

Chapter 2 – Combinational Digital Circuits

Part 3 – Additional Gates and Circuits

Overview

- **