# Boolean Algebra

ECE 152A – Summer 2009

## Reading Assignment

- Brown and Vranesic
  - 2 Introduction to Logic Circuits
    - 2.5 Boolean Algebra
      - 2.5.1 The Venn Diagram
      - 2.5.2 Notation and Terminology
      - 2.5.3 Precedence of Operations
    - 2.6 Synthesis Using AND, OR and NOT Gates
      - 2.6.1 Sum-of-Products and Product of Sums Forms

# Reading Assignment

- Brown and Vranesic (cont)
  - 2 Introduction to Logic Circuits (cont)
    - 2.7 NAND and NOR Logic Networks
    - 2.8 Design Examples
      - 2.8.1 Three-Way Light Control
      - 2.8.2 Multiplexer Circuit



# Reading Assignment

- Roth
  - 2 Boolean Algebra
    - 2.3 Boolean Expressions and Truth Tables
    - 2.4 Basic Theorems
    - 2.5 Commutative, Associative, and Distributive Laws
    - 2.6 Simplification Theorems
    - 2.7 Multiplying Out and Factoring
    - 2.8 DeMorgan's Laws

# Reading Assignment

- Roth (cont)
  - 3 Boolean Algebra (Continued)
    - 3.1Multiplying Out and Factoring Expressions
    - 3.2 Exclusive-OR and Equivalence Operation
    - 3.3 The Consensus Theorem
    - 3.4 Algebraic Simplification of Switching Expressions



# Reading Assignment

- Roth (cont)
  - 4 Applications of Boolean Algebra
    Minterm and Maxterm Expressions
    - 4.3 Minterm and Maxterm Expansions
  - 7 Multi-Level Gate Circuits
    NAND and NOR Gates
    - 7.2 NAND and NOR Gates
    - 7.3 Design of Two-Level Circuits Using NAND and NOR Gates
    - 7.5 Circuit Conversion Using Alternative Gate Symbols

# Boolean Algebra

- Axioms of Boolean Algebra
  - Axioms generally presented without proof

$$0 \cdot 0 = 0 \qquad 1 + 1 = 1$$
$$1 \cdot 1 = 1 \qquad 0 + 0 = 0$$
$$0 \cdot 1 = 1 \cdot 0 = 0 \qquad 1 + 0 = 0 + 1 = 1$$
$$\text{if } X = 0\text{, then } X' = 1 \qquad \text{if } X = 1\text{, then } X' = 0$$



# Boolean Algebra

- The Principle of Duality

  from Zvi Kohavi, Switching and Finite Automata Theory

  *"We observe that all the preceding properties are grouped in pairs. Within each pair one statement can be obtained from the other by interchanging the OR and AND operations and replacing the constants 0 and 1 by 1 and 0 respectively. Any two statements or theorems which have this property are called dual, and this quality of duality which characterizes switching algebra is known as the principle of duality. It stems from the symmetry of the postulates and definitions of switching algebra with respect to the two operations and the two constants. The implication of the concept of duality is that it is necessary to prove only one of each pair of statements, and its dual is henceforth proved."*

## Boolean Algebra

- Single-Variable Theorems
  - Theorems can be proven with truth tables
    - Truth table proof a.k.a., "Perfect Induction"

$$X \cdot 0 = 0 \qquad X + 1 = 1$$
$$X \cdot 1 = X \qquad X + 0 = X$$
$$X \cdot X = X \qquad X + X = X$$
$$X \cdot X' = 0 \qquad X + X' = 1$$
$$(X')' = X$$



## Boolean Algebra

- Two- and Three-Variable Properties
  - Commutative

$$X \cdot Y = Y \cdot X \qquad X + Y = Y + X$$

  - Associative

$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z \qquad X + (Y + Z) = (X + Y) + Z$$

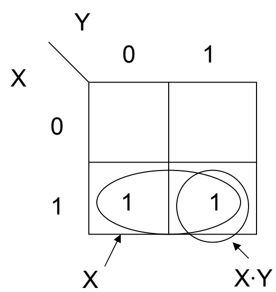
  - Distributive

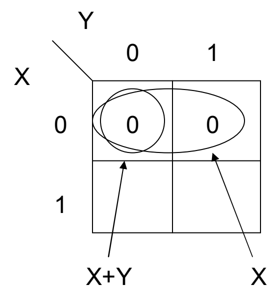
$$X \cdot (Y + Z) = X \cdot Y + X \cdot Z \qquad X + (Y \cdot Z) = (X + Y)\cdot(X + Z)$$

# Boolean Algebra

- Absorption (Simplification)

