

## Lecture 2



信息科学与技术学院  
School of Information Science and Technology

# Digital Logic Basis

Haoyu Wang  
ShanghaiTech University



## Outline



信息科学与技术学院  
School of Information Science and Technology

- Truth Table
- Basic Logic Operation and Gates
- Logic Circuits
- NOR Gates and NAND Gates
- Boolean Theorems
- Demorgan's Theorems



## Outline

- **Truth Table**
- Basic Logic Operation and Gates
- Logic Circuits
- NOR Gates and NAND Gates
- Boolean Theorems
- Demorgan's Theorems



## Boolean Constants and Variables

- Boolean algebra allows only two values—0 and 1

Logic 0	Logic 1
False	True
Off	On
LOW	HIGH
No	Yes
Open switch	Closed switch

- The 3 basic logic operations:
  - » OR, AND, and NOT.



# Truth Table

- Describes the **relationship** between the **input** and **output** of a logic circuit.
- # of entries corresponds to # of inputs.
  - » A 2-input table:  $2^2 = 4$  entries
  - » A 3-input table:  $2^3 = 8$  entries

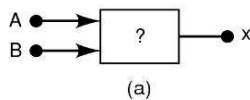


## Examples of truth tables with 2, 3, and 4 inputs.

Output

Inputs

A	B	x
0	0	1
0	1	0
1	0	1
1	1	0



A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b)

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

(c)



## Outline

- Truth Table
- **Basic Logic Operation and Gates**
- Logic Circuits
- NOR Gates and NAND Gates
- Boolean Theorems
- Demorgan's Theorems



## OR Operation with OR Gates

- The Boolean expression for the **OR** operation

is:  $X = A + B$  — Read as “X equals A OR B”

The + sign does *not* stand for ordinary addition—it stands for the OR operation

- The OR operation is similar to addition, but when  $A = 1$  and  $B = 1$ , the OR operation produces:

$$1 + 1 = 1 \text{ not } 1 + 1 = 2$$

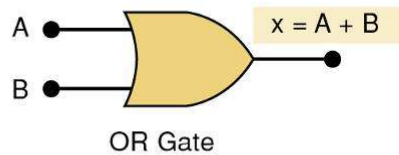
In the Boolean expression  $x = 1 + 1 + 1 = 1...$   
*x is true (1) when A is true (1) OR B is true (1) OR C is true (1)*

- An **OR gate** is a circuit with two or more inputs, whose output is equal to the OR combination of the inputs.

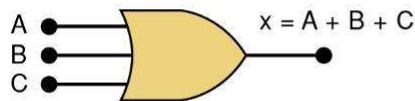
Truth table/circuit symbol for a **2 input OR gate**.

OR

A	B	$x = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

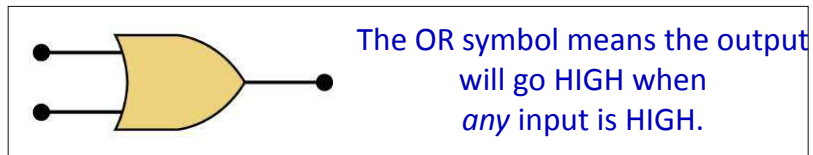
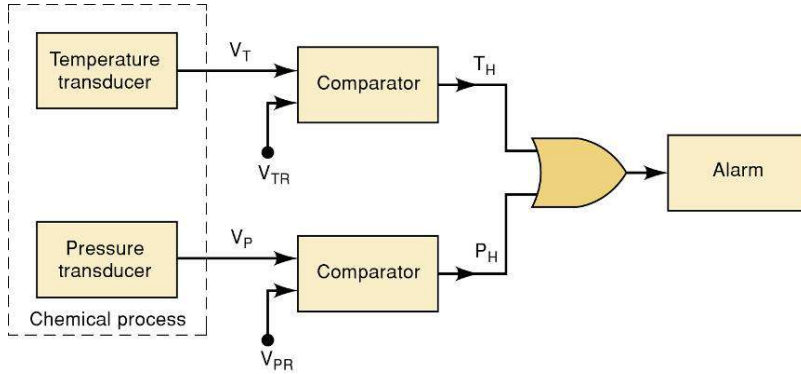


Truth table/circuit symbol for a **3 input OR gate**.



