PDS0101

Introduction to Digital Systems

Combinational Logic Analysis I

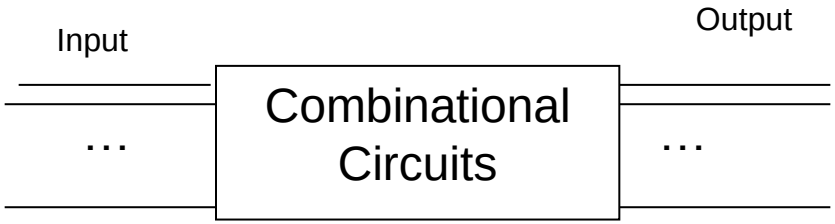
MMU
MULTIMEDIA UNIVERSITY

Lecture outcome

- By the end of today's lecture you should know
  - how to convert between boolean expressions and logic circuit diagrams
  - simplify expressions using boolean algebra to create more efficient circuits
  - SOP and POS forms and their standard forms
  - how to convert between SOP and POS forms
  - use truth tables to create logic circuits/boolean expressions

Implementing Combinational Logic

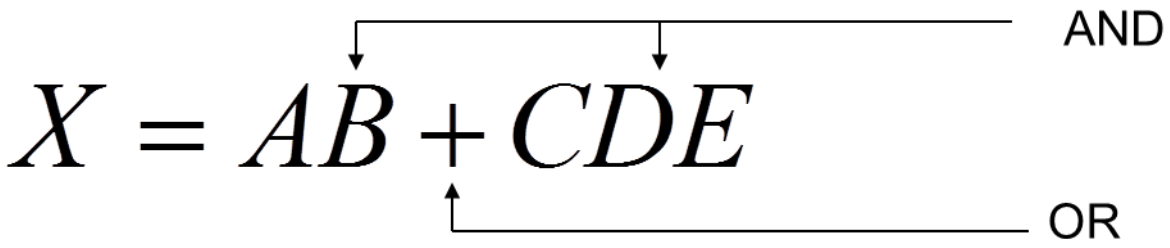
- Combinational logic is defined as that class of digital circuits where,at any given time, the state of all outputs depends only upon the values of the inputs at that time and not upon any previous states.
- A combinational logic circuit may be regarded as a black box having N input lines and P output lines, each of which carries a digital function which can have only two possible values, commonly denoted as 0 or 1

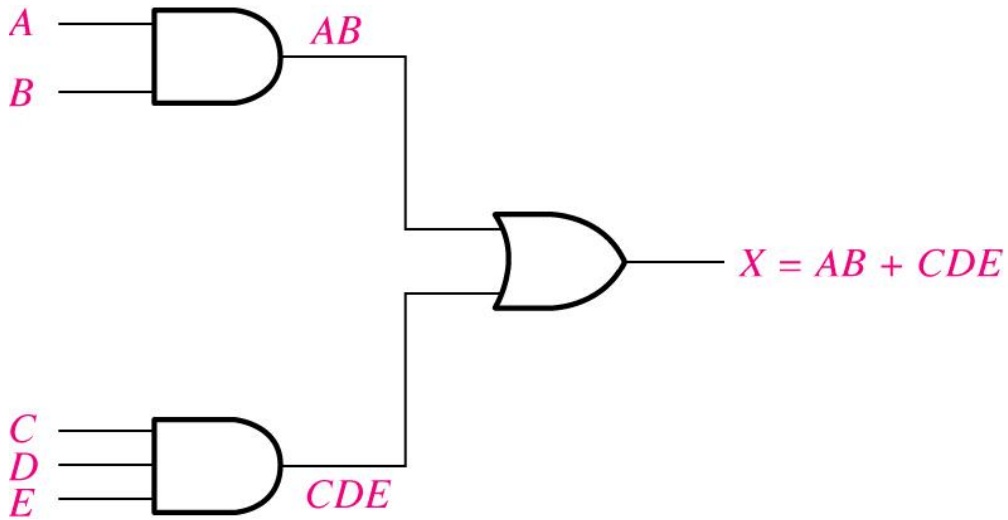


- There are various ways to create combinational circuits → using boolean expression/equations, truth tables, etc.

Boolean Expression to Logic Circuit

- From the following Boolean Expression below the AND operations between AB and CDE are performed before the resulting terms can be ORed





SOP and POS forms

- Boolean expressions can be written in the sum-of-products form (SOP) or in the product-of-sums form (POS).
- These forms can simplify the implementation of combinational logic, particularly with PLDs.
- In both forms, an overbar cannot extend over more than one variable.

- An expression is in SOP form when two or more product terms are summed as in the following examples:

$\overline{A}\,\overline{B}\,\overline{C} + A\,B$ $\qquad$ $A\,B\,\overline{C} + C\,\overline{D}$ $\qquad$ $C\,D + \overline{E}$

- An expression is in POS form when two or more sum terms are multiplied as in the following examples

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

Sum of product (SOP) forms

- When two or more product terms are summed by Boolean addition, the resulting expression is in sum-of-products (SOP) form. E.g.

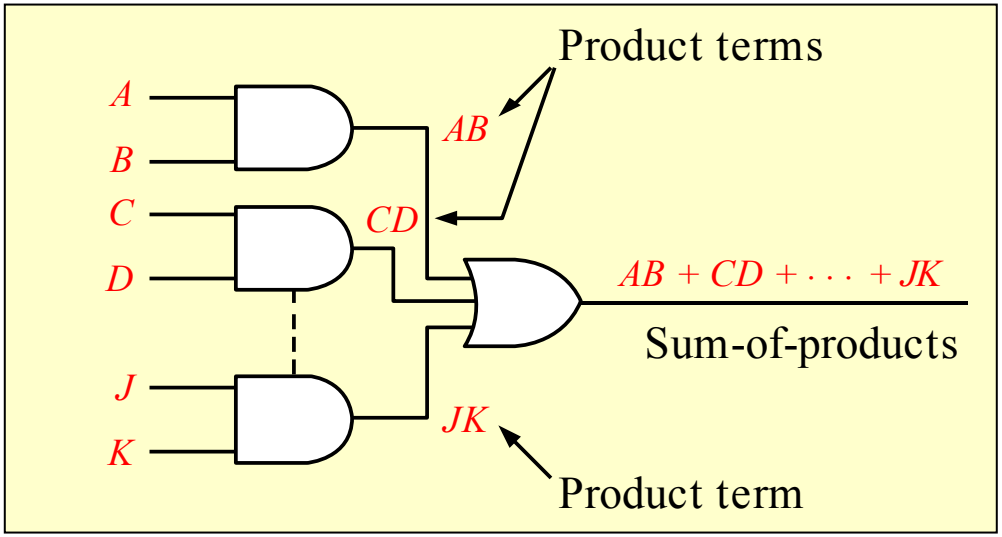
$$ABC + \overline{A}B\overline{C}$$

$$A + AB + C\overline{D} + EF + GK + H\overline{I}$$

- Each of these sum of products expressions consists of 2 or more AND terms (products) that are ORed together
- Each AND term consists of one or more variables appearing in either complemented or uncomplemented form - also known as a *literal*
- With complements, the overbar cannot extend over more than one variable, although more than one variable in a term can have an overbar
  - Can have the term $\overline{A}B\overline{C}$ but not $\overline{ABC}$

