Shawnee State University
Industrial & Engineering Technologies

Computer Engineering Technology

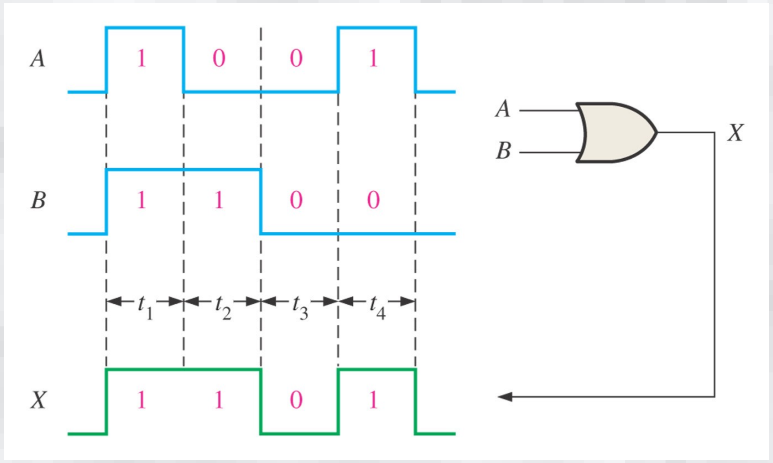
ETEC 2301 Programmable Logic Devices

Chapter 4 Boolean Algebra and Logic Simplification

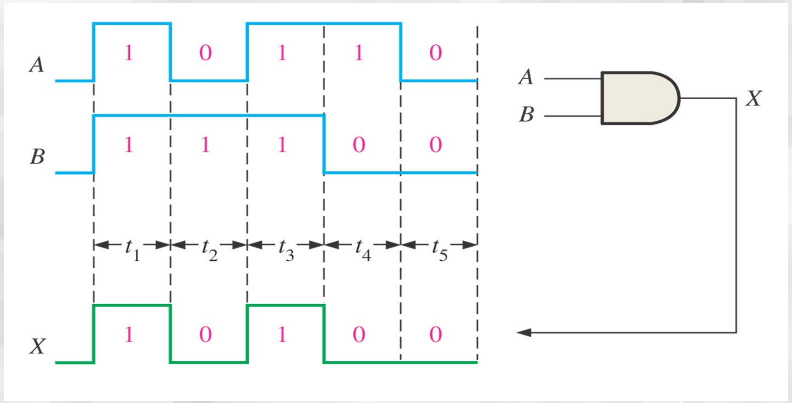
Shawnee State University
Department of Industrial and Engineering Technologies

Boolean Operations and Expressions

- Boolean Addition
  - The OR Operation



- Boolean Multiplication
  - The AND Operation



Laws and Rules of Boolean Algebra

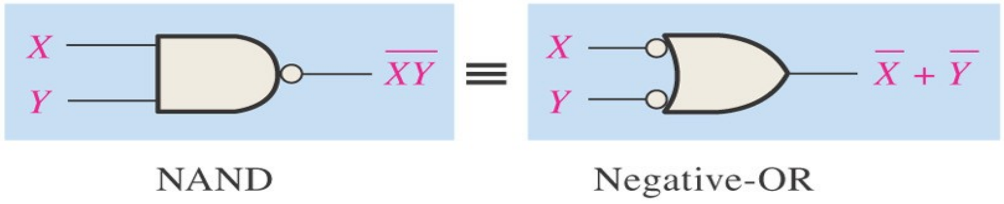
- Laws of Boolean Algebra
  - Commutative Law
    - Commutative Law of Addition: A + B = B + A
    - Commutative Law of Multiplication: AB = BA
  - Associative Law
    - Associative Law of Addition: A + (B + C) = (A + B) + C
    - Associative Law of Multiplication: A(BC) = (AB)C
  - Distributive Law
    - A(B + C) = AB + AC

Laws and Rules of Boolean Algebra (continued)

- Laws of Boolean Algebra (Continued)
  - The 12 Rules of Boolean Algebra
    - $A + 0 = A$
    - $A + 1 = 1$
    - $A \cdot 0 = 0$
    - $A \cdot 1 = A$
    - $A + A = A$
    - $A + \overline{A} = 1$
    - $A \cdot A = A$
    - $A \cdot \overline{A} = 0$
    - $\overline{\overline{A}} = A$
    - $A + AB = A$
    - $A + \overline{A}B = A + B$
    - $(A + B)(A + C) = A + BC$