A	B	C	$x = A + B + C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

### Example of the use of an OR gate in an alarm system.



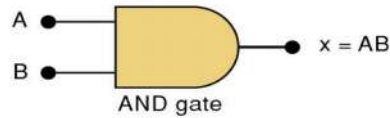
## AND Operations with AND gates

- The **AND** operation is similar to multiplication:

$$X = A \cdot B \cdot C \text{ — Read as "X equals A AND B AND C"}$$

The  $\cdot$  sign does *not* stand for ordinary multiplication—it stands for the AND operation.  
*x is true (1) when A AND B AND C are true (1)*

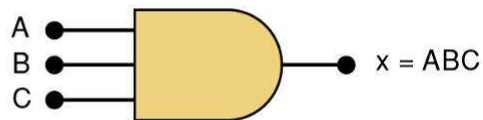
AND		
A	B	$x = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1



Truth table — Gate symbol.

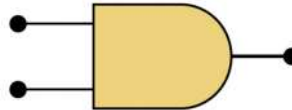
Truth table/circuit symbol for a 3 input AND gate.

A	B	C	$x = ABC$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



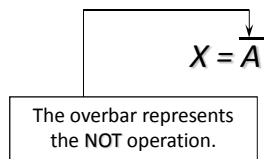


The AND symbol on a logic-circuit diagram tells you output will go HIGH only when all inputs are HIGH.

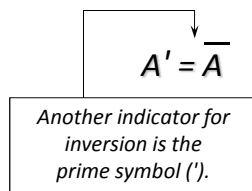


## NOT Operation

- The Boolean expression for the **NOT** operation:



— Read as: “X equals NOT A”  
“X equals the *inverse* of A”  
“X equals the *complement* of A”



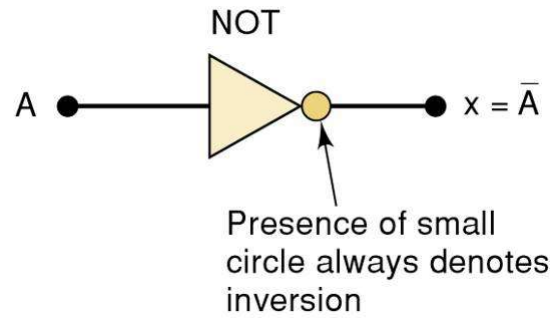
NOT

A	x = $\overline{A}$
0	1
1	0

NOT Truth Table

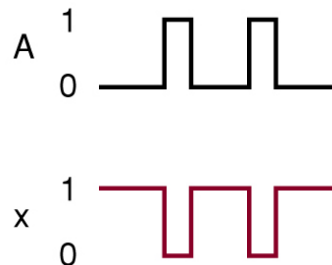


A NOT circuit—commonly called an INVERTER.



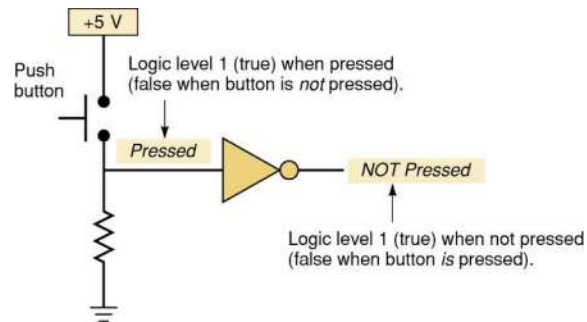
This circuit *always* has only a single input, and the output logic level is always *opposite* to the logic level of this input.

The INVERTER inverts (*complements*) the input signal at all points on the waveform.



Whenever the input = 0, output = 1, and vice versa.

## Typical application of the NOT gate.



This circuit provides an expression that is true when the button is not pressed.

## Outline

- Truth Table
- Basic Logic Operation and Gates
- **Logic Circuits**
- NOR Gates and NAND Gates
- Boolean Theorems
- Demorgan's Theorems



## Boolean Operations

### Summarized rules for **OR**, **AND** and **NOT**

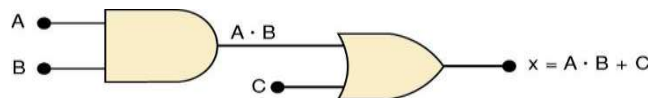
<i>OR</i>	<i>AND</i>	<i>NOT</i>
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\overline{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\overline{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

These three basic Boolean operations  
can describe any logic circuit.

