following expression into standard SOP form: AB'C+A'B'+ABC'DSol)
domain={A,B,C,D}, AB'C(D'+D)+A'B'(C'+C)(D'+D)+ABC'D
=AB'CD'+AB'CD+A'B'C'D'+A'B'C'D+A'B'CD'+A'B'CD+ABC'D
=1010+1011+0000+0001+0010+0011+1101 =0+1+2+3+10+11+13 =Σ(0,1,2,3,10,11,13)

Standard POS Form(Canonical POSForm)

– For all the missing variables, apply (x'x)=0 to the ORtermsoftheexpression
– List all the max-terms in forms of the complete set of

variablesinascending order

Ex : Convert the following expression into standard POSform: (A+B’+C)(B’+C+D’)(A+B’+C’+D)

Sol) domain={A,B,C,D}, (A+B’+C)(B’+C+D’)(A+B’+C’+D)=(A+B’+C+D’D)(A’A+B’+C+D’)(A+ B’+C’+D)

=(A+B’+C+D’)(A+B’+C+D)(A’+B’+C+D’)(A+B’+C+D’)(A+B’+C’+D)= (0100))(0101)(0110)(1101)= Π(4,5,6,13)

Converting Standard SOP to StandardPOS

Step 1. Evaluate each product term in the SOP expression. Determine thebinarynumbers that represent the product terms

Step 2. Determine all of the binary numbers not included in the evaluationinStep1Step 3. Write in equivalent sum term for each binary number Step 2 andexpressioninPOS form

Ex : Convert the following SOP to POS

Sol) SOP= A’B’C’+A’BC’+A’BC+AB’C+ABC=0+2+3+5+7 =Σ(0,2,3,5,7) POS=(1)(4)(6)

= Π(1, 4, 6) (=(A+B+C’)(A’+B+C)(A’+B’+C))

❑ SOP and POS Observations

– Canonical Forms (Sum-of-minterms, Product-of-Maxterms), or other standardforms (SOP, POS) differ in complexity

– Boolean algebra can be used to manipulate equations into simpler forms– Simpler

equations lead to simpler implementations

Summary of Minterms and Maxterms • There are 2^n minterms and

maxterms for Boolean functions with *n* variables. • Minterms and maxterms are

indexed from 0 to $2^n - 1$

• Any Boolean function can be expressed as a logical sum of minterms and as alogical product of maxterms

• The complement of a function contains those minterms not included in theoriginal function

• The complement of a sum-of-minterms is a product-of-maxterms with the

sameindices**Dual of a Boolean Expression**

• **To changing 0 to 1 and + operator to – vise versa for a given booleanfunction❑**

Example: F = (A + C) · B + 0

dual F = (A · C + B) · 1 = A · C + B

❑ Example: G = X · Y + (W + Z) dual G =

✔ Unless it happens to be self-dual, the dual of an expression does not equal theexpression itself

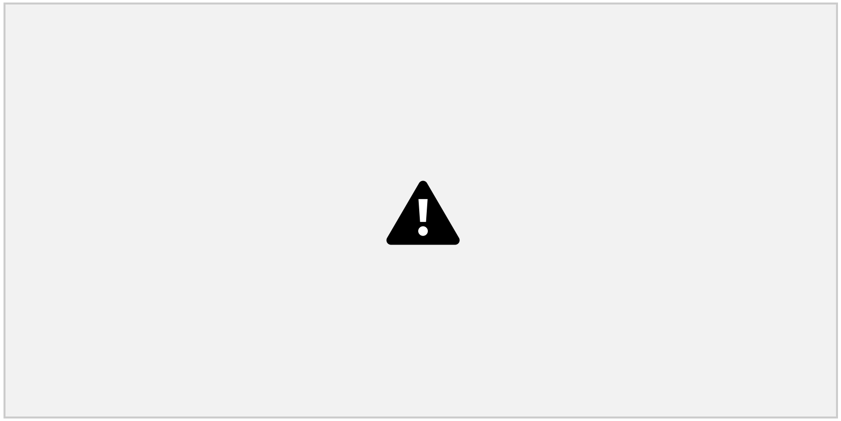
✔ Are any of these functions self-dual? (A+B)(A+C)(B+C)=(A+BC)(B+C)=AB+AC+BC