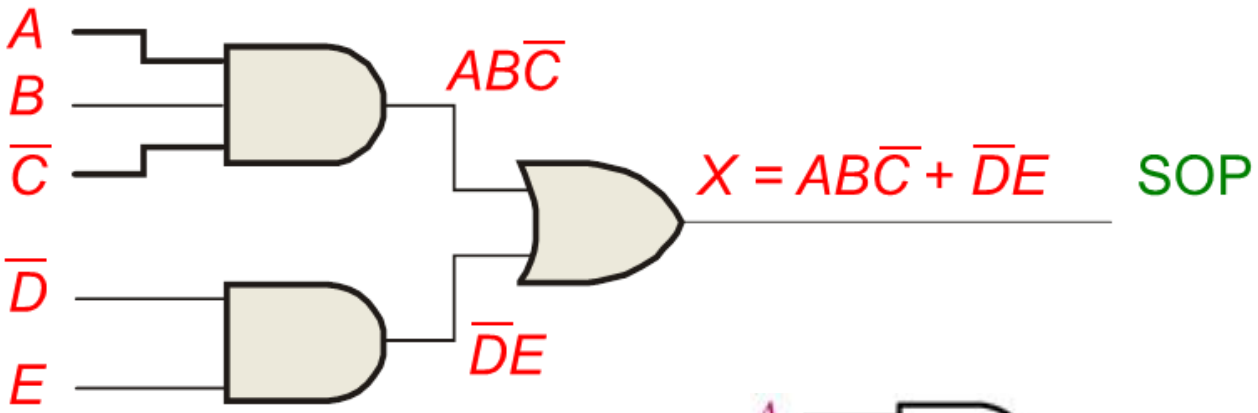
SOP Combinational Logic Circuits

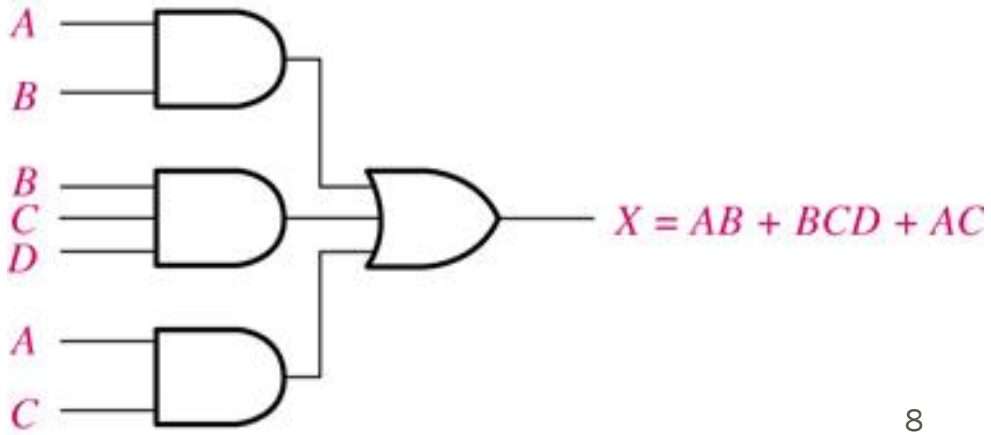
- In Sum-of-Products (SOP) form, basic combinational circuits can be directly implemented with AND-OR combinations if the necessary complement terms are available
  - The product terms of a number of AND gates (equal to the number of product terms in the expression) feed into the inputs of an OR gate



SOP Combinational Logic Circuits

- Examples of an SOP implementation to their logic circuits are shown below.
- The SOP expression is an AND-OR combination of the input variables and the appropriate complements





SOP standard form

- In SOP ***standard form***, every variable in the domain must appear in each term.
- This form is useful for constructing truth tables or for implementing logic in PLDs
- You can expand a nonstandard term to standard form by multiplying the term by a term consisting of the sum of the missing variable and its complement
- Example

Convert $X = A\overline{B}\,\overline{} + A\,B\,C$ to standard form.

The first term does not include the variable C. Therefore, multiply it by the $(C + \overline{C})$, which = 1:

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

Product-of-sum (POS) forms

- When two or more terms are multiplied, the resulting expression is in a product-of-sum (POS) form
  - It consists of 2 or more OR terms(sums) that are ANDed together.
  - Each OR term contains one or more variables in complemented or non complemented form

- Examples

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

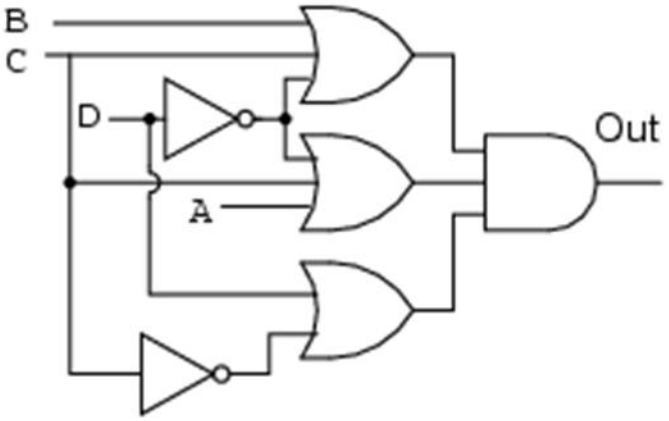
$$(A+\overline{B})(A+\overline{B}+C)(\overline{A}+C)$$

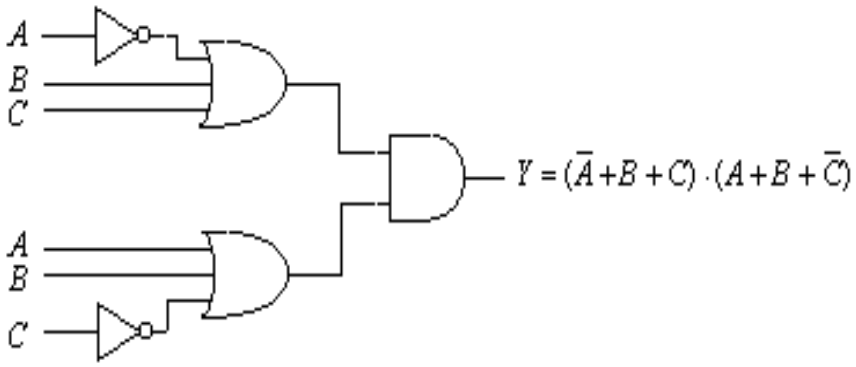
$$(\overline{A}+\overline{B}+\overline{C})(C+\overline{D}+E)(\overline{B}+C+D)$$

POS Combinational Logic Circuits

- In Product-Of-Sum (POS) form, basic combinational circuits can be directly implemented with AND-OR combinations if the necessary complement terms are available
  - The product terms of a number of OR gates (equal to the number of product terms in the expression) feed into the inputs of an AND gate
  - Examples are shown below of POS logic circuits

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





Implementation of the POS expression

POS standard form

- In POS ***standard form***, every variable in the domain must appear in each sum term of the expression.
- You can expand a nonstandard POS expression to standard form by adding the product of the missing variable and its complement and applying rule which states that (A + B)(A + C) = A + BC.
- Example

