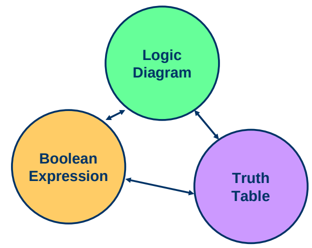
Chapter 3

Boolean Algebra and Combinational Logic

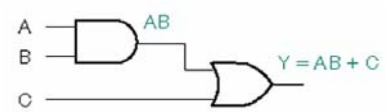
Objective



Logic Gate Network

- Two or more logic gates connected together
- Described by truth table, logic diagram or Boolean expression

Logic Gate Network



Boolean Expression from Logic Gate Network

- Similar to finding the expression for a single gate
- Inputs may be compound expressions that represent outputs from previous gates

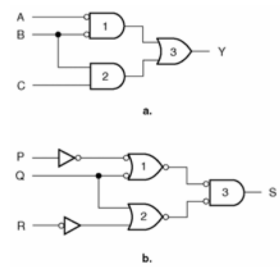
Bubble-to-Bubble Convention

- Choose gate symbols so that outputs with bubbles connect to inputs with bubbles
- Results in a cleaner notation and a clearer idea of the circuit function

Simplification by Double Inversion

- **In logic circuits, when two bubbles touch, they cancel out**
- **In Boolean expressions, bars of the same length cancel**

Bubble-to-Bubble / Double Inversion (example)



$Y = \bar{A}\bar{B}+BC$

$S = [\overline{\overline{\bar{\bar{P}}+\bar{Q}}}][\overline{\overline{Q+\bar{R}}}]$

$= (P+\bar{Q})(Q+\bar{R})$

Order of Precedence

- **Unless otherwise specified, in Boolean expressions AND functions are performed first, followed by ORs**

  Example: Y = A & B + C + D

- **To change the order of precedence, use parentheses**

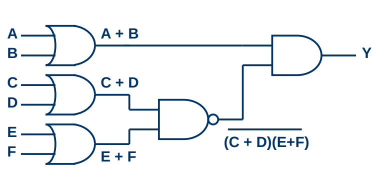
  Example: Y = A & (B + C + D)

Logic Diagrams from Boolean Expressions

- **Called *synthesis***
- **Use order of precedence**
  - **A bar over a group of variables is the same as having those variables in parentheses**
- **Create levels of gating based on orders of precedence**

Logic Diagrams from Boolean Expressions (example)

Synthesize: $Y = (A + B)\overline{(C + D)(E + F)}$



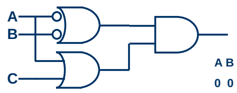
Truth Tables from Logic Diagrams or Boolean Expressions

- **Two methods:**

  1) **Combine individual truth tables from each gate into a final output truth table**
  2) **Develop a Boolean expression and use it to fill in the truth table**

  Which is more thorough? Which is more efficient?

Truth Table (Method 1)



A B C	$(\bar{A} + \bar{B})$	$(A + C)$	$(\bar{A} + \bar{B})(A + C)$
0 0 0	1	0	0
0 0 1	1	1	1
0 1 0	1	0	0
0 1 1	1	1	1
1 0 0	1	1	1
1 0 1	1	1	1
1 1 0	0	1	0
1 1 1	0	1	0

Circuit Description Using Boolean Expressions

- **Definitions:**
  - **A *product term* is a part of a Boolean expression where one or more true or complement variables are *ANDed***
  - **A *sum term* is a part of a Boolean expression where one or more true or complement variables are ORed**

SOP and POS

- **A *sum-of-products (SOP)* is a Boolean expression where several product terms are summed (*ORed*) together**

  Example: $Y = AB + CD + AD$

- **A *product-of-sums (POS)* is a Boolean expression where several sum terms are multiplied (*ANDed*) together**

  Example: $P = (A + B)(C + D)(E + F)$

Minterms and Maxterms

- **Minterm**
  - **A product term in a Boolean expression where all possible input variables appear once in true or complemented form**

    Examples: ABC; $\bar{A}B\bar{C}$

- **Maxterm**
  - **A sum term in a Boolean expression where all possible variables appear once in true or complemented form**