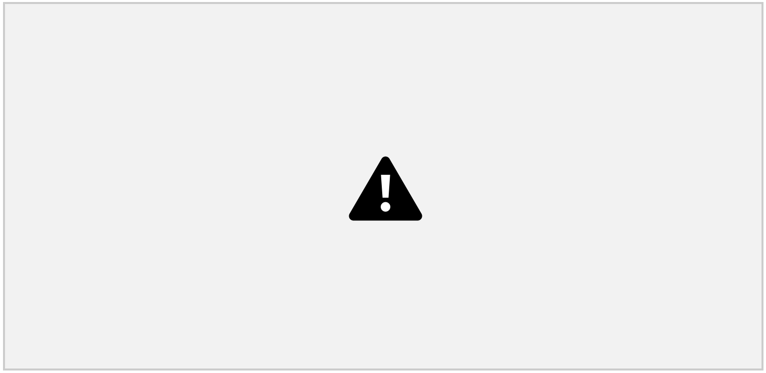
KarnaughMap

- Simplification methods – Boolean algebra(algebraic method) – Karnaugh map(map method)) – Quine-McCluskey(tabular method)

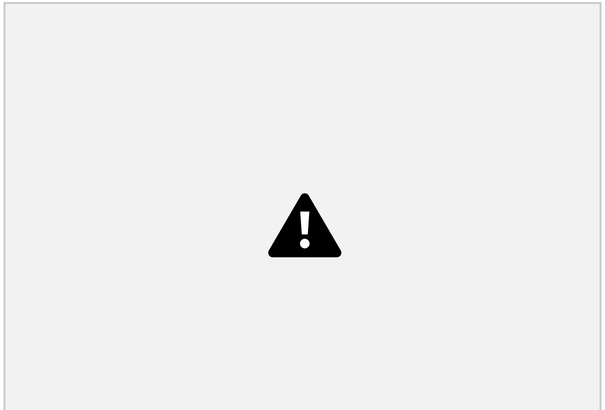
$XY+XY'=X(Y+Y')=X$

  – Three- and Four-input Kanaugh maps





Graycode



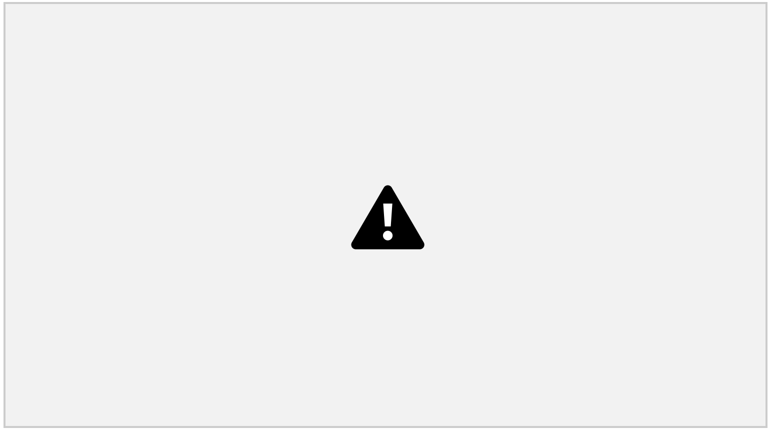
Dr. V. Krishnanaik *Ph.D*

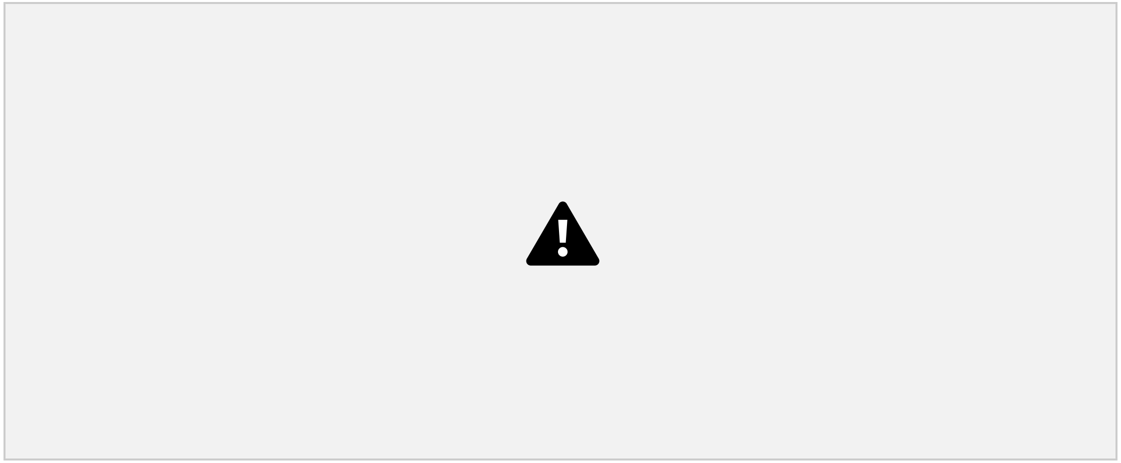
Karnaugh Map (K- Map) Steps

1. Sketch a Karnaugh map grid for the given problem.in power of 2^N Squares
2. Fill in the 1's and 0's from the truth table of sop or pos Boolean function3. Circle groups of 1's.

- Circle the largest groups of 2, 4, 8, etc. first.
- Minimize the number of circles but make sure that every 1is inacircle.4. Write an equation using these circles.

Example) F(X,Y,Z)=Σm(2,3,4,5) =X'Y+XY' *Example) F(X,Y,Z)=Σm(0,2,4,6) = X'Z'+XZ ' =Z'(X'+X)=Z'*





