EE 102: INTRODUCTION TO COMPUTER ENGINEERING

Lecture 2: Boolean Algebra
Basic Logic Gates, Simplification and K-Maps

4/7/2010

Avinash Kodi, kodi@ohio.edu

Course Administration

2

- Homework 1 will be posted today, due Wednesday 4/7 (next week)

Why Boolean Algebra?

3

- The Boolean Algebra allows us
  - Assign a binary variable to an event/outcome of interest
    - True/False, On/Off, Yes/No, 1/0
  - Then set identities and relationships between events
    - Logic operations: AND (.), OR (+), NOT (~ or ‘)
- Events/outcome examples
  - A buzzer that sounds when car keys are in ignition and the door is open
  - A fire alarm that goes off if it senses heat or smoke

Notation Examples

4

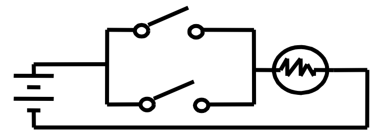
- Examples
  - Y = A . B is read “Y is equal to A AND B”
  - Z = X + Y is read “Z is equal to X OR Y”
  - X = A’ is read “X is equal to NOT A”

Logic Function Implementation

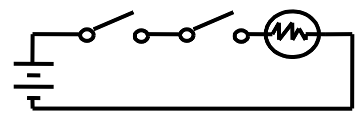
5

- Using switches
  - For inputs
    - Logic 1 is switch closed
    - Logic 0 is switch open
  - For outputs
    - Logic 1 is light ON
    - Logic 0 is light OFF
  - NOT uses a switch such that
    - Logic 1 is switch open
    - Logic 0 is switch closed

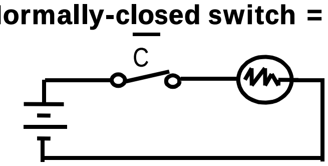
Switches in parallel => OR



Switches in series => AND

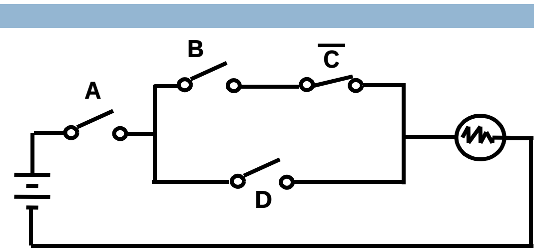


Normally-closed switch => NOT



Switch Example

6



- Light is ON (L = 1) for
  - $L(A, B, C, D) = A\bullet(B\bullet\overline{C} + D) = A(B\overline{C}+D)$
- And off otherwise
- Useful model for relay circuits and for CMOS circuits, the foundation of current digital logic

Summary of Logic Gates - I

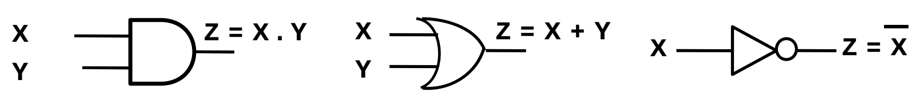
7

- Truth Table of 3 basic gates

X	Y	Z = X . Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	Z = X + Y
0	0	0
0	1	1
1	0	1
1	1	1

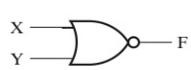
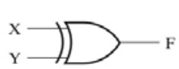
X	Z = X'
0	1
1	0



AND Gate **OR Gate** **NOT Gate**

Summary of Logic Gates - II

8

			X Y | F)
NAND		$F = \overline{X \cdot Y}$	0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y = X \oplus Y$	0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\overline{Y} = \overline{X \oplus Y}$	0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Boolean Algebra: Formal Definition