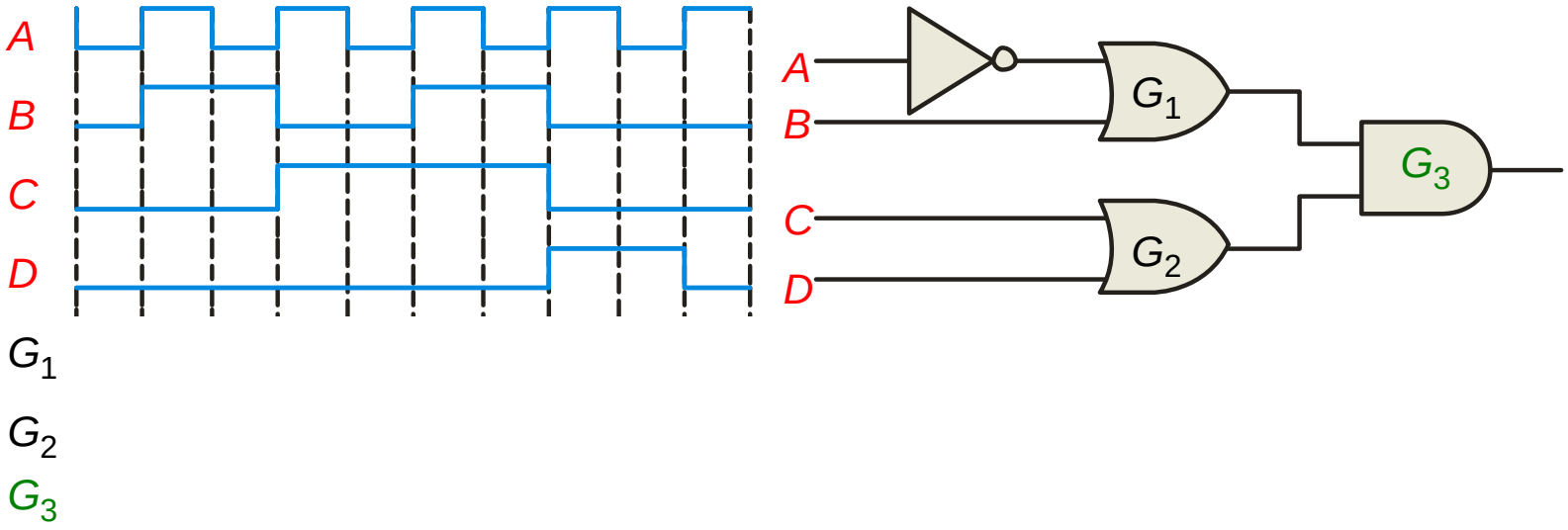
Convert $X = (\overline{A} + B)(\overline{A} + B + C)$ to standard form.

The first sum term does not include the variable *C*.
Therefore, add $C.\overline{C}$ and expand the result by rule 12.

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

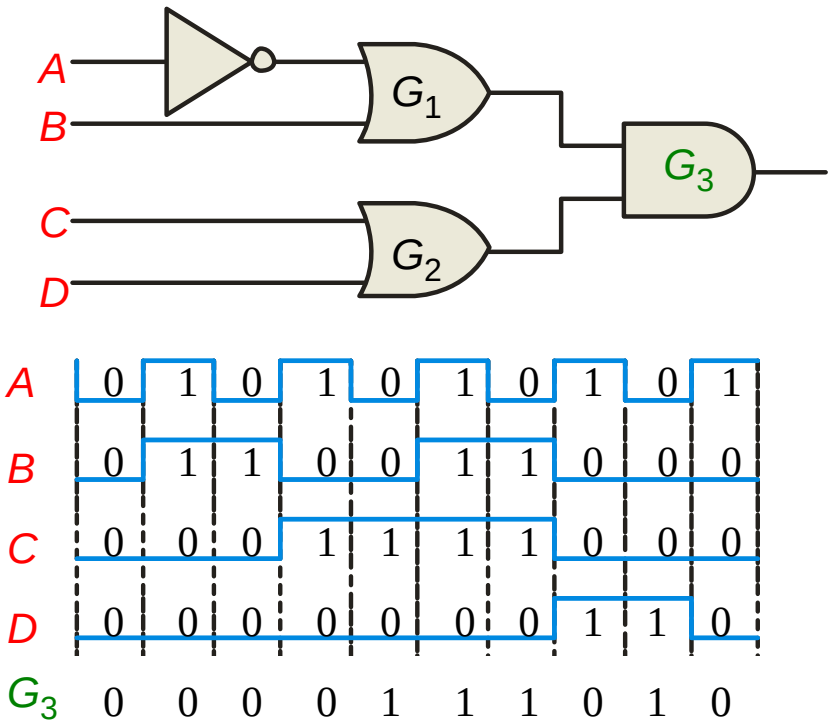
Validating logic circuits

- For combinational circuits with pulsed inputs, the output can be predicted/validated by developing intermediate outputs and combining the result.
- For example, the circuit shown can be analyzed at the outputs of the OR gates

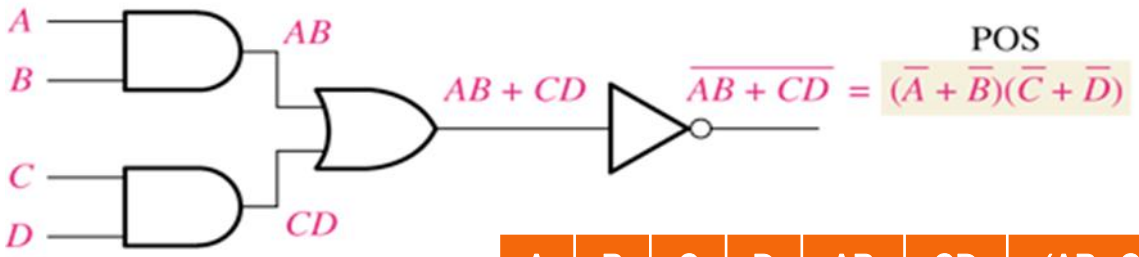


Validating logic circuits

- Alternatively, instead of timing diagrams you can develop the truth table for the circuit and enter 0's and 1's on the waveforms.
- Then read the output from the table



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



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

Truth Tables to Logic Circuits

- A truth table is a convenient way to symbolically represent (and validate) a logic function.
- All possible combinations of input variable values are presented in tabular column (usually in a ascending order) and for each unique combinations of inputs,the output variables are listed with a separate column assigned to each variable.
- The truth table thus constitutes a complete specification of the combinational logic to be designed.

Truth table example

- Design a three input logic circuit whose output will be high when a majority of the inputs are high

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

Truth table to logic circuit

- Using the earlier example (slide 15), write the AND term for each case where the output is 1

$$\bar{A}BC,\ A\bar{B}C,\ AB\bar{C},\ and\ ABC$$

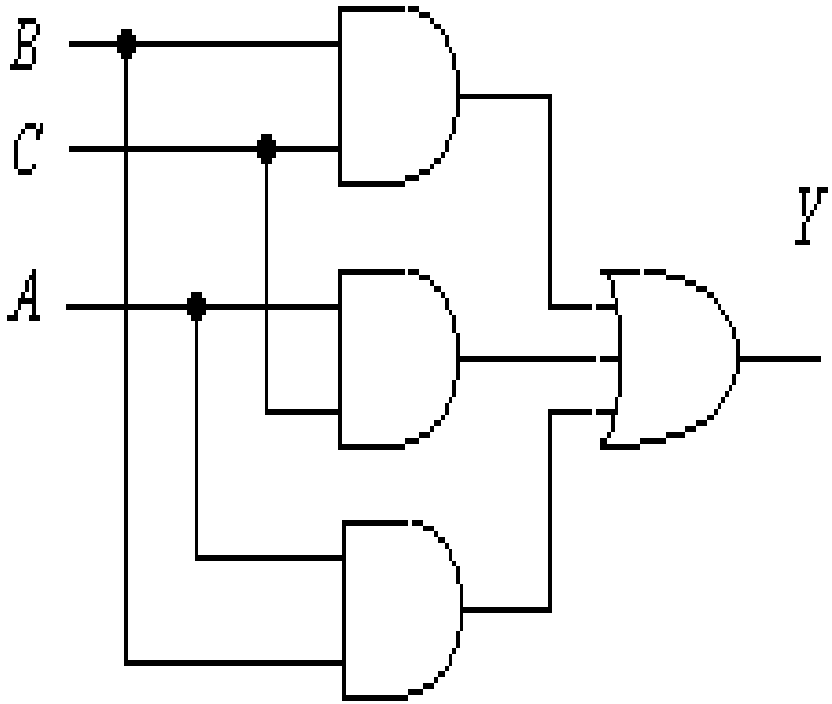
- Using boolean algebra rules and theorem, simplify the output expression