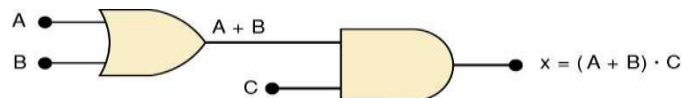


## Describing Logic Circuits Algebraically

- If an expression contains both **AND** and **OR** gates, the **AND** operation will be performed first.

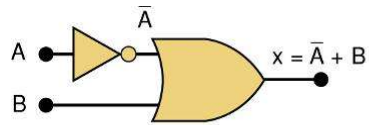


- Unless there is a parenthesis in the expression.

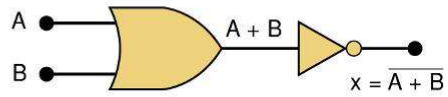


- Whenever an **INVERTER** is present, output is equivalent to input, with a bar over it.

– Input A through an inverter equals  $\bar{A}$ .

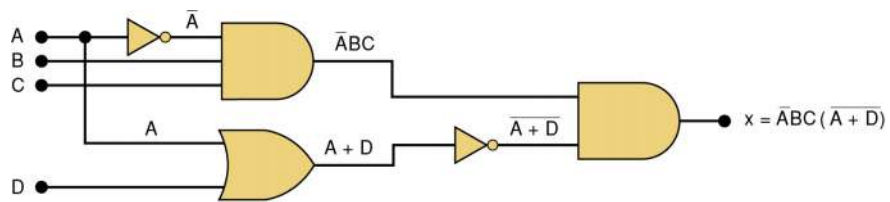


(a)



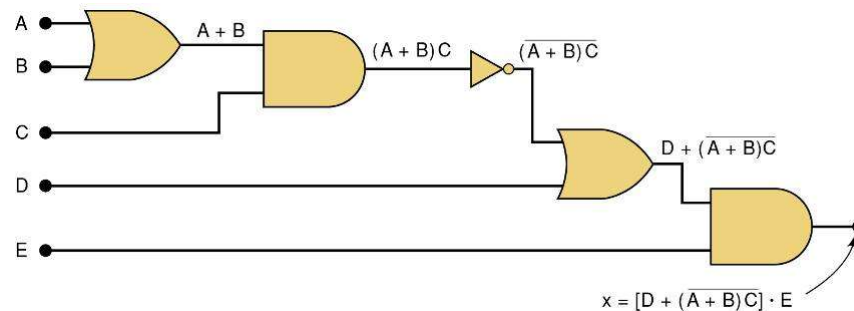
(b)

- Further examples...





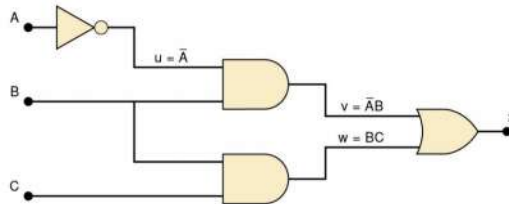
- Further examples...



## Evaluating Logic Circuit Outputs

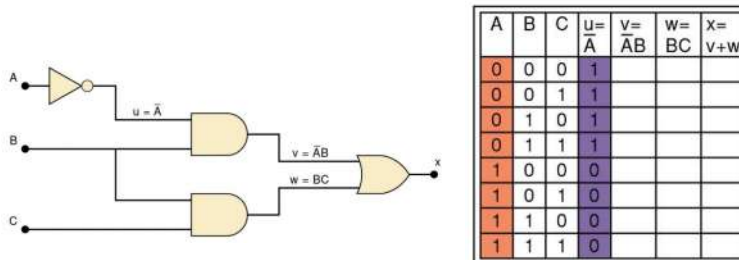
- Rules for evaluating a Boolean expression:
  - » Perform all **inversions** of **single** terms.
  - » Perform all operations within **parenthesis**.
  - » Perform **AND** operation before an **OR** operation unless parenthesis indicate otherwise.
  - » If an expression has a **bar** over it, perform operations inside the expression, and then invert the result.

- The best way to analyze a circuit made up of multiple logic gates is to use a **truth table**.
  - » It allows you to analyze one gate or logic combination at a time.
  - » It allows you to easily double-check your work.
  - » When you are done, you have a table of tremendous benefit in troubleshooting the logic circuit.