Four-Variable K-Map : 16 minterms : m_0~m_{15}Rectangle

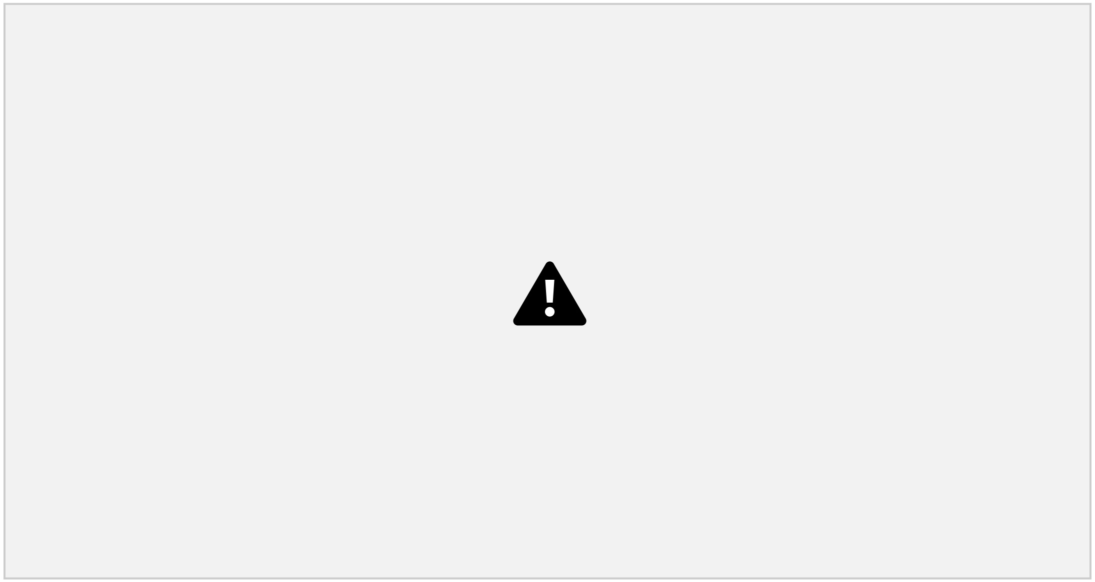
group

– 2-squares(minterms) : 3-literals product term– 4-squares

: 2-literals product term

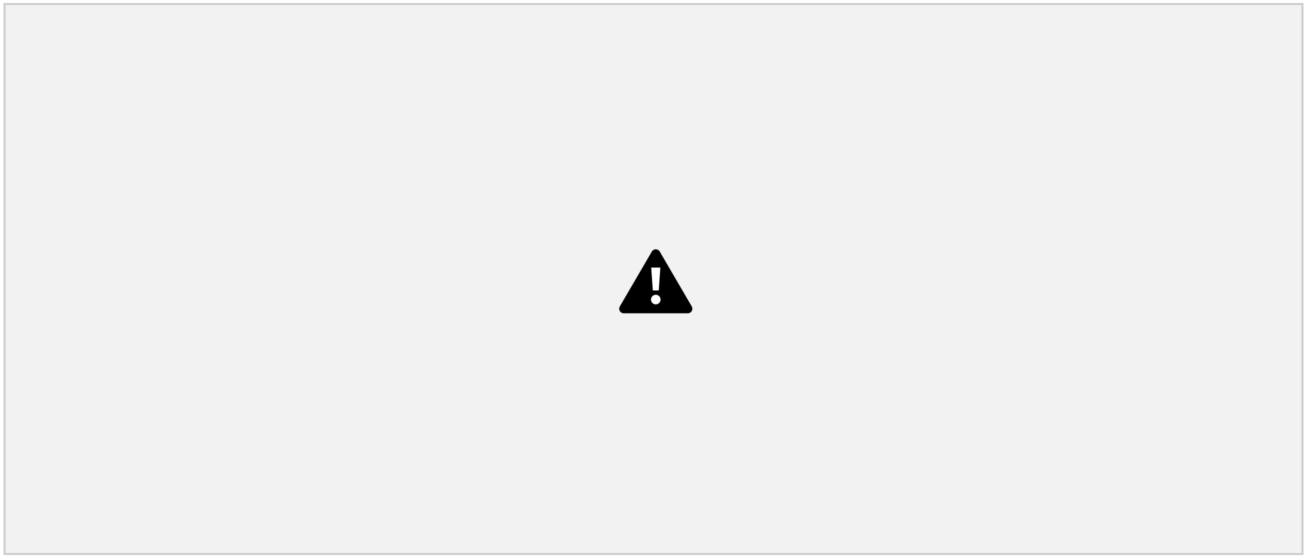
– 8-squares : 1-literals product term

– 16-squares : logic 1



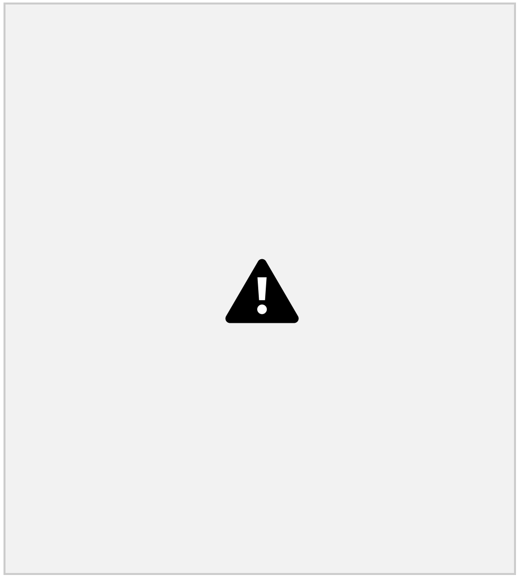
29

$$F(W, X, Y, Z) = \Sigma m(0,2,7,8,9,10,11) = WX' + X'Z' + W'XYZ$$



Ex 4-28) Minimize the following expression

AB’C+A’BC+A’B’C+A’B’C’+AB’C’



Sol) B’+A’C

Ex Minimize the following expression

B’C’D’+A’BC’D’+ABC’D’+A’B’CD+AB’CD+A’B’CD’+A’BCD’
+ABCD’+AB’CD’Sol) D’+B’C

❑**Don’t Care Conditions**

• it really does not matter since they will never occur(its outputiseither‘0’or ‘1’)