9

- A Boolean algebra (BA) is a set *A* together with
  - binary operations + and *(dot) and a unary operation -
  - and elements 0, 1 of *A*
  - such that the following laws hold:
    - commutative [x+y = y+x] and
    - associative [x+(y+z)=(x+y)+z] laws for addition and multiplication,
    - distributive laws both for
      - multiplication over addition [x+(y•z)=(x+y)(x+z)] and
      - addition over multiplication [x•(y+z)=(x•y)+(x•z)]
  - and the following special laws:

x + (x • y) = x
x • (x + y) = x
x + (x') = 1
x • (x') = 0

The Truth Table

10

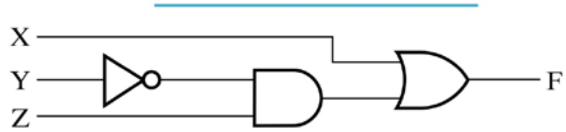
- Shows a logic circuit's output response to various input combinations
- Using logic 1 for true and 0 for false
- All input permutations are listed on the left and output on the right

Variables (Inputs)

Truth Table for the Function $F = X + \overline{Y}Z$

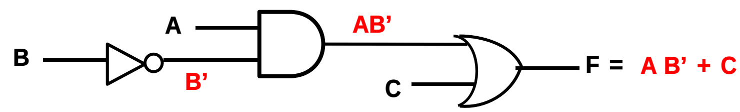
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

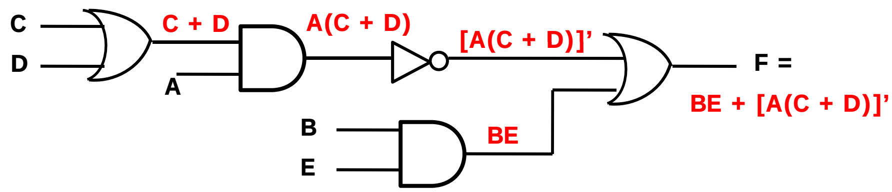
Possible combinations



Reading of a Digital Circuit

11





Truth Table – More Variables

12

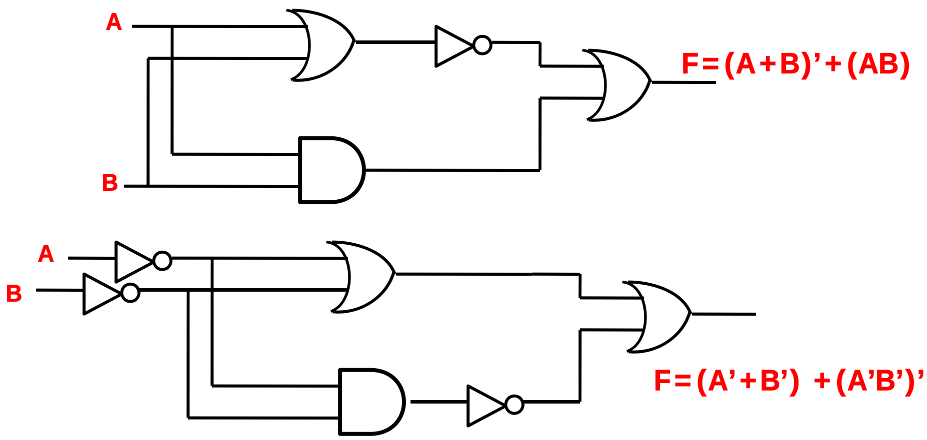
AB' + C = (A + C)(B' +C)

A B C	B'	AB'	AB'+C	A+C	B'+C	(A+C)(B'+C)
0 0 0	1	0	0	0	1	0
0 0 1	1	0	1	1	1	1
0 1 0	0	0	0	0	0	0
0 1 1	0	0	1	1	1	1
1 0 0	1	1	1	1	1	1
1 0 1	1	1	1	1	1	1
1 1 0	0	0	0	1	0	0
1 1 1	0	0	1	1	1	1

More Examples

13

- Find the Boolean expression and construct the truth table



Boolean Operator Precedence

14

- The order of Boolean operations:
  - Parentheses
  - NOT
  - AND
  - OR

- Consequence: Parentheses appear around OR expressions
- Example: F = A (B + C) (C + D')

