

ENGG2020 Digital Logic and Systems

Chapter 2: Logic Gates and Boolean Algebra

The Chinese University of Hong Kong

Logic Gates

Boolean algebra allows only two values—0 and 1.

Logic 0 can be: *false, off, low, no, open switch.*

Logic 1 can be: *true, on, high, yes, closed switch.*

The three basic logic operations:

OR, AND, and NOT

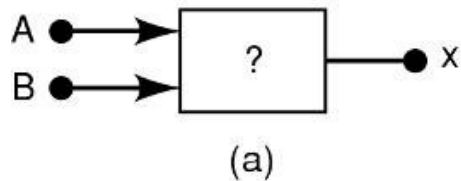
Logic 0	Logic 1
False	True
Off	On
LOW	HIGH
No	Yes
Open switch	Closed switch

Logic Gates

A truth table describes the relationship between the input and output of a logic circuit.

A 2-input table have $2^2 = 4$ entries. A 3-input table have $2^3 = 8$ entries.

Inputs		Output
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)

Logic Gates

The Boolean expression for the **OR** operation is:

$$\mathbf{X = A + B} \text{ — Read as “X equals A OR B”}$$

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:

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

In the Boolean expression $\mathbf{x = 1 + 1 + 1 = 1...}$

*x is true (1) when A is true (1) **OR** B is true (1) **OR** C is true (1)*

Logic Gates

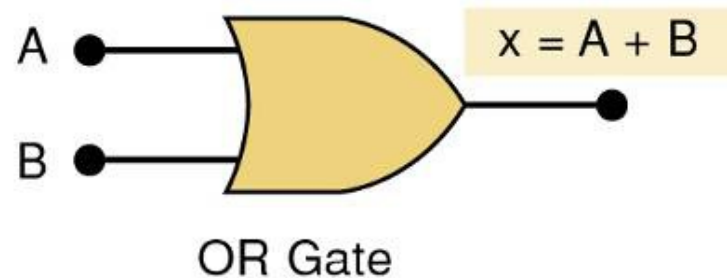
An **OR gate** is a circuit with two or more inputs, whose output is equal to the **OR** combination of the inputs.

Truth table/circuit symbol for a two input OR gate.

OR

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

(a)

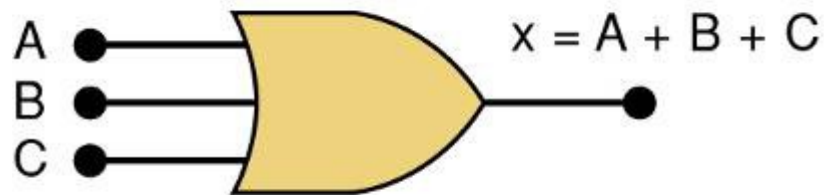


(b)

Logic Gates

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 three 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

Logic Gates

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

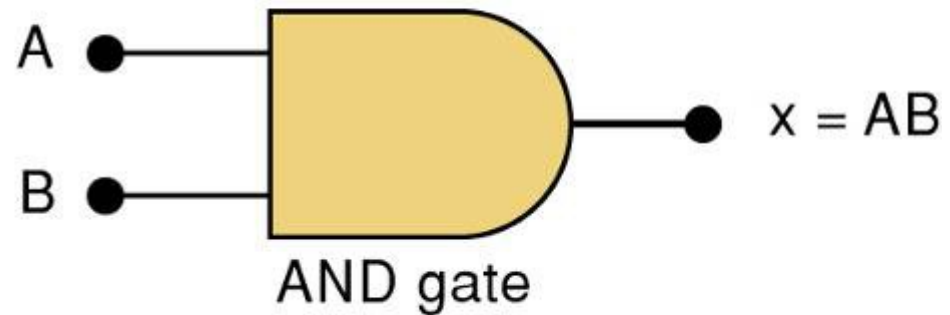
$X = A \cdot B \cdot C$ — Read as “ **X** equals **A AND B AND C** ”

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

(a)

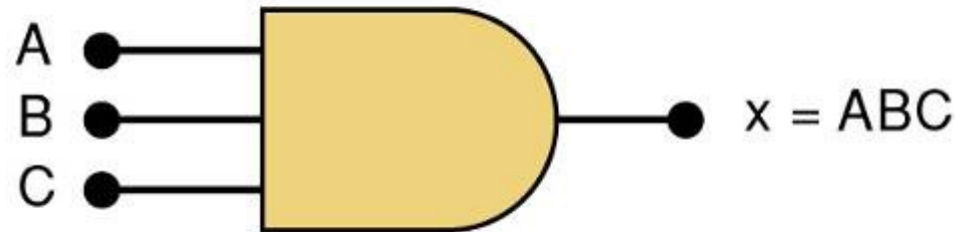


(b)

Logic Gates

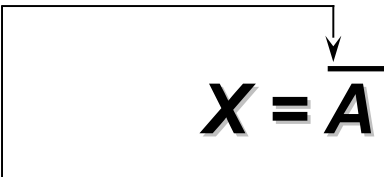
Truth table/circuit symbol for a three input AND gate.

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

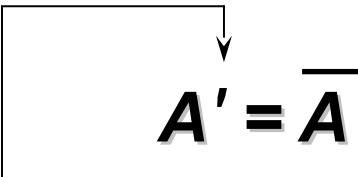


Logic Gates

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


$$X = \overline{A}$$

The overbar represents the **NOT** operation.


$$A' = \overline{A}$$

Another indicator for inversion is the prime symbol ($'$).

“**X** equals **NOT A**”

“**X** equals the *inverse* of **A**”

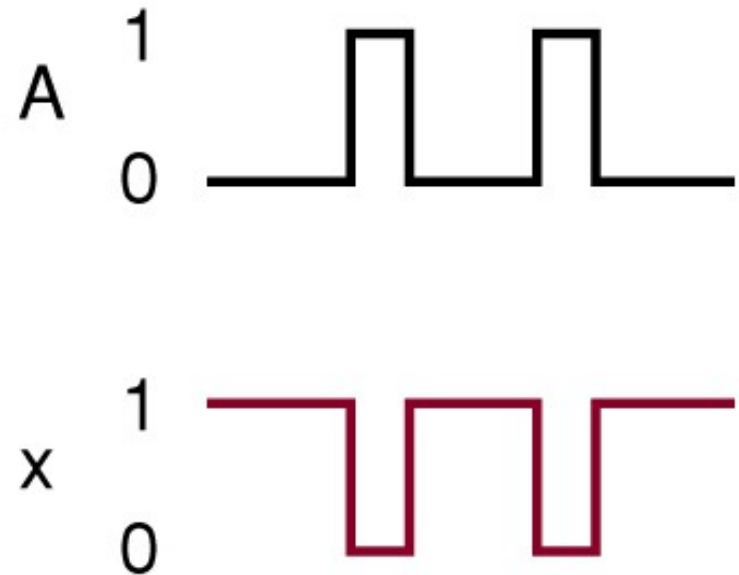
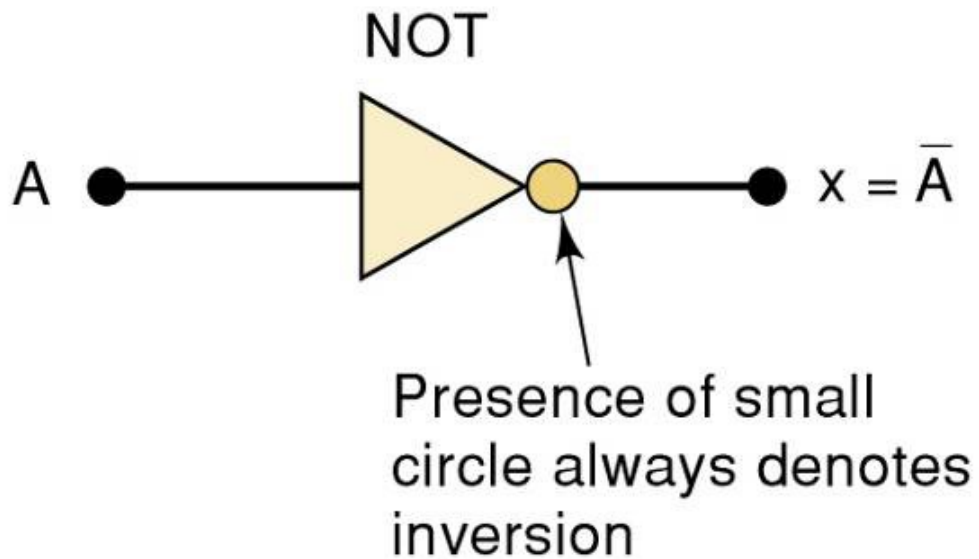
“**X** equals the *complement* of **A**”

NOT

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

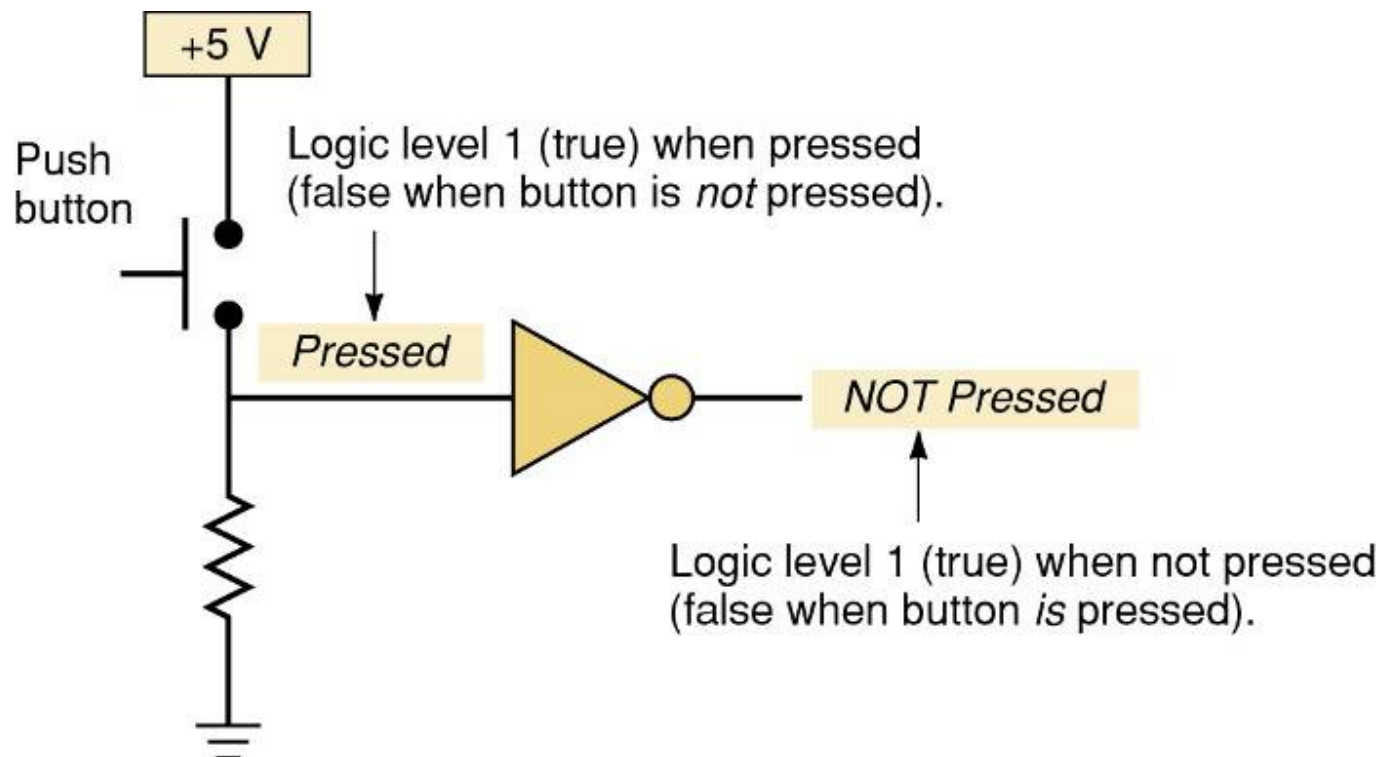
Logic Gates

A NOT circuit—commonly called an INVERTER.



