CS430 Computer Architecture

Spring 2015

Chapter 11
Digital Logic

- Reading: pp.365-384
- Good Problems to Work: 11.1 a., 11.3 a., 11.4, 11.5, 11.6, 11.8

Boolean Algebra

- Boolean algebra is a convenient tool for:
  1. describing the function of digital circuitry
  2. simplifying the implementation of said function
- Boolean algebra uses
  1. logical variables (values are 0 or 1)
  2. operations on logical variables

Boolean Operators of Two Input Variables

P	Q	NOT P $(\overline{P})$	P AND Q $(P \bullet Q)$	P OR Q $(P + Q)$	P NAND Q $(\overline{P \bullet Q})$	P NOR Q $(\overline{P + Q})$	P XOR Q $(P \oplus Q)$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

Truth table of each logical operation

Logical Operators

- AND can be $A * B, A \cdot B, or\ AB$
- OR is $A + B$
- NOT can be $A' or\ \bar{A}$

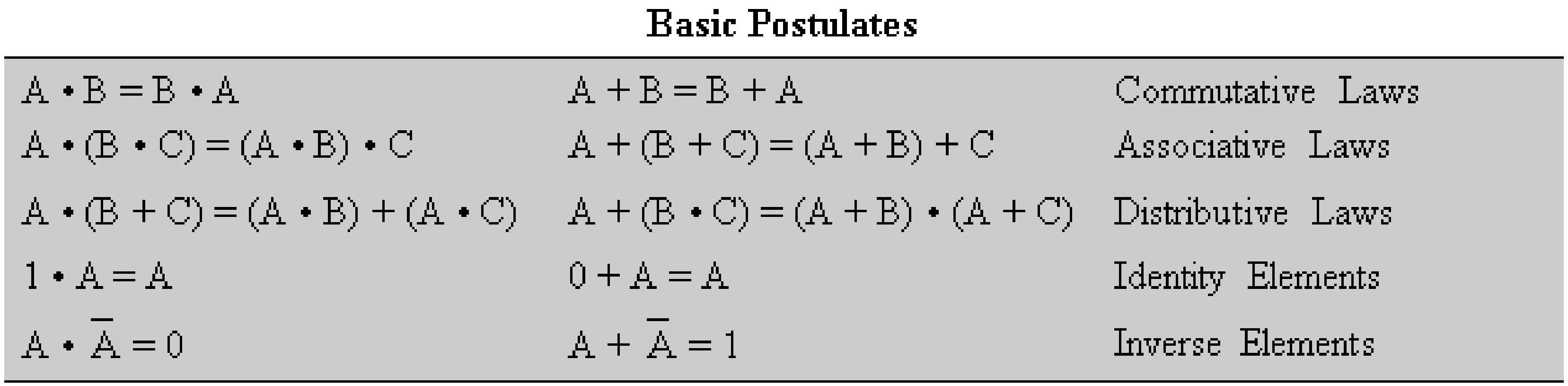
- Note: In the absence of parentheses, AND takes precedence over OR

Boolean Operators of n Input Variables

Operation	Expression	Output = 1 if
AND	$A \bullet B \bullet \ldots$	All of the set {A, B, …} are 1.
OR	$A + B + \ldots$	Any of the set {A, B, …} are 1.
NAND	$\overline{A \bullet B \bullet \ldots}$	Any of the set {A, B, …} are 0.
NOR	$\overline{A + B + \ldots}$	All of the set {A, B, …} are 0.
XOR	$A \oplus B \oplus \ldots$	The set {A, B, …} contains an odd number of ones.