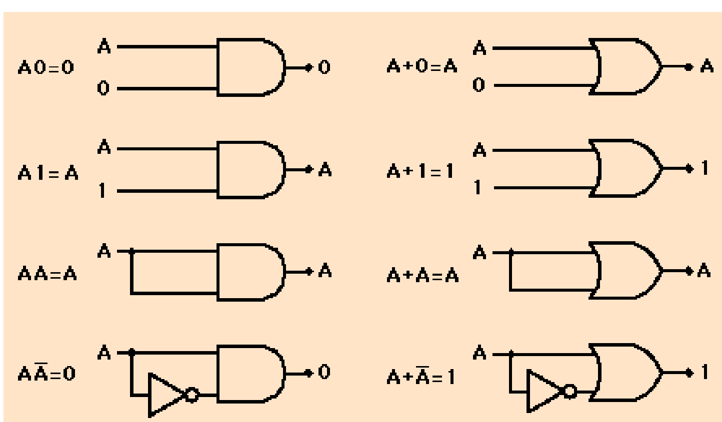
Basic Theorems – Single Variables

15

Operations with 0 and 1	X + 0 = X X + 1 = 1	X * 1 = X X * 0 = 0
Idempotent laws	X + X = X	X * X = X
Involution law	(X')' = X	
Laws of complementarity	X + X' = 1	X * X' = 0

Basic Theorems – Single Variables

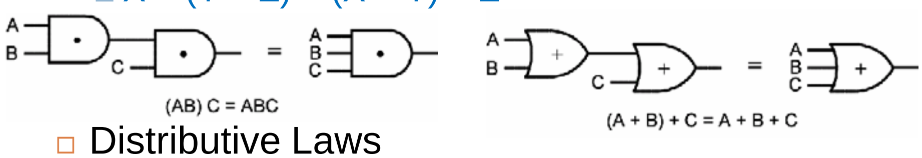
16



Commutative, Associative & Distributive Laws

17

- Commutative Laws
  - XY = YX
  - X + Y = Y + X
- Associative Laws
  - (XY)Z = XYZ
  - X + (Y + Z) = (X + Y) + Z



- Distributive Laws
  - X(Y + Z) = XY + XZ
  - X + YZ = (X + Y)(X + Z)

Algebraic Proof

18

- **X + YZ = (X + Y)(X + Z) (Distributive Identity)**

(X + Y)(X + Z) = XX + XZ + YX + YZ
= X + XZ + XY + YZ
= X·(1+Z+Y) + YZ
= X + YZ

- **AB + A'C + BC = AB + A'C (Consensus Theorem)**

AB + A'C + BC = AB + A'C + 1·BC
= AB + A'C + (A+A')·B C
= AB + A'C + ABC + A'BC
= AB·(1+C) + A'C·(1+B)
= AB + A'C

Simplification Theorems

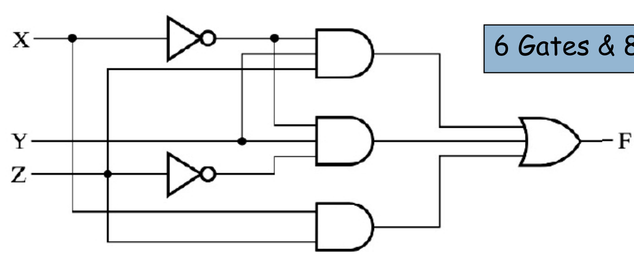
19

- The logic form in the truth table will work, but it is often inefficient and takes an unnecessarily large number of gates.
- Logic expressions can often be simplified algebraically, and although there is no fixed procedure, the following rules are often helpful.
- Use previous theorems to put the original expression in a form involving only a sum of products.
- Check the form for common factors and use the single variable theorems below to eliminate terms after factoring.

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

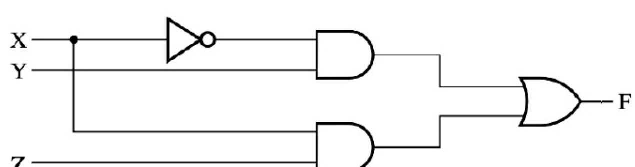
Proof of Simplification

20



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

6 Gates & 8 literals



(b) $F = \overline{X}Y + XZ$

4 Gates & 4 literals

X	Y	Z	(a) F	(b) F
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

DeMorgan's Laws

21

The most important logic theorem for digital electronics: any logical binary expression remains unchanged if we

- Change all variables to their complements.
- Change all AND operations to ORs.
- Change all OR operations to ANDs.
- Take the complement of the entire expression.