$X + X \cdot Y = X$ $\qquad$ $X \cdot (X + Y) = X$



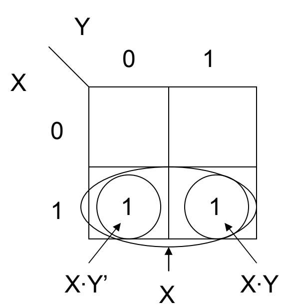



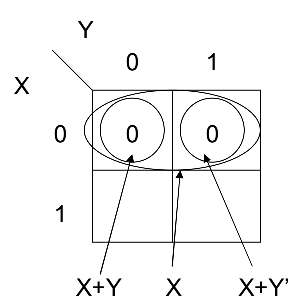



# Boolean Algebra

- Combining (Simplification)

$X \cdot Y + X \cdot Y' = X$ $\qquad$ $(X + Y) \cdot (X + Y') = X$

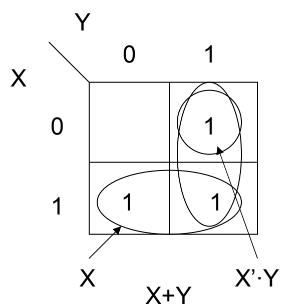


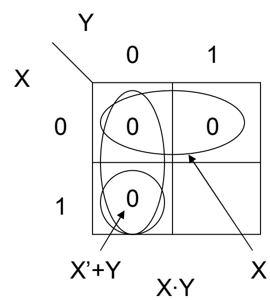



# Boolean Algebra

- Redundant Coverage (simplification)

$$X + X' \cdot Y = X + Y \qquad X \cdot (X' + Y) = X \cdot Y$$



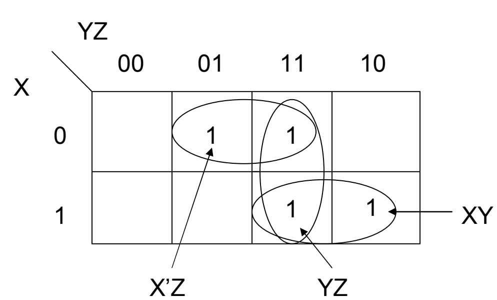






# Boolean Algebra

- The Consensus Theorem

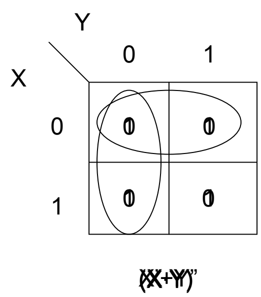
$$XY + X'Z + YZ = XY + X'Z$$

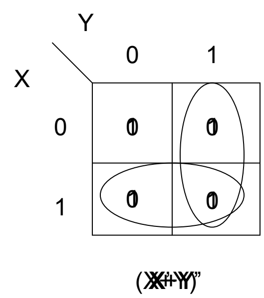


# Boolean Algebra

- DeMorgan's Theorem

$(X \cdot Y)' = X' + Y'$ $\qquad$ $(X + Y)' = X' \cdot Y'$






# Boolean Expressions

- Precedence of Operations
  - Order of evaluation
    - 1. NOT
    - 2. AND
    - 3. OR
      - Or forced by parentheses
  - Example: F = ab'c + a'b + a'bc' + b'c'
    - a=0, b=0 and c=1
      - NOT: 011 + 10 + 100 + 10
      - AND: 0 + 0 + 0 + 0
      - OR: 0

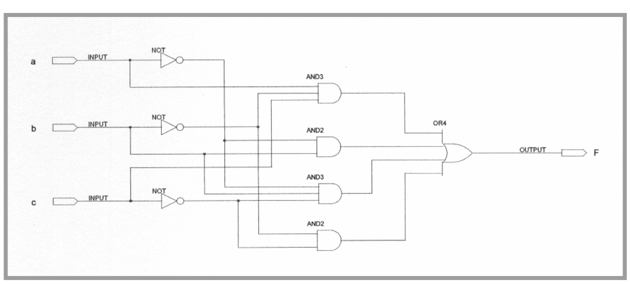
## Boolean Expressions, Logic Networks, Karnaugh Maps, Truth Tables & Timing Diagrams

- Derive Logic Network, Karnaugh Map, Truth Table and Timing Diagram from:
  - F = ab'c + a'b + a'bc' + b'c'
    - 3 variables, 10 literals, 4 product terms
    - Expression is in Standard Sum-of-Products form
      - i.e., the function is the sum (or logical OR) or the four product (or logical AND) terms
      - The alternative standard form is Product-of-Sums
    - The expression "implies" structure
      - Direct realization with AND, OR and NOT functions



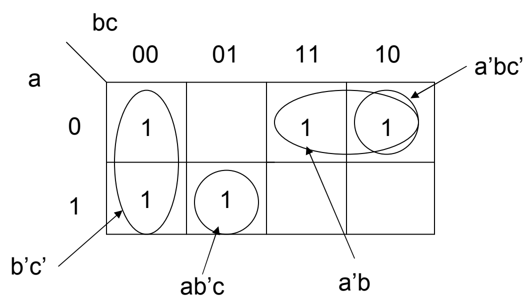
## Boolean Expressions, Logic Networks, Karnaugh Maps, Truth Tables & Timing Diagrams