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

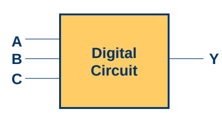
Deriving a SOP Expression from a Truth Table

1) Each line of the truth table with a 1 (HIGH) output represents a product term (a minterm) in the truth table's Boolean expression

2) Write all input variables for each minterm in true or complement form
- **If an input variable's value is 0, write it in complement form (with a bar over it)**
- **If an input variable's value is 1, write it in true form (no bar)**

3) Combine all minterms as a sum (OR them)

SOP (example)



A B C	Y
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	0

Apply all possible inputs to fill out truth table

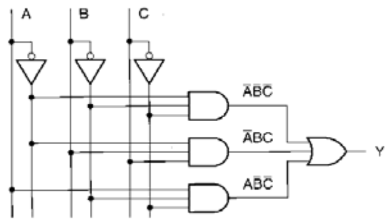
Find Minterms:

$\bar{A}\bar{B}\bar{C}$, $\bar{A}BC$, $A\bar{B}\bar{C}$

Bus Form

Bus Form:

A schematic convention in which each variable is available in true or complement form (or *double-rail*) at any point along a conductor



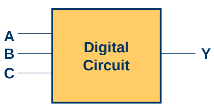
Deriving a POS Expression from a Truth Table

1) Each line of the truth table with a 0 (LOW) output represents a sum term (a maxterm) in the truth table's Boolean expression

2) Write all input variables for each maxterm in true or complement form
   - If a variable's value is 1, write it in complement form (with a bar over it)
   - If a variable's value is 0, write it in true form (no bar)

3) Combine all maxterms as a product (AND them)

POS (example)



A B C	Y
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	0

Apply all possible inputs to fill out the truth table

Find Maxterms:

$(A + B + \bar{C})$, $(A + \bar{B} + C)$, $(\bar{A} + B + \bar{C})$

$(\bar{A} + \bar{B} + C)$, $(\bar{A} + \bar{B} + \bar{C})$

Exercise

- Derive the SOP form for an XOR gate

Theorems of Boolean Algebra

- 24 theorems
- Used to minimize a Boolean expression to reduce the number of logic gates in a network

Commutative Property

- An operation is commutative if it can be applied to its operands in any order without effecting the result
- AND and OR are commutative
  - Theorem 1: xy = yx.
  - Theorem 2: x + y = y + x

Associative Property

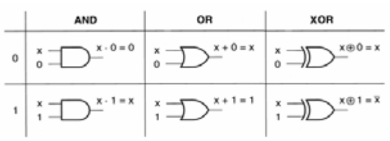
- An operation is associative if its operands can be grouped in any order without effecting the result
- AND and OR are associative
  - Theorem 3: (xy)z = x(yz) = (xz)y
  - Theorem 4: (x + y) + z = x + (y + z) = (x + z) + y

Distributive Property (of Multiplication over Addition)

- The property allows us to distribute (multiply through) an AND across several OR functions
  - Theorem 5: x(y + z) = xy + xz
  - Theorem 6: (x + y)(w + z) = xw + xz + yw + yz

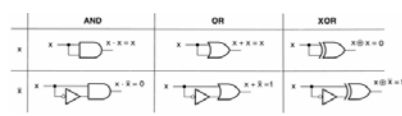
Operations with 1 and 0

- Theorem 7: $x \cdot 0 = 0$
- Theorem 8: x + 0 = x
- Theorem 9: $x \oplus 0 = x$
- Theorem 10: $x \cdot 1 = x$
- Theorem 11: x + 1 = 1
- Theorem 12: $x \oplus 1 = \overline{x}$



Operations with One Variable

- The same variable
  - Theorem 13: $x \cdot x = x$
  - Theorem 14: x + x = x
  - Theorem 15: $x \oplus x = 0$
- The complement of the variable
  - Theorem 16: $x \cdot \overline{x} = 0$
  - Theorem 17: $x + \overline{x} = 1$
  - Theorem 18: $x \oplus \overline{x} = 1$



Double Inversion

- Theorem 19: $x = \overline{\overline{x}}$

DeMorgan's Theorems

- Theorem 20: $\overline{x \cdot y} = \overline{x} + \overline{y}$
- Theorem 21: $\overline{x + y} = \overline{x} \cdot \overline{y}$