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

$$Y = BC(\bar{A} + A) + AC(\bar{B} + B) + AB(\bar{C} + C)$$

$$Y = BC + AC + AB$$

- Finally, convert the expression into the logic circuit



$$Y = BC + AC + AB$$

SOP Expressions from Truth Table

- Set up the truth table
- List the binary values of the input variables for which output is 1.
  - Convert the binary values to the corresponding product term by replacing each 1 with the corresponding product term.
  - Write the AND term for each case where the output is a 1.
- Write the sum of products expression for the output → this is the expression in SOP standard form
- Simplify the output expression
- Implement the circuit for the final expression

Example 1

- Develop a truth table for the standard SOP expression

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

- There are three variables in the domain, so there are eight possible combinations of binary value of the variables as listed in the left three columns of the table →
- The binary values that make the product terms in the expression equal to 1 are $\overline{A}\overline{B}C$:001, $A\overline{B}\overline{C}$:100 and ABC:111.
- For each these binary values, a 1 is placed in the output column as shown in the table.
- For each of the remaining binary combinations, a 0 is placed in the output column.

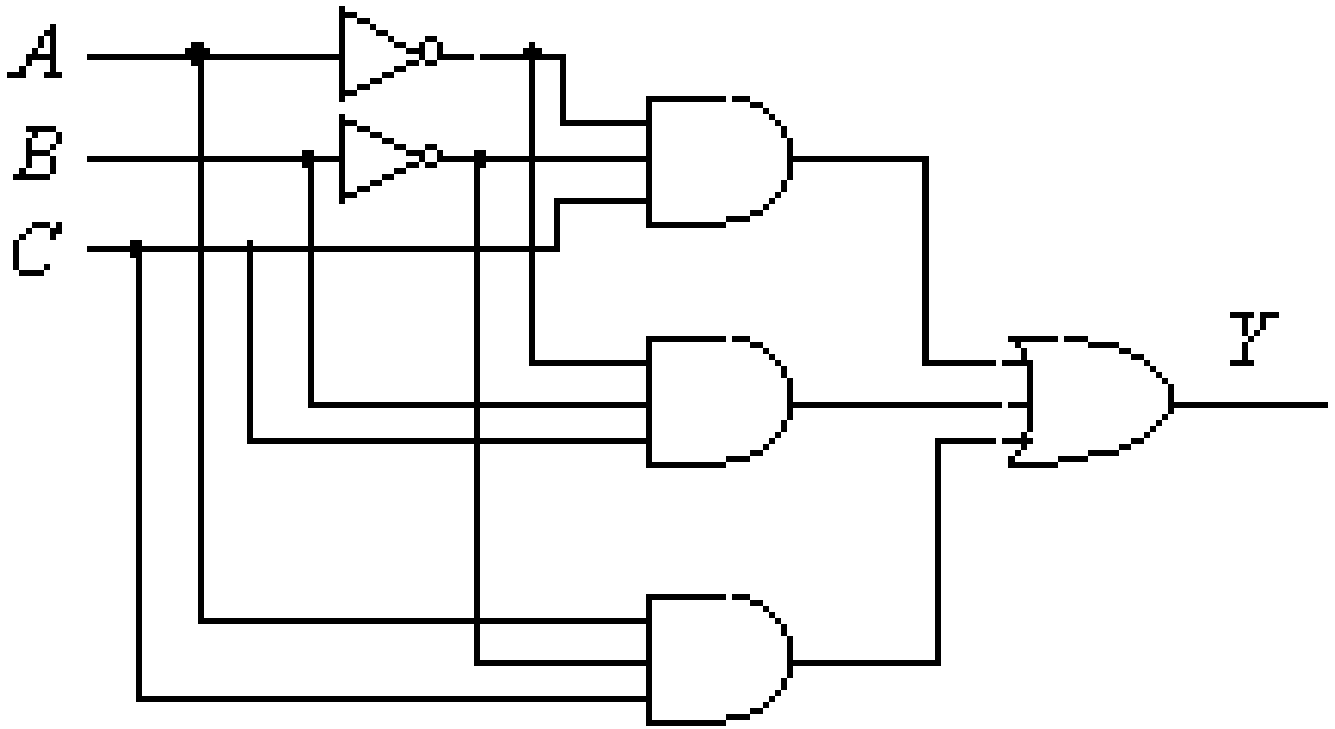
INPUT			OUTPUT
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

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

Example 2

A	B	C	Y	
0	0	0	0	
0	0	1	1	$\rightarrow \bar{A}\bar{B}C$
0	1	0	0	
0	1	1	1	$\rightarrow \bar{A}BC$
1	0	0	0	
1	0	1	1	$\rightarrow A\bar{B}C$
1	1	0	0	
1	1	1	0	

Truth Table



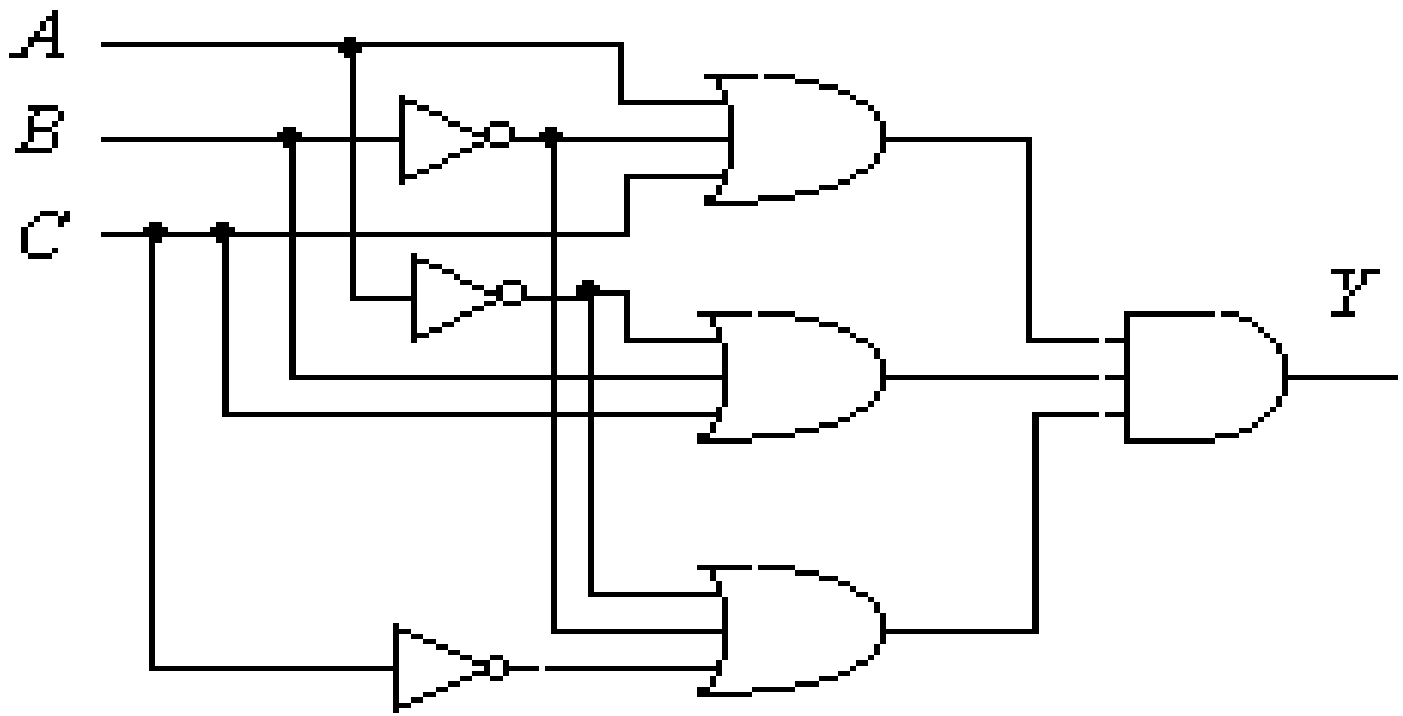
Equivalent AND-OR circuit

POS Expressions from Truth Table

- List the binary values of the input variables for which output is 0.
- Convert the binary values to the corresponding sum term by replacing each 1 with the corresponding variable complement.
- Replace each 0 from step 1 with the corresponding variable
- Complete the POS expression by ANDing all variables