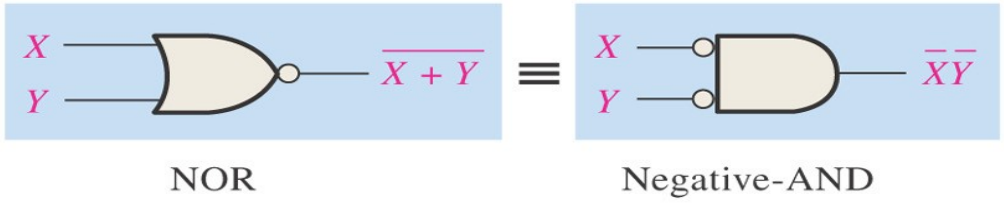
Demorgan's Theorems

$\overline{XY} = \overline{X} + \overline{Y}$



Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

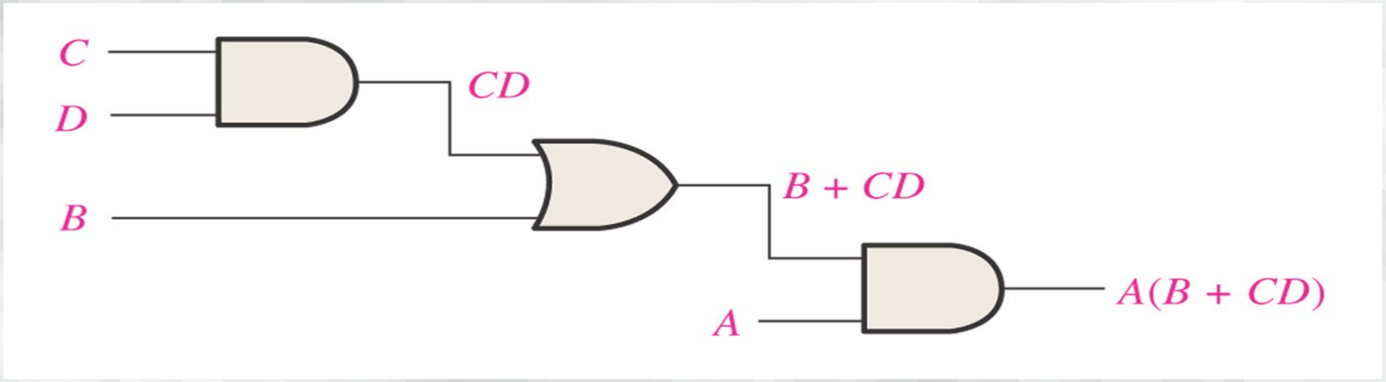
$\overline{X + Y} = \overline{X}\,\overline{Y}$



Inputs		Output	
X	Y	$\overline{X + Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Boolean Analysis of Logic Circuits

- Boolean Expression for a Logic Circuit
  - Boolean expressions are written by starting at the left-most gate, working toward the final output, and writing the expression for each gate:



Boolean Analysis of Logic Circuits (continued)

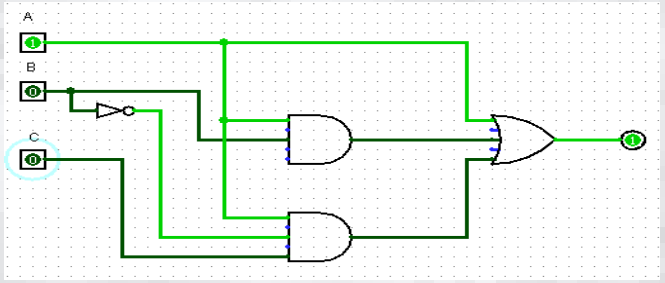
- Constructing a Truth Table for a Logic Circuit:
- A(B + CD)

INPUTS				OUTPUT
A	B	C	D	A(B + CD)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Simplification Using Boolean Algebra

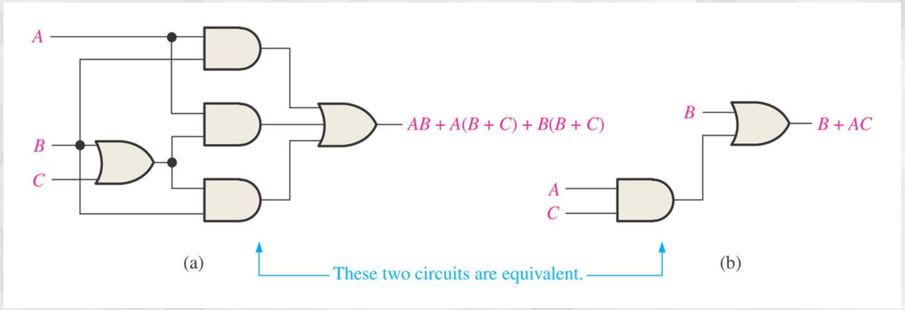
- Simplify $A + AB + A\overline{B}C$
  - Use the 12 rules and DeMorgan's theorems.

$$A + AB + A\overline{B}C$$
$$A + A\overline{B}C \quad \text{Apply Rule 10}$$
$$A \quad \text{Apply Rule 10}$$



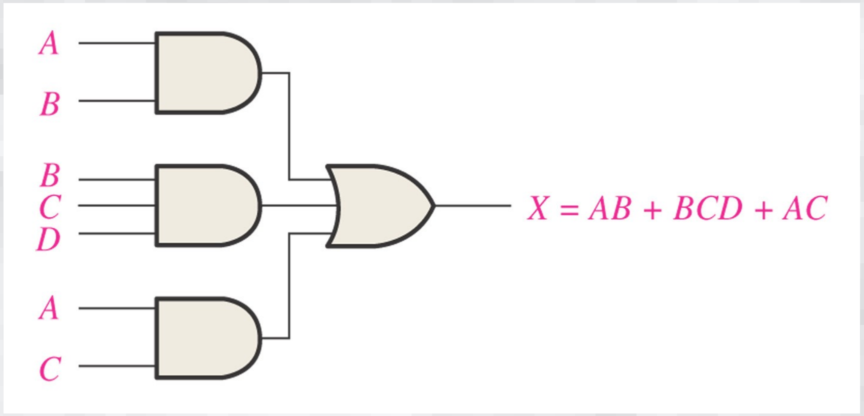
- Simplify AB + A(B + C) + B(B + C)

$$AB + AB + AC + BB + BC$$
$$AB + AC + B + BC$$
$$AB + B + AC$$
$$B + AC$$

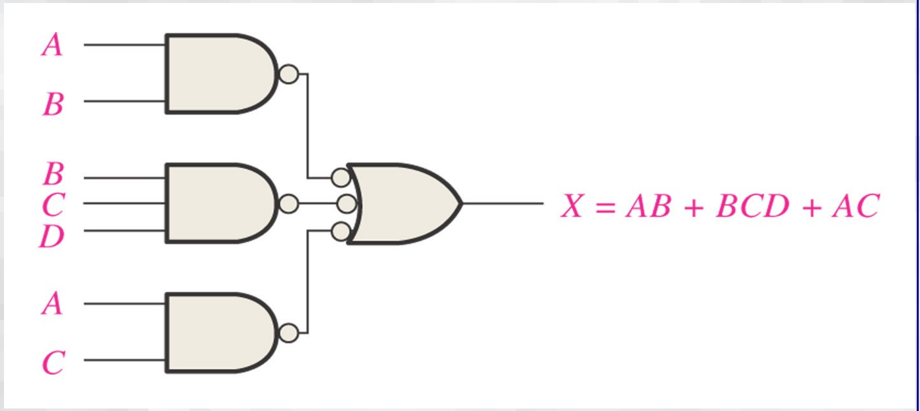


Standard Forms of Boolean Expressions

- Sum-of-Products(SOP) Form
  - e.g. AB + ABC, ABC + CDE + $\overline{B}C\overline{D}$
  - Domain of a Boolean Expression = the set of variables contained in the expression. e.g. For $\overline{A}B + A\overline{B}C$ the domain is A, B, and C
  - AND/OR implementation of an SOP expression
    - ORing the output of two or more AND gates



