Santa Clara University

COEN 210

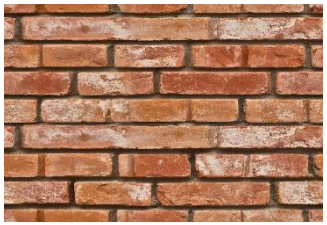
Chapter 3, Part 1 – Logic Gates

Yan Cui

ycui4@scu.edu

Transistors and Logic Gates

- Transistors are the physical foundation of computer chips
- Logic gates are the building bricks



- **Goal:**

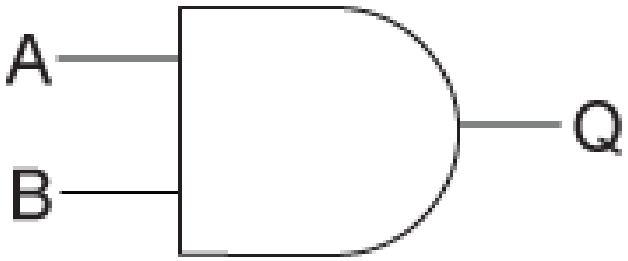
To understand how digital a computer can work, at the lowest level.

Basic Logic Gates

Logic Gates

- Input and Output signals are binary.
  - binary: always in one of two possible states;
  - typically treated as:
    - On / Off (electrically)
    - 1 / 0
    - True / False
- There is a delay between when a change happens at a logic gates inputs and when the output changes, called gate switching time.
- The True or False view is most useful for thinking about the meaning of the basic logic gates.

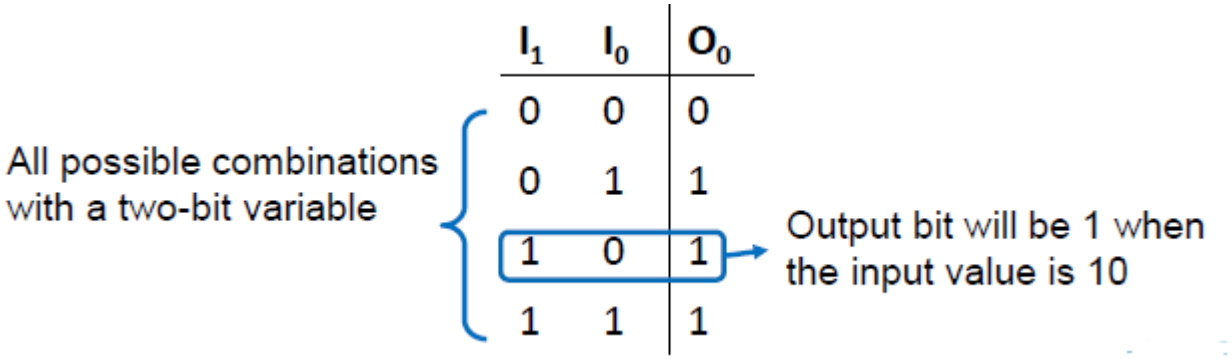
Gate – An Electric Perspective



- What does it mean for a hardware device to represent a Boolean function (or truth table), say AND gate?
  - Place on the two input lines voltages representing logical values (T or F).
  - After a short delay, the output line will stabilize to a voltage representing the logical result of the inputs.

Truth Table - Recap

- Used to describe behavior of a digital circuit
- Show all possible combinations of inputs as a separate row in the table
  - N inputs → 2^N combinations or rows
- For each row, show the output value
- Example : A 2-bit input and 1-bit output truth table

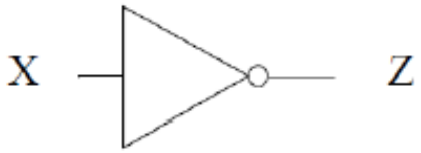


I_1	I_0	O_0
0	0	0
0	1	1
1	0	1
1	1	1

Three Logic Operations

- OR
  - A + B
  - 1 if either of the inputs is 1
  - A.k.a. logical sum
- AND
  - A * B or AB
  - 1 only if both input signal are 1
  - A.k.a. logical product
- NOT
  - A or A’ or ~A
  - 1 only if the inputs is 0
  - A.k.a. inversion or negation