Boolean Algebra

- Basic Postulates



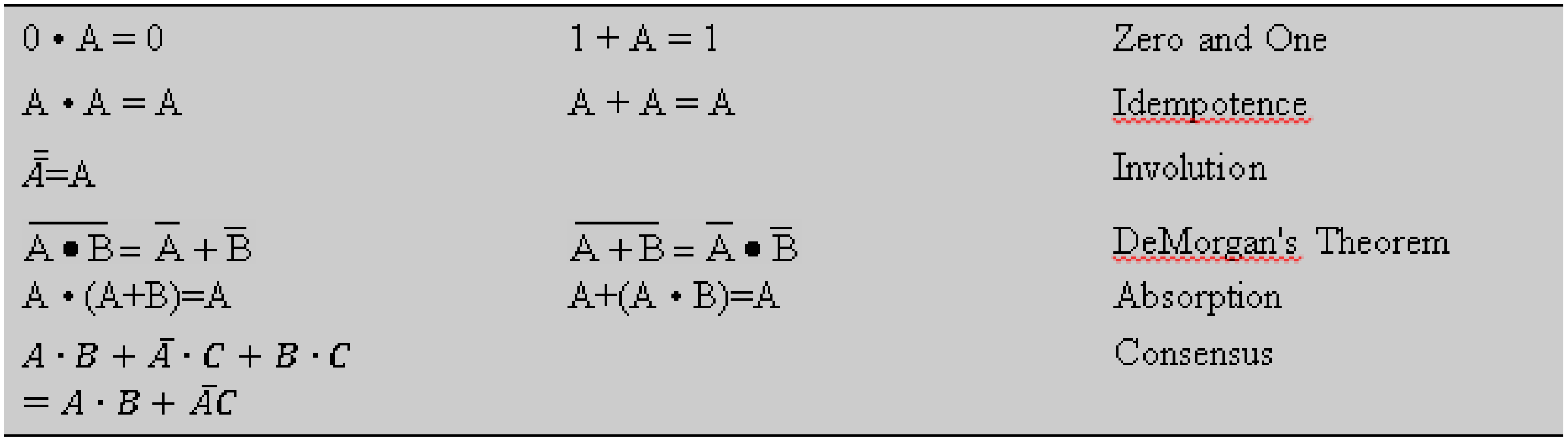
Basic Postulates

$A \bullet B = B \bullet A$	$A + B = B + A$	Commutative Laws
$A \bullet (B \bullet C) = (A \bullet B) \bullet C$	$A + (B + C) = (A + B) + C$	Associative Laws
$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$	$A + (B \bullet C) = (A + B) \bullet (A + C)$	Distributive Laws
$1 \bullet A = A$	$0 + A = A$	Identity Elements
$A \bullet \overline{A} = 0$	$A + \overline{A} = 1$	Inverse Elements

Boolean Algebra

- Theorems that can be derived from the postulates

Theorems



$0 \bullet A = 0$	$1 + A = 1$	Zero and One
$A \bullet A = A$	$A + A = A$	Idempotence
$\bar{\bar{A}} = A$		Involution
$\overline{A \bullet B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \bullet \bar{B}$	DeMorgan's Theorem
$A \bullet (A+B) = A$	$A + (A \bullet B) = A$	Absorption
$A \cdot B + \bar{A} \cdot C + B \cdot C = A \cdot B + \bar{A}C$		Consensus

Problem

1. Prove $A \cdot A = A$

2. Prove $A + AB = A$

3. Prove $A + AB = A$ using perfect induction with truth tables

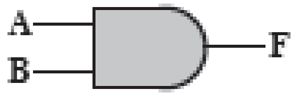
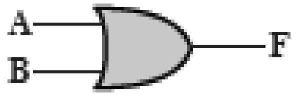
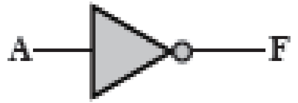
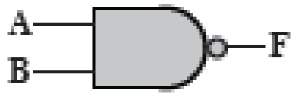
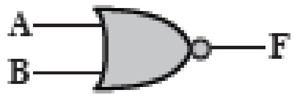
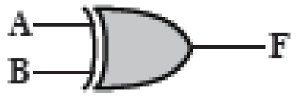
Problem

- The complement of any Boolean expression can be found using DeMorgan's Theorem
  1. Find the complement of $X + (YZ)$
  2. Using a truth table, show you really found the complement