Logic Gates

Typical application of the NOT gate.



Logic Gates

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

OR

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

AND

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

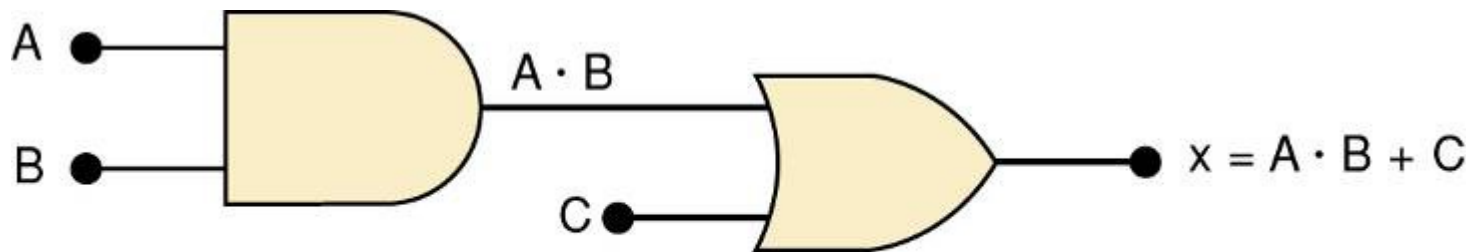
NOT

$$\overline{0} = 1$$

$$\overline{1} = 0$$

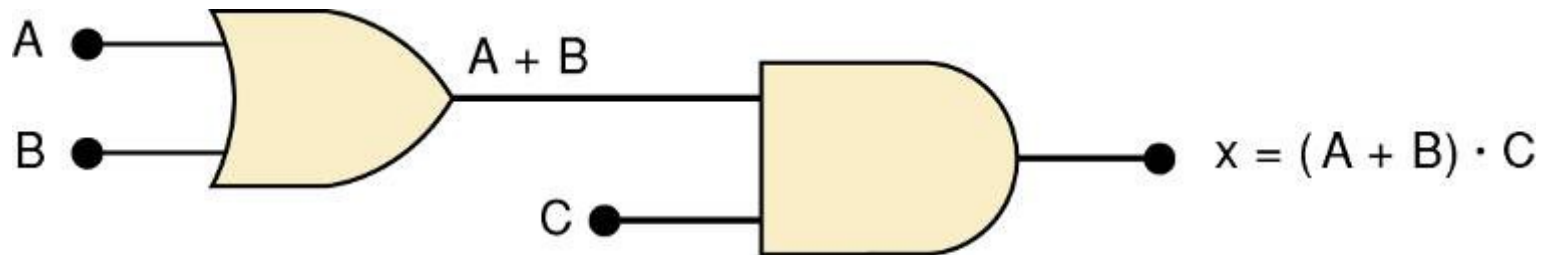
Logic Circuits

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



(a)

Unless there is a parenthesis in the expression.

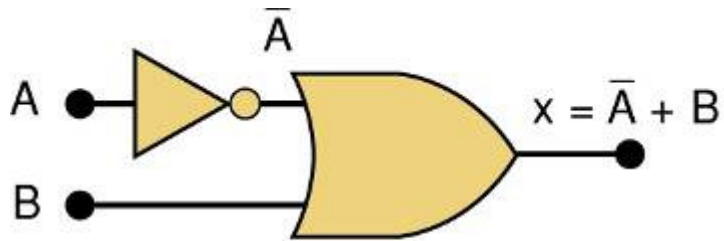


(b)

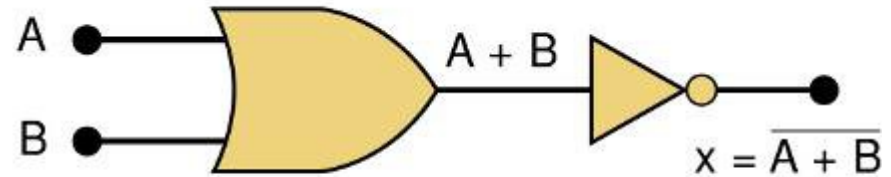
Logic Circuits

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

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



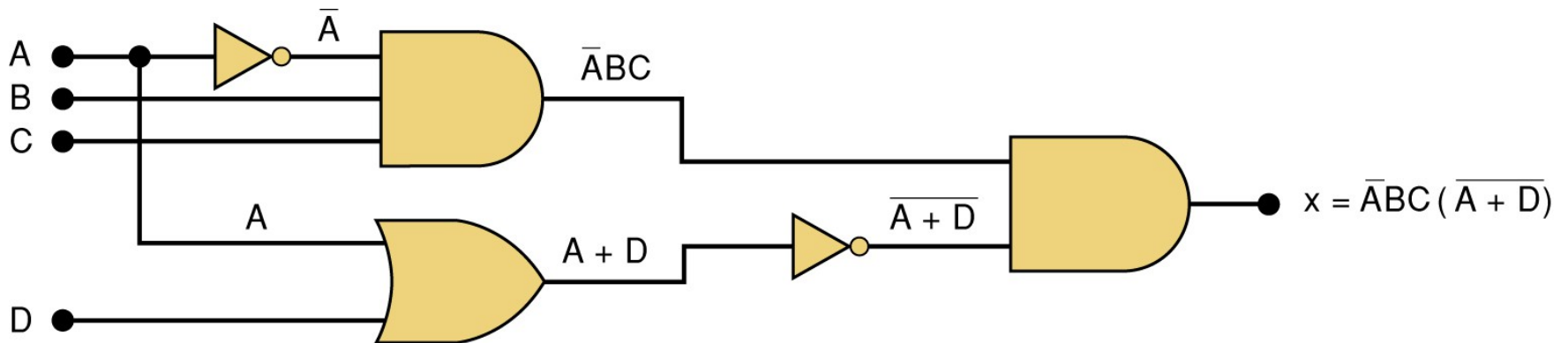
(a)



(b)

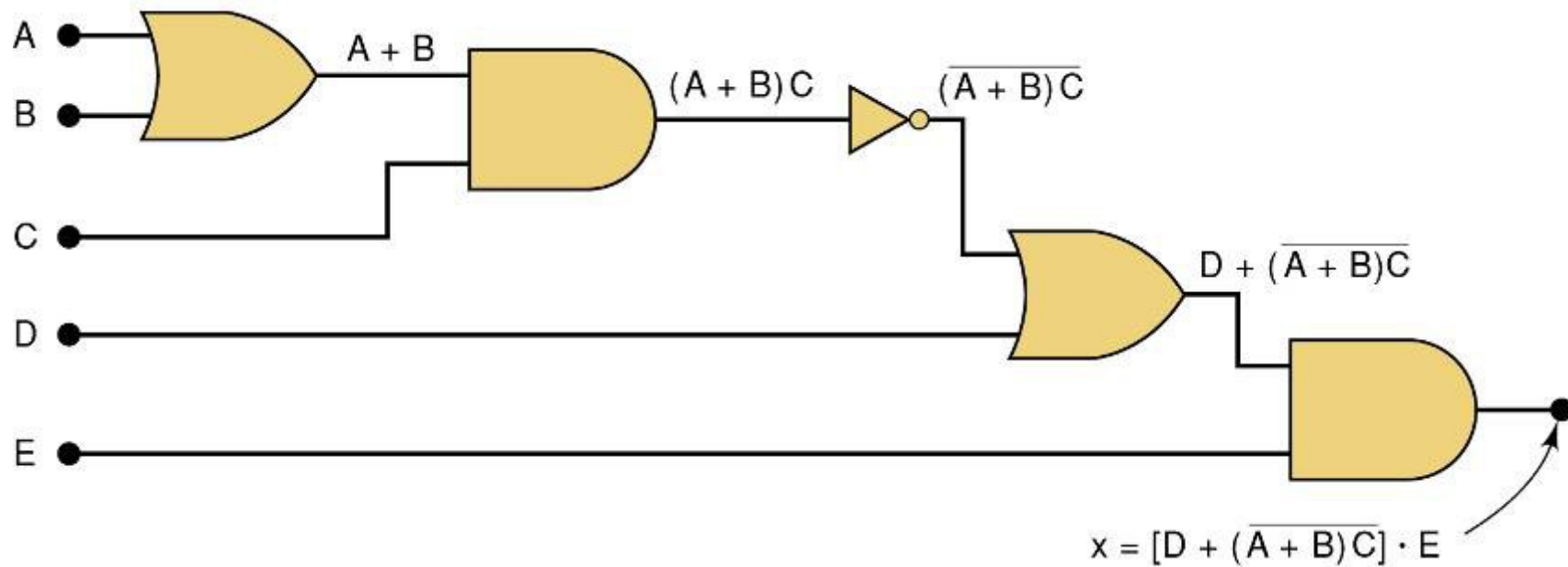
Logic Circuits

Further examples...



Logic Circuits

Further examples...



Logic Circuits

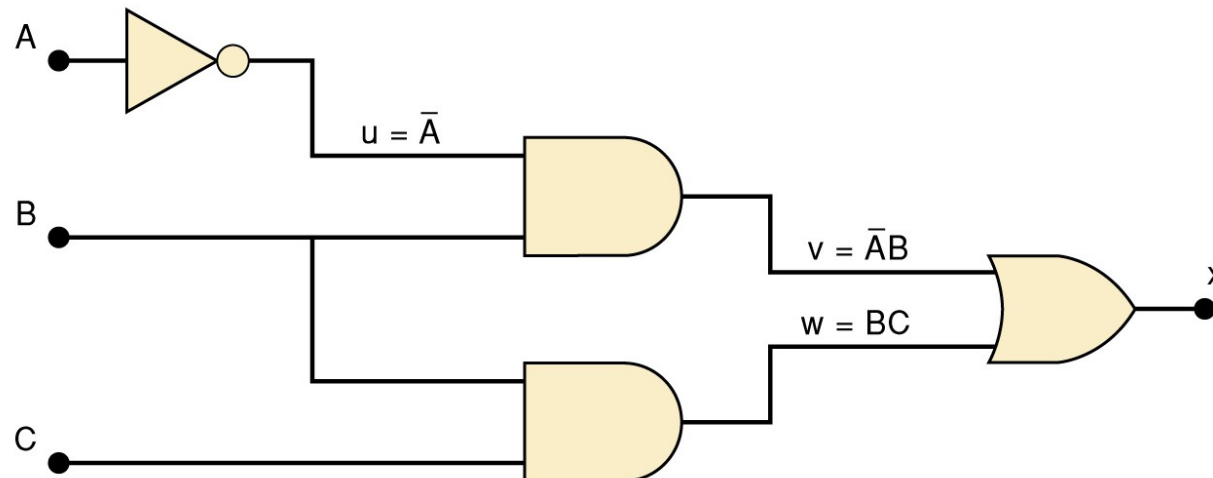
Rules for evaluating a Boolean expression:

- 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.