## Step 1

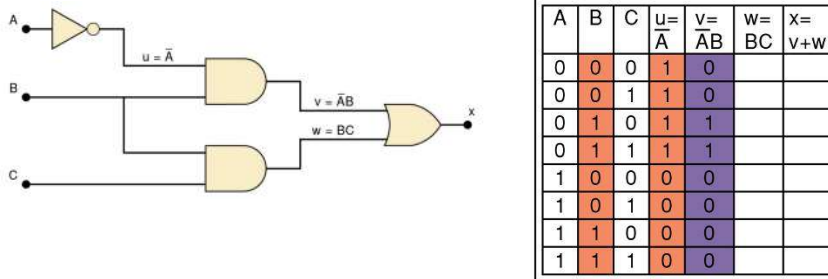
- List all input combinations.
- Create a column in the truth table for each node.
- Fill the values for  $u$ .



Node  $u$  has been filled as the complement of  $A$

## Step 2

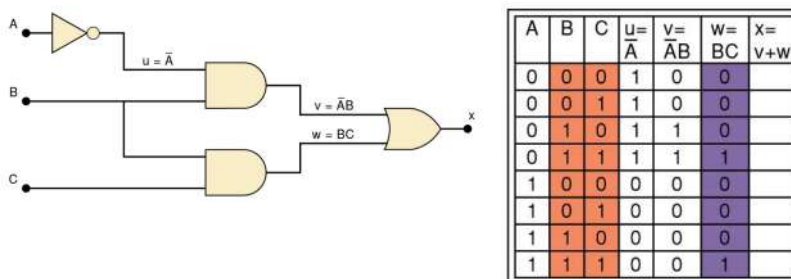
- Fill in the values for column  $v$ .



$v = \bar{A}B$  — Node  $v$  should be HIGH when  $A$  (node  $u$ ) is HIGH AND  $B$  is HIGH

## Step 3

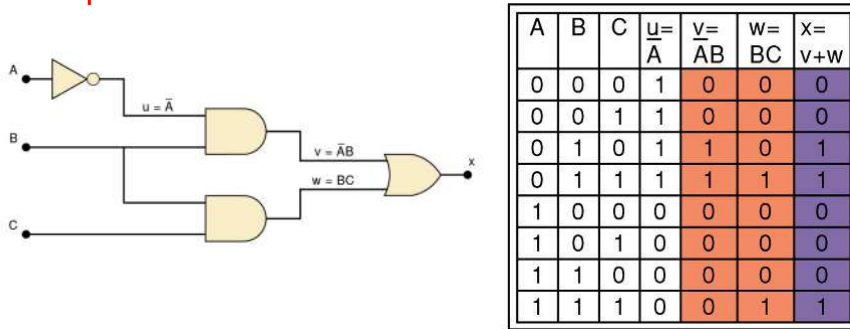
- Predict the values at node  $w$  which is the logical product of  $BC$ .



This column is HIGH whenever  $B$  is HIGH AND  $C$  is HIGH

## Step 4

- Logically combine columns  $v$  and  $w$  to predict the output  $x$ .



Since  $x = v + w$ , the  $x$  output will be HIGH when  $v$  OR  $w$  is HIGH

## Evaluating Logic Circuit Outputs

- Output logic levels can be determined directly from a circuit diagram.
  - » Output of each gate is noted until final output is found.
    - Technicians frequently use this method.



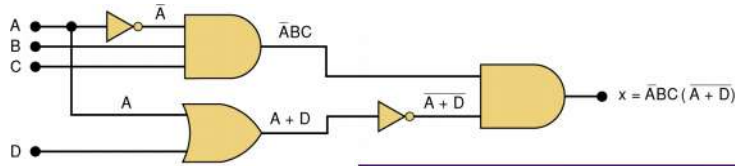


Table of logic state  
at each node of the  
circuit shown.

A	B	C	D	t = ABC	u = A + D	v = A + D	x = tv
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	1	0	0
1	1	1	1	0	1	0	0

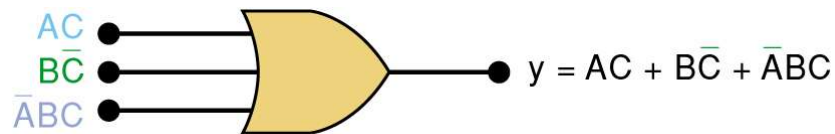


## Draw a Circuit From a Boolean Expression

- $X = A \cdot B \cdot C$   
» A 3-input **AND** gate.
- $X = A + \bar{B}$   
» A 2-input **OR** gate with an **INVERTER** on one of the inputs.