A	B	C	Y	
0	0	0	1	
0	0	1	1	
0	1	0	0	$A+\overline{B}+C$
0	1	1	1	
1	0	0	0	$\overline{A}+B+C$
1	0	1	1	
1	1	0	1	
1	1	1	0	$\overline{A}+\overline{B}+\overline{C}$

Truth Table



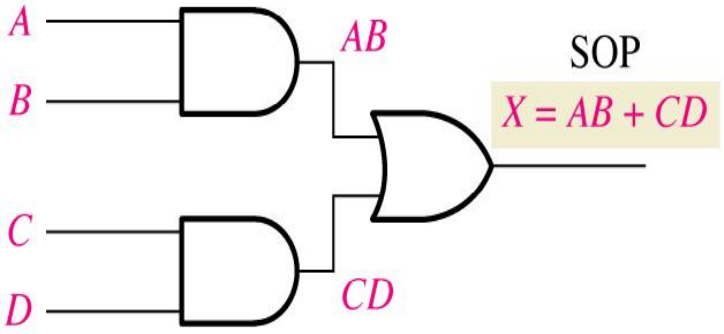
Equivalent OR-AND circuit

Revisiting logic gates

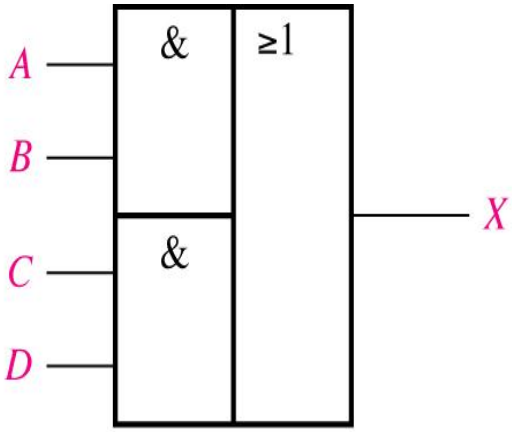
- Earlier topics have covered basic logic gates
- Using only AND, OR and NOT gates it is possible to construct the whole range of logic gates with the methods shown in previous slides

AND-OR Logic gate

- Basic AND-OR gate ANDs multiple input groups (product) and ORs the results of the products → this creates a SOP
- Figure a shows an AND-OR circuit consisting of two 2-input AND gates and one 2-input OR gate. Figure b shows the ANSI standard rectangular outline symbol.



(a) Logic diagram (ANSI standard distinctive shape symbols)

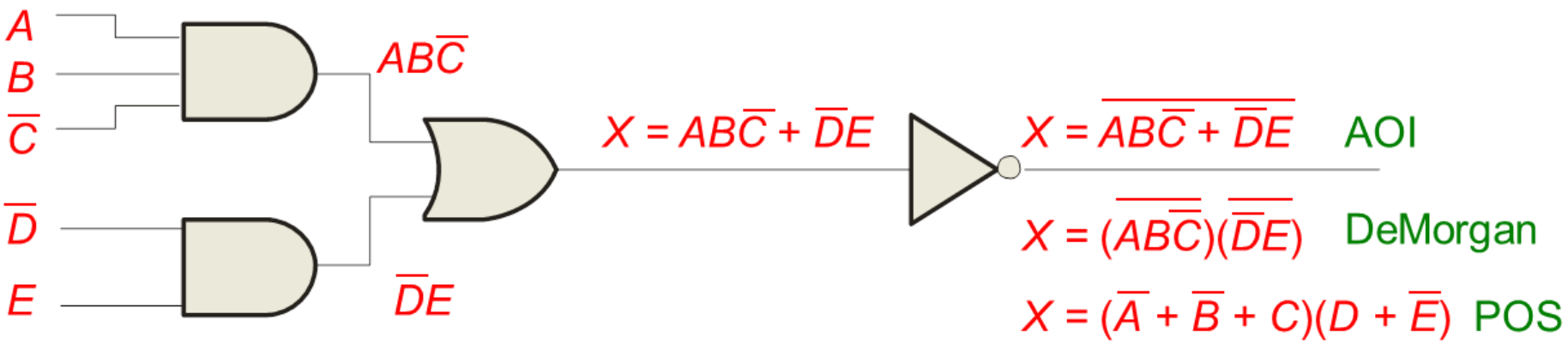


(b) ANSI standard rectangular outline symbol

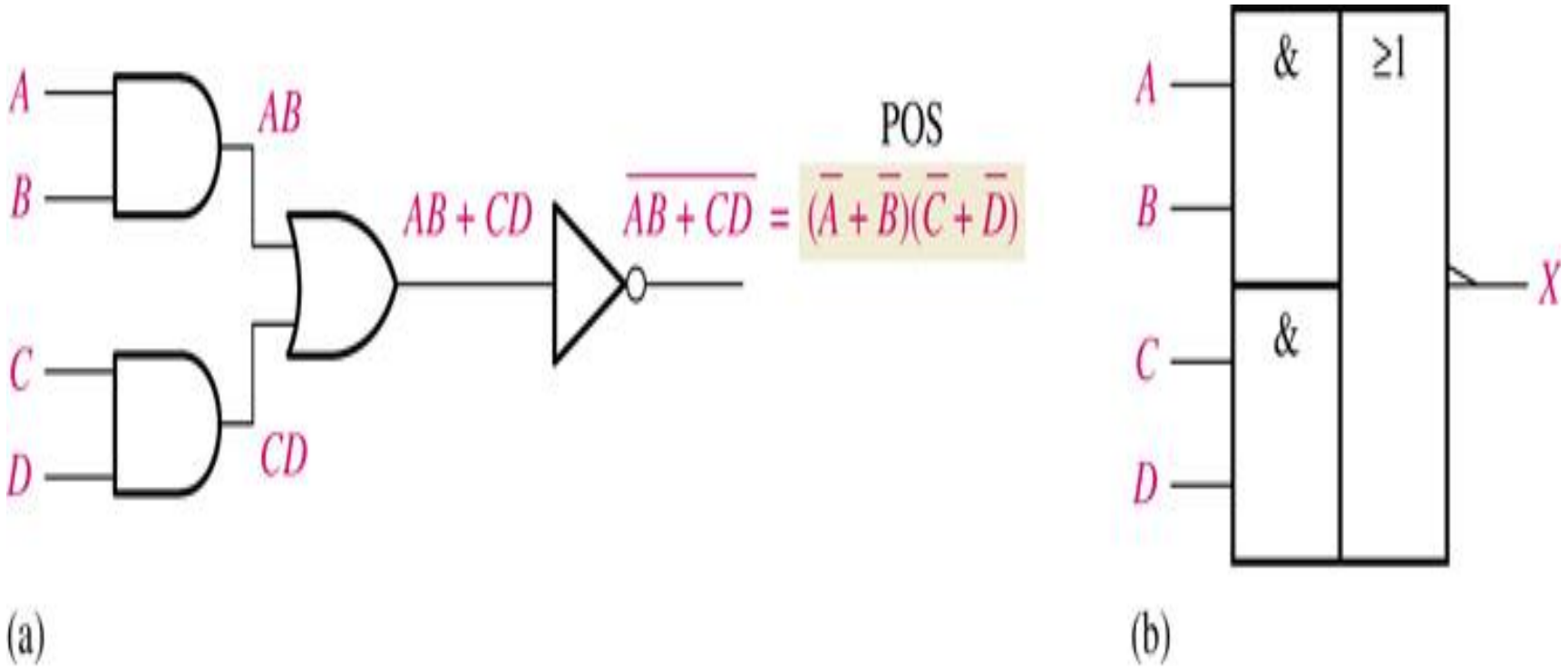
- In general, AND-OR circuit can have any number of AND gates each with any number of inputs