Truth Tables and Logic Gates

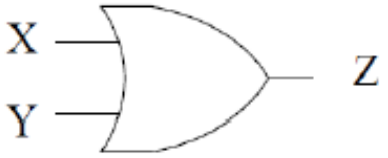


NOT
Z = X' or $\bar{X}$ or ~X

X	Z
0	1
1	0

AND
Z = X · Y

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

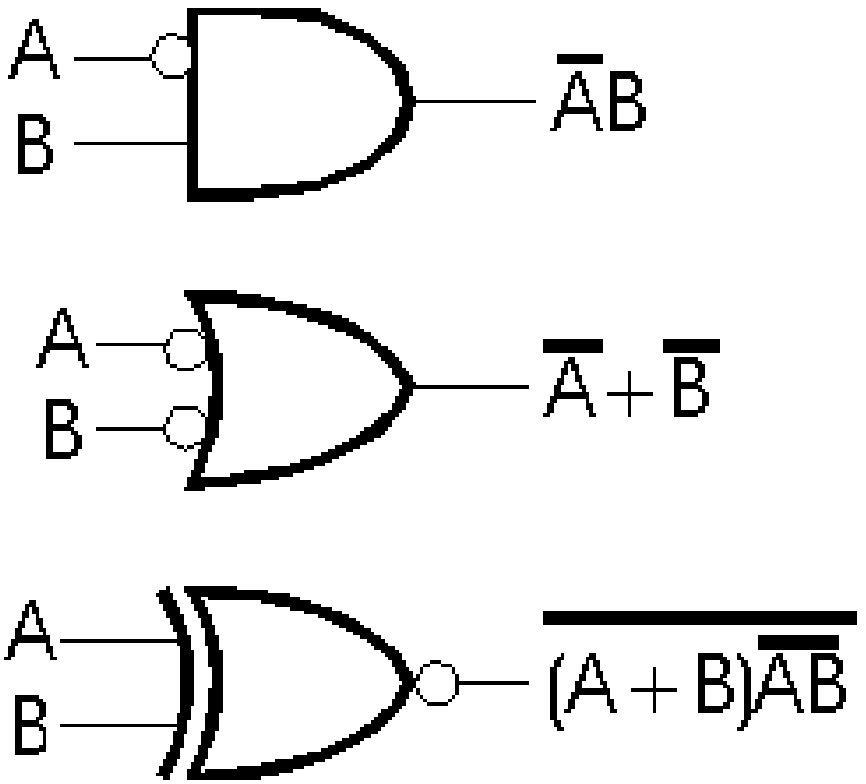


OR
Z = X + Y

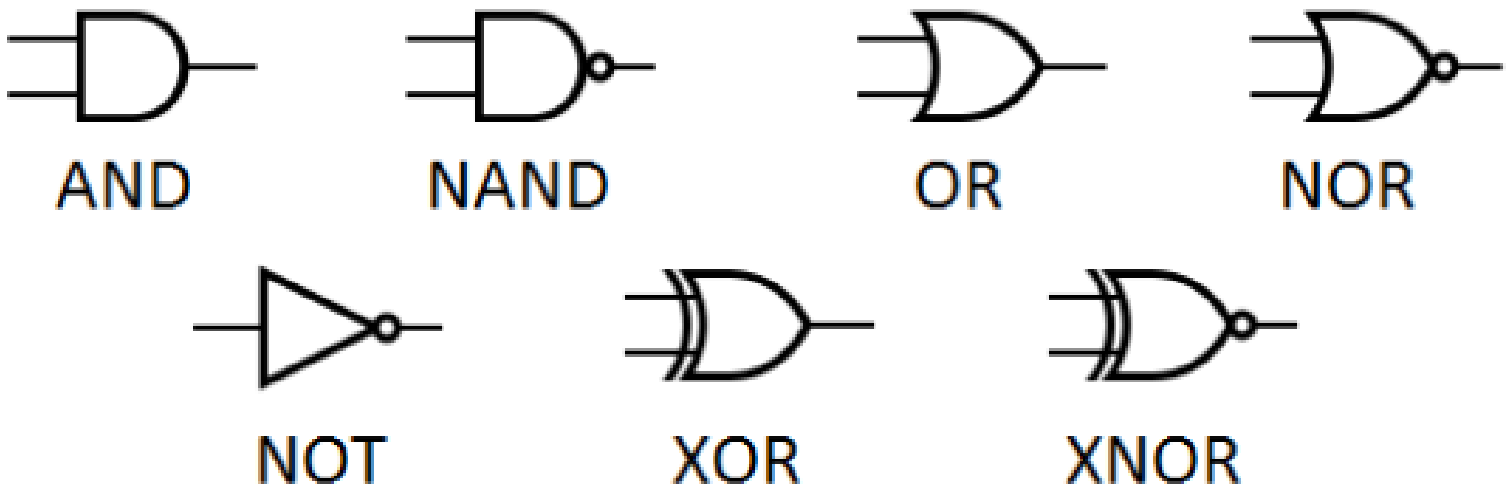
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Inverted Signals