AND/OR Implementation of SOP



NAND/NAND Implementation of SOP

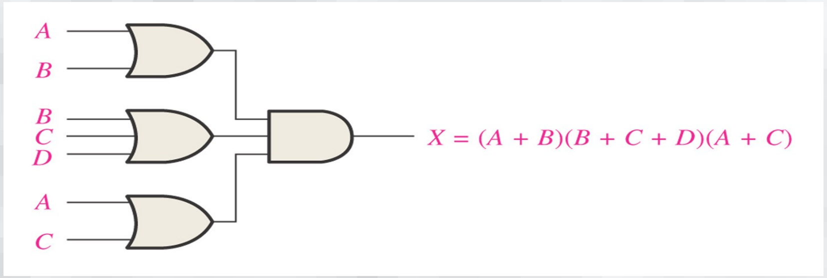
Standard Forms of Boolean Expressions

- Conversion of a General Expression to SOP Form
  - e.g. Convert AB + B(CD + EF) to SOP
    - AB + BCD + BEF
- The Standard SOP Form
  - All of the variables in the domain appear in each product term
    - e.g. $A\bar{B}CD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D}$ is in standard form
  - If a variable is missing from any term, it and its negative must be added to that term.
    - e.g. Convert $A\bar{B}C+\bar{A}\bar{B}+AB\bar{C}D$ to standard SOP form:

$$A\bar{B}C=A\bar{B}C(D+\bar{D})=A\bar{B}CD+A\bar{B}C\bar{D}$$
$$\bar{A}\bar{B}=\bar{A}\bar{B}(C+\bar{C})=\bar{A}\bar{B}C+\bar{A}\bar{B}\bar{C}$$
$$\bar{A}\bar{B}=\bar{A}\bar{B}C+\bar{A}\bar{B}\bar{C}=\bar{A}\bar{B}C(D+\bar{D})+\bar{A}\bar{B}\bar{C}(D+\bar{D})=\bar{A}\bar{B}CD+\bar{A}\bar{B}C\bar{D}+\bar{A}\bar{B}\bar{C}D+\bar{A}\bar{B}\bar{C}\bar{D}$$
$$A\bar{B}C+\bar{A}\bar{B}+AB\bar{C}D=A\bar{B}CD+A\bar{B}C\bar{D}+\bar{A}\bar{B}CD+\bar{A}\bar{B}C\bar{D}+\bar{A}\bar{B}\bar{C}D+\bar{A}\bar{B}\bar{C}\bar{D}+AB\bar{C}D$$

Standard Forms of Boolean Expressions (continued)

- Product-of-Sums (POS) Form
  - e.g. $(\bar{A}+B)(A+B+\bar{C})$
  - Implementation of a POS expression:
    - (A + B)(B + C + D)(A + C)



- The Standard POS Form
  - Must contain all of the terms in the domain of the expression
  - e.g. (A + B + C)(A + B + D)(A + B + C + D) is missing D in the first term and C in the second term. These need to be added in to get the expression into standard form.

Standard Forms of Boolean Expressions (continued)

- Converting a sum term to standard POS:
  - Add each nonstandard product term a term made up of the product of the missing variable and its complement.
  - Apply rule 12 => A + BC = (A + B)(A + C)
  - Apply to all terms missing variables.
  - e.g. Convert: $A(A+\bar{C})(A+B)$
    - First, lets just reduce it:

$$(AA+A\bar{C})(A+B)$$
$$(A+A\bar{C})(A+B)$$
$$A(A+B)$$
$$AA+AB$$
$$A+AB$$
$$A$$

Standard Forms of Boolean Expressions (continued)

- Converting a sum term to standard POS (continued)
  - Now let's put it into standard form:

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

$$\left.\begin{array}{c} A+B\bar{B}=(A+B)(A+\bar{B}) \\ (A+B+C\bar{C})=(A+B+C)(A+B+\bar{C}) \\ A+\bar{B}+C\bar{C}=(A+\bar{B}+C)(A+\bar{B}+\bar{C}) \end{array}\right\} \text{First term: A}$$

$$\left.A+\bar{C}+B\bar{B}=(A+B+\bar{C})(A+\bar{B}+\bar{C})\right\} \text{Second Term: (A + C)}$$

$$\left.A+B+C\bar{C}=(A+B+C)(A+B+\bar{C})\right\} \text{Third Term: (A + B)}$$

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

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

Boolean Expressions and Truth Tables

- Converting SOP Expressions to Truth Table Format
  - For a sum term we need to determine where the term = 1
  - e.g. Develop the truth table for: $\overline{A}B\overline{C}+A\overline{B}C$

Inputs			Output	Product Term
A	B	C	X	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\overline{A}B\overline{C}$
0	1	1	0	
1	0	0	0	
1	0	1	1	$A\overline{B}C$
1	1	0	0	
1	1	1	0	

An SOP = 1 only if one or more of the product terms = 1

$\overline{0}\,1\,\overline{0}=1\,1\,1=1$

$1\,\overline{0}\,1=1\,1\,1=1$

Boolean Expressions and Truth Tables (continued)

- Converting POS Expressions to Truth Table Format
  - For a product term we need to determine where the term = 0
  - e.g. Develop the truth table for: $(A+\overline{B}+C)(A+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$

Inputs			Output	Sum Term
A	B	C	X	
0	0	0	1	
0	0	1	0	$(A+B+\overline{C})$
0	1	0	0	$(A+\overline{B}+C)$
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	0	$(\overline{A}+\overline{B}+\overline{C})$

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

Boolean Expressions and Truth Tables (continued)

- Determining Standard Expressions from a Truth Table
  - Given a truth table, the SOP expression is developed from where the output is 1 and the POS expression is developed from where the output is 0
  - e.g. Develop the SOP and POS expressions from the following:

Inputs			Output
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

For the SOP expression: Look or an output of 1. Then write the terms:

$$011 \rightarrow \overline{A}BC$$
$$100 \rightarrow A\overline{B}\,\overline{C}$$
$$110 \rightarrow AB\overline{C}$$
$$111 \rightarrow ABC$$
$$X = \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + + ABC$$

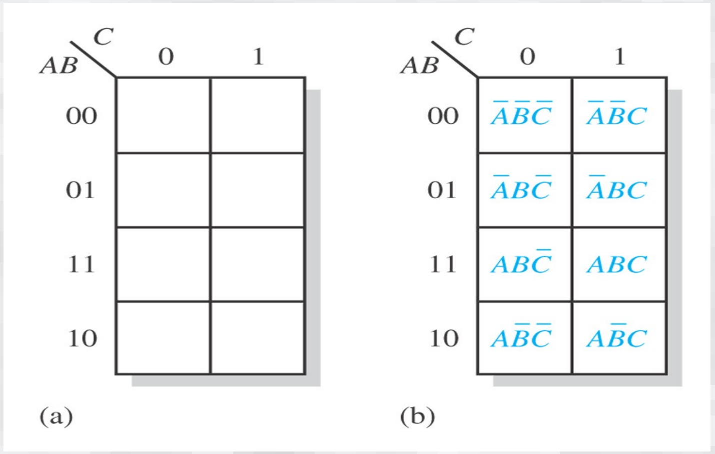
For the POS expression: Look for an output of 0. Then write the terms:

$$000 \rightarrow A+B+C$$
$$001 \rightarrow A+B+\overline{C}$$
$$010 \rightarrow A+\overline{B}+C$$
$$101 \rightarrow \overline{A}+B+\overline{C}$$
$$X = (A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+B+\overline{C})$$

The Karnaugh Map

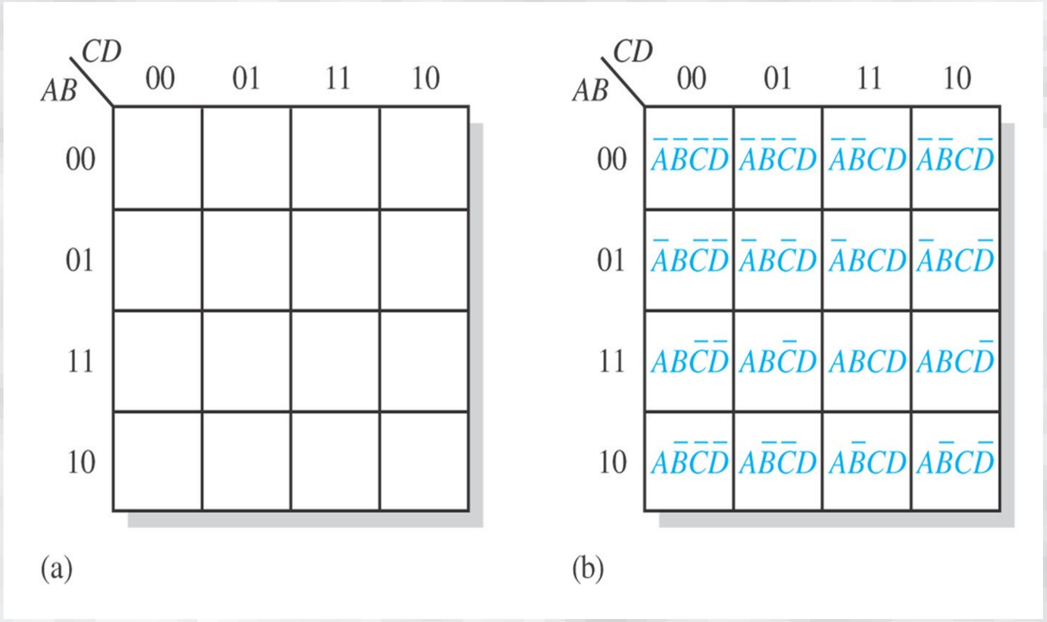
- Provides a method for simplifying Boolean expressions
- It will produce the simplest SOP and POS expressions
- Works best for less than 6 variables
- Similar to a truth table => it maps all possibilities
- A Karnaugh map is an array of cells arranged in a special manner
- The number of cells is 2^n where n = number of variables
- A 3-Variable Karnaugh Map:

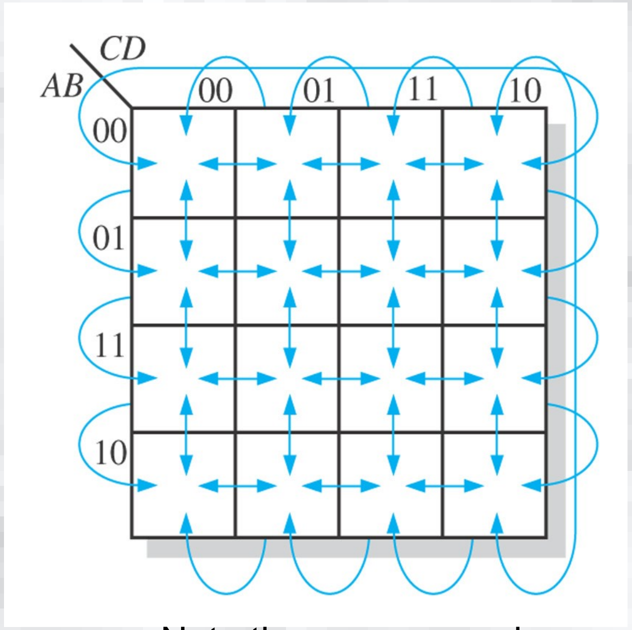
Note the order of these values they are reverse of the usual order. They are arranged this way so that only one variable changes at a time.