A practical way to look at DeMorgan's Theorem is:

- the inversion bar of an expression may be broken at any point and the operation at that point replaced by its opposite (i.e., AND replaced by OR or vice versa).

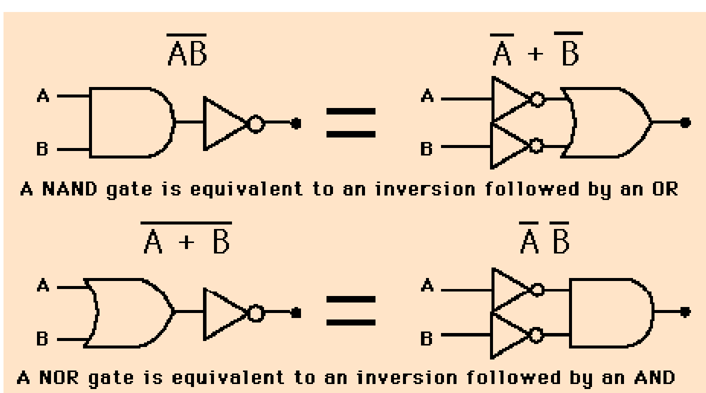
Different notations →

$\overline{(X + Y)} = \bar{X}\bar{Y}$ $\overline{(XY)} = \bar{X} + \bar{Y}$

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

DeMorgan's Laws - Logic Circuits

22



Complementing Functions

23

- Use DeMorgan's Theorem to complement a function (inverts all outcomes):
  1. Interchange AND and OR operators
  2. Complement each constant value and literal
- Example: Complement $F = x'yz' + xy'z'$

  $F' = (x + y' + z)(x' + y + z)$
- Example: Complement $G = (a + b'c)d + e'$

  $G' =$

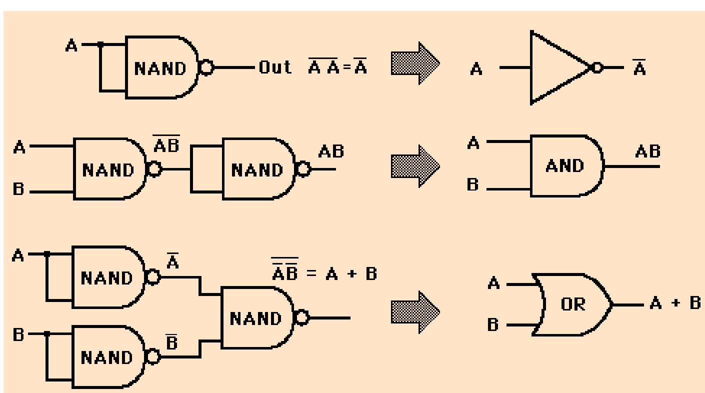
  $(a+b'c)d)' \bullet e = ((a+b'c)'+d')e$

  $= (a' \bullet (b+c')+d')e = (a'(b+c')+d')e$

DeMorgan's Laws – NAND Applications

24

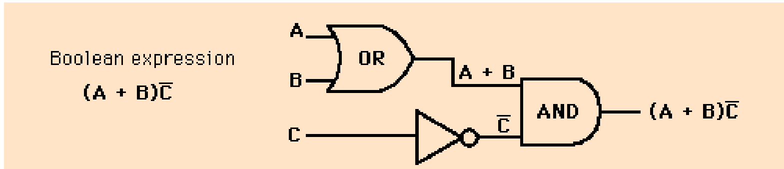
The NAND gate is called a universal gate because combinations of it can be used to accomplish all the basic functions.