Logic Circuits

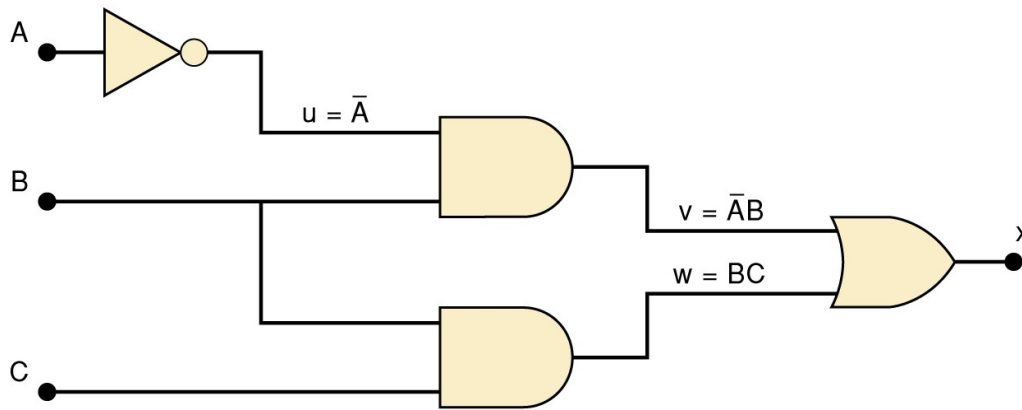
The best way to analyze a circuit made up of multiple logic gates is to use a truth table.

- 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.



Logic Circuits

The first step after listing all input combinations is to create a column in the truth table for each intermediate signal (node).

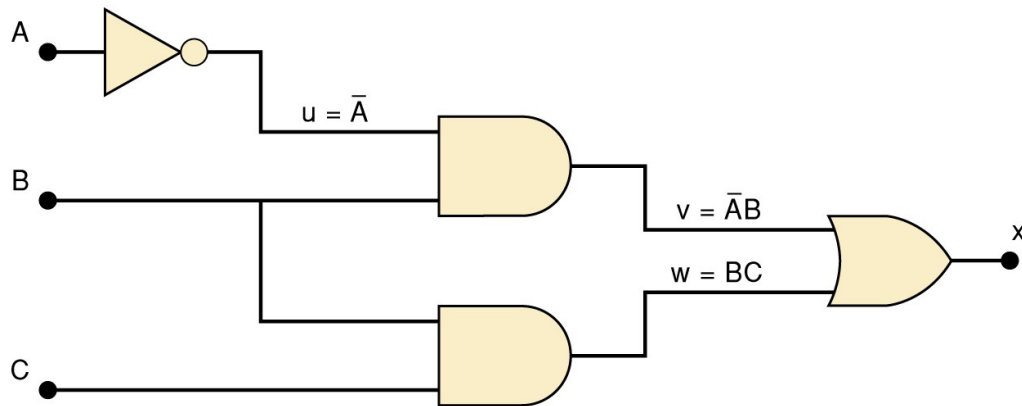


A	B	C	$u = \bar{A}$	$v = \bar{A}B$	$w = BC$	$x = v + w$
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	0			
1	0	1	0			
1	1	0	0			
1	1	1	0			

Logic Circuits

The next step is to fill in the values for column v .

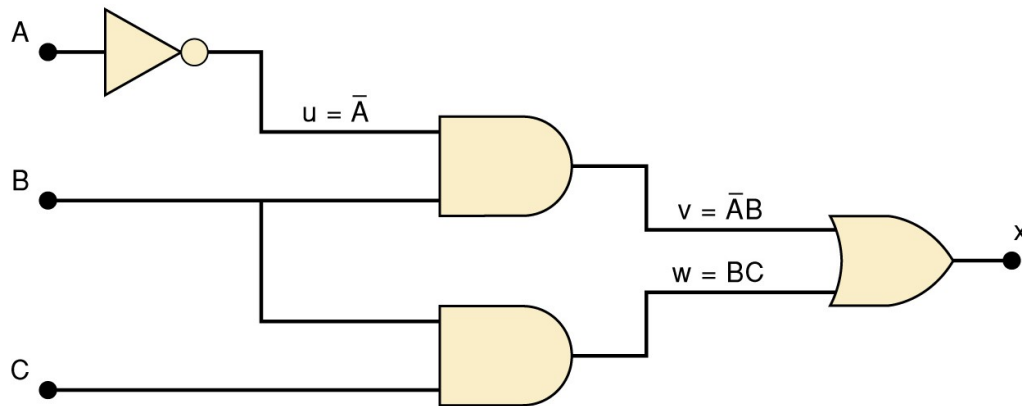
The third step is to predict the values at node w which is the logical product of BC .



A	B	C	$u = \bar{A}$	$v = \bar{A}B$	$w = BC$	$x = v + w$
0	0	0	1	0	0	
0	0	1	1	0	0	
0	1	0	1	1	0	
0	1	1	1	1	1	
1	0	0	0	0	0	
1	0	1	0	0	0	
1	1	0	0	0	0	
1	1	1	0	0	1	

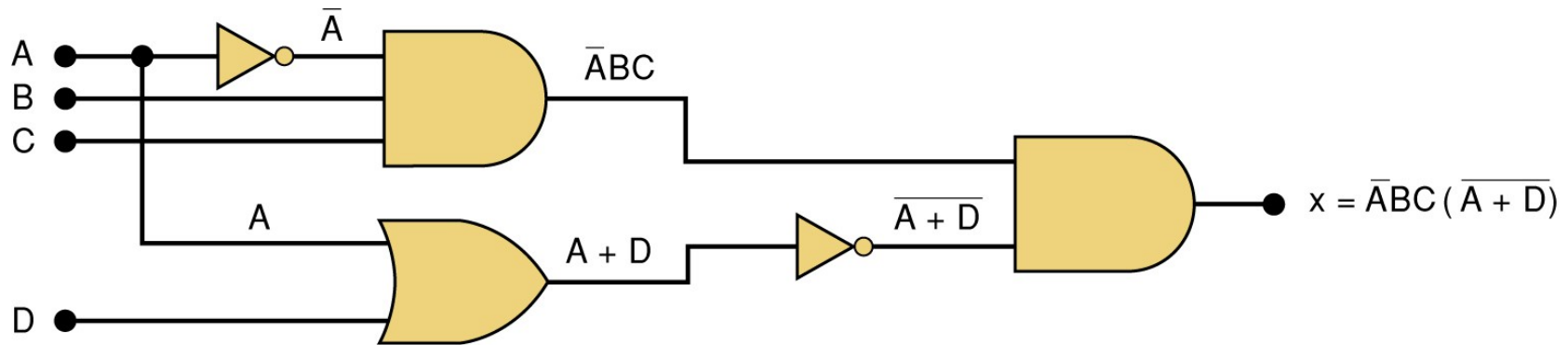
Logic Circuits

The final step is to logically combine columns v and w to predict the output x .



A	B	C	$\underline{u} = \bar{A}$	$\underline{v} = \bar{A}B$	$\underline{w} = BC$	$\underline{x} = v + w$
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

Logic Circuits



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

A	B	C	D	$t = \bar{A}BC$	$u = A + D$	$v = \overline{A + D}$	$x = tv$
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

Logic Circuits

It is important to be able to draw a logic circuit from a Boolean expression.

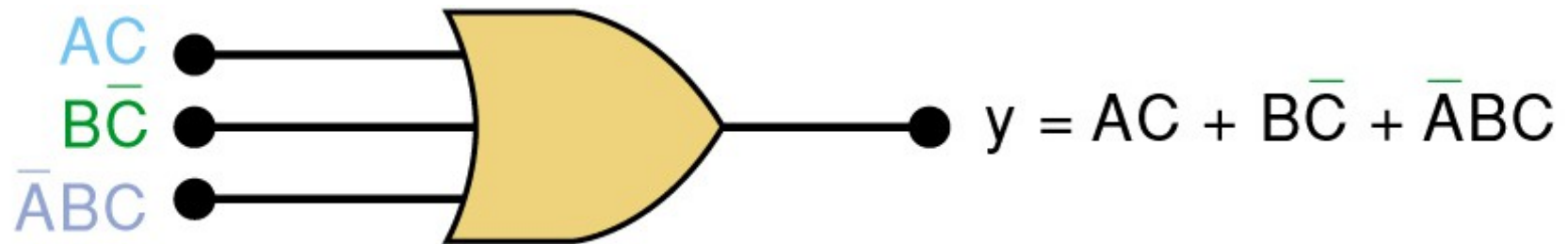
The expression $X = A \cdot B \cdot C$, could be drawn as a three input **AND** gate.

A circuit defined by $X = A + B$, would use a two-input **OR** gate with an INVERTER on one of the inputs.

Logic Circuits

A circuit with output $y = AC + BC + ABC$ contains three terms which are **OR**ed together.

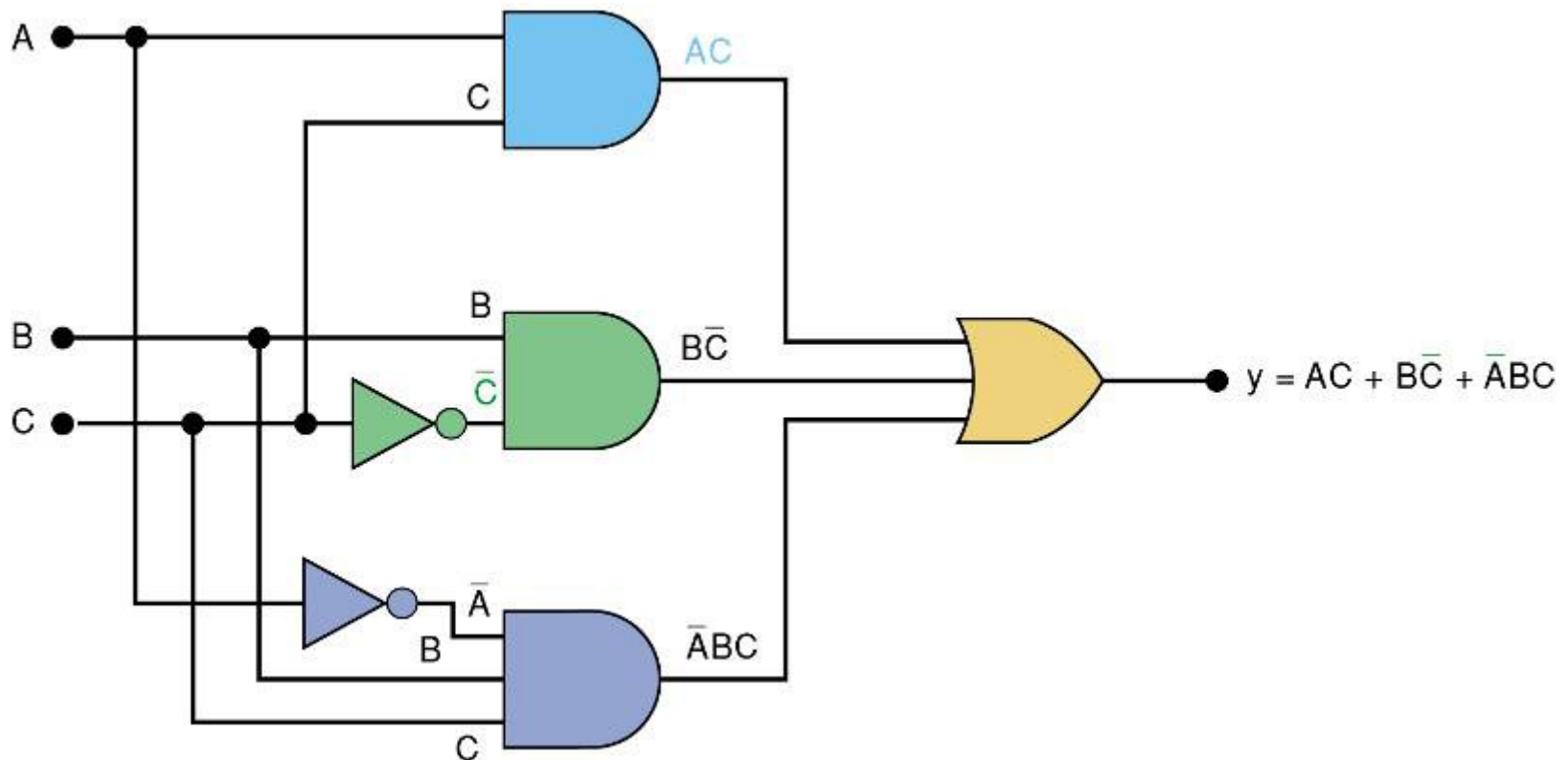
It requires a three-input **OR** gate.