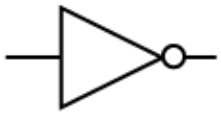
- A small circle on either the input or output of a gate means that that signal is inverted.
  - That is, it's as if there were an inverter (not) gate there



Seven Basic Gates



Truth Tables for Basic Gates



NOT

Input	Output
I	F
0	1
1	0

AND

Inputs		Output
A	B	F
0	0	0
1	0	0
0	1	0
1	1	1

NAND

Inputs		Output
A	B	F
0	0	1
1	0	1
0	1	1
1	1	0

OR

Inputs		Output
A	B	F
0	0	0
1	0	1
0	1	1
1	1	1

NOR

Inputs		Output
A	B	F
0	0	1
1	0	0
0	1	0
1	1	0

EXCLUSIVE OR

XOR

Inputs		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

EXCLUSIVE NOR

XNOR

Inputs		Output
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

NAND

NAND (Not AND)

$F(x,y) = (x\ y)'$

Table 4.9 Truth table for NAND.

x	y	F(x,y)
0	0	1
0	1	1
1	0	1
1	1	0

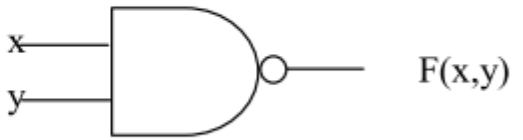
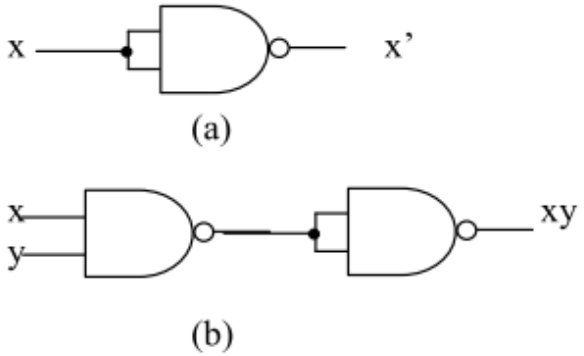
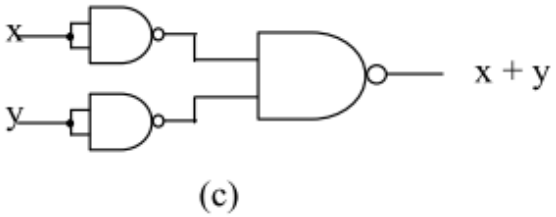


Figure 4.4 Logic symbol for NAND gate.

