

Logic Gates and Circuits

CSIM601251

Instructor: Tim Dosen DDAK

Slide By : Erdefi Rakun

Fasilkom UI



Outline:

- **Other Gate Types and Symbols: NAND + NOR**
- Exclusive-OR Operator and Gates
- Universal Gates
- SOP and NAND Circuits
- POS and NOR Circuits

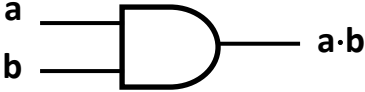

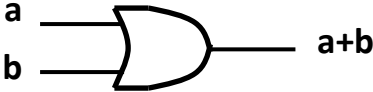
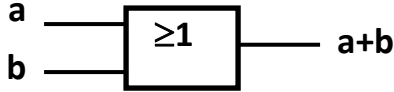
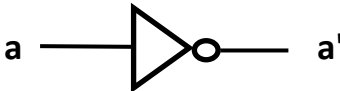
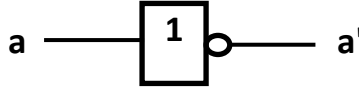
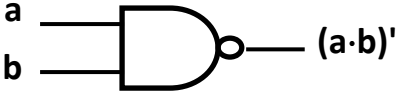
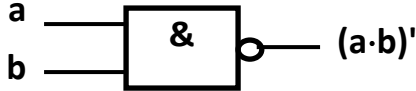
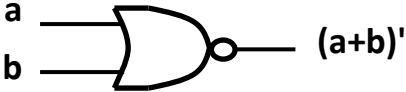
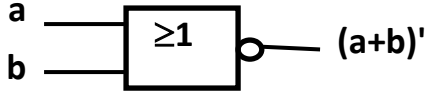
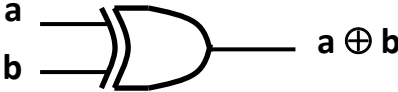
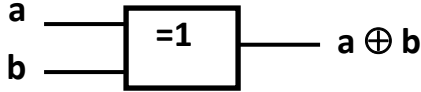
Note: Portion of this material are taken from Aaron Tan's slide and other portions of this material © 2008 by Pearson Education, Inc

Other Gate Types

- Why?
 - Implementation feasibility and low cost
 - Power in implementing Boolean functions
 - Convenient conceptual representation
- Gate classifications
 - Primitive gate - a gate that can be described using a single primitive operation type (AND or OR) plus an optional inversion(s).
 - Complex gate - a gate that requires more than one primitive operation type for its description
- Primitive gates will be covered first

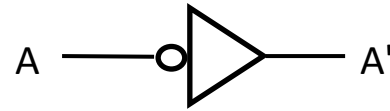
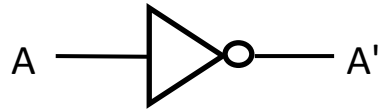
Logic Gates

- Gate symbols

	Symbol set 1	Symbol set 2 (ANSI/IEEE Standard 91-1984)
AND		
OR		
NOT		
NAND		
NOR		
EXCLUSIVE OR		

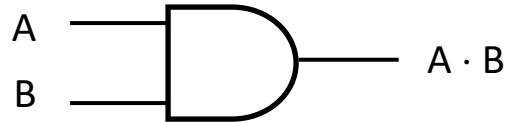
Inverter, AND, OR Gates - review

- Inverter (NOT gate)