AO Invert gates (SOP to POS)

- When the output of a SOP form is inverted, the circuit is called an AND-OR-Invert circuit.
- The AOI configuration lends itself to product-of-sums (POS) implementation
- An example of an AOI implementation is shown.
  - The output expression can be changed to a POS expression by applying DeMorgan's theorem twice



- For a 4-input AND-OR-Invert logic circuit (figure below), the output X is LOW (0) if both input A and input B are HIGH (1) or both input C and input D are HIGH (1)

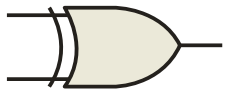


- This can be proven with a truth table or timing diagrams

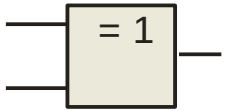
XOR Logic gate

- The truth table for an XOR gate is shown below with ANSI symbols

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

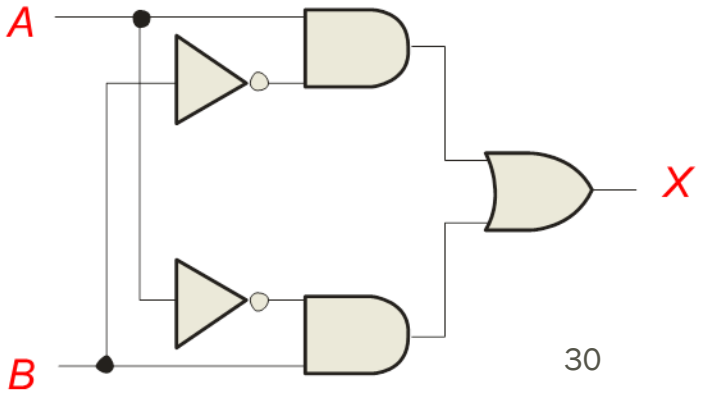


Distinctive shape outline



Rectangular

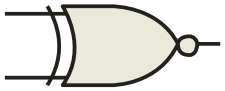
- Notice that the output is HIGH whenever A and B disagree.
- The Boolean expression to represent this, X = A'B + AB'
- This makes the XOR gate actually a combination of other gates – two AND gates, one OR gate and two inverters

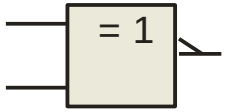


XNOR Logic gate

- The truth table for an exclusive-NOR gate and ANSI symbols

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

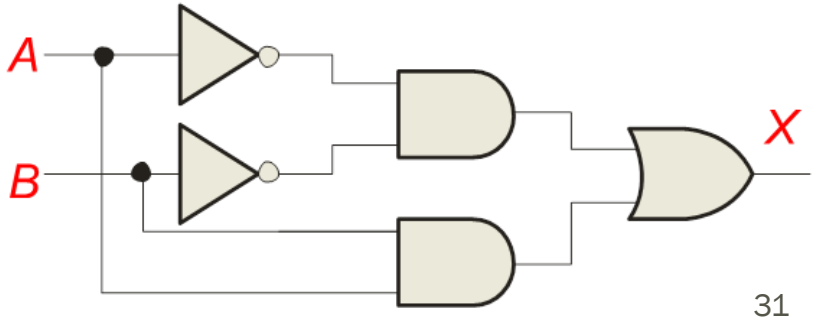




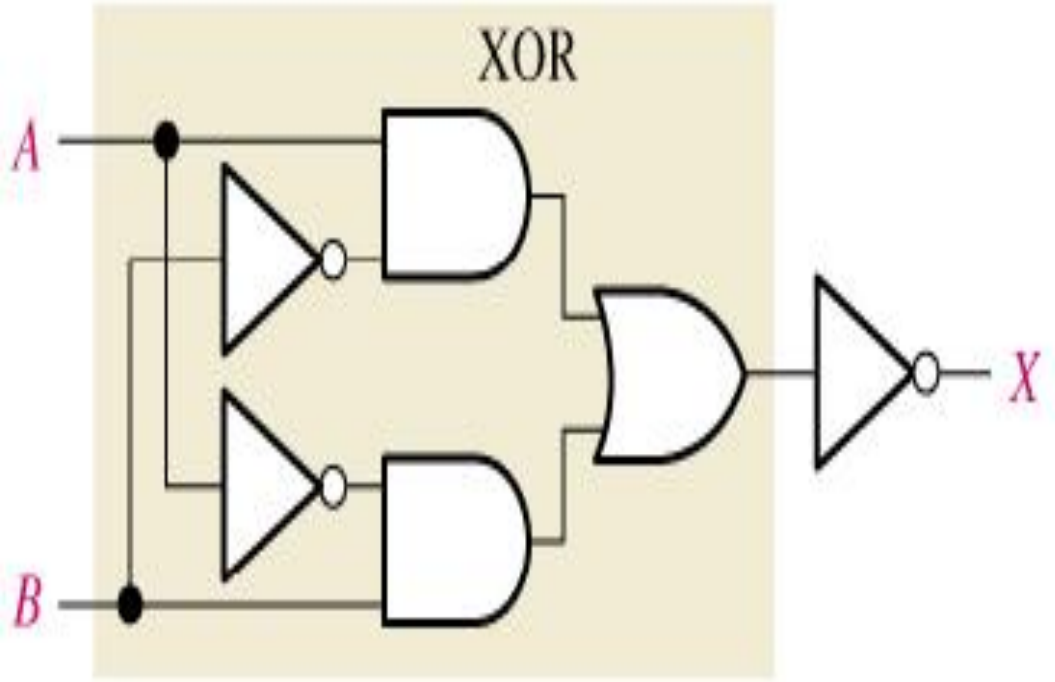
- The complement of the exclusive-OR function is the exclusive-NOR, which can be derived as follows

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

- Notice that the output is HIGH whenever A and B agree
- Thus, the circuit can be drawn as



- Alternatively, using AOI method the following circuit is also valid

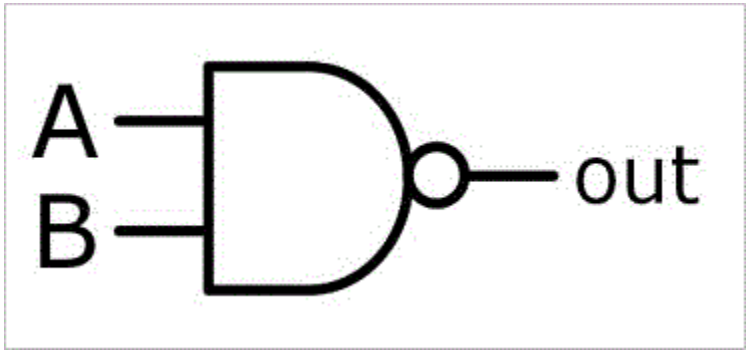


(a) $X = \overline{A\bar{B} + \bar{A}B}$

NAND Logic gate

- Recap: The NAND gate outputs high when the inputs are low or not identical

Input A	Input B	Output (AB)
Low (0)	Low (0)	High (1)
Low (0)	High (1)	High (1)
High (1)	Low (0)	High (1)
High (1)	High (1)	Low (0)