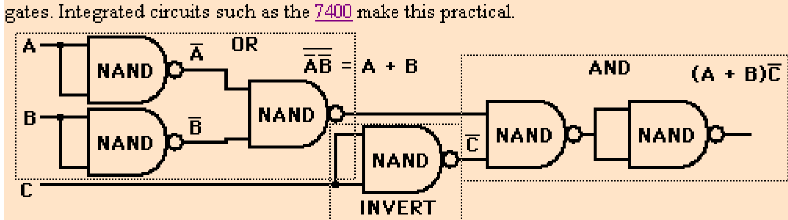
DeMorgan’s Laws – NAND Applications

25

Suppose you want a high output when either A or B is high but C is low. The Boolean expression and straightforward gate version of this are:



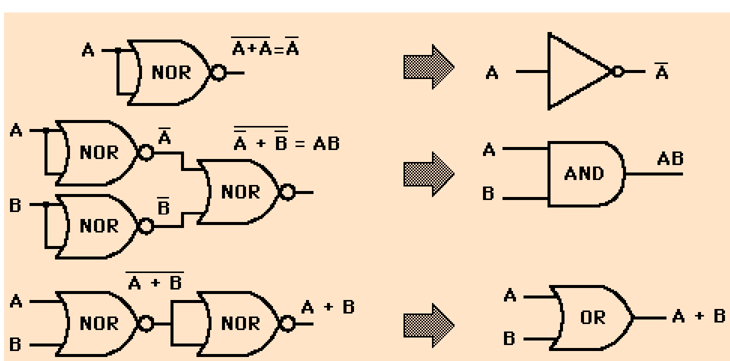
But the same task can be accomplished with NAND gates only since NAND's are universal gates. Integrated circuits such as the 7400 make this practical.



DeMorgan’s Laws – NOR Applications

26

The NOR gate is called a universal gate because combinations of it can be used to accomplish all the basic functions.

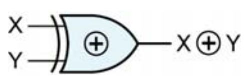


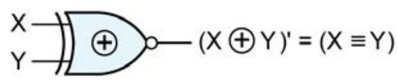
Exclusive-OR and XNOR

27

The exclusive-OR (XOR), denoted by⊕, and its invert (XNOR) are logical operations that performs the functions:

$$X \oplus Y \qquad \overline{X \oplus Y}$$





$$X \oplus Y = X\overline{Y} + \overline{X}Y \qquad \overline{X \oplus Y} = XY + \overline{X}\,\overline{Y}$$

XOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

XOR and XNOR Operations

28

$$\overline{X \oplus Y} = \overline{X\overline{Y} + \overline{X}Y} = (\overline{X} + Y)(X + \overline{Y}) = XY + \overline{X}\,\overline{Y}$$

Identities

$$X \oplus 0 = X \qquad X \oplus X = 0$$

$$X \oplus 1 = \overline{X} \qquad X \oplus \overline{X} = 1$$

$$X \oplus \overline{Y} = \overline{X \oplus Y} \qquad \overline{X} \oplus Y = \overline{X \oplus Y}$$

Laws

commutative $A \oplus B = B \oplus A$

associative $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

Summary of Boolean Identities

29

•All operations and simplifications in Boolean Algebra are based on the following identities:

1.	$X+0=X$	2.	$X\cdot 1=X$	
3.	$X+1=1$	4.	$X\cdot 0=0$	
5.	$X+X=X$	6.	$X\cdot X=X$	
7.	$X+\overline{X}=1$	8.	$X\cdot \overline{X}=0$	
9.	$\overline{\overline{X}}=X$			
10.	$X+Y=Y+X$	11.	$XY=YX$	Commutative
12.	$X+(Y+Z)=(X+Y)+Z$	13.	$X(YZ)=(XY)Z$	Associative
14.	$X(Y+Z)=XY+XZ$	15.	$X+YZ=(X+Y)(X+Z)$	Distributive
16.	$\overline{X+Y}=\overline{X}\cdot\overline{Y}$	17.	$\overline{X\cdot Y}=\overline{X}+\overline{Y}$	DeMorgan's

Logic Reduction

30

- Simplification Rules

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

- DeMorgan's Law

$\overline{(X+Y)} = \overline{X}\,\overline{Y}$	$\overline{(XY)} = \overline{X} + \overline{Y}$
(X + Y)' = X'Y'	**(XY)' = X' + Y'**

- Distributive Identity and Consensus Theorem

X + YZ = (X + Y)(X +Z) **(X + Y)(X' + Z) = XZ + X'Y**
XY + YZ + X'Z = XY + X'Z

Problem with Boolean Reduction

31

- No standard procedure
- No way to know if answer is truly the "simplest form"
- Time consuming
- Lots of room for errors – We will look at better solutions (Karnaugh Maps next week)