A	A'
0	
1	

- AND gate



A	B	A · B
0	0	
0	1	
1	0	
1	1	

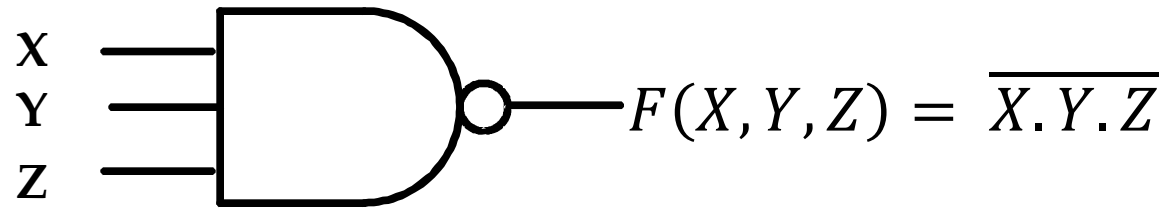
- OR gate



A	B	A + B
0	0	
0	1	
1	0	
1	1	

NAND Gate

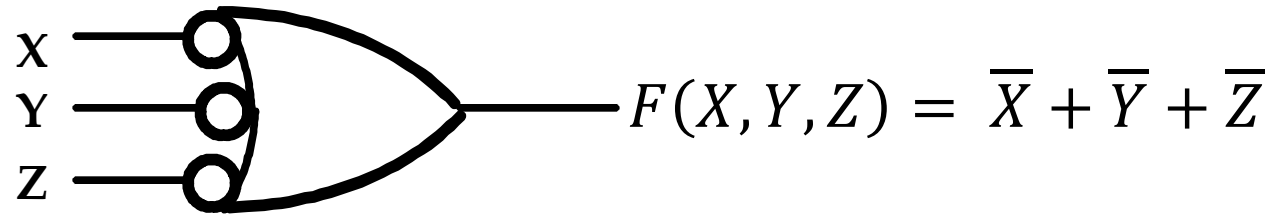
- The basic NAND gate has the following symbol, illustrated for three inputs:
 - AND-Invert (NAND)



- NAND represents NOT AND, i. e., the AND function with a NOT applied. The symbol shown is an AND-Invert. The small circle (“bubble”) represents the invert function.

NAND Gates (continued)

- Applying DeMorgan's Law gives Invert-OR (NAND)



- This NAND symbol is called Invert-OR, since inputs are inverted and then ORed together.
- AND-Invert and Invert-OR both represent the NAND gate. Having both makes visualization of circuit function easier.
- A NAND gate with one input degenerates to an inverter.

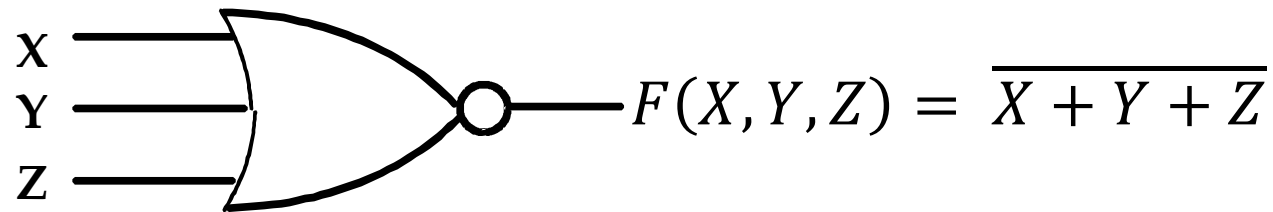
NAND Gates (continued)

- The NAND gate is the natural implementation for CMOS technology in terms of chip area and speed.
- *Universal gate* - a gate type that can implement any Boolean function.
- The NAND gate is a universal gate as shown in Figure 2-24 of the text.
- NAND usually does not have a operation symbol defined since
 - the NAND operation is not associative, and
 - we have difficulty dealing with non-associative mathematics!

NOR Gate

- The basic NOR gate has the following symbol, illustrated for three inputs:

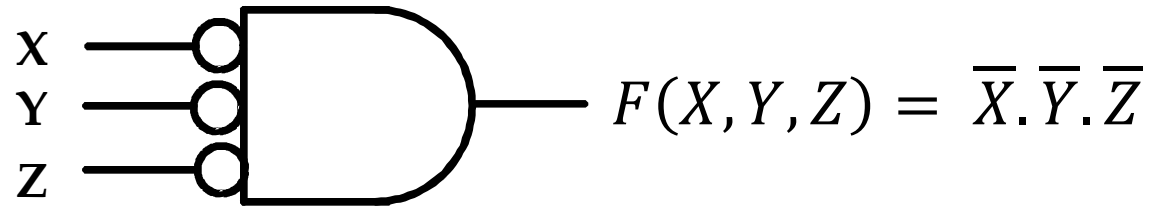
–OR-Invert (NOR)



- NOR represents NOT - OR, i. e., the OR function with a NOT applied. The symbol shown is an OR-Invert. The small circle (“bubble”) represents the invert function.