Logic Circuits

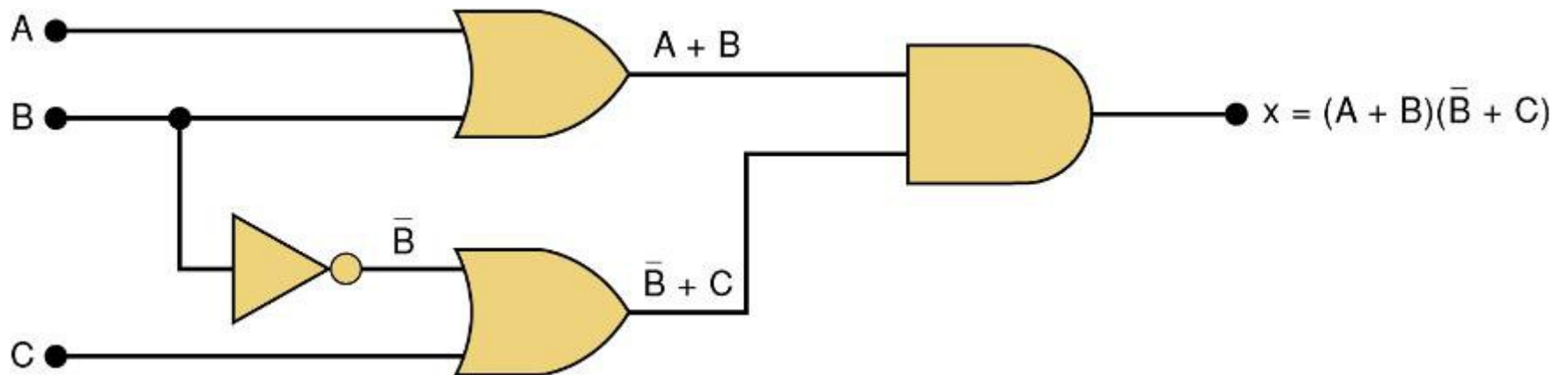
Each **OR** gate input is an **AND** product term.

An **AND** gate with appropriate inputs can be used to generate each of these terms.



Logic Circuits

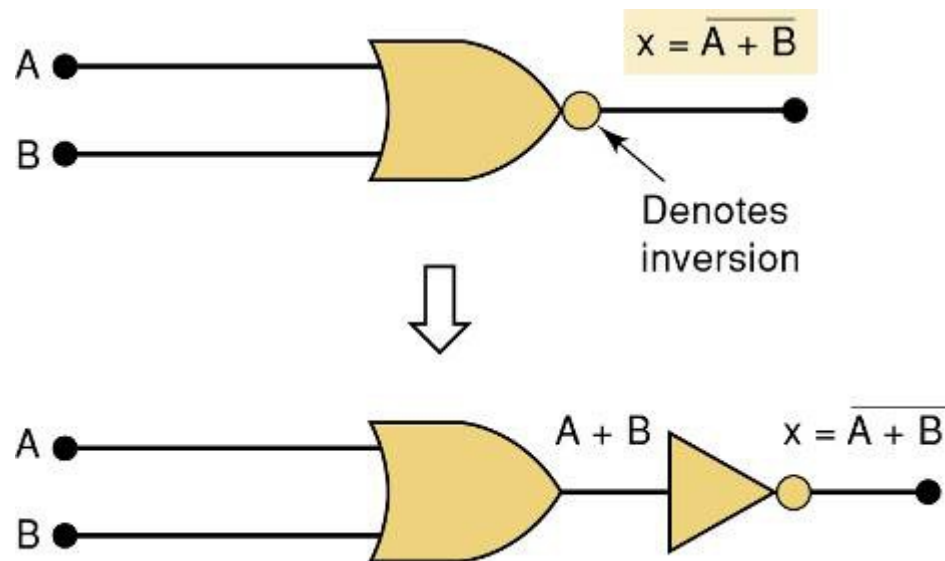
Circuit diagram to implement $x = (A + B)(\bar{B} + C)$



Logic Circuits

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

An inversion “bubble” is placed at the output of the **OR** gate, making the Boolean output expression $x = \overline{A + B}$

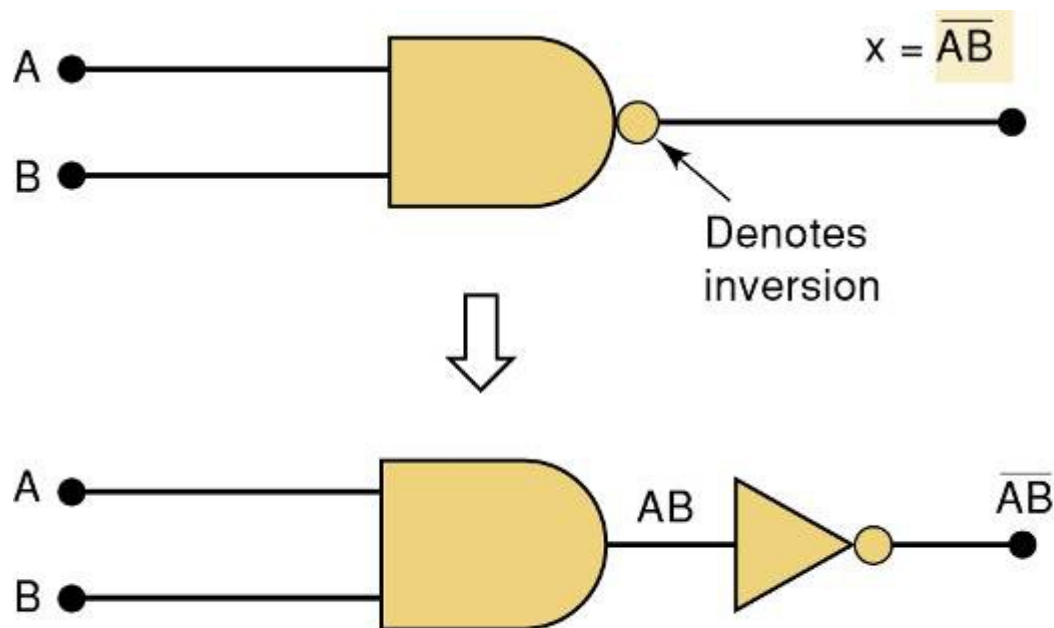


		OR		NOR	
A	B	$A + B$		$\overline{A + B}$	
0	0	0		1	
0	1	1		0	
1	0	1		0	
1	1	1		0	

Logic Circuits

The **NAND** gate is an inverted **AND** gate.

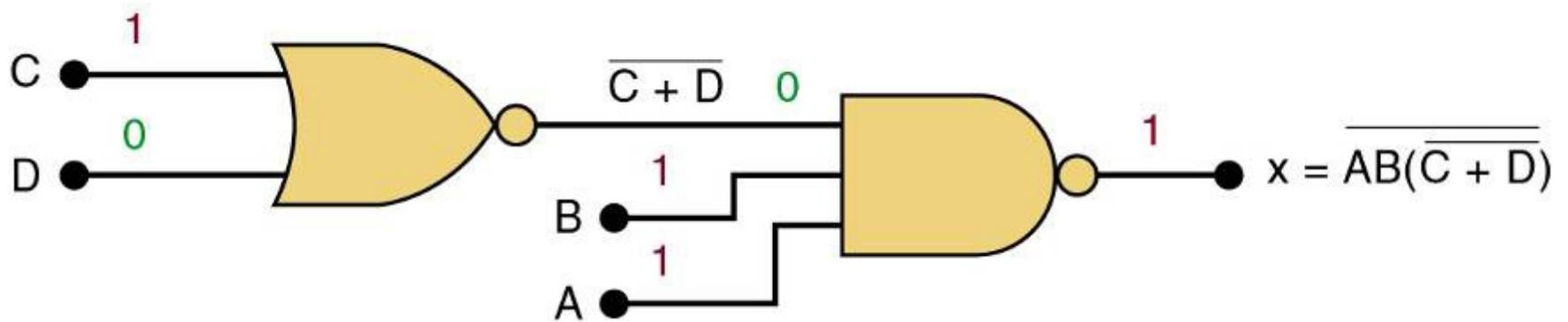
An inversion “bubble” is placed at the output of the **AND** gate, making the Boolean output expression $x = \overline{AB}$



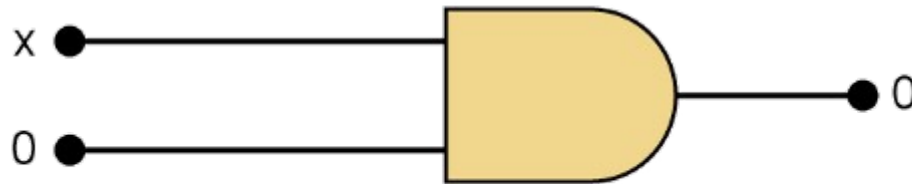
		AND		NAND	
A	B	AB		\overline{AB}	
0	0	0		1	
0	1	0		1	
1	0	0		1	
1	1	1		0	

Logic Circuits

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



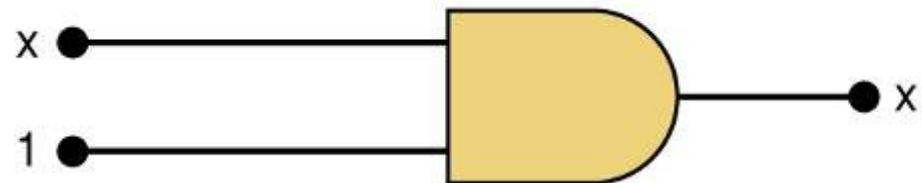
Boolean Theorems



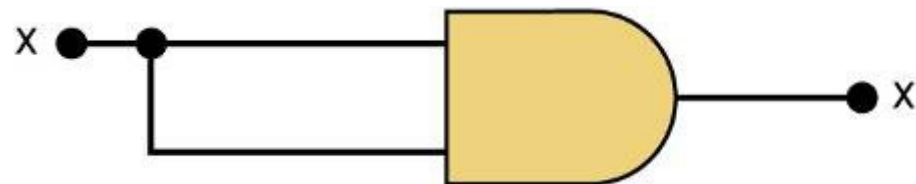
(1) $x \cdot 0 = 0$

Theorem (1) states that if any variable is **ANDed** with 0, the result must be 0.