Gates

- gate
  1. an electronic circuit that operates on one or more signals producing an output signal
  2. fundamental building block of ALL digital logic circuits
- interconnections of these gates allow us to build simple or complicated logical functions

Basic Logic Gates

Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \cdot B$ or $F = AB$	A B F 0 0 0 0 1 0 1 0 0 1 1 1
OR		$F = A + B$	A B F 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$F = \overline{A}$ or $F = A'$	A F 0 1 1 0
NAND		$F = \overline{AB}$	A B F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$F = \overline{A + B}$	A B F 0 0 1 0 1 0 1 0 0 1 1 0
XOR		$F = A \oplus B$	A B F 0 0 0 0 1 1 1 0 1 1 1 0

Logical Circuit

- When designing a logical circuit, the designer works from two sets of known values
  1. the input states that the logical circuit can take
  2. the output states for each input

Logical Circuit

- A logical network has two inputs, X and Y, and a single output Z. The relationship between the inputs and the outputs is:
  1. When X and Y are 0, Z is 0
  2. When X and Y are 1, Z is 0
  3. When X is 0 and Y is 1, Z is 0
  4. When X is 1 and Y is 0, Z is 1

Logical Circuit

- Expressing this relationship, we have the following truth table

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

Boolean Expressions

- Certain types of Boolean expressions lead to circuits that are more desirable from an implementation viewpoint.
- product term – a single variable or the logical product of several variables. The variables may or may not be complemented.
- sum term – a single variable or the logical sum of several variables. The variables may or may not be complemented.

Boolean Expressions

- X is both a sum term and a product term
  X+Y is a sum term
  XY is a product term
  X+YX is neither
- sum-of-products - a product term or several product terms logically added
- product-of-sums - a sum term or several sum terms logically multiplied

Logical Circuit

- Add a row of product terms to the truth table

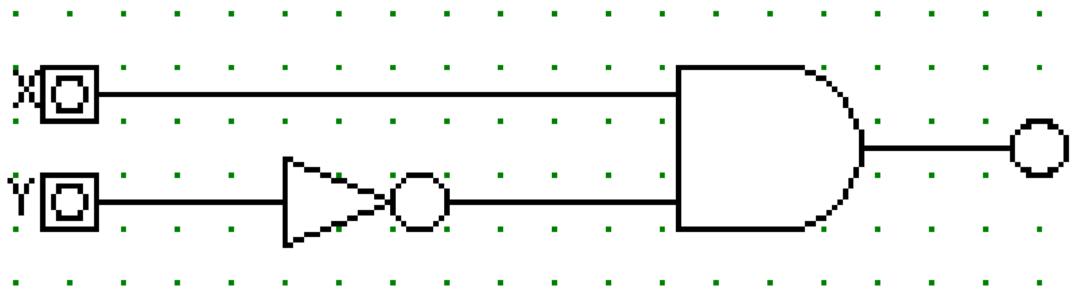
X	Y	Z	Product Terms
0	0	0	X'Y'
0	1	0	X'Y
1	0	1	XY'
1	1	0	XY

Logical Circuit

- When Z equals 1, the product term is used in the sum-of-products
- A product term containing all input variables is called **canonic**.
- A canonical expansion is the logical sum of the product terms where the output (Z in this case) is a 1.
- The canonical expansion for the logical network is therefore Z=XY’

Logical Circuit Realized

- Draw the logical circuit for Z=XY' using OR gates, AND gates, and inverters



- We are interested in finding the least expensive logic network; therefore, we might use some Boolean algebra to simplify the canonical expansion

Problem

- Write the Boolean expression (in sum-of-products form) for a logic network that will have a 1 output when: X=1,Y=0,Z=0; X=1,Y=1,Z=0; and X=1,Y=1,Z=1. All other combinations of input will produce an output of 0. Simplify the expression derived and draw a block diagram for the simplified expression.
- Note: We are interested in finding the least expensive logic network; therefore, we might use some Boolean algebra to simplify the canonical expansion.

product-of-sum expressions

- How to do this?