NOR Gate (continued)

- Applying DeMorgan's Law gives Invert-AND (NOR)



- This NOR symbol is called Invert-AND, since inputs are inverted and then ANDed together.
- OR-Invert and Invert-AND both represent the NOR gate. Having both makes visualization of circuit function easier.
- A NOR gate with one input degenerates to an inverter.

NOR Gate (continued)

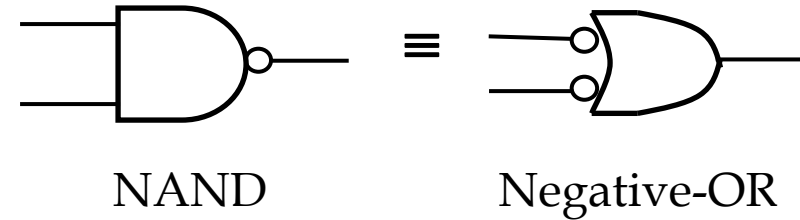
- The NOR gate is a natural implementation for some technologies other than CMOS in terms of chip area and speed.
- The NOR gate is a universal gate
- NOR usually does not have a defined operation symbol since
 - the NOR operation is not associative, and
 - we have difficulty dealing with non-associative mathematics!

NAND, NOR Gates

- **NAND gate**



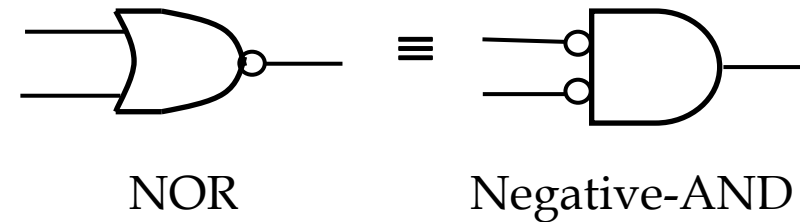
A	B	$(A \cdot B)'$
0	0	
0	1	
1	0	
1	1	



- **NOR gate**



A	B	$(A + B)'$
0	0	
0	1	
1	0	
1	1	



Outline:

- Other Gate Types and Symbols: NAND + NOR
- **Exclusive-OR Operator and Gates**
- Universal Gates
- SOP and NAND Circuits
- POS and NOR Circuits

Note: Portion of this material are taken from Aaron Tan's slide and other portions of this material © 2008 by Pearson Education, Inc

Exclusive OR/ Exclusive NOR

- The *eXclusive OR* (XOR) function is an important Boolean function used extensively in logic circuits.
- The XOR function may be;
 - implemented directly as an electronic circuit (truly a gate) or
 - implemented by interconnecting other gate types (used as a convenient representation)
- The *eXclusive NOR* function is the complement of the XOR function
- By our definition, XOR and XNOR gates are complex gates.

Exclusive OR/ Exclusive NOR

- Uses for the XOR and XNORs gate include:
 - Adders/subtractors/multipliers
 - Counters/incrementers/decrementers
 - Parity generators/checkers
- Definitions
 - The XOR function is: $X \oplus Y = X.\bar{Y} + \bar{X}.Y$
 - The eXclusive NOR (XNOR) function, otherwise known as *equivalence* is: $\overline{X \oplus Y} = X.Y + \bar{X}.\bar{Y}$
- Strictly speaking, XOR and XNOR gates do not exist for more than two inputs. Instead, they are replaced by odd and even functions.