C
AB
0
1
00
01
11
10
(a)
C
AB
0
1
00
01
11
10
$\overline{A}\,\overline{B}\,\overline{C}$
$\overline{A}\,\overline{B}C$
$\overline{A}B\overline{C}$
$\overline{A}BC$
$AB\overline{C}$
ABC
$A\overline{B}\,\overline{C}$
$A\overline{B}C$
(b)

The Karnaugh Map (continued)

- Cell Adjacency



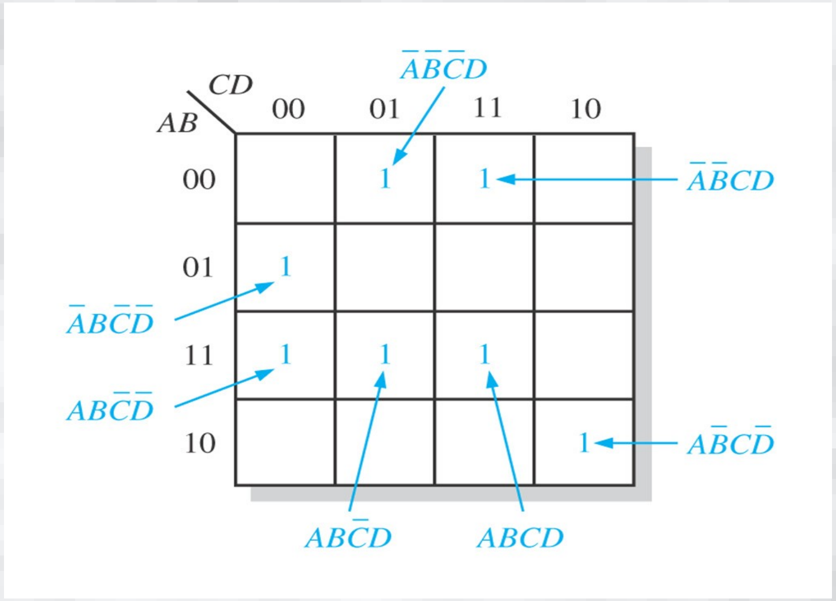


Note the wrap-around from side to side and top to bottom.

Karnaugh Map SOP Minimization

- Mapping a Standard SOP expression
  - Place a 1 in the Karnaugh map where each product term = 1
  - e.g. Map the following:

$$\overline{A}\,\overline{B}\,C\,D+\overline{A}\,B\,\overline{C}\,\overline{D}+A\,B\,\overline{C}\,D+ABCD+AB\overline{C}\,\overline{D}+\overline{A}\,\overline{B}\,\overline{C}\,D+A\,\overline{B}\,C\,\overline{D}$$


$\overline{A}\overline{B}\overline{C}D$
CD
AB
00
01
11
10
00
01
11
10
1
1
$\overline{A}\overline{B}CD$
1
$\overline{A}B\overline{C}\overline{D}$
1
1
1
$AB\overline{C}\overline{D}$
1
$A\overline{B}C\overline{D}$
$AB\overline{C}D$
$ABCD$

Note that this expression is in standard SOP form.

Karnaugh Map SOP Minimization

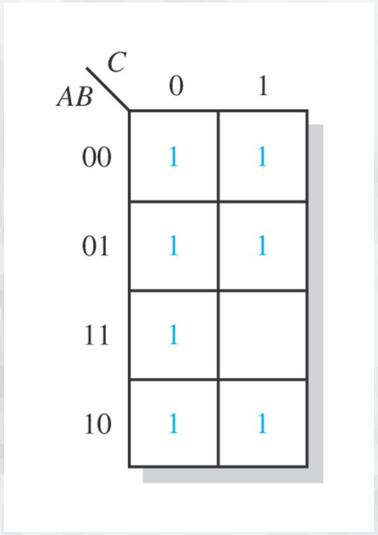
- Mapping a Nonstandard SOP expression
  - Each nonstandard part of the expression must be expanded
  - e.g. Map the SOP expression: $\overline{A}+A\overline{B}+AB\overline{C}$

$\overline{A}$
$\overline{A}BC=011$
$\overline{A}B\overline{C}=010$
$\overline{A}\overline{B}C=001$
$\overline{A}\overline{B}\overline{C}=000$

$A\overline{B}$
$A\overline{B}C=101$
$A\overline{B}\overline{C}=100$

$AB\overline{C}=110$

Place 1s in the Karnaugh map that correspond with each of the above terms.