- Logic Network
  - F = ab'c + a'b + a'bc' + b'c'



# Boolean Expressions, Logic Networks, Karnaugh Maps, Truth Tables & Timing Diagrams

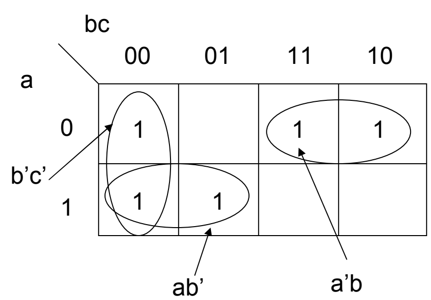
- Karnaugh Map
  - F = ab’c + a’b + a’bc’ + b’c’






# Boolean Expressions, Logic Networks, Karnaugh Maps, Truth Tables & Timing Diagrams

- Note possible simplification
  - Redundant coverage (eliminates literal) and absorption (eliminates product term)



## Boolean Expressions, Logic Networks, Karnaugh Maps, Truth Tables & Timing Diagrams

- Truth Table
  - F = ab’c + a’b + a’bc’ + b’c’

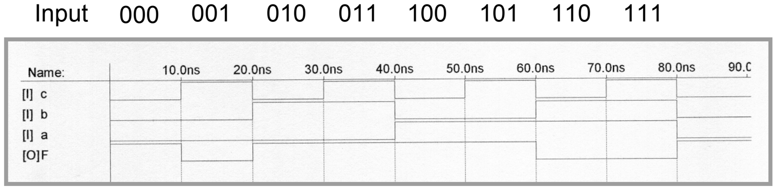
| a | b | c | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



## Boolean Expressions, Logic Networks, Karnaugh Maps, Truth Tables & Timing Diagrams

- Timing Diagram (Functional Simulation)
  - F = ab’c + a’b + a’bc’ + b’c’

Input 000 001 010 011 100 101 110 111




Output 1 0 1 1 1 1 0 0

# Minterms and Maxterms

- Minterm
  - A product term which contains each of the *n* variables as factors in either complemented or uncomplemented form is called a *minterm*
    - Example for 3 variables: ab'c is a *minterm*; ab' is not
- Maxterm
  - A sum term which contains each of the *n* variables as factors in either complemented or uncomplemented form is called a *maxterm*
    - For 3 variables: a'+b+c' is a *maxterm*; a'+b is not



# Minterms and Maxterms

- Minterm and Maxterm Expansion
  - Three variable example:

| Row number | $x_1$ | $x_2$ | $x_3$ | Minterm | Maxterm |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$ | $M_0 = x_1 + x_2 + x_3$ |
| 1 | 0 | 0 | 1 | $m_1 = \overline{x}_1\overline{x}_2x_3$ | $M_1 = x_1 + x_2 + \overline{x}_3$ |
| 2 | 0 | 1 | 0 | $m_2 = \overline{x}_1x_2\overline{x}_3$ | $M_2 = x_1 + \overline{x}_2 + x_3$ |
| 3 | 0 | 1 | 1 | $m_3 = \overline{x}_1x_2x_3$ | $M_3 = x_1 + \overline{x}_2 + \overline{x}_3$ |
| 4 | 1 | 0 | 0 | $m_4 = x_1\overline{x}_2\overline{x}_3$ | $M_4 = \overline{x}_1 + x_2 + x_3$ |
| 5 | 1 | 0 | 1 | $m_5 = x_1\overline{x}_2x_3$ | $M_5 = \overline{x}_1 + x_2 + \overline{x}_3$ |
| 6 | 1 | 1 | 0 | $m_6 = x_1x_2\overline{x}_3$ | $M_6 = \overline{x}_1 + \overline{x}_2 + x_3$ |
| 7 | 1 | 1 | 1 | $m_7 = x_1x_2x_3$ | $M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$ |

$$(m_i)' = M_i \ldots and \ldots (M_i)' = m_i$$

## Sum-of-Products Form

- Canonical Sum-of-Products (or Disjunctive Normal) Form
  - *The sum of all minterms derived from those rows for which the value of the function is 1 takes on the value 1 or 0 according to the value assumed by f. Therefore this sum is in fact an algebraic representation of f. An expression of this type is called a canonical sum of products, or a disjunctive normal expression.*