- The don’t care terms can be

used to advantage on the

Karnaugh map

31

Ex K- Map for POS (B+C+D)(A+B+C’+D)(A’+B+C+D’)(A+B’+C+D)(A’+B’+C+D) Sol)

(B+C+D)=(A'A+B+C+D)=(A'+B+C+D)(A+B+C+D) (1+0+0+0)(0+0+0+0)(0+0+1+0)

(1+0+0+1)(0+1+0+0)(1+1+0+0)

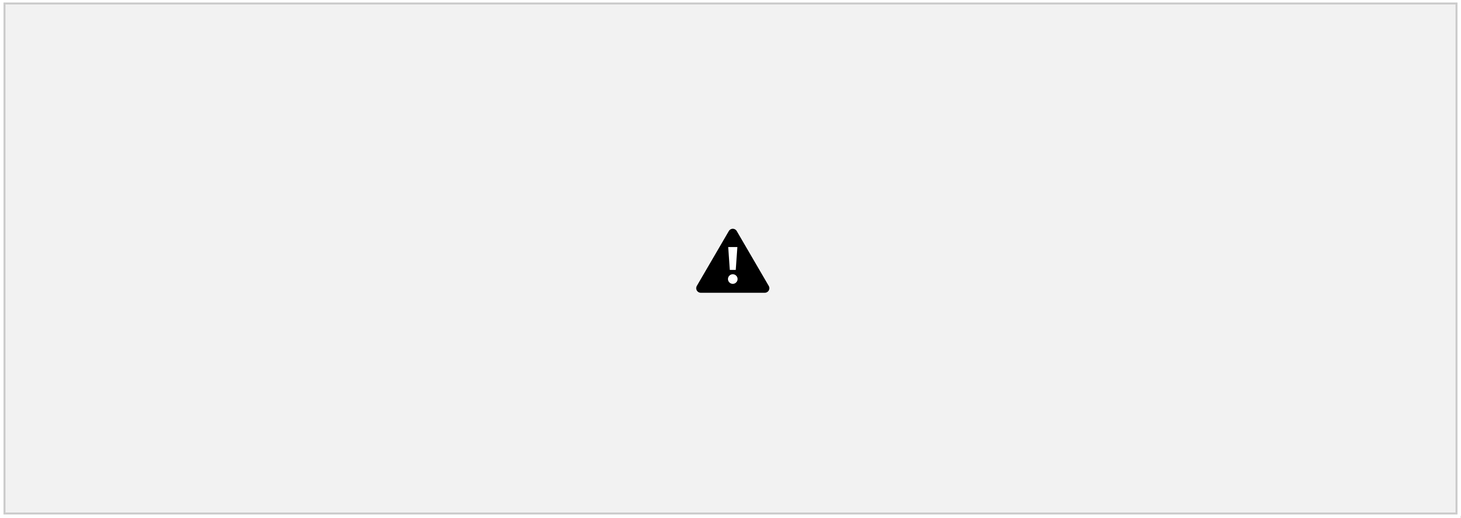
F=(C+D)(A'+B+C)(A+B+D)

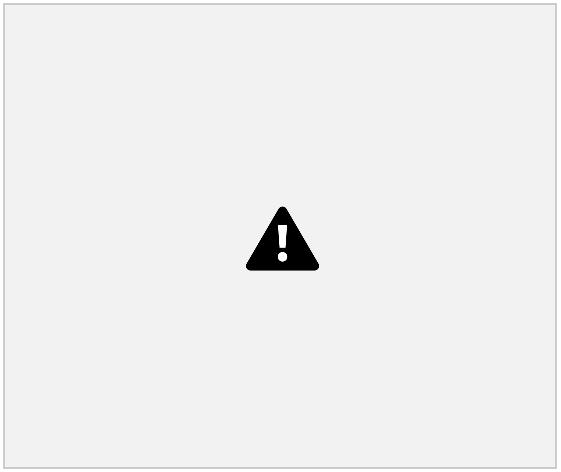
❑ Converting Between POS and SOP Using the K-map