C
AB
0
1
00
01
11
10
1

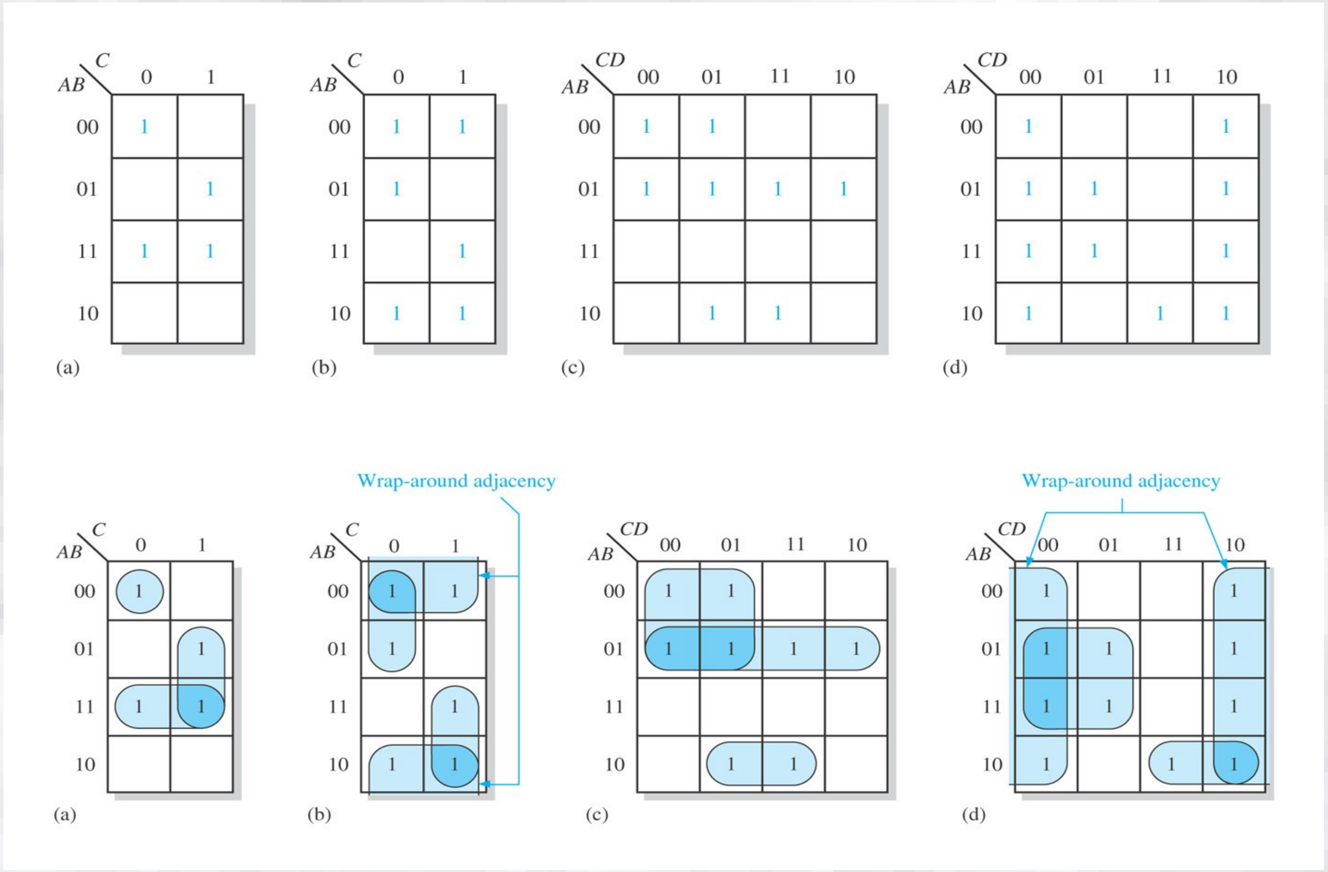
AB \ C	0	1
00	1	1
01	1	1
11	1	
10	1	1

Karnaugh Map SOP Minimization (continued)

- Karnaugh Map Simplification of SOP Expressions
  - Finding the minimum SOP expression after an SOP expression has been mapped
  - Process is to group the 1s in adjacent cells
    - A group must contain either 1, 2, 4, 8, or 16 cells (a power of 2)
    - Each cell in a group must be adjacent to 1 or more cells.
    - Always include the largest possible number of 1s in a group but it must be 1, 2, 4, 8, or 16 cells (a power of 2)
    - Each 1 on the map must be included in at least one group. Groups may overlap
  - Be sure to remember the adjacency from side to side and top to bottom when grouping the 1s.
  - In some cases there may be more than one way to group the 1s.

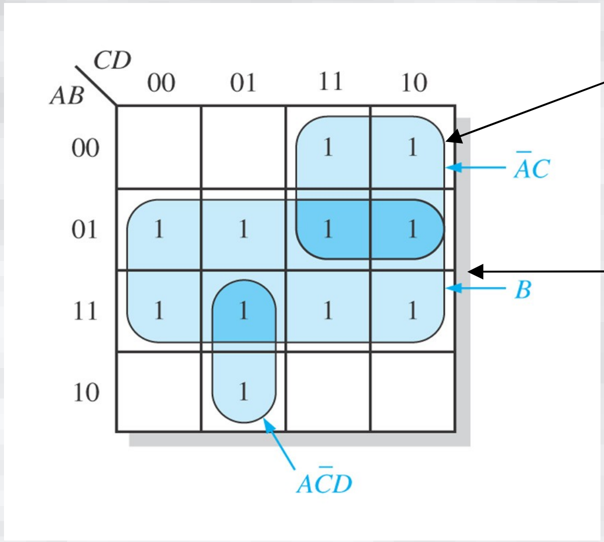
Karnaugh Map SOP Minimization (continued)

- Simplification example:



Karnaugh Map SOP Minimization (continued)

- Determining the Minimum SOP Expression from the Map.
  - After grouping the 1s, look for the variables that don't change in each group. These will remain in our product term. The variables that do change drop out.
  - e.g. Develop the SOP for the following Karnaugh Map:



Note that B takes on both 0 and 1 values.

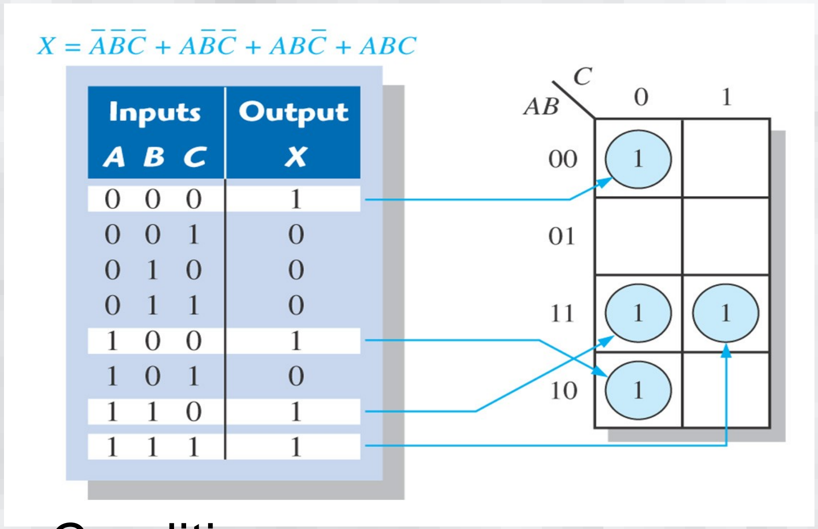
Here A and C assume both 0 and 1 values so they drop out.

The resulting SOP is the sum of each group:

$$B+\overline{A}C+A\overline{C}D$$

Karnaugh Map SOP Minimization (continued)

- Mapping Directly from a Truth Table:
  - e.g.



$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$

Inputs			Output
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\begin{array}{c|cc} AB \backslash C & 0 & 1 \\ \hline 00 & 1 & \\ 01 & & \\ 11 & 1 & 1 \\ 10 & 1 & \end{array}$$

- Don't Care Conditions
  - Sometimes certain variable combinations are not allowed. In that case, we represent them with an “X” in the truth table and Karnaugh map.