$x' = (x\ x)'$ $\quad x\ y = [\ (x\ y)'\ ]'$ $\quad x + y = (x'\ y'\ )'$





Implementation of NOT, AND, OR using NAND gates

NOR

NOR (Not OR) $F(x,y) = (x + y)'$

Table 4.10 Truth table for NOR.

x	y	F(x,y)
0	0	1
0	1	0
1	0	0
1	1	0

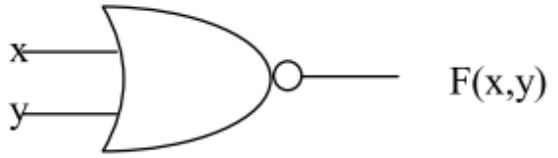
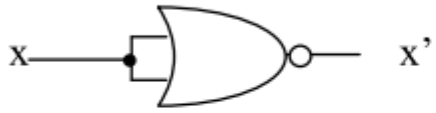
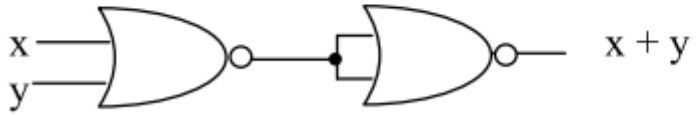


Figure 4.6 Logic symbol for NOR gate.

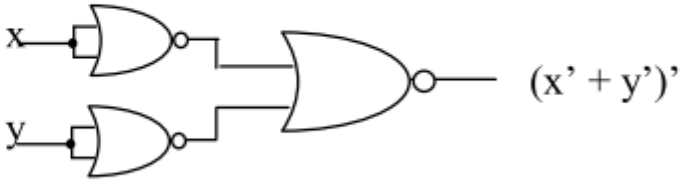
$x' = (x + x)'$ $x + y = [(x + y)']'$ $x\,y = (x' + y')'$



(a)



(b)



(c)

XOR Gate

EXCLUSIVE-OR (XOR)

$F(x, y) = x \oplus y$

Table 4.11 Truth table for EXCLUSIVE-OR.

x	y	F(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

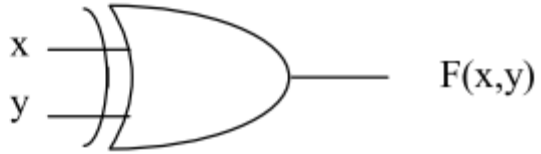
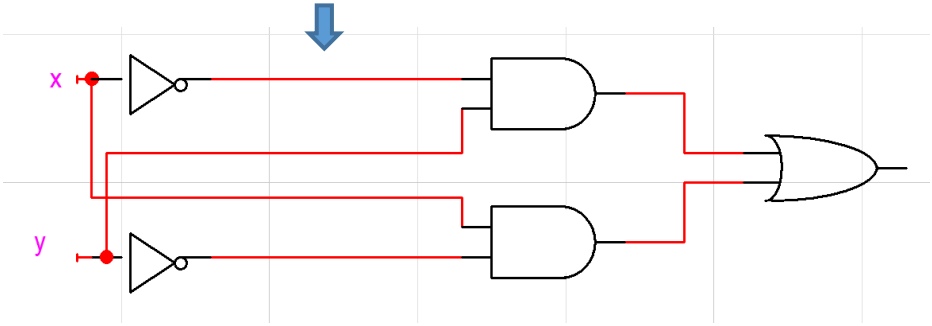


Figure 4.8 Logic symbol for EXCLUSIVE-OR gate.

$$x \oplus y = x'y + xy' = (x' + y')(x + y)$$



The *Output* signal from an **XOR** gate is *True* (on, 1) if **either** *Input* signal are *True* (on, 1).

The *Output* signal from an **XOR** gate is *False* (off, 0) if **both** *Input* signals are *False* (off, 0) or **both** input signals are True(on, 1).