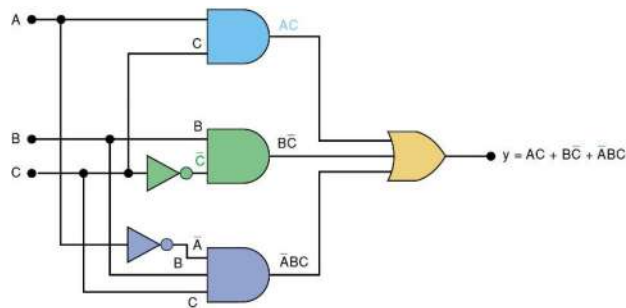
A circuit with output  $y = AC + BC\bar{C} + \bar{A}BC$  contains three terms which are **OR**ed together.



...and requires a three-input **OR** gate.

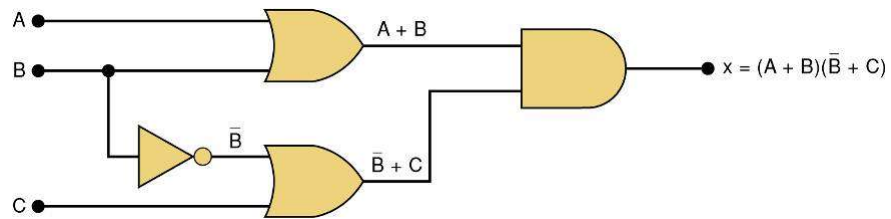


- Each **OR** gate input is an **AND** product term,
  - » An **AND** gate with appropriate inputs can be used to generate each of these terms.





Circuit diagram to implement  $x = (A + B)(\bar{B} + C)$



## Outline

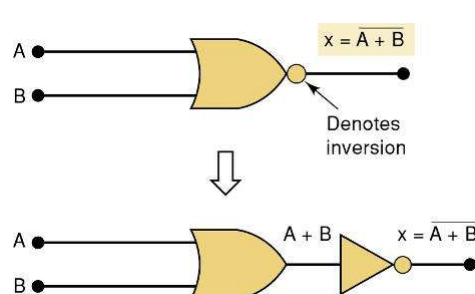
- Truth Table
- Basic Logic Operation and Gates
- Logic Circuits
- **NOR Gates and NAND Gates**
- Boolean Theorems
- Demorgan's Theorems

## NOR Gates and NAND Gates

- Combine basic **AND**, **OR**, and **NOT** operations.
  - » Simplifying the writing of Boolean expressions
- Output of **NAND** and **NOR** gates may be found by determining the output of an **AND** or **OR** gate, and inverting it.
  - » The truth tables for **NOR** and **NAND** gates show the complement of truth tables for **OR** and **AND** gates.

- The **NOR** gate is an inverted **OR** gate.

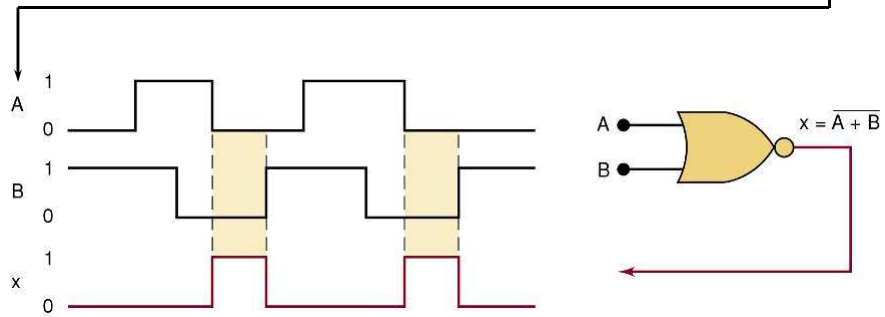
$$x = \overline{A + B}$$



		OR		NOR	
A	B	$A + B$	$\overline{A + B}$	$A + B$	$\overline{A + B}$
0	0	0	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	1	0	1	0

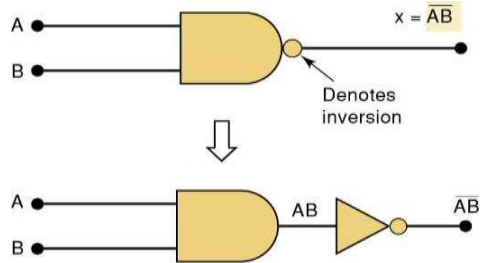
Only if both inputs are 0, the output will be 1

Output waveform of a **NOR** gate for the input waveforms shown here.