Truth Tables for XOR/XNOR

- Operator Rules:

XOR

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

X	Y	$\overline{X \oplus Y}$ or $X \equiv Y$
0	0	1
0	1	0
1	0	0
1	1	1

- The XOR function means:
X OR Y, but NOT BOTH
- Why is the XNOR function also known as the *equivalence* function, denoted by the operator \equiv ?

XOR/XNOR (Continued)

- The XOR function can be extended to 3 or more variables. For more than 2 variables, it is called an *odd function* or *modulo 2 sum* (Mod 2 sum), not an XOR:

$$X \oplus Y \oplus Z = \bar{X}.\bar{Y}.Z + \bar{X}.Y.\bar{Z} + X.\bar{Y}.\bar{Z} + X.Y.Z$$

- The complement of the odd function is the even function.
- The XOR identities:

$$X \oplus 0 = X$$

$$X \oplus X = 0$$

$$X \oplus Y = Y \oplus X$$

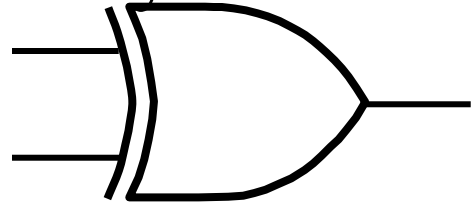
$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

$$X \oplus 1 = \bar{X}$$

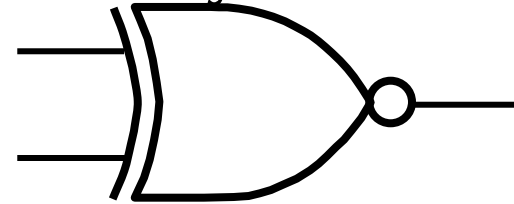
$$X \oplus \bar{X} = 1$$

Symbols For XOR and XNOR

- XOR symbol:



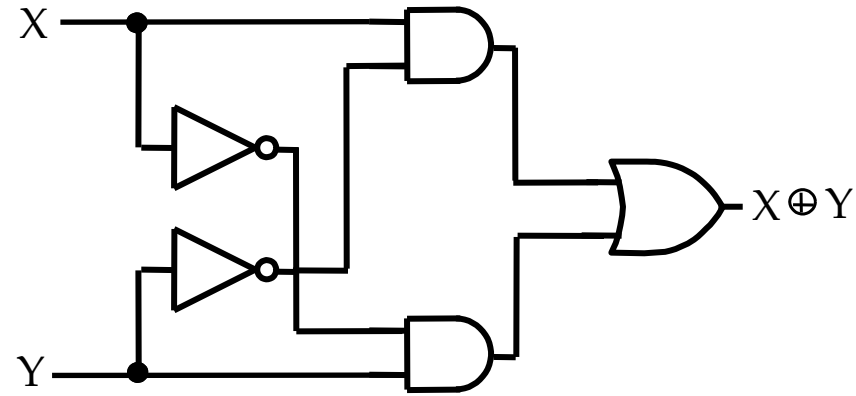
- XNOR symbol:



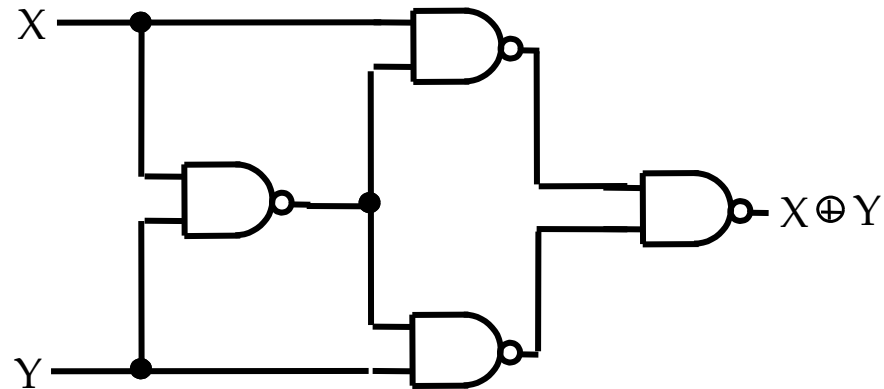
- Shaped symbols exist only for two inputs