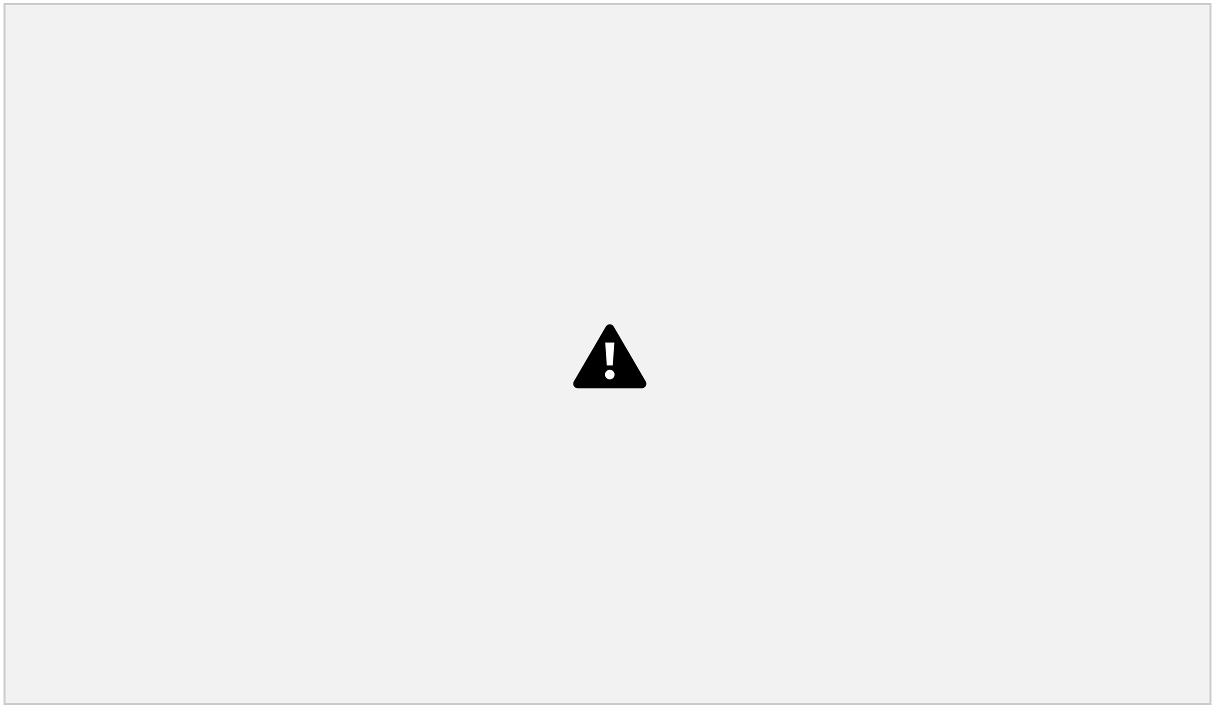
Ex 4-33) (A'+B'+C+D)(A+B'+C+D)

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

(A+B+C'+D)







Quine-McCluskey - Tabular Method

- **Step 1** − Arrange the given min terms in an **ascending order** and make the groups basedonthenumber of ones present in their binary representations. **- ‘n+1’ groups**

- **Step 2** − Compare the min terms present in **successive groups**. If there is a changeinonlyone-bitposition, then take the pair of those two min terms. Place this symbol ‘_’ in the differedbitpositionand keep the remaining bits as it is.

- **Step 3** − Repeat step2 with newly formed terms till we get all **prime implicants**.

- **Step 4** − Formulate the **prime implicant table**. It consists of set of rows and columns. Place‘1’inthe cells corresponding to the min terms that are covered in each prime implicant.

- **Step 5** − Find the essential prime implicates by observing each column. Those essential primeimplicants will be part of the simplified Boolean function.

- **Step 6** − Reduce the prime implicant table by removing the row of each essential primeimplicantand the columns corresponding to the min terms that are covered in that essential primeimplicant.Repeat step 5 for Reduced prime implicant table. Stop this process when all mintermsofgivenBoolean function are over.