Theorem (2) is also obvious by comparison with ordinary multiplication.



(2) $x \cdot 1 = x$



(3) $x \cdot x = x$

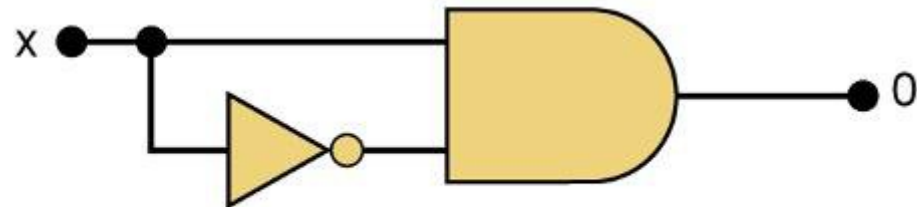
Prove Theorem (3) by trying each case.

If $x = 0$, then $0 \cdot 0 = 0$

If $x = 1$, then $1 \cdot 1 = 1$

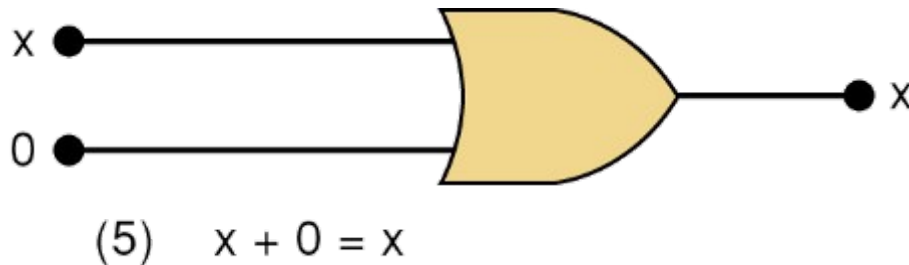
Thus, $x \cdot x = x$

Theorem (4) can be proved in the same manner.



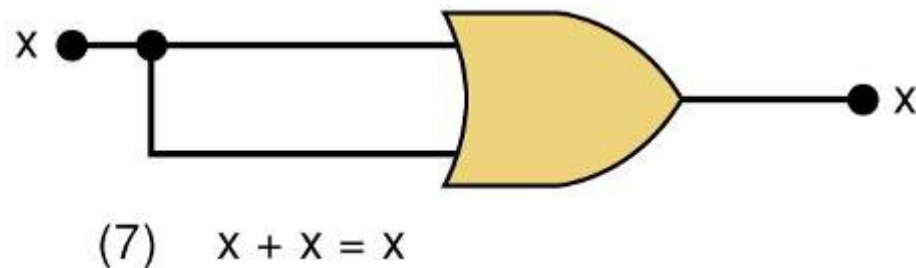
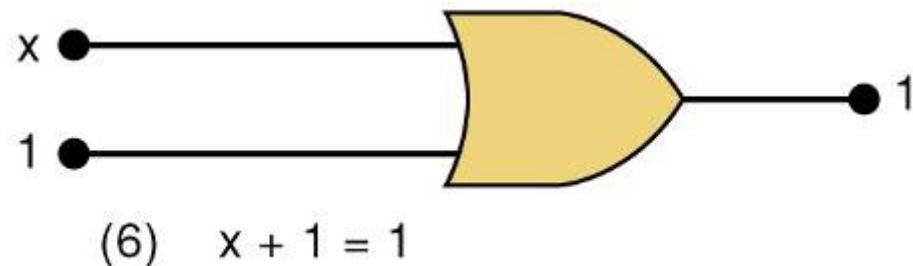
(4) $x \cdot \bar{x} = 0$

Boolean Theorems



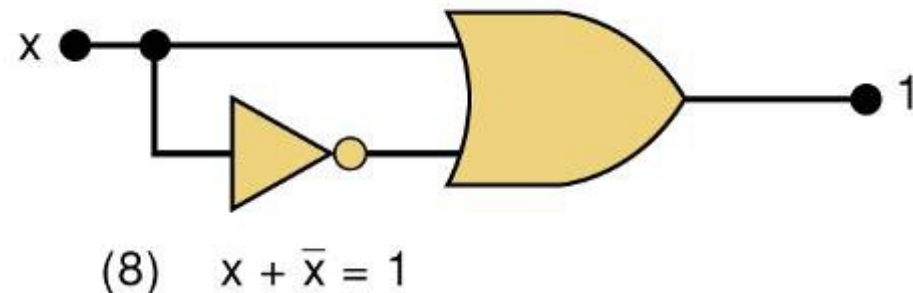
Theorem (5) is straightforward, as 0 *added* to anything does not affect value, either in regular addition or in OR addition.

Theorem (6) states that if any variable is ORed with 1, the is always 1.
Check values: $0 + 1 = 1$ and $1 + 1 = 1$.



Theorem (7) can be proved by checking for both values of x:
 $0 + 0 = 0$ and $1 + 1 = 1$.

Theorem (8) can be proved similarly.



Boolean Theorems

Multivariable Theorems

Commutative laws

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

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

Associative laws

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

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

Distributive law

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

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

Boolean Theorems

Multivariable Theorems

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

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

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

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

$$\begin{aligned} x + xy &= x(1 + y) \\ &= x \cdot 1 \\ &= x \end{aligned}$$

[using theorem (6)]
[using theorem (2)]

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

Boolean Theorems

DeMorgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted.

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

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

Theorem (16) says inverting the OR sum of two variables is the same as inverting each variable individually, then ANDing the inverted variables.

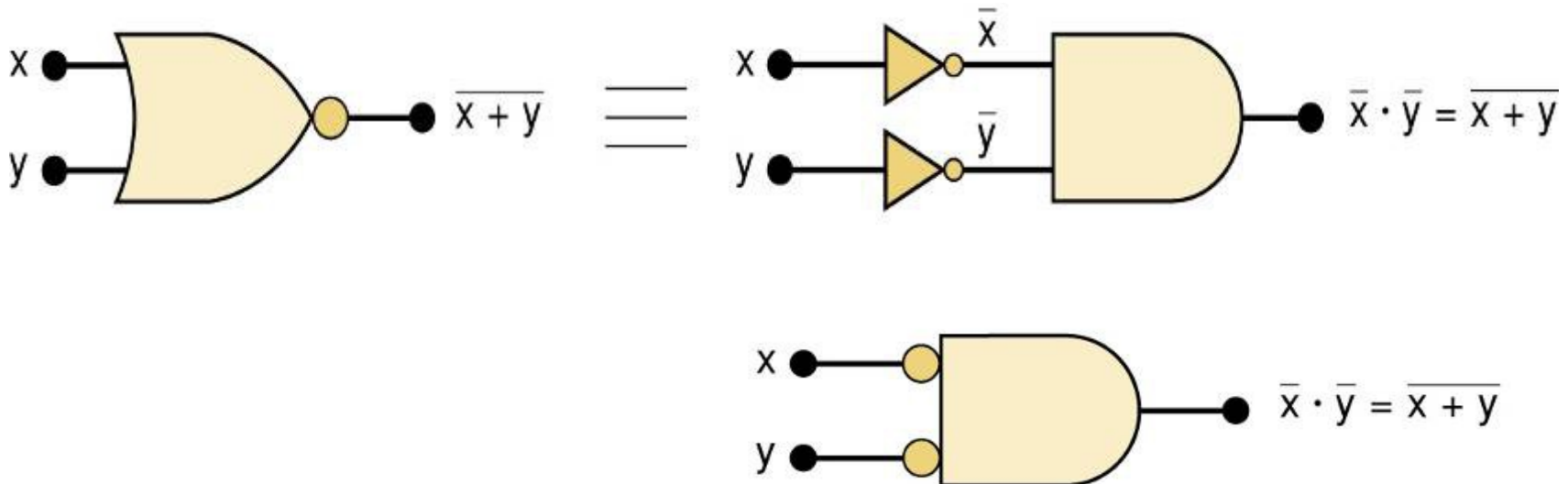
$$(17) \quad \overline{(x \cdot y)} = \bar{x} + \bar{y}$$

Theorem (17) says inverting the AND product of two variables is the same as inverting each variable individually and then ORing them.

Boolean Theorems

Equivalent circuits implied by Theorem (16)

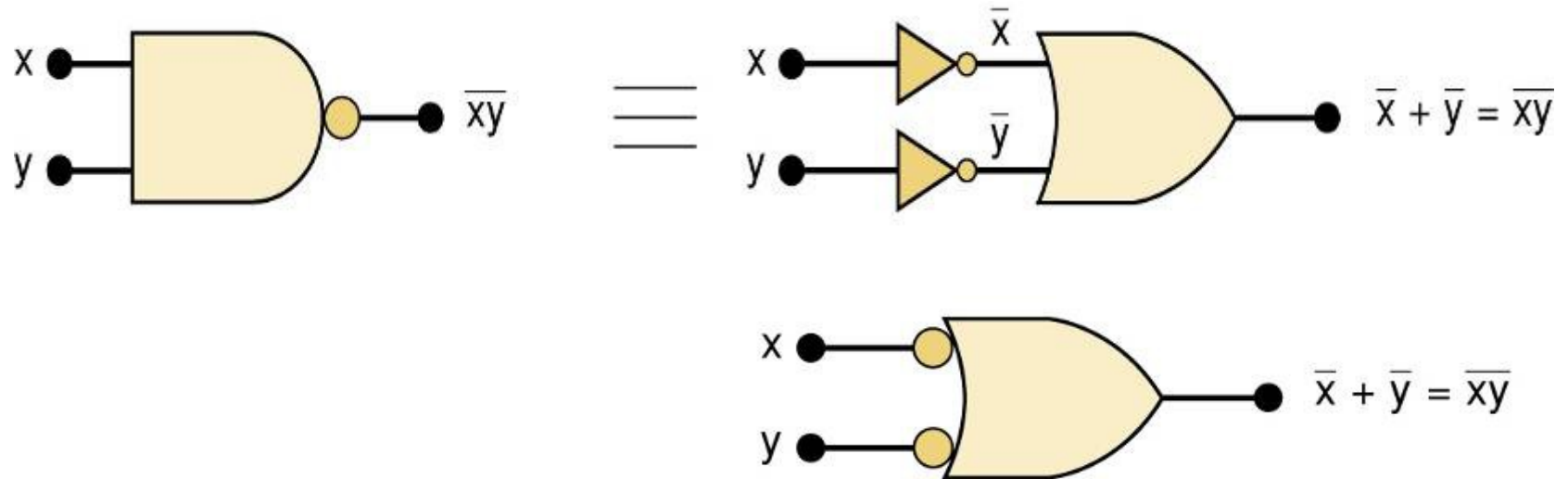
$$(16) \quad \overline{(x + y)} = \bar{x} \cdot \bar{y}$$



Boolean Theorems

Equivalent circuits implied by Theorem (17)

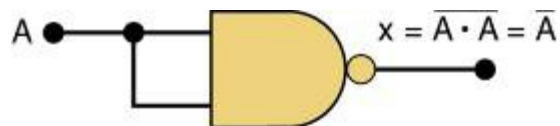
$$(17) \quad \overline{(x \cdot y)} = \bar{x} + \bar{y}$$



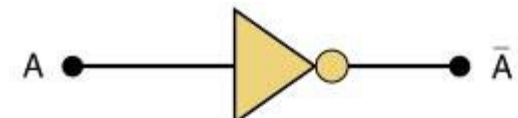
Universality of NAND and NOR Gates

NAND or NOR gates can be used to create the three basic logic expressions: OR, AND, and INVERT.