XOR Implementations

- The simple SOP implementation uses the following structure:



- A NAND only implementation is:

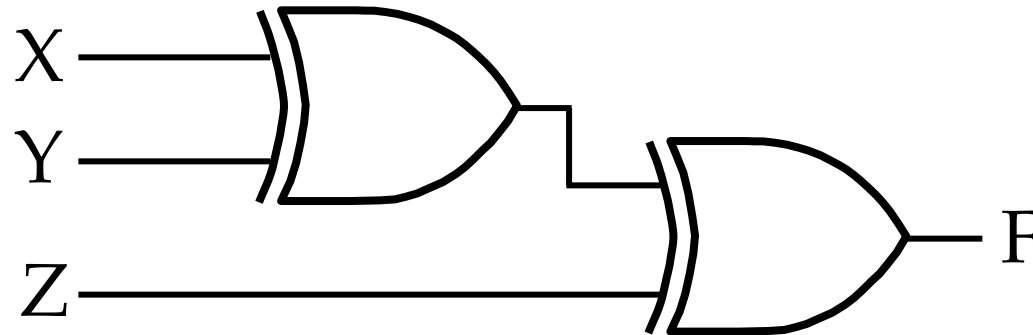


Odd and Even Functions

- The odd and even functions on a K-map form “checkerboard” patterns.
- The 1s of an odd function correspond to minterms having an index with an odd number of 1s.
- The 1s of an even function correspond to minterms having an index with an even number of 1s.
- Implementation of odd and even functions for greater than four variables as a two-level circuit is difficult, so we use “trees” made up of :
 - 2-input XOR or XNORs
 - 3- or 4-input odd or even functions

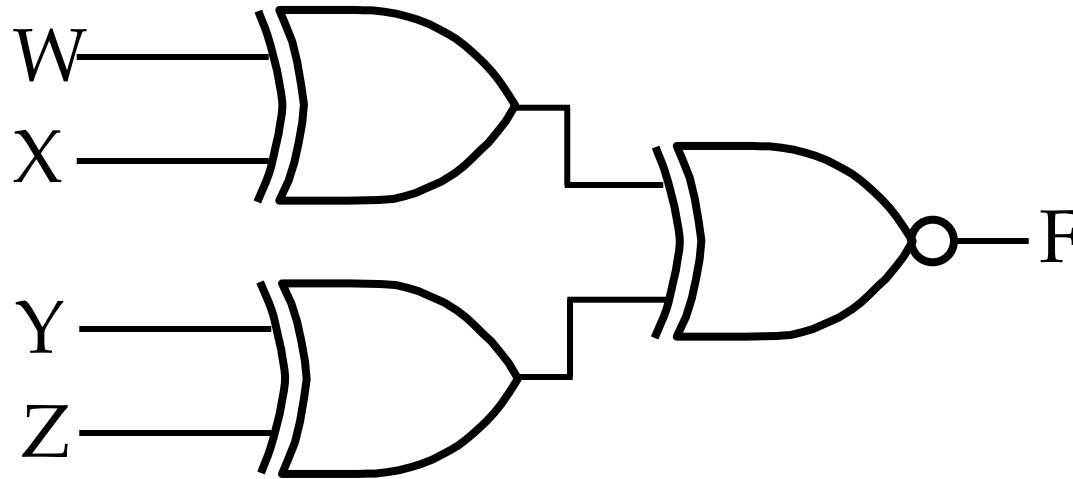
Example: Odd Function Implementation

- Design a 3-input odd function $F = X \oplus Y \oplus Z$ with 2-input XOR gates
- Factoring, $F = (X \oplus Y) \oplus Z$
- The circuit:



Example: Even Function Implementation

- Design a 4-input odd function $F = \overline{W \oplus X \oplus Y \oplus Z}$ with 2-input XOR and XNOR gates
- Factoring, $F = \overline{(W \oplus X) \oplus (Y \oplus Z)}$
- The circuit:



Outline:

- Other Gate Types and Symbols: NAND + NOR
- Exclusive-OR Operator and Gates
- **Universal Gates**
- SOP and NAND Circuits
- POS and NOR Circuits

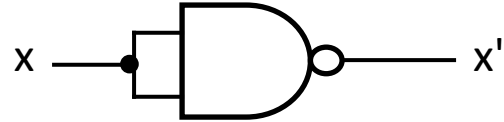
Note: Portion of this material are taken from Aaron Tan's slide and other portions of this material © 2008 by Pearson Education, Inc

Universal Gates

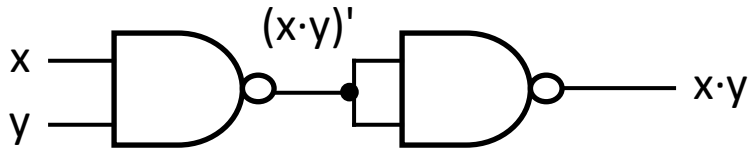
- AND/OR/NOT gates are sufficient for building any Boolean function.
- We call the set {AND, OR, NOT} a **complete set of logic**.
- However, other gates are also used:
 - Usefulness (eg: XOR gate for parity bit generation)
 - Economical
 - Self-sufficient (eg: NAND/NOR gates)

NAND Gates

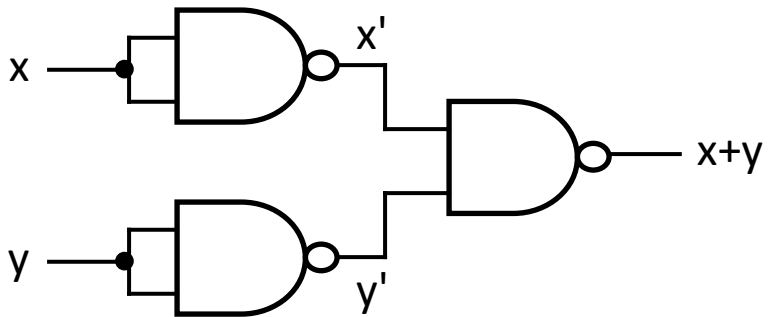
- {NAND} is a complete set of logic.
- Proof: Implement NOT/ AND/OR using only NAND gates.



$$(x \cdot x)' = x' \quad (\text{idempotency})$$



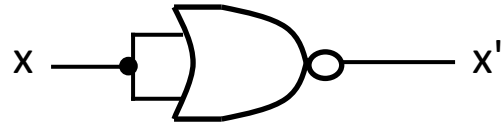
$$\begin{aligned} ((x \cdot y)' \cdot (x \cdot y)')' &= ((x \cdot y)')' \quad (\text{idempotency}) \\ &= x \cdot y \quad (\text{involution}) \end{aligned}$$



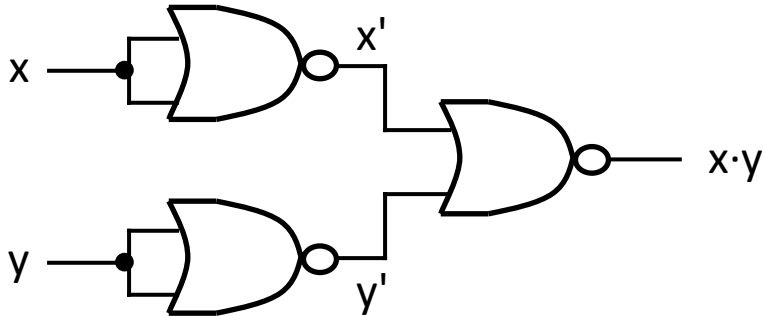
$$\begin{aligned} ((x \cdot x)' \cdot (y \cdot y)')' &= (x' \cdot y')' \quad (\text{idempotency}) \\ &= (x')' + (y')' \quad (\text{DeMorgan}) \\ &= x + y \quad (\text{involution}) \end{aligned}$$