Boolean Reduction – An Approach

32

(1) Convert with DeMorgan's if needed

(A + B)' = A'B'; (AB)' = A' + B'

(2) Expand terms

C(A+B) = CA + CB

(3) Simplify "simple" terms

AA'BC = 0; ABC'C' = ABC'

A + AB = A; 1 + X = 1

(4) Group terms and reduce

AB'C + AB' = AB'(C+1) = AB'

(5) Look for any terms with A + A'B "2nd law"

ABC + B'C = C(AB + B') = C(A + B')(B' + B)=C(A + B')

(6) Possibly consider adding "1"

AB = AB(C+C') = ABC + ABC'

Reduction Example 1

33

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

$= (X'Y')Z + XY'$ — DeMorgan's Law
$= Y'(X'Z + X)$ — Distributive Identity
$= Y'(X + Z)(X + X')$
$= Y'(X + Z)$

Reduction Example 2

34

- $AB + A'CD + A'BD + A'CD' + ABCD$

$= AB + ABCD + A'CD + A'CD' + A'BD$
$= AB(1 + CD) + A'C(D + D') + A'BD$
$= AB + A'C + A'BD$
$= B(A + A'D) + A'C$
$= B(A + D) + A'C = AB + BD + A'C$

Reduction Examples 3

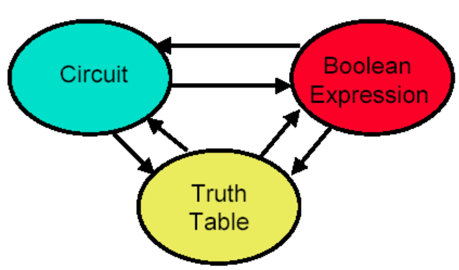
35

- Factor the expression to obtain a product-of-sum (PoS)
  - AB + C'D'
- Simplify by applying one of the theorems:
  - (A' + BC)(D'E + F)' + (DE' + F)
- Show the following equation is valid:
  - A'BC'D + (A' + BC)(A + C'D') + BC'D + A'BC' = ABCD + A'C'D' + ABD + ABCD' + BC'D

Representation Conversion

36

- Need to transition between Boolean expression, truth table and logic circuit (symbols)
- Converting from expression to truth table is easy
- Converting between expression and circuit is easy
- More difficult to convert from truth table



Truth Table to Expression

37

- Converting a truth table to an expression
- Each row with output 1 becomes a Product term
- Sum-of-Product (SOP) terms are put together

X	Y	Z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

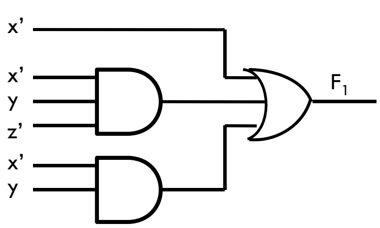
Any boolean expression can be represented as a sum-of-products form

xyz + xyz' + x'yz

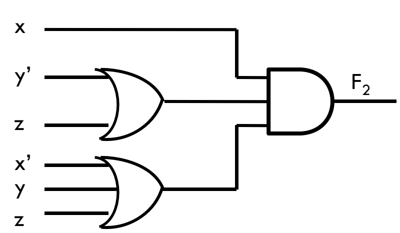
Representation of Circuits

38

- All logic circuits can be represented in 2-level format
  - Circuits can be reduced to minimal 2-level representation
  - Sum-of-Products representation is the most common in industry



Sum-of-Products (SOP)
F_1 = x' + x'yz + x'y



Product-of-Sums (POS)
F_2 = (x)(y'+z)(x'+y+z)

Minterms and Maxterms

39

- Standard forms (Canonical forms)
  - Facilitate the simplification procedures for Boolean expressions
  - Allow comparison for equality
  - Relates directly to the truth tables
  - Often results in more desirable (simple) circuits
- Sum-of-Products (SOP) (Minterms)
  - Sum of AND combination of literals
    - Example: F = A'BC + ABC' + A'B'C'
- Product-of-Sums (POS) (Maxterms)
  - Product of OR combination of literals
    - Example: F = (A+B'+C')(A'+B'+C)(A+B+C)