Kohavi



## Minterms and Maxterms

- Truth Table from earlier example
  - F = ab’c + a’b + a’bc’ + b’c’

| | | a | b | c | F |
|---|---|---|---|---|---|
| $m_0$ | $M_0$ | 0 | 0 | 0 | 1 = $a_0$ |
| $m_1$ | $M_1$ | 0 | 0 | 1 | 0 = $a_1$ |
| $m_2$ | $M_2$ | 0 | 1 | 0 | 1 = $a_2$ |
| $m_3$ | $M_3$ | 0 | 1 | 1 | 1 = $a_3$ |
| $m_4$ | $M_4$ | 1 | 0 | 0 | 1 = $a_4$ |
| $m_5$ | $M_5$ | 1 | 0 | 1 | 1 = $a_5$ |
| $m_6$ | $M_6$ | 1 | 1 | 0 | 0 = $a_6$ |
| $m_7$ | $M_7$ | 1 | 1 | 1 | 0 = $a_7$ |

## Sum-of-Products

- Canonical Sum-of-Products
  - F = ab'c + a'b + a'bc' + b'c'

$$F = a_0 m_0 + a_1 m_1 + a_2 m_2 + a_3 m_3 + a_4 m_4 + a_5 m_5 + a_6 m_6 + a_7 m_7$$
$$F = 1m_0 + 0m_1 + 1m_2 + 1m_3 + 1m_4 + 1m_5 + 0m_6 + 0m_7$$
$$F = 1m_0 + 1m_2 + 1m_3 + 1m_4 + 1m_5$$
$$F = a'b'c' + a'bc' + a'bc + ab'c' + ab'c$$
$$F = \sum m(0,2,3,4,5)$$



## Product-of-Sums Form

- Canonical Product-of-Sums (or Conjunctive Normal) Form
  - *An expression formed of the product of all maxterms for which the function takes on the value 0 is called a canonical product of sums, or a conjunctive normal expression.*

## Product-of-Sums

- Canonical Product-of-Sums
  - F = ab'c + a'b + a'bc' + b'c'

$$F = (a_0 + M_0)(a_1 + M_1)(a_2 + M_2)(a_3 + M_3)(a_4 + M_4)(a_5 + M_5)(a_6 + M_6)(a_7 + M_7)$$

$$F = (1 + M_0)(0 + M_1)(1 + M_2)(1 + M_3)(1 + M_4)(1 + M_5)(0 + M_6)(0 + M_7)$$

$$F = (0 + M_1)(0 + M_6)(0 + M_7)$$

$$F = (a + b + c')(a' + b' + c)(a' + b' + c')$$

$$F = \prod M(1,6,7)$$



## General Sum-of-Product (SOP) and Product-of-Sums (POS) Forms

- $a_i$ is the Boolean value of the function in the $i^{th}$ row of an *n*-variable Truth Table

$$F = a_0 m_0 + a_1 m_1 + m_1 + \cdots + a_i m_i = \sum_{i=0}^{2^n - 1} a_i m_i$$

$$F = (a_0 + M_0)(a_1 + M_1) \cdots (a_i + M_i) = \prod_{i=0}^{2^n - 1} (a_i + M_i)$$

$$F' = \sum_{i=0}^{2^n - 1} a'_i \, m_i = \prod_{i=0}^{2^n - 1} (a'_i + M_i)$$

## Equivalence of SOP and POS Forms

- Minterm / Maxterm Lists

$$F = \sum_{i=0}^{2^n-1} a_i m_i = \prod_{i=0}^{2^n-1} (a_i + M_i)$$

$$F_{example} = \sum m(0,2,3,4,5) = \prod M(1,6,7)$$

$$and$$

$$F' = \sum_{i=0}^{2^n-1} a'_i\, m_i = \prod_{i=0}^{2^n-1} (a'_i + M_i)$$

$$F'_{example} = \sum m(1,6,7) = \prod M(0,2,3,4,5)$$

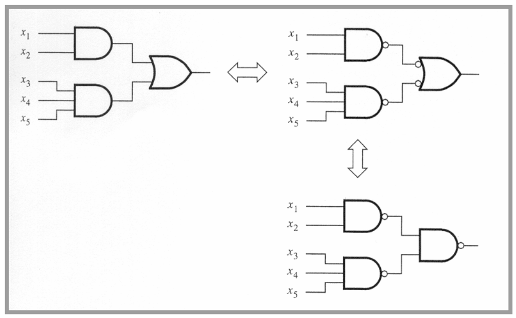


## Functionally Complete Operations

- *A set of operations is said to be functionally complete (or universal) if and only if every switching function can be expressed entirely by means of operations from this set*
  - *[Since] every switching function can be expressed in a canonical sum-of-products [and product-of-sums] form, where each expression consists of a finite number of switching variables, constants and the operations AND, OR and NOT [this set of operations is functionally complete]*

## SOP Realization with NAND/NAND

- The NAND operation is functionally complete






## POS Realization with NOR/NOR

- The NOR operation is functionally complete