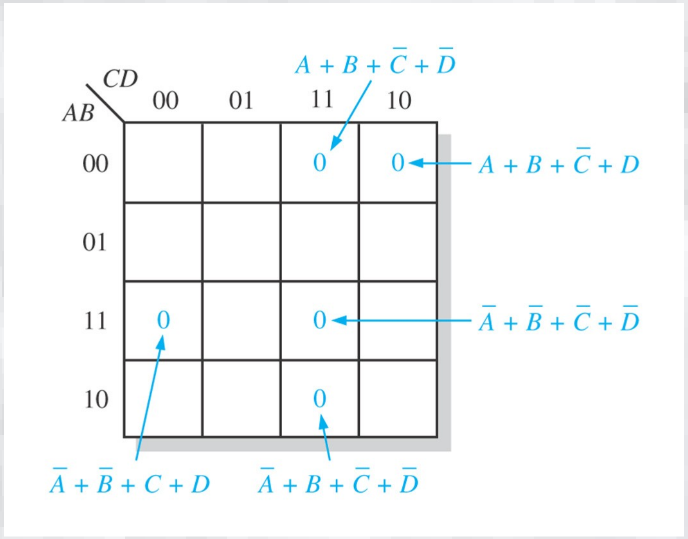
Karnaugh Map POS Minimization

- Mapping a Standard POS Expression
  - A 0 is placed on the Karnaugh map for each sum term in the expression.
  - The values are mapped:
    - Determine the binary value of each sum term in the standard expression
    - Place a 0 on the Karnaugh map in the corresponding cell
  - e.g. Map the expression:

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(\overline{A}+\overline{B}=\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$

Binary value of each Sum term:

$$(\overline{A}+\overline{B}+C+D)=1100$$
$$(\overline{A}+B+\overline{C}+\overline{D})=1011$$
$$(A+B+\overline{C}+D)=0010$$
$$(\overline{A}+\overline{B}+\overline{C}+\overline{D})=1111$$
$$(A+B+\overline{C}+\overline{D})=0011$$


CD
AB
00
01
11
10
0
$A+B+\overline{C}+\overline{D}$
$A+B+\overline{C}+D$
$\overline{A}+\overline{B}+\overline{C}+\overline{D}$
$\overline{A}+\overline{B}+C+D$
$\overline{A}+B+\overline{C}+\overline{D}$

Karnaugh Map POS Minimization (continued)

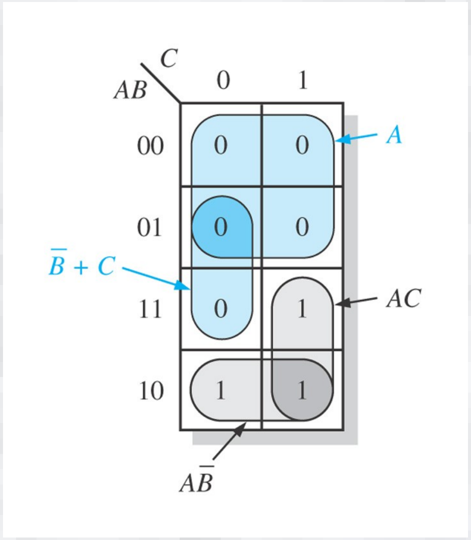
- Karnaugh Map Simplification of POS Expressions
  - Basically the same as for the POS expression except that we will group 0s to produce the minimum sum terms instead of grouping 1s.
  - e.g. Use a Karnaugh Map to minimize:

$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$

Binary value of each Product term:

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

$$X=A(\overline{B}+C)$$



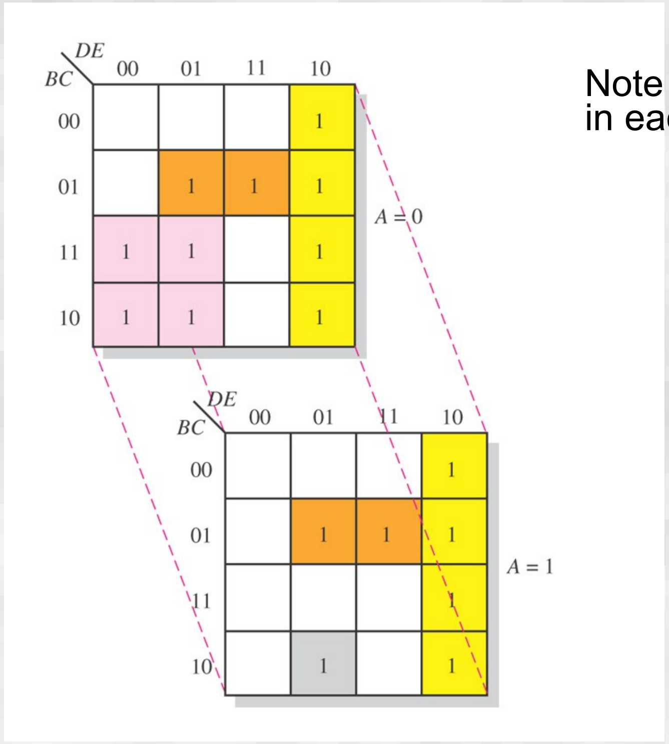
We can also derive the SOP expression by looking at the 1s: $X=AC+A\overline{B}=A(\overline{B}+C)$

Karnaugh Map POS Minimization (continued)

- Converting Between POS and SOP using the Karnaugh Map
  - Once an expression or truth table has been mapped each cell contains a 1 or a 0 (or x)
  - By grouping the 0s we can derive the minimum POS expression
  - By grouping the 1s we can derive the minimum SOP expression

Five Variable Karnaugh Maps

- We can simplify 5 variable expressions by using two 4 variable 32-cell Karnaugh Maps.
- Each of the maps is a value of the 5th variable and adjacencies exist between cells in the same location on each map



Note that we still look for variables that remain the same in each group. Those will appear in our final expression.