XNOR

EQUIVALENCE, Exclusive-NOR (XNOR) $F(x, y) = (x \oplus y)' = x \odot y$

Table 4.12 Truth table for EXCLUSIVE-NOR.

x	y	F(x,y)
0	0	1
0	1	0
1	0	0
1	1	1

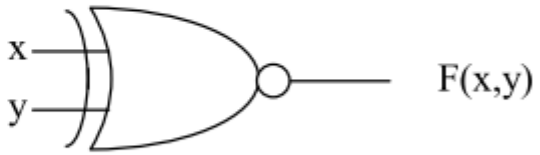
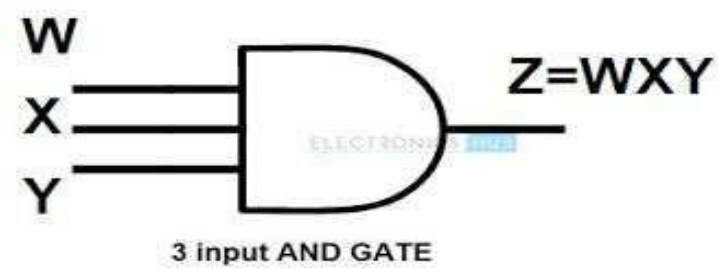
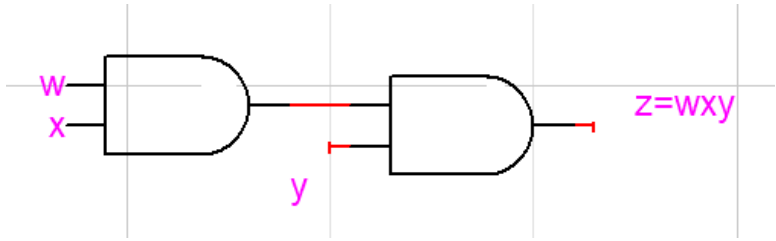


Figure 4.9 Logic symbol for EXCLUSIVE-NOR gate.

$$x \odot y = (x \oplus y)' = x'y' + xy = (x' + y)(x + y')$$

3-Input Versions of Basic Gates





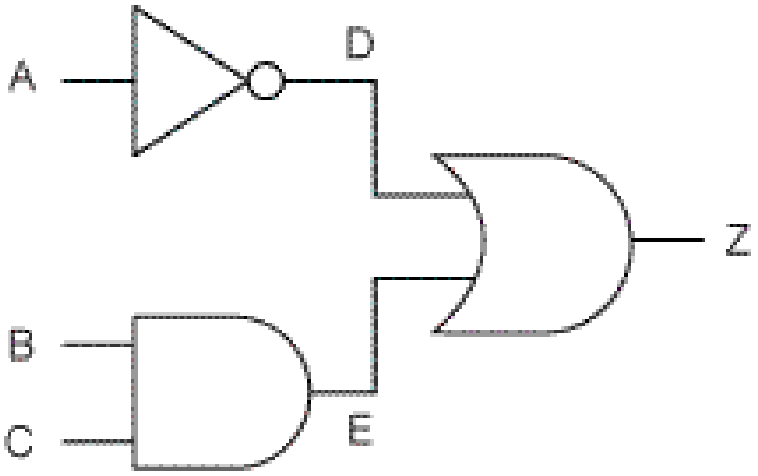
- Some gates allow multiple inputs.
  - For example, a 3-input AND is essentially just a cascade of two 2-input ANDs.
- *What else gates can have 3 inputs?*
  - AND, NAND, OR, NOR
- *What gates cannot have 3 inputs?*
  - XOR, XNOR, NOT

A Complex Function

- Primitive boolean functions may be implemented by logic gates
- More complex functions can be implemented by combinations of gates.