- NAND gates can function as either a NAND or a negative-OR because by DeMorgan's theorem which states

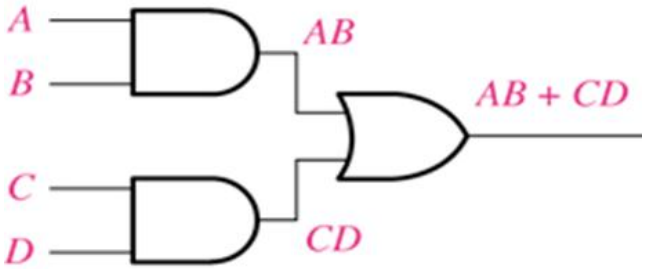
$$\overline{AB} = \overline{A} + \overline{B}$$

NAND → $\overline{AB}$; $\overline{A} + \overline{B}$ ← Negative-OR

- **NOTE: negative-OR is NOT the same as NOR**

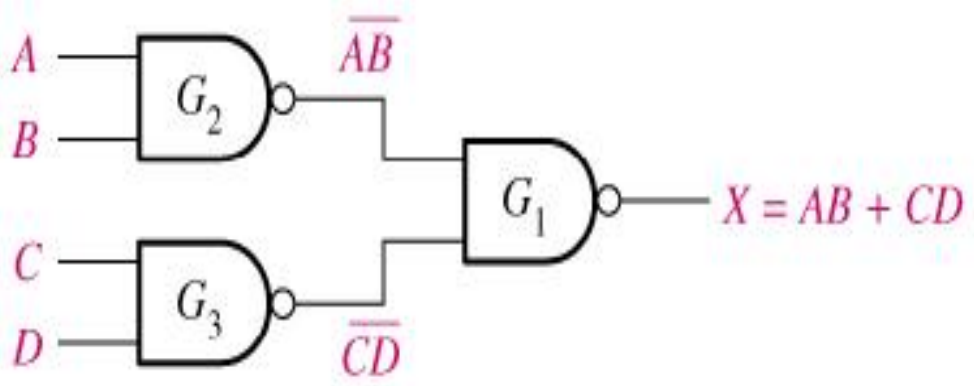
NAND Logic

- Consider the logic circuit in the figure below which is from slide 27



- Instead of different gates, the circuit can be implemented using just NAND gates. The output expression is developed in the following steps:

$$
\begin{aligned}
X &= AB + CD \\
&= \overline{\overline{AB}} + \overline{\overline{CD}} \\
&= \overline{(\overline{A} + \overline{B})} + \overline{(\overline{C} + \overline{D})} \\
&= \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} \\
&= \overline{(\overline{AB})(\overline{CD})}
\end{aligned}
$$

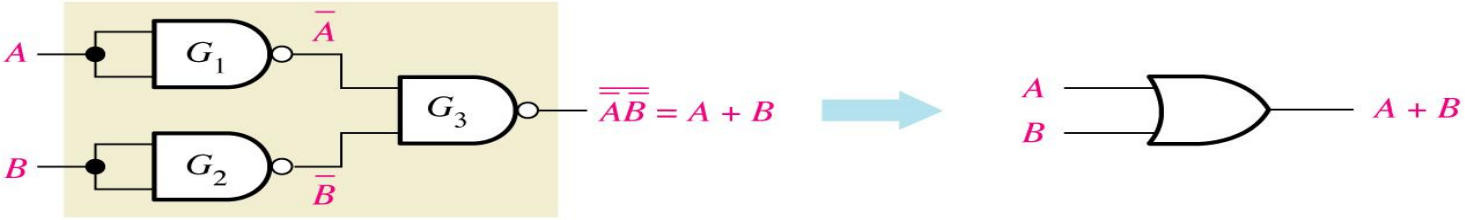


Universal gate

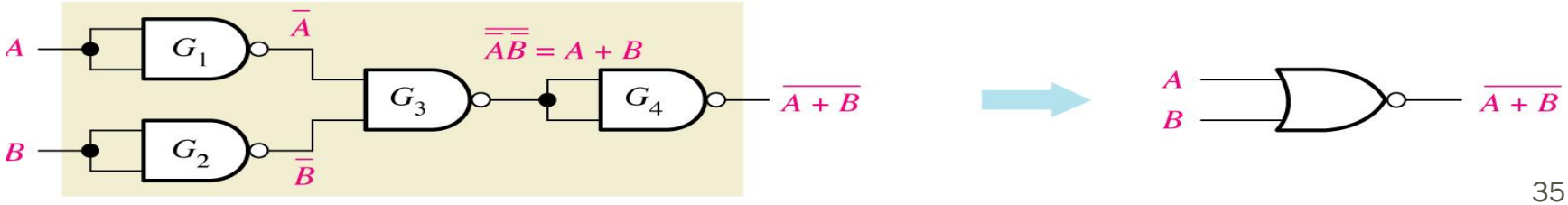
- NAND gates are sometimes called ***universal gates*** because they can be used to produce the other basic Boolean functions.

(a) A NAND gate used as an inverter

(b) Two NAND gates used as an AND gate



(c) Three NAND gates used as an OR gate



(d) Four NAND gates used as a NOR gate

NOR Logic gate

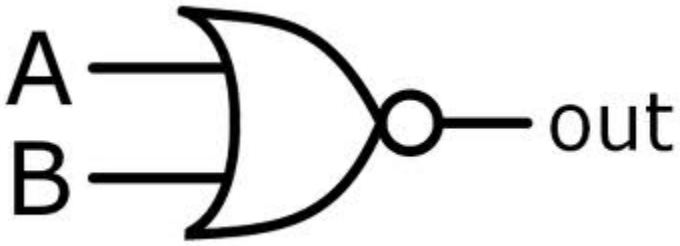
- Conversely, the NOR gate can function as either a NOR or a negative-AND, as shown by DeMorgan's theorem

$$\overline{A + B} = \overline{A}\,\overline{B}$$

NOR — Negative-AND

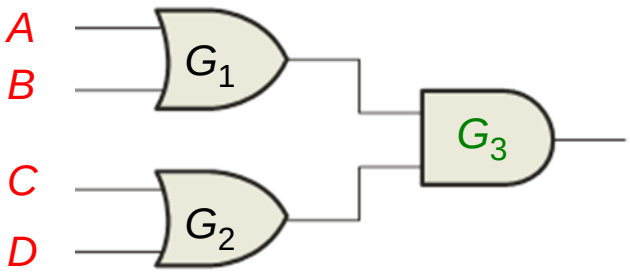
- Recap: The NOR gate outputs high when the inputs are identical and not high

Input A	Input B	Output (A+B)'
Low (0)	Low (0)	High (1)
Low (0)	High (1)	Low (0)
High (1)	Low (0)	Low (0)
High (1)	High (1)	Low (0)



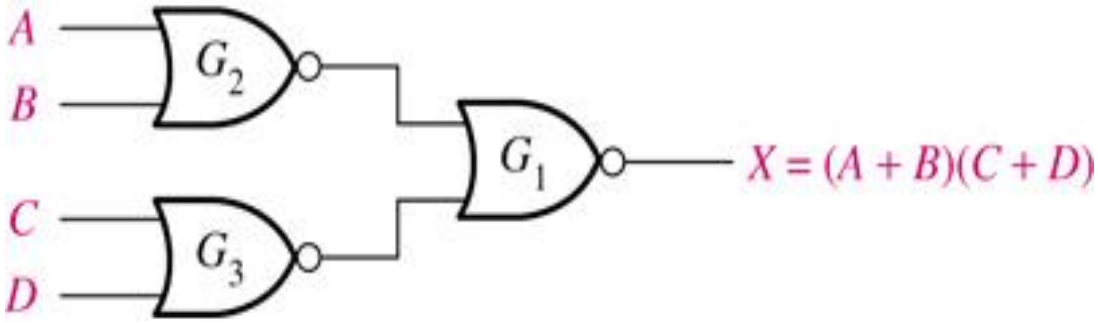
NOR logic

- Consider the logic circuit in the figure below



- Applying the same logic as before, the circuit can be implemented using just only NOR gates