1. Construct a table of input and output values
2. Input value of 1 complements variable and 0 does not
3. Desired expression is the product of sum terms whose output is 0

Problem

- Consider the following truth table

Inputs		Output
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

- Give a sum-of-products expression and a product-of-sums expression for the output F and draw the logical circuitry that realizes each expression that represents the output F

NAND Gate Design

- NAND gates are widely used in the design of modern computers. In general, NAND gates take less resistors than AND and NOR gates take less resistors than OR.
- To design a NAND-to-NAND two level gate network, we produce the sum-of-products expression.
- A sum-of-products expression can be drawn using AND gates at the first level and an OR gate at the second level. Finally, we can replace all AND and OR gates with NAND gates

Problem

- Develop the sum-of-products that describe the function of the following truth table (X,Y, and Z are inputs, A is output):

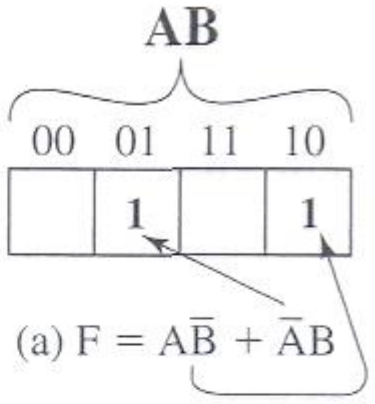
X	Y	Z	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- Draw the logical circuitry using only AND and OR logic. Then design the logical circuitry using only NAND gates

Karnaugh Map (K-map)

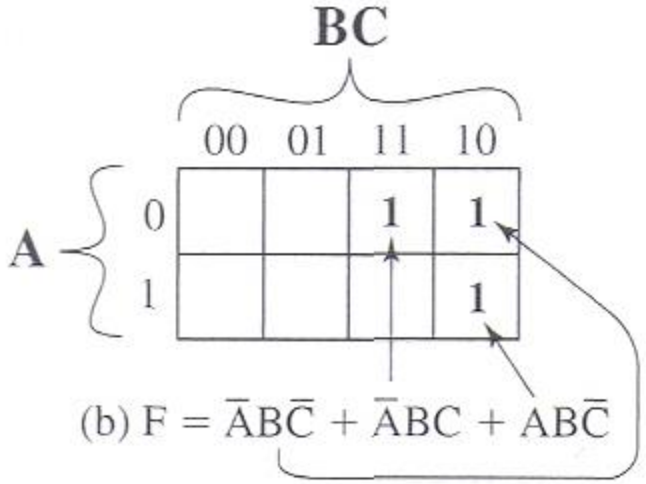
- A Karnaugh Map is a way of representing and simplifying a Boolean function for up to 4 variables.
- The map represents all possible combinations of n binary variables and thus contains 2^n possible squares (where n is the number of variables).