NOR Gates

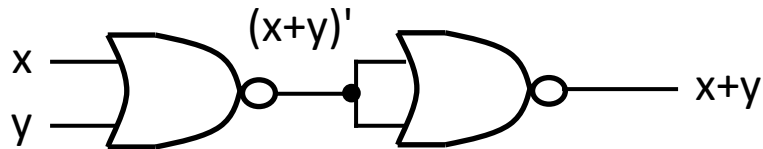
- {NOR} is a complete set of logic.
- Proof: Implement NOT/AND/OR using only NOR gates.



$$(x+x)' = x' \quad (\text{idempotency})$$



$$\begin{aligned} ((x+x)' + (y+y'))' &= (x' + y')' \quad (\text{idempotency}) \\ &= (x')' \cdot (y')' \quad (\text{DeMorgan}) \\ &= x \cdot y \quad (\text{involution}) \end{aligned}$$



$$\begin{aligned} ((x+y)' + (x+y'))' &= ((x+y))' \quad (\text{idempotency}) \\ &= x+y \quad (\text{involution}) \end{aligned}$$

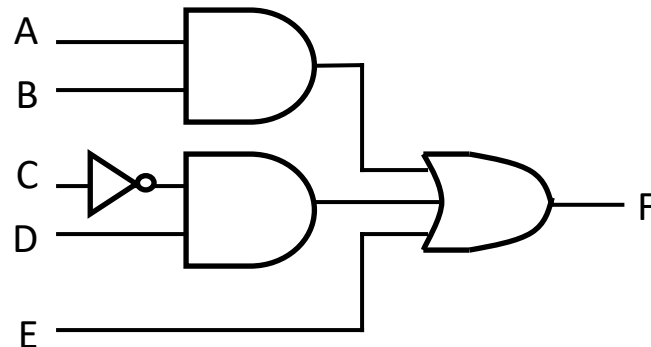
Outline:

- Other Gate Types and Symbols: NAND + NOR
- Exclusive-OR Operator and Gates
- Universal Gates
- **SOP and NAND Circuits**
- **POS and NOR Circuits**

Note: Portion of this material are taken from Aaron Tan's slide and other portions of this material © 2008 by Pearson Education, Inc