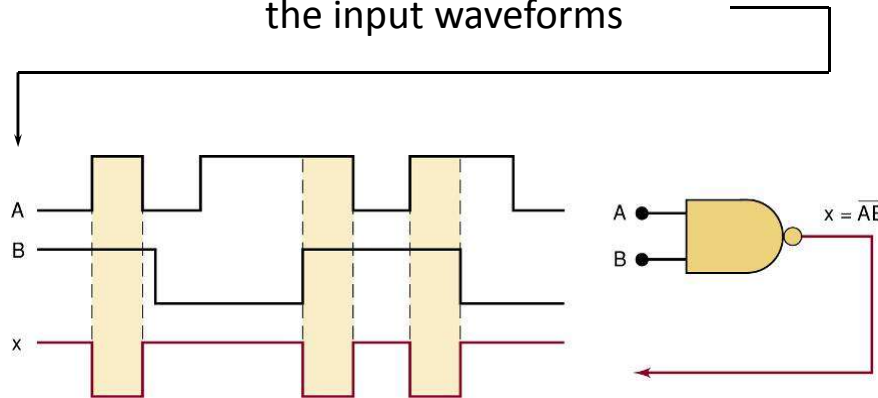
- The NAND gate is an inverted AND gate.

–  $x = \overline{AB}$

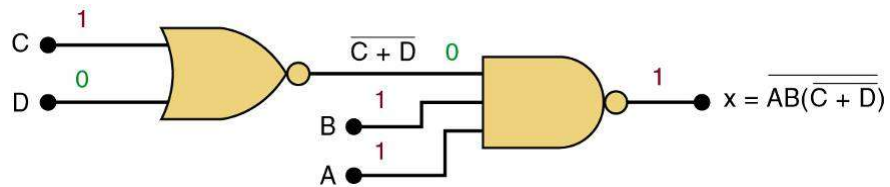


		AND		NAND	
A	B	AB		AB	
0	0	0		1	
0	1	0		1	
1	0	0		1	
1	1	1		0	

### Output waveform of a **NAND** gate for the input waveforms



Logic circuit with the expression  $x = \overline{AB \cdot (\overline{C + D})}$  using only NOR and NAND gates.



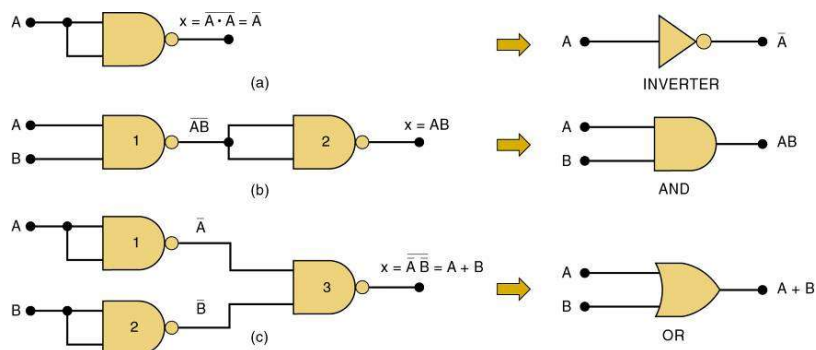


## Universality of NAND and NOR Gates

- **NAND** or **NOR** gates can be used to create the three basic logic expressions.
  - » **OR**, **AND**, and **INVERT**.
    - Provides flexibility—very useful in logic circuit design.

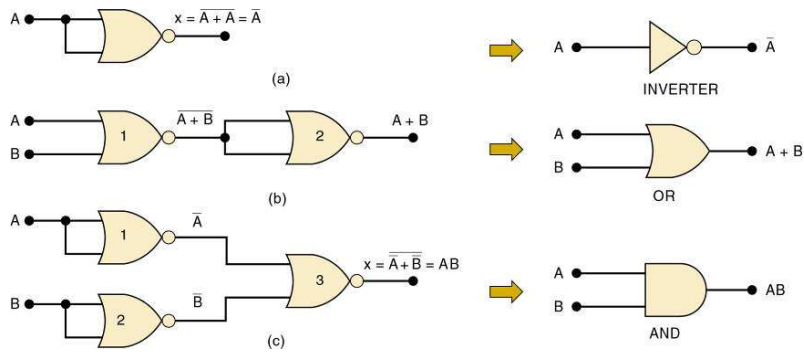


### How combinations of NANDs are used to create the three logic functions.



It is possible to implement any logic expression using *only* NAND gates and no other type of gate, as shown.