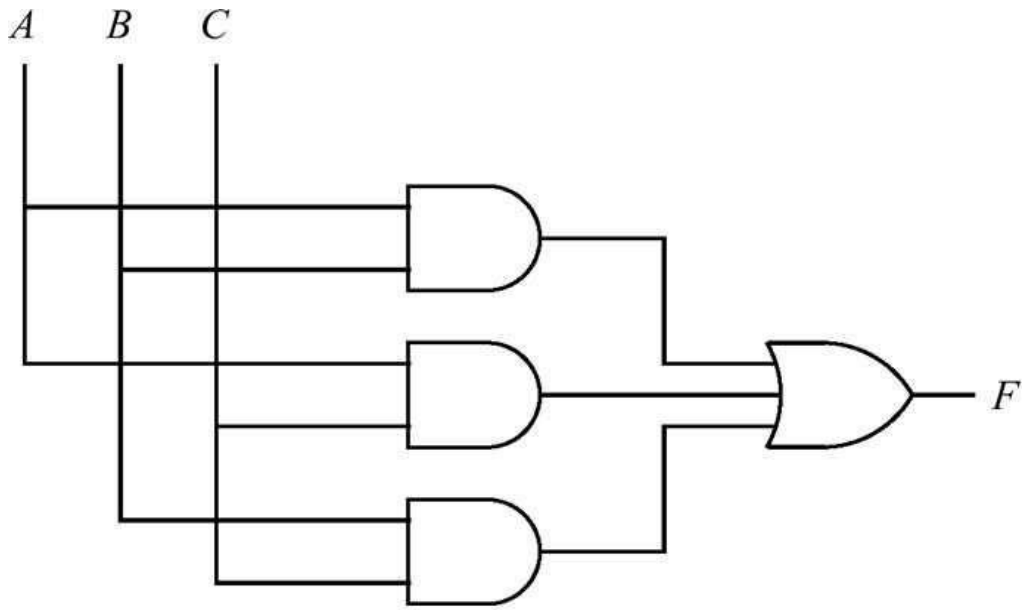
```
Z = !A || (B && C);
```

$$Z = A' + BC$$



A	B	C	Z
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	1

What Does This Logic Do?

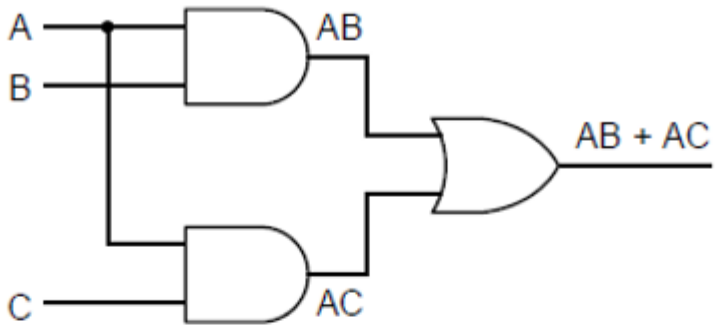


Boolean Expression: $F = AB + AC + BC$

- This is called *Majority Circuit*.
  - The output F will be 1 (true, on), whenever more than half of its inputs are 1 (True, on)

Circuit Equivalence 1.a

- AB + AC

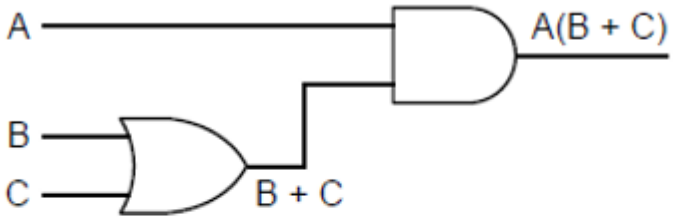


A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)

Circuit Equivalence 1.b

- A(B + C)



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

Useful Laws of Boolean Algebra

Name	AND form	OR form
Identity law	$A \cdot 1 = 1$	$A + 0 = A$
Zero/one law	$A \cdot 0 = 0$	$A + 1 = 1$
Inverse law	$A\ A' = 0$	$A + A' = 1$
Commutative law	$A \cdot B = B \cdot A$	$A + B = B + A$
Associative law	$A \cdot (B \cdot C) = (A * B) * C$	$A + (B + C) = (A + B) + C$
Distributive laws	$A * (B + C) = (A * B) + (A * C)$	$A + (B * C) = (A + B) + (A * C)$
Idempotency law	$A * A = A$	$A + A = A$
Absorption law	$A * (A + B) = A$	$A + (A * B) = A$
Double negation	$(A')' = A$	

Reference Readings

- Patterson, "Computer Organization and Design"
  - Appendix A.1, A.2
- Tanenbaum, "Structured Computer Organization"
  - Sec 3.1