2 Variable K-map



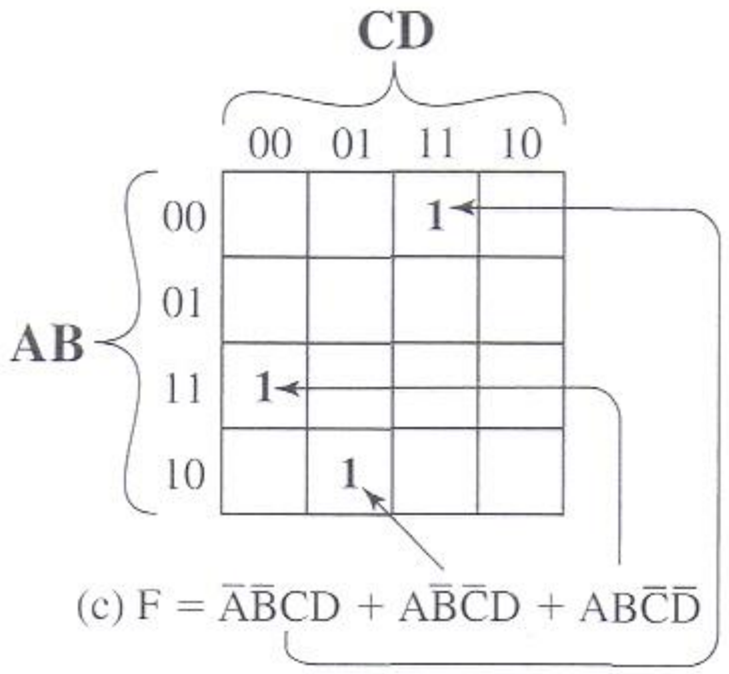
(a) $F = A\overline{B} + \overline{A}B$

3 Variable K-map



(b) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$

4 Variable K-map



(c) $F = \overline{A}\overline{B}CD + A\overline{B}\overline{C}D + AB\overline{C}\overline{D}$