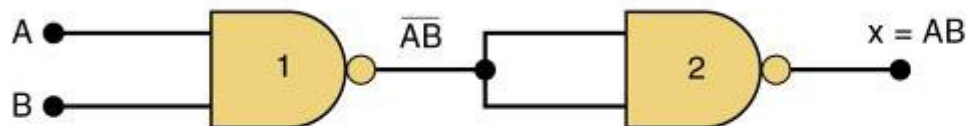
It is possible, however, to implement any logic expression using *only* NAND gates



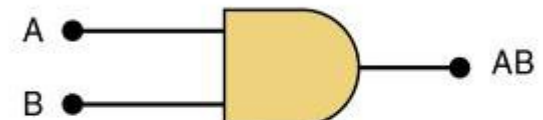
(a)



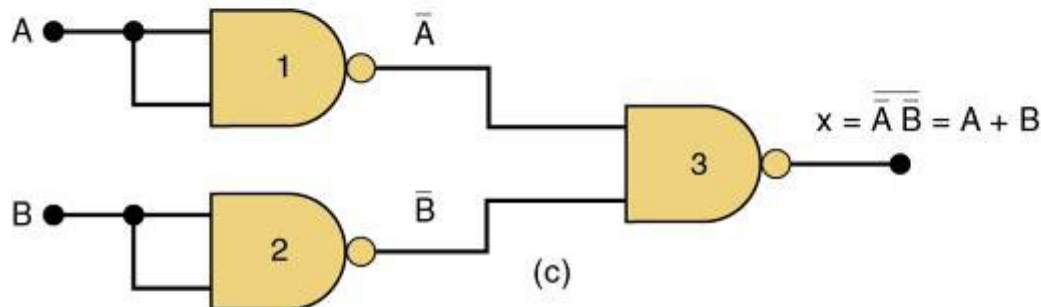
INVERTER



(b)



AND



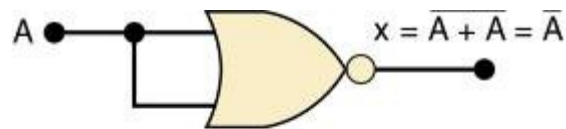
(c)



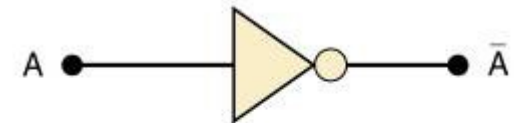
OR

Universality of NAND and NOR Gates

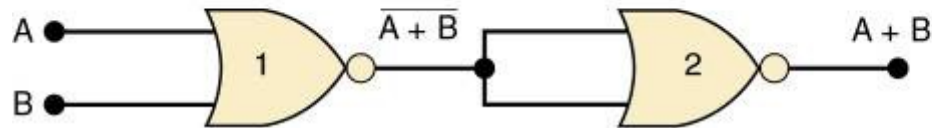
NOR gates can be arranged to implement any of the Boolean operations, as shown.



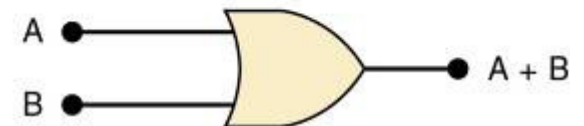
(a)



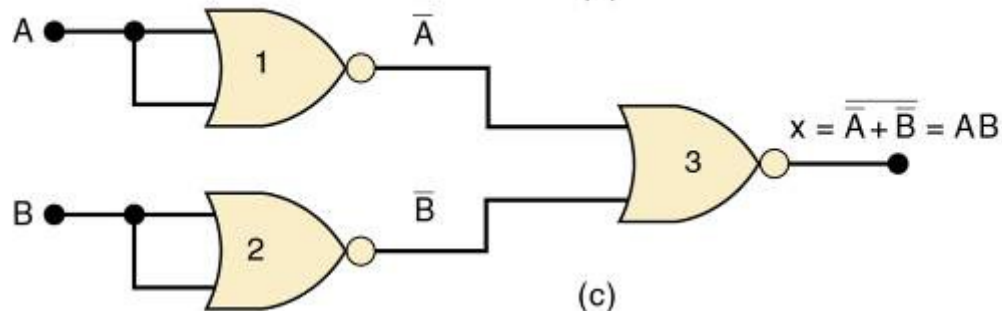
INVERTER



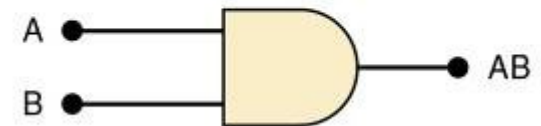
(b)



OR



(c)



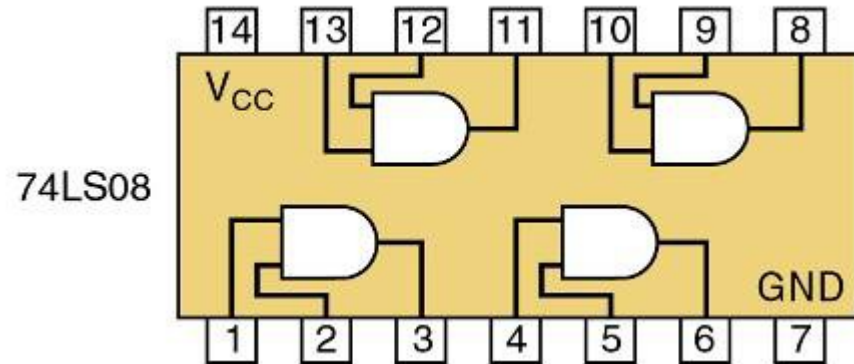
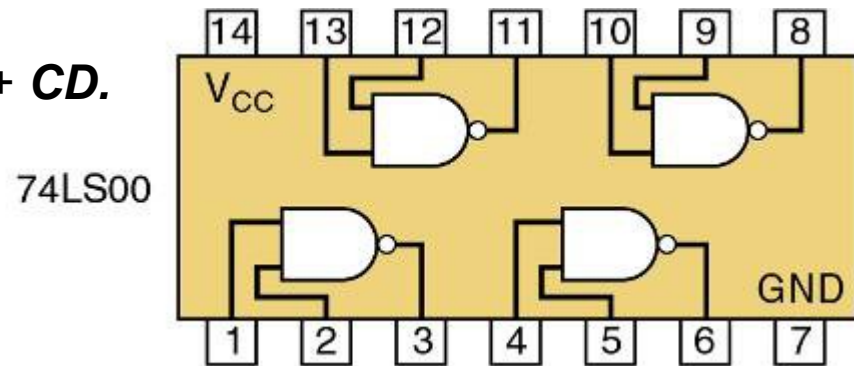
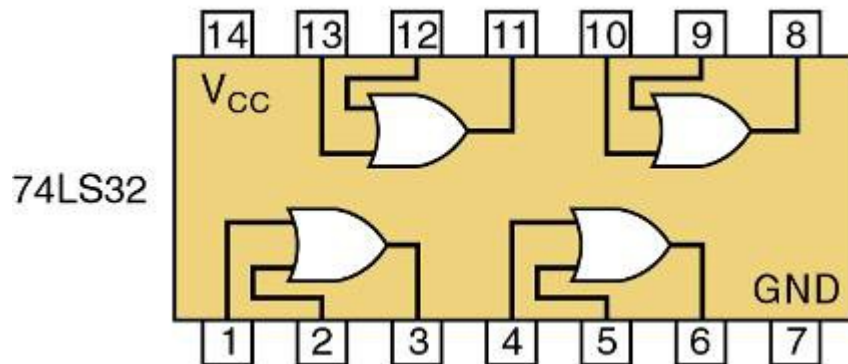
AND

Universality of NAND and NOR Gates

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.

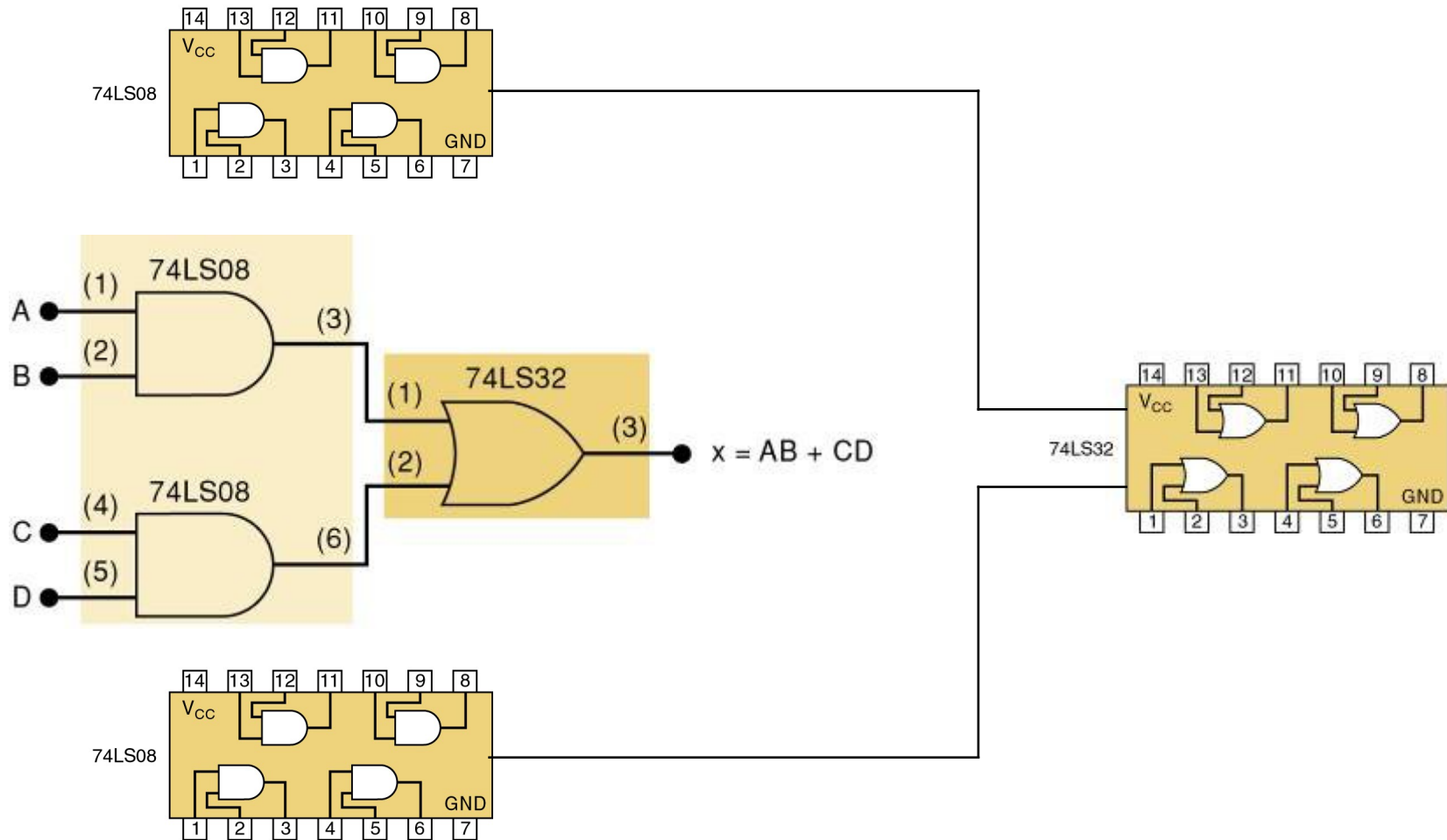
The logic expression will be $x = AB + CD$.