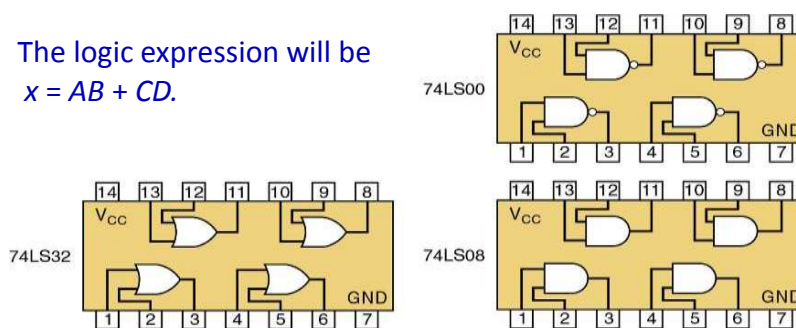
**How combinations of NORs are used to create the three logic functions.**



NOR gates can be arranged to implement any of the Boolean operations, as shown.

**A logic circuit to generate a signal  $x$ , that will go HIGH whenever conditions  $A$  and  $B$  exist simultaneously, or whenever conditions  $C$  and  $D$  exist simultaneously.**

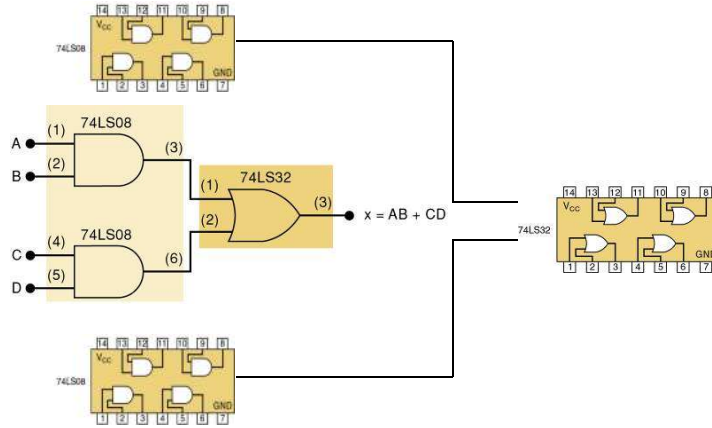
The logic expression will be  $x = AB + CD$ .



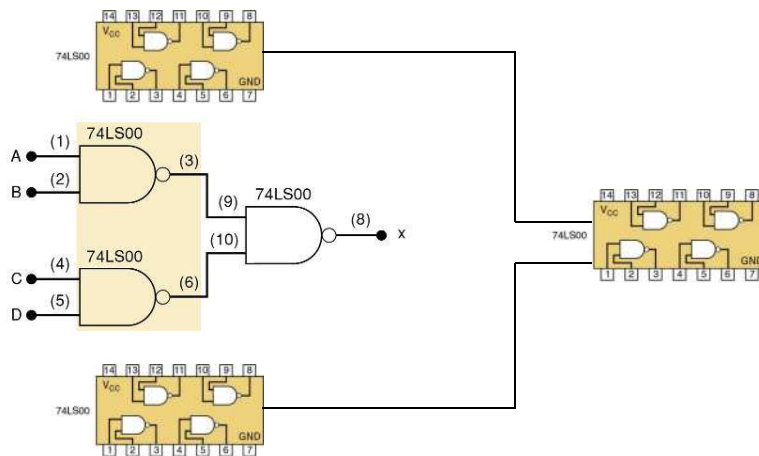




### Possible Implementations # 1



### Possible Implementations #2



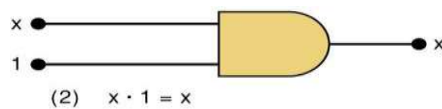
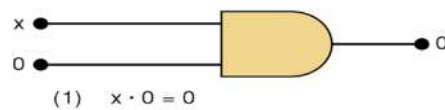