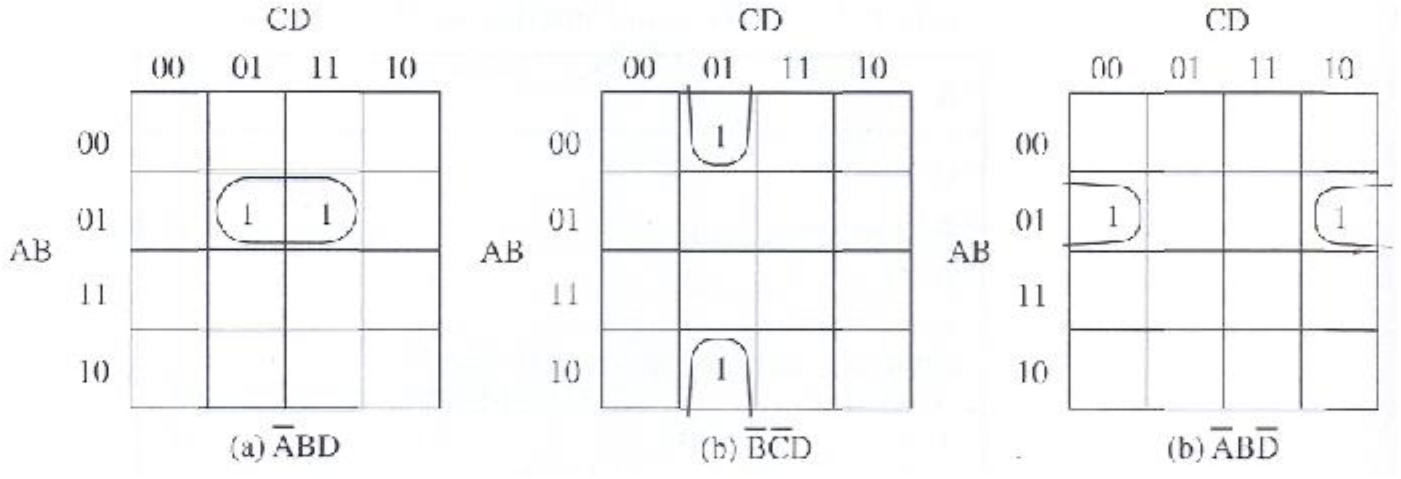
Simplification using K-maps

1. Among the squares with a 1, find those that belong to the largest "block" of 1, 2, 4, or 8. Circle those blocks.

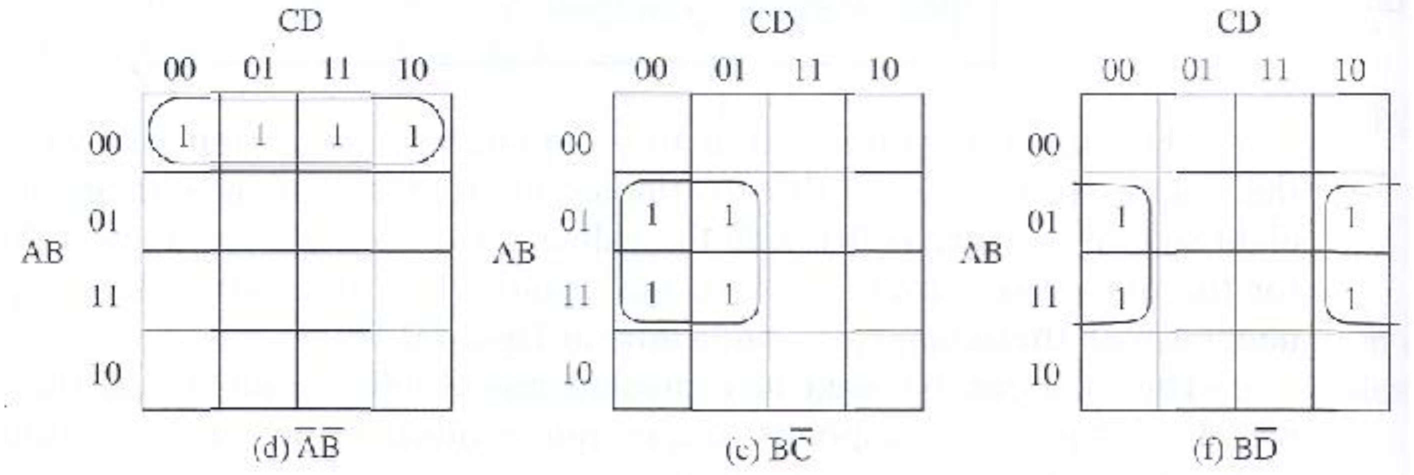
2. Select additional blocks so that every 1 in the square is a member of some block. 1's can be included in multiple blocks.

3. A nice explanation of K-maps can be found at: http://www.facstaff.bucknell.edu/mastascu/elessonshtml/Logic/Logic3.html

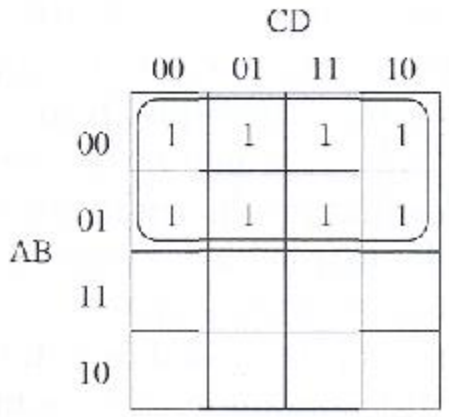
Simplification using K-maps



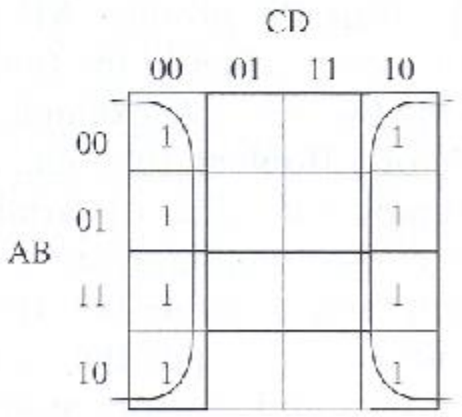
Simplification using K-maps



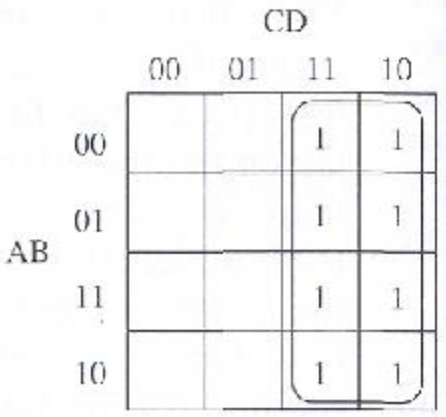
Simplification using K-maps



(g) $\overline{A}$



(h) $\overline{D}$



(i) C

Problem

1. Draw the K-map (A, B, C are inputs; F is output)
2. b) Give the Boolean function for the simplest sum-of-products form
3. c) Draw the circuit using all NANDs

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	0
1	1	0	0
1	1	1	0