Each of the TTL ICs shown here will fulfill the function. Each IC is a *quad*, with *four* identical gates on one chip



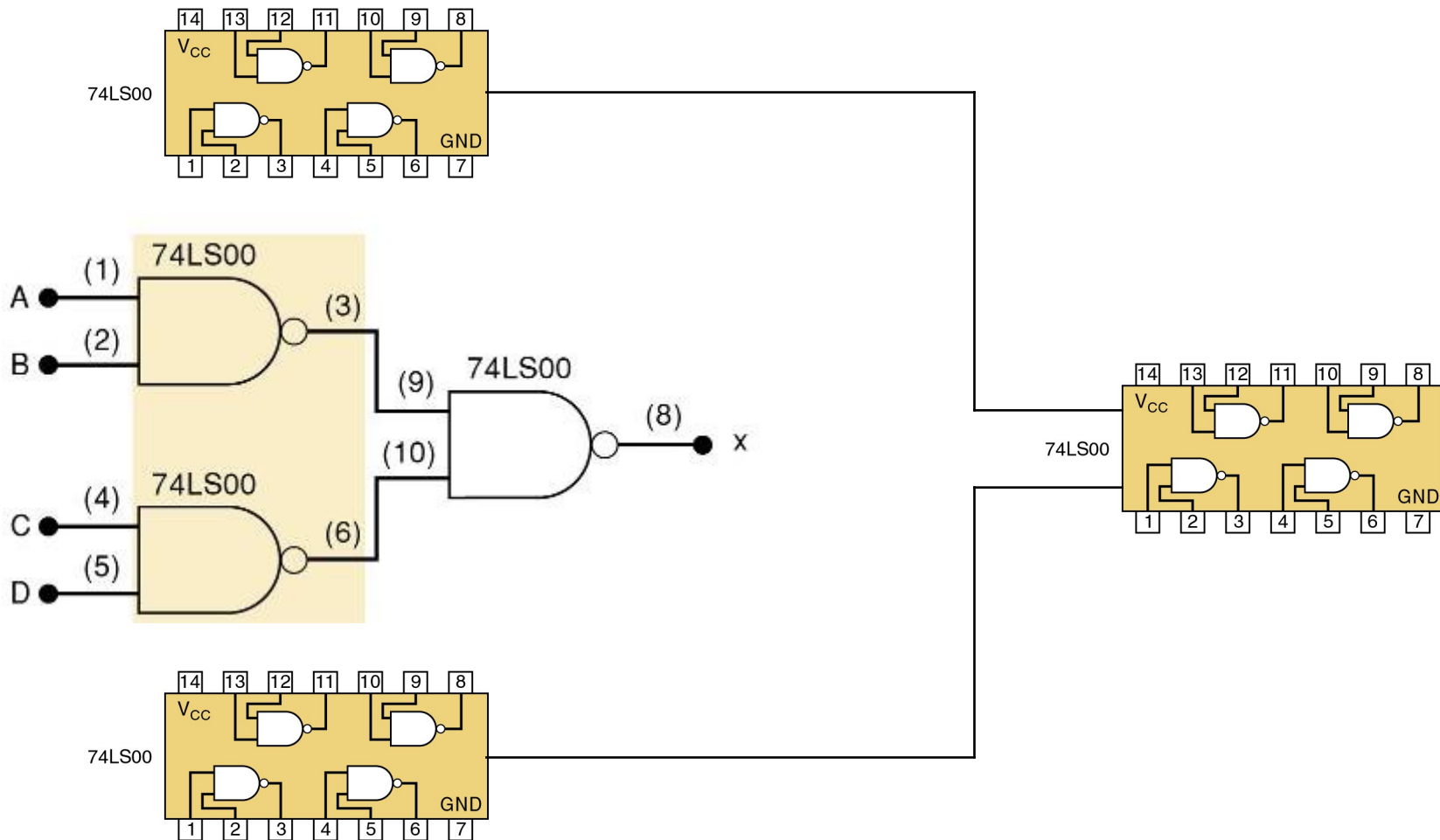
Universality of NAND and NOR Gates

Possible Implementations # 1



Universality of NAND and NOR Gates

Possible Implementations #2



Alternate Logic-Gate Representations

To convert a standard symbol to an alternate:

- Invert each input and output in standard symbols.
- Add an inversion bubble where there are none.
- Remove bubbles where they exist.



Alternate Logic-Gate Representations

Points regarding logic symbol equivalences:

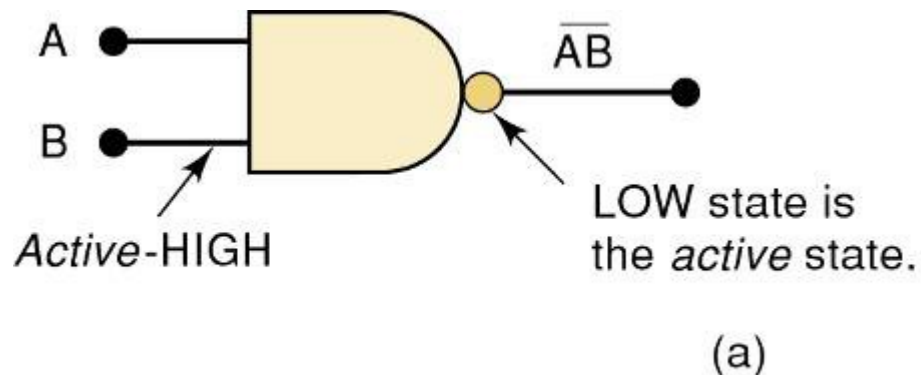
- The equivalences can be extended to gates with *any* number of inputs.
- None of the standard symbols have bubbles on their inputs, and all the alternate symbols do.
- Standard & alternate symbols for each gate represent the same physical circuit.
- **NAND** and **NOR** gates are inverting gates: Both the standard and the alternate symbols for each will have a bubble on *either* the input or the output.
- **AND** and **OR** gates are *noninverting* gates: The alternate symbols for each will have bubbles on *both* inputs and output.

Alternate Logic-Gate Representations

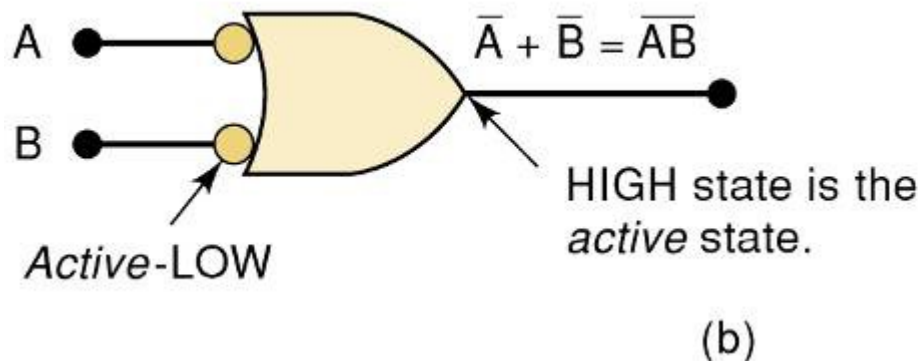
Active-HIGH – an input/output has *no* inversion bubble.

Active-LOW – an input or output has an inversion bubble.

Interpretation of the two **NAND** gate symbols.



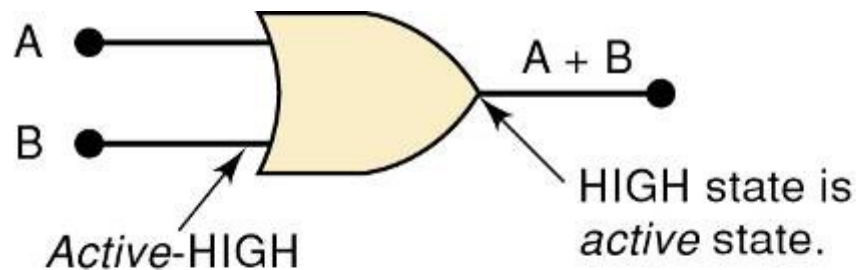
Output goes LOW only when *all* inputs are HIGH.



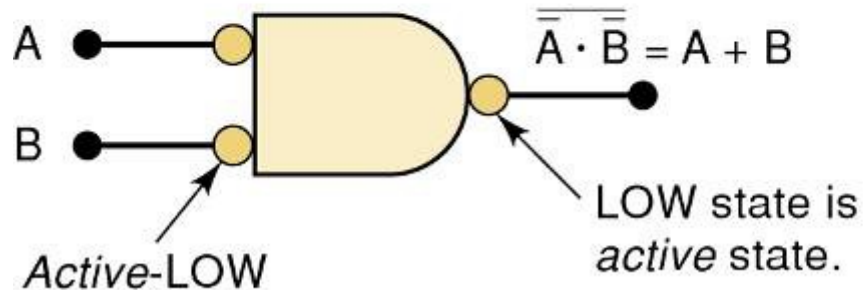
Output is HIGH when *any* input is LOW.

Alternate Logic-Gate Representations

Interpretation of the two **OR** gate symbols.



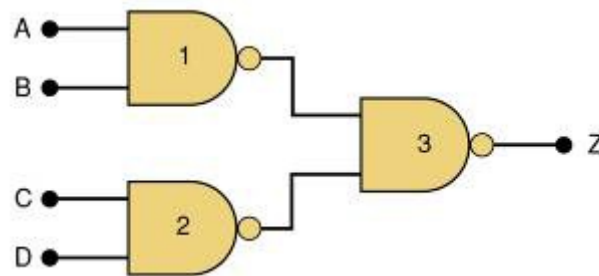
Output goes HIGH when *any* input is HIGH.



Output goes LOW only when *all* inputs are LOW.

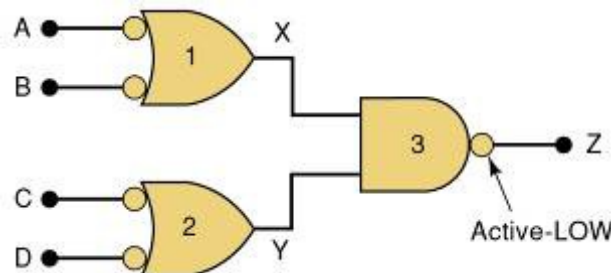
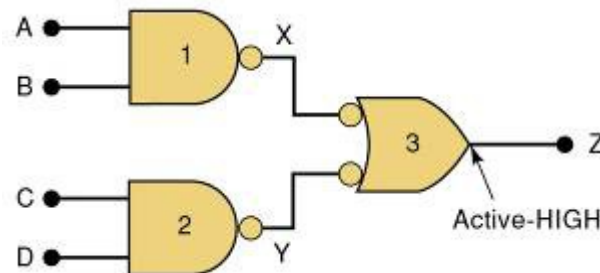
Alternate Logic-Gate Representations

Proper use of alternate gate symbols in the circuit diagram can make circuit operation much clearer.