Multivariable Theorems

- Theorem 22: $x + xy = x$
- Theorem 23: $(x + y)(x + z) = x + yz$
- Theorem 24: $x + \overline{x}y = x + y$

Simplification by Karnaugh Mapping

- A Karnaugh map, called a k-map, is a graphical tool used for simplifying Boolean expressions
- A variation of the Venn diagram

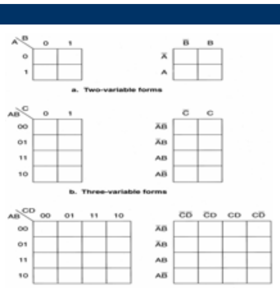
Construction of a Karnaugh Map

- Square or rectangle divided into cells
- Each cell represents a line in the truth table
- Cell contents are the value of the output variable on that line of the truth table

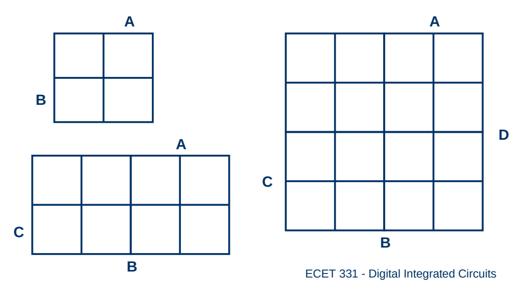
K-Map Cell Locations

- Adjacent cells differ by only one variable
- Grouping adjacent cells allows canceling variables in their true and complement forms

K-Map Forms



Other K-Map Forms



Loading a K-Map from a Truth Table

- Each line in a truth table has a numerical according to the binary combination of the inputs
- Each cell in a K-map has a binary address – although not in order due to the need for logical cell adjacency
- Load the cells address of the K-map with the function output for the same line of the truth table

Loading a K-map from a truth table

A B C	#	Y
0 0 0	0	1
0 0 1	1	0
0 1 0	2	0
0 1 1	3	1
1 0 0	4	1
1 0 1	5	0
1 1 0	6	0
1 1 1	7	0

Cell Addresses (A, B, C)

	0	2	6	4
C	1	3	7	5

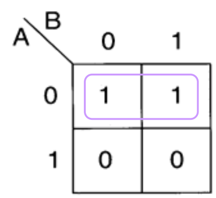
Truth Table Outputs (A, B, C)

	1	0	0	1
C	0	1	0	0

Grouping Cells

- Cells can be grouped as singles, pairs, quads and octets or hexadectets (i.e. in powers of 2)
- A pair cancels 1 variable
- A quad cancels 2 variables
- An octet cancels 3 variables
- A hexadectet cancels 4 variables

Grouping Cells (example)

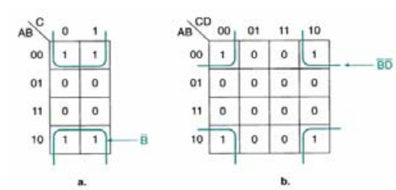


$Y = \overline{A}$

Grouping Cells Along Outside Edges

- The cells along an outside edge are adjacent to cells along the opposite edge
- In a four-variable map, the four corner cells are adjacent

Grouping Cells Along Outside Edges (example)



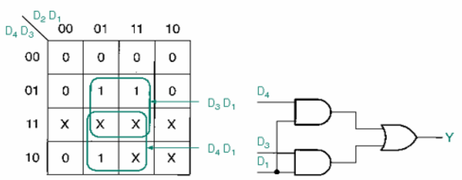
Multiple Groups

- Each group is a term in the maximum SOP simplified expression
- A cell may be grouped more than once as long as every group has at least one cell that does not belong to any other group

Don't Care States

- The output state of a circuit for a combination of inputs that will never occur
- We can regard this state as a 1 or 0; whichever will yield maximum simplification
- Shown in a K-map as an "x"

Don't Care States (example)



a. Karnaugh map   b. Logic diagram

POS Simplification Using K-maps

- Group those cells with values of 0
- Use the complements of the cell coordinates as the sum term