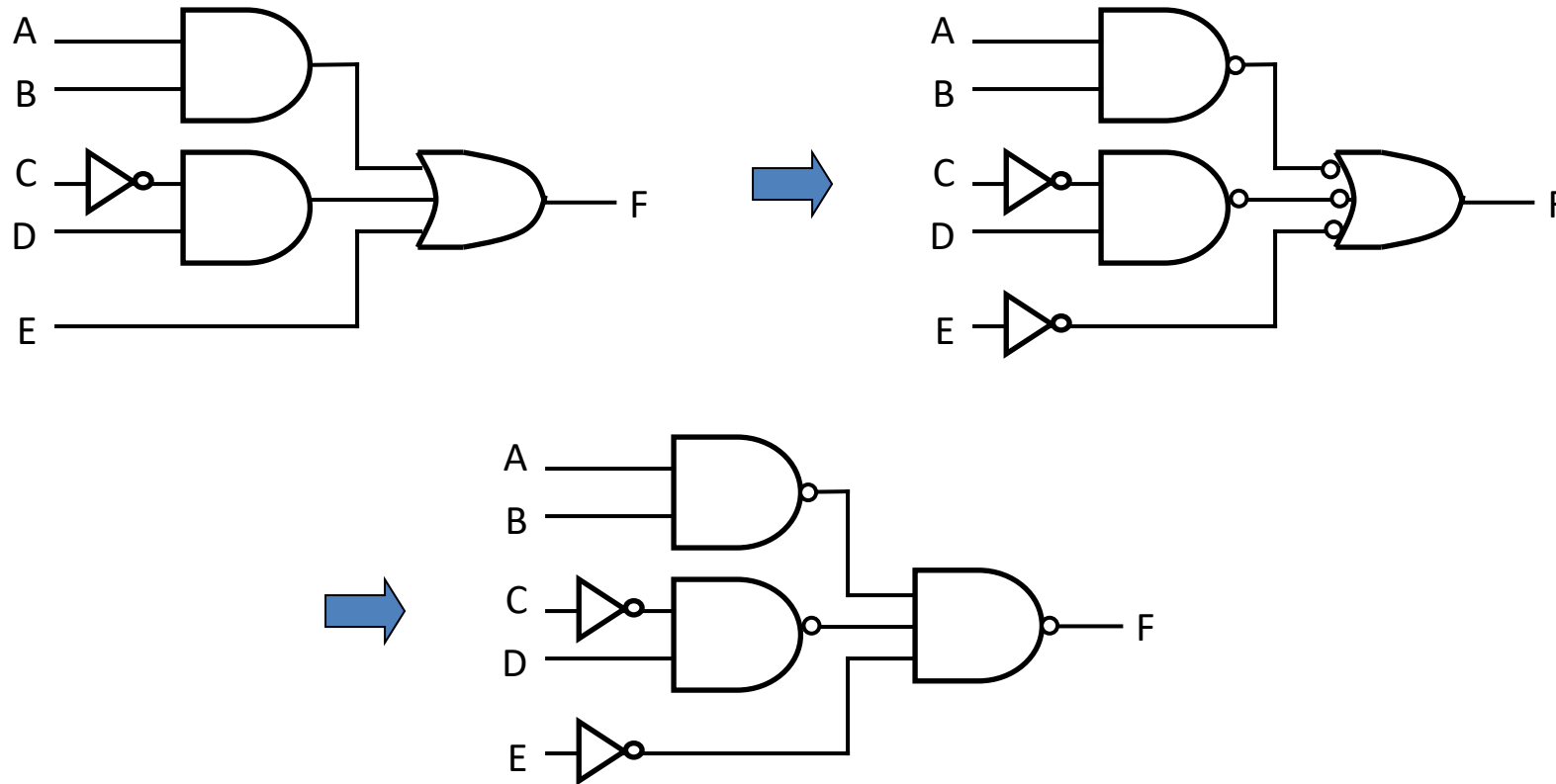
SOP and NAND Circuits (1/2)

- An SOP expression can be easily implemented using
 - 2-level AND-OR circuit
 - 2-level NAND circuit
- Example: $F = A \cdot B + C' \cdot D + E$
 - Using 2-level AND-OR circuit



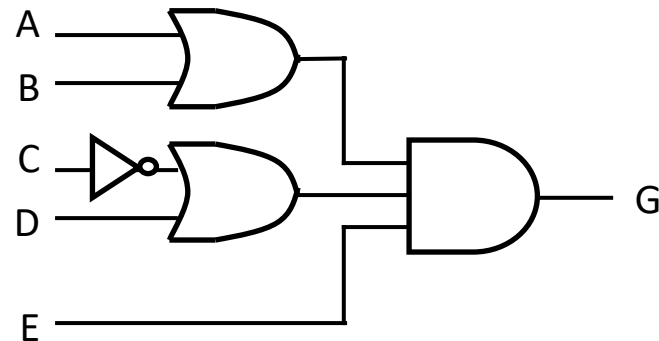
SOP and NAND Circuits (2/2)

- Example: $F = A \cdot B + C' \cdot D + E$
 - Using 2-level NAND circuit



POS and NOR circuits (1/2)

- A POS expression can be easily implemented using
 - 2-level OR-AND circuit
 - 2-level NOR circuit
- Example: $G = (A+B) \cdot (C'+D) \cdot E$
 - Using 2-level OR-AND circuit



POS and NOR circuits (2/2)

- Example: $G = (A+B) \cdot (C'+D) \cdot E$
 - Using 2-level NOR circuit