The term for the yellow group is $D\overline{E}$

The term for the orange group is $\overline{B}C E$

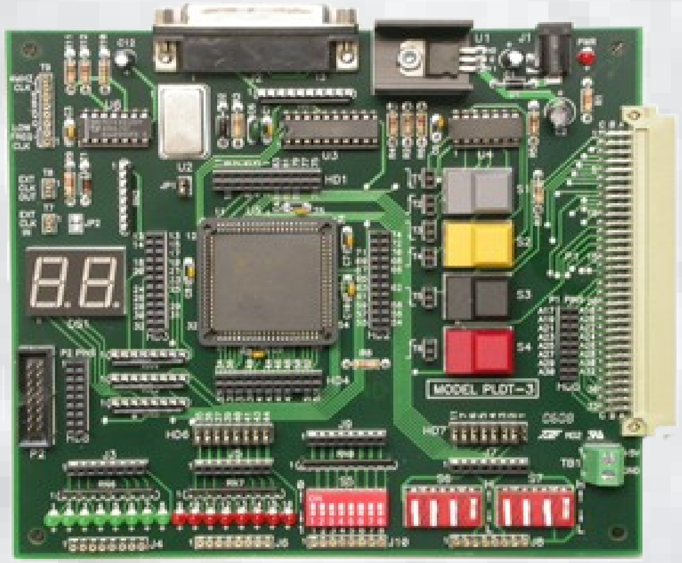
The term for the light red group is $\overline{A}B\overline{D}$

The term for the gray cell grouped with the red cell is $B\overline{C}\overline{D}E$

$$x = D\overline{E} + \overline{B}C E + \overline{A}B\overline{D} + B\overline{C}\overline{D}E$$

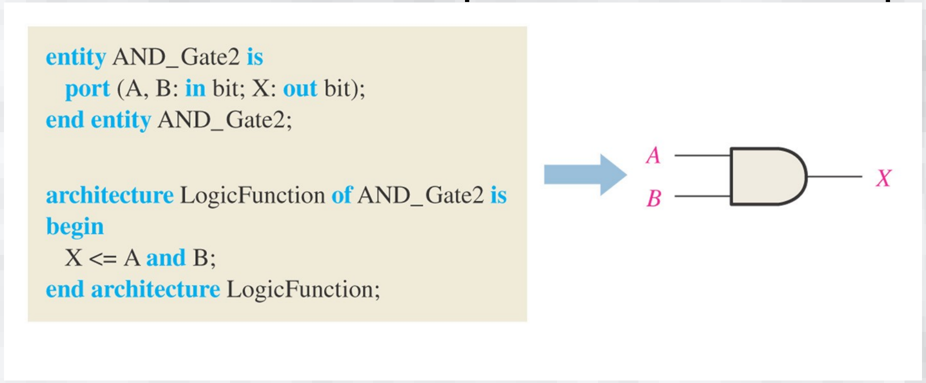
Hardware Description Languages (HDLs)

- We use HDL to program some of the functions of the Xilinx CPLD chips on the PLDT-3 board.
- HDLs are a quick and versitile method of describing logic designs for downloading into the programmable logic devices.
- Like any other programming language, there is a syntax and reserved words that are used in describing the logic functions



Hardware Description Languages (HDLs)

- VHDL
  - Abbreviation for Very High Speed Integrated Circuit Hardware Description Language
  - Using VHDS, we can describe the circuit in one of three ways: behavior, data flow, or structure.
  - We use the keywords: and, or, not, nand, nor, xor, and xnor to describe the parts of the logic circuits.
  - Also we need to describe the inputs, outputs and the internal operation of the logic function (architecture)
  - An entity is described by at least 3 statements: Name, Port list (inputs and outputs), and the end statement
  - The architectural component is other component.

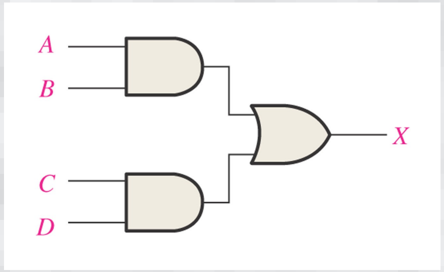


```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```

Hardware Description Languages (HDLs) (continued)

- Writing Boolean Expressions in VHDL
  - Using the keywords, we can describe the logic function
  - e.g. Write a VHDL program to describe:



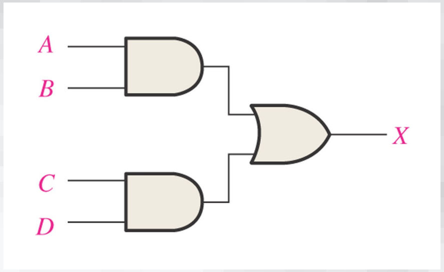
$X = AB + CD$

```
entity AND_OR is
     port( A, B, C, D: in bit; X: out bit );
end entity AND_OR;

architecture LogicFunction of AND_OR is
begin
     X <=(A and B)or(C and D);
end architecture LogicFunction;
```

Hardware Description Languages (HDLs) (continued)

- Verilog
  - Similar to VHDL it is an IEE standard in design
  - It uses different syntax, but basically does the same thing – defines the inputs, outputs, and the logic associated with the module
  - Uses symbols for the logical operators: ! (not), && (and), || (or)
  - Uses symbols for the bitwise operators: ~ (not), & (and), ~& (nand), | (or), ~| (nor), ^ (xor), ~^(xnor)
  - e.g. Same as for VHDL to see how they compare:



$$X = AB + CD$$

```
module AndOR(X, A, B, C, D);
input A, B, C, D;
output X;
        assign X = (A & B) || (C & D);
endmodule
```

Hardware Description Languages (HDLs) (continued)

- ABEL
  - Nonstandard proprietary HDL that stands for Advanced Boolean Expression
  - Can use either equations, truth tables, or state diagrams to describe the logic function
  - It also uses symbols for the logic operations: ! (not), & (and), # (or), $ (xor)
- AHDL
  - Also a nonstandard proprietary HDL that stands for Altera Hardware Description Language.
  - Can use either text or schematic entry for logic functions.
  - It alos uses symbols for the logic operations: ! (not), & (and), !& (nand), # (or), !# (nor), $ (xor), !$ (xnor)