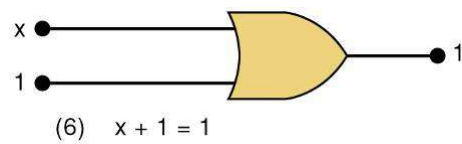
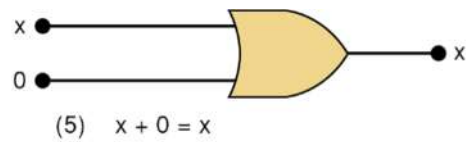
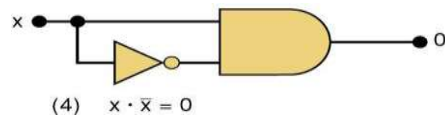
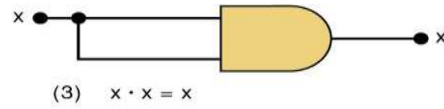
## Outline

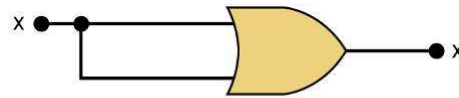
- Truth Table
- Basic Logic Operation and Gates
- Logic Circuits
- NOR Gates and NAND Gates
- **Boolean Theorems**
- Demorgan's Theorems



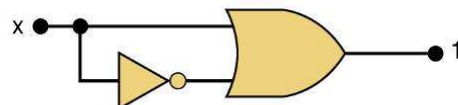
## Boolean Theorems







(7)  $x + x = x$



(8)  $x + \bar{x} = 1$



## Multivariable Theorems

### Commutative laws

(9)  $x + y = y + x$

(10)  $x \cdot y = y \cdot x$

### Associative laws

(11)  $x + (y + z) = (x + y) + z = x + y + z$

(12)  $x(yz) = (xy)z = xyz$

### Distributive law

(13a)  $x(y + z) = xy + xz$

(13b)  $(w + x)(y + z) = wy + xy + wz + xz$



Theorems (14) and (15) do not have counterparts in ordinary algebra. Each can be proved by **trying all possible cases for  $x$  and  $y$ .**

(14)  $x + \overline{xy} = x$

Analysis table & factoring for Theorem (14)

(15a)  $\overline{x} + \overline{xy} = \overline{x} + y$

(15b)  $\overline{x} + xy = \overline{x} + y$

$$\begin{aligned} x + xy &= x(1 + y) \\ &= x \cdot 1 && \text{[using theorem (6)]} \\ &= x && \text{[using theorem (2)]} \end{aligned}$$

x	y	xy	x + xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1



## Outline

- Truth Table
- Basic Logic Operation and Gates
- Logic Circuits
- NOR Gates and NAND Gates
- Boolean Theorems
- **Demorgan's Theorems**



# DeMorgan's Theorems

$$(16) \quad \overline{(x + y)} = \bar{x} \cdot \bar{y}$$

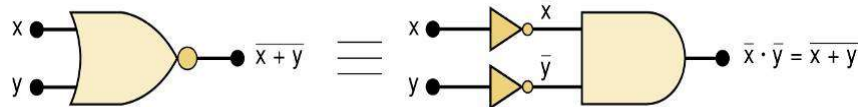
$$(17) \quad \overline{(x \cdot y)} = \bar{x} + \bar{y}$$

Each of DeMorgan's theorems can readily be proven by checking for all possible combinations of  $x$  and  $y$ .

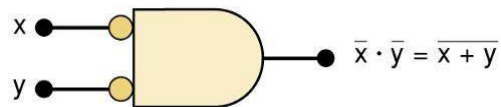


## Equivalent circuits implied by Theorem (16)

$$(16) \quad \overline{(x + y)} = \bar{x} \cdot \bar{y}$$



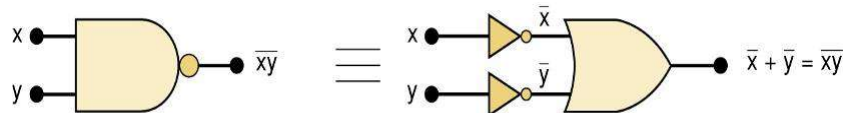
The alternative symbol for the NOR function.



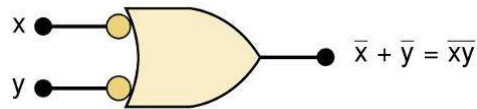


### Equivalent circuits implied by Theorem (17)

$$(17) \quad \overline{(x \cdot y)} = \bar{x} + \bar{y}$$



The alternative symbol  
for the NAND function.



## References

- David M. Harris and Sarah L. Harris, *Digital Design and Computer Architecture*
- Ronald J. Tocci and Neal Widmer, *Digital Systems Principles and Applications*, 11<sup>th</sup> edition