Minterms (mi) and Maxterms (Mi)

40

- Examples: Two variable Minterms and Maxterms

Index i	Minterms	Maxterms
0	X'Y'	X + Y
1	X'Y	X + Y'
2	XY'	X' + Y
3	XY	X' + Y'

Note that

$m_i = \overline{M_i}$

- For minterms look at what combination gives you a "1" – this gives you the index
- For maxterms, look at what combination gives you a "0" – this gives you the index

Purpose of the Index

41

- The index for the minterm and maxterm
  - Expressed as a binary number
  - Used to determine if the variable is in true form or complemented form
- For minterms
  - "1" means the variable is "True"
  - "0" means the variable is "False"
- For maxterms
  - "0" means the variable is "True"
  - "1" means the variable is "False"

Index Examples: Four Variables

42

Index i	Binary Pattern	Minterms	Maxterms
0	0000	A'B'C'D'	A+B+C+D
1	0001	A'B'C'D	A+B+C+D'
3	0011	A'B'CD	A+B+C'+D'
5	0101	A'BC'D	A+B'+C+D'
7	0111	A'BCD	A+B'+C'+D'
10	1010	AB'CD'	A'+B+C'+D
13	1101	ABC'D	A'+B'+C+D'
15	1111	ABCD	A'+B'+C'+D'

Minterm Representation

43

- Focus on the TRUE (1) cases for the function
  - Identify minterms involved
  - Sum (OR) minterms for a shorthand representation

X	Y	Z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$G = x'yz + xyz' + xyz$$

↓

$$G = m_3 + m_6 + m_7$$

$$= \Sigma(3, 6, 7)$$

Maxterm Representation

44

- Focus on the FALSE (0) cases for the function
  - Identify maxterms involved
  - Multiply (AND) maxterms for a shorthand representation

X	Y	Z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$G = (x + y + z)(x + y + z')(x + y' + z)$$
$$(x' + y + z)(x' + y + z')$$

↓

$$G = M_0\ M_1\ M_2\ M_4\ M_5$$

$$= \Pi(0, 1, 2, 4, 5)$$

Conversion Between Canonical Forms

45

- To convert SoP (minterms) to PoS (maxterms) representations
  - Pick the terms missing from the full set

X	Y	Z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

G = x'yz + xyz' + xyz

↓

$G = m_3 + m_6 + m_7 = \Sigma(3, 6, 7)$ (SOP)

↓

$G = M_0 M_1 M_2 M_4 M_5$

$= \Pi(0, 1, 2, 4, 5)$ (POS)

↓

G = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)(x' + y + z')

Complementing Functions

46

- To obtain the complement of function
  - Select the missing minterms in the SOP form
  - Or, simply use the PoS with the <u>same</u> indices

X	Y	Z	G	G'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

(1) $F(x, y, z) = \Sigma_m(1,3,5,7)$

$F'(x, y, z) = \Sigma_m(0,2,4,6)$

$F'(x, y, z) = \Pi_M(1,3,5,7)$

(2) G = xyz + xyz' + x'yz

G'=(xyz + xyz' + x'yz)'

G'=(xyz)'(xyz')'(x'yz)'

=(x'+y'+z')(x'+y'+z)(x+y'+z')

$=M_7M_6M_3=M_3M_6M_7$

Minterm and Maxterm Expansions

47

- Example: Find the minterm expansion of f(a,b,c,d) = a'(b'+d)+acd'

- F = a'b' + a'd + acd' (Not a standard SOP form)

F = a'b'c'd' + a'b'cd' + a'b'c'd + a'b'cd + a'b'c'd + a'b'cd + a'bc'd + a'bcd + abcd' + ab'cd'

= 0000 + 0010 + 0001 + 0011 + 0001 + 0011 + 0101 + 0111 + 1110 + 1010

= Σ (0, 1, 2, 3, 5, 7, 10, 14)

Maxterm f(a,b,c,d) = Π(4, 6, 8, 9, 11, 12, 13, 15)