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

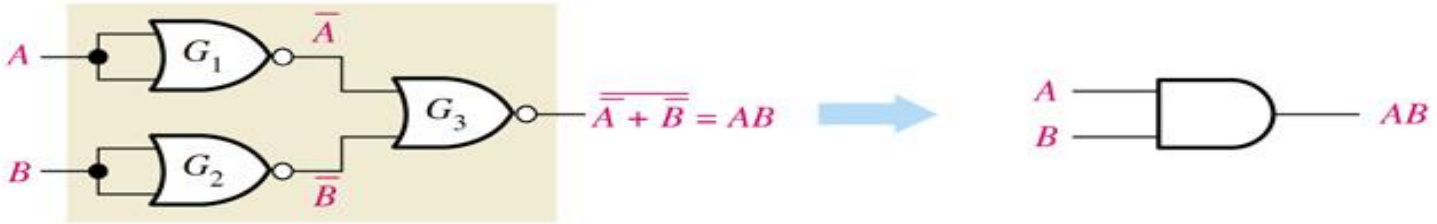


NOR Logic gate

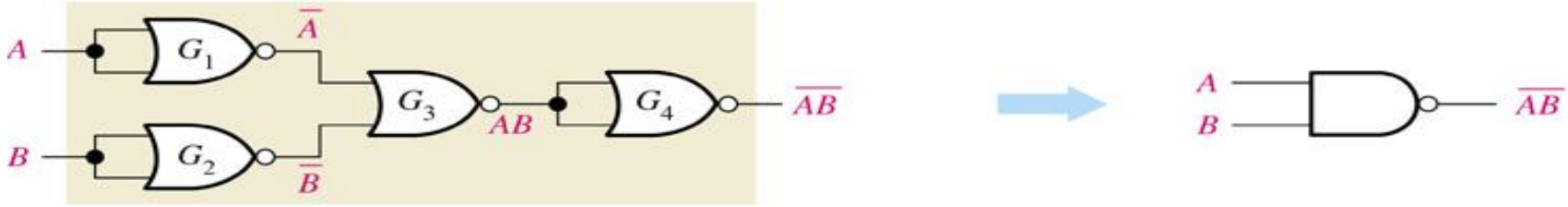
- NOR gates are also ***universal gates*** and can form other basic gates

(a) A NOR gate used as an inverter

(b) Two NOR gates used as an OR gate



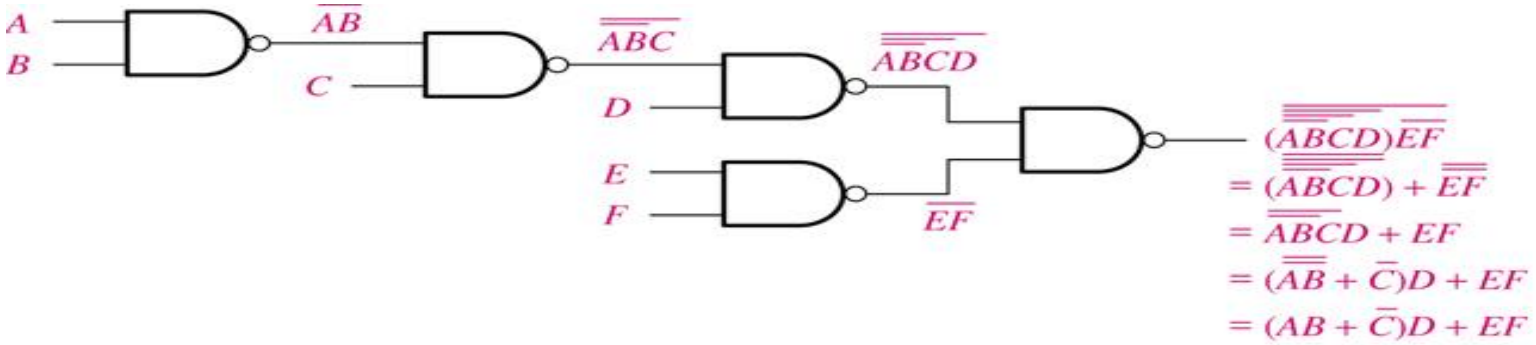
(c) Three NOR gates used as an AND gate



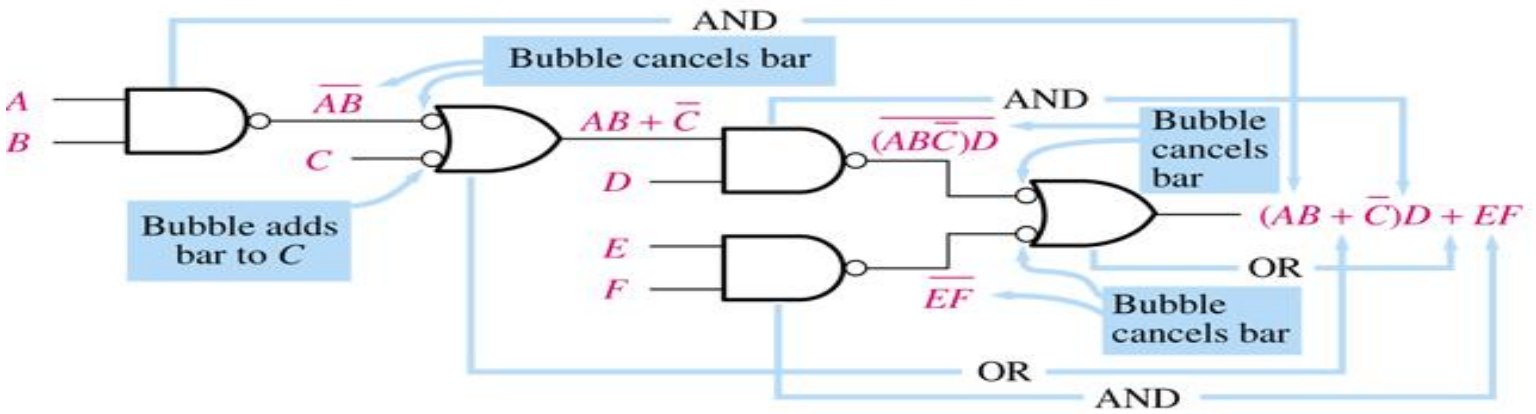
(d) Four NOR gates used as a NAND gate

NAND and NOR combinational logic

- Logic diagrams using NAND gates can be drawn with each gate represented by either NAND symbol or the equivalent negative-OR symbol to reflect the operation or the gate within the logic circuit.
- Similarly, the NOR can be replaced with the negative-AND



(a) Several Boolean steps are required to arrive at final output expression.



(b) Output expression can be obtained directly from the function of each gate symbol in the diagram.

Summary

- AND-OR logic produces an output expression in SOP form
- AOI logic produces a complemented SOP form which is actually POS form
- To do analysis of a logic circuit, you can start with the logic circuit, develop the Boolean output expression and/or truth table
- Implementation of logic circuit is the process in which you start with the Boolean output expression/truth table and develop a logic circuit that produces that output
- Two negation indicators (bubbles) will cancel each other out in logic circuit