Original circuit using standard **NAND** symbols.

Equivalent representation where output Z is active-HIGH.



Equivalent representation where output Z is active-LOW.

Alternate Logic-Gate Representations

When a logic signal is in the *active* state (HIGH or LOW) it is said to be *asserted*.

When a logic signal is in the *inactive* state (HIGH or LOW) it is said to be *unasserted*.

A bar over a signal
means asserted
(active) LOW.

\overline{RD}

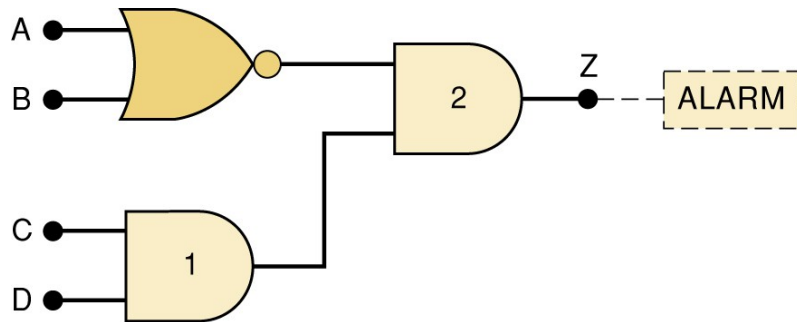
Absence of a bar
means asserted
(active) HIGH

RD

Alternate Logic-Gate Representations

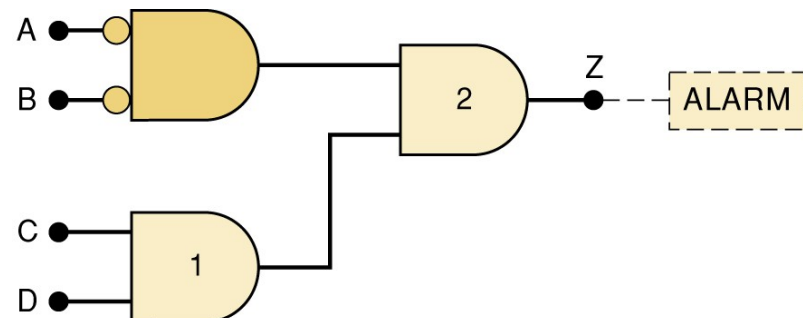
When possible, choose gate symbols so bubble outputs are connected to bubble input. Nonbubble outputs connected to nonbubble inputs.

The logic circuit shown activates an alarm when output Z goes HIGH.



Modify the circuit diagram so it represents the circuit operation more effectively.

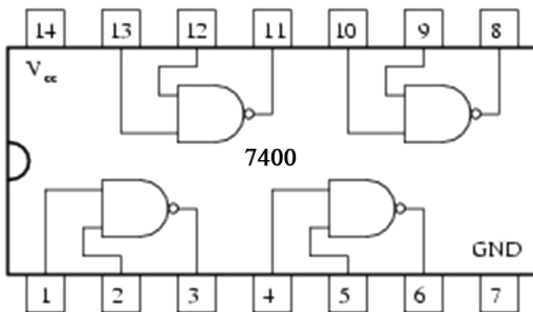
The NOR gate symbol should be changed to the alternate symbol with a nonbubble (active-HIGH) output to match the nonbubble input of AND gate 2.



Physical Characteristics

74LS00

SN74LS00



DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage			0.8	V	Guaranteed Input LOW Voltage for All Inputs
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V	$V_{CC} = \text{MIN}$, $I_{IN} = -18 \text{ mA}$
V_{OH}	Output HIGH Voltage	2.7	3.5		V	$V_{CC} = \text{MIN}$, $I_{OH} = \text{MAX}$, $V_{IN} = V_{IH}$ or V_{IL} per Truth Table
V_{OL}	Output LOW Voltage		0.25	0.4	V	$I_{OL} = 4.0 \text{ mA}$
			0.35	0.5	V	$I_{OL} = 8.0 \text{ mA}$
I_{IH}	Input HIGH Current			20	μA	$V_{CC} = \text{MAX}$, $V_{IN} = 2.7 \text{ V}$
				0.1	mA	$V_{CC} = \text{MAX}$, $V_{IN} = 7.0 \text{ V}$
I_{IL}	Input LOW Current			-0.4	mA	$V_{CC} = \text{MAX}$, $V_{IN} = 0.4 \text{ V}$
I_{OS}	Short Circuit Current (Note 1)	-20		-100	mA	$V_{CC} = \text{MAX}$
I_{CC}	Power Supply Current Total, Output HIGH			1.6	mA	$V_{CC} = \text{MAX}$
	Total, Output LOW			4.4		

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS ($T_A = 25^\circ\text{C}$)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
t_{PLH}	Turn-Off Delay, Input to Output		9.0	15	ns	$V_{CC} = 5.0 \text{ V}$ $C_L = 15 \text{ pF}$
t_{PHL}	Turn-On Delay, Input to Output		10	15	ns	

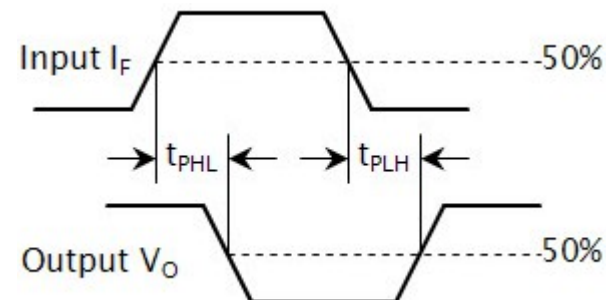
Physical Characteristics

Propagation Delay

Due to limitations in transistor switching speeds caused by undesirable internal capacitive stored charges

A delay time for an input wave (at 50% point of input) to propagate through an IC to the output (at 50% point of output)

- Time propagation from HIGH to LOW
- Time propagation from LOW to HIGH



Physical Characteristics

Noise Margin

Noise is all undesirable voltage variations that are superimposed on normal operating voltage levels

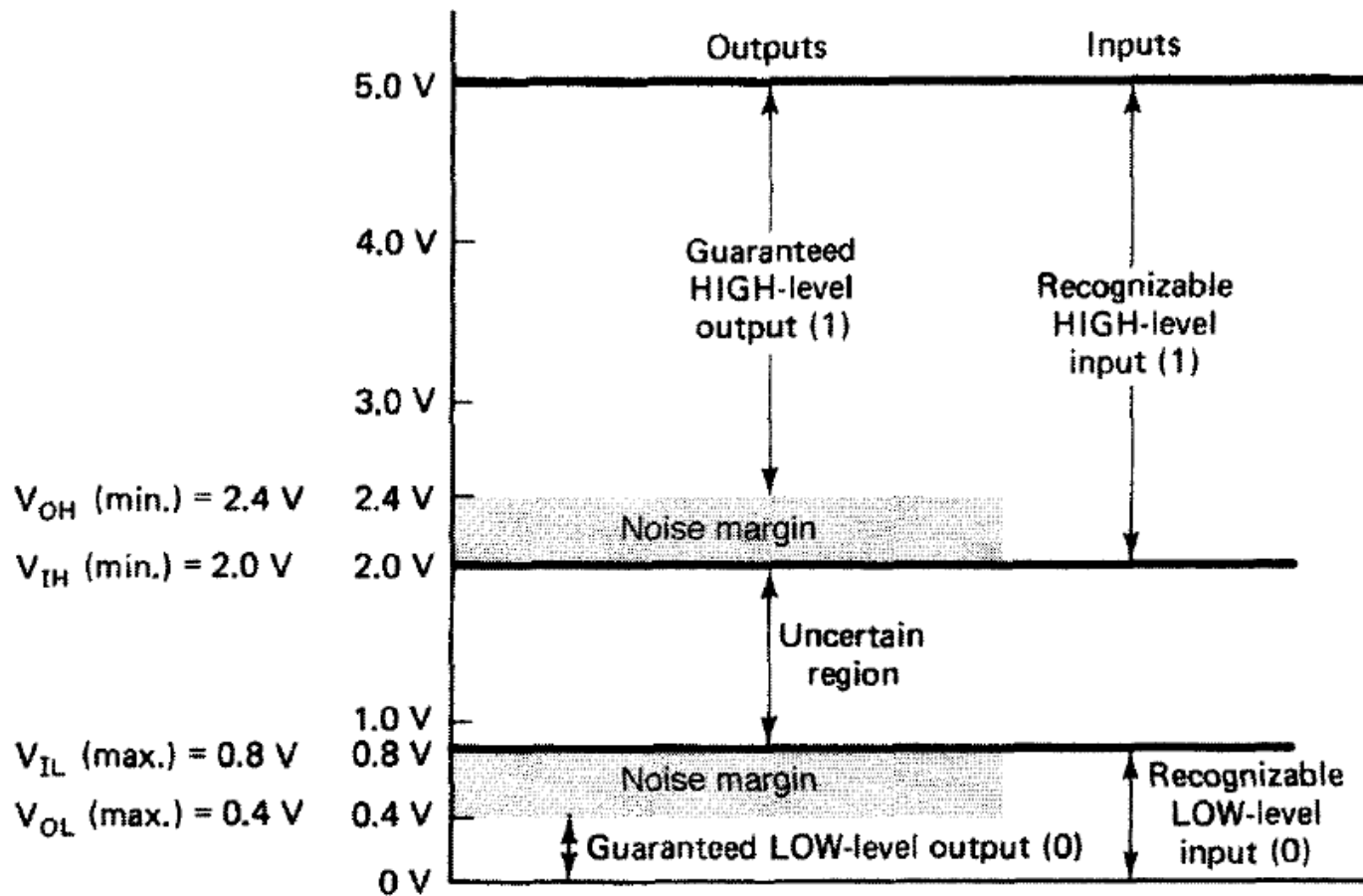
Noise Margin is the maximum noise voltage level that a logic signal can tolerate without errors

- Noise Margin for LOW output = $V_{ILmax} - V_{OLmax}$
- Noise Margin for HIGH output = $V_{OHmin} - V_{OLmin}$

Parameter	Minimum	Typical	Maximum
V_{OL}		0.2 V	0.4 V
V_{IL}			0.8 V
V_{OH}	2.4 V	3.4 V	
V_{IH}	2.0 V		

Physical Characteristics

Noise Margin



Physical Characteristics

Fan-in: Number of inputs available on a gate

Fan-out is a standard load (i.e. the number of inputs) that each output can drive

$$\text{Fan-out} = \min \left\{ \frac{I_{OH}}{I_{IH}}, \frac{I_{OL}}{I_{IL}} \right\}$$

Example: $I_{OH} = 600\mu A, I_{IH} = 40\mu A, I_{OL} = 16\mu A, I_{IL} = 1.6\mu A \Rightarrow \min \left\{ \frac{600}{40}, \frac{16}{1.6} \right\} = 10$

I_{OH} = Output Current of a gate when its output is HIGH

I_{OL} = Output Current of a gate when its output is LOW

I_{IH} = Input Current of a gate when its output is HIGH

I_{IL} = Input Current of a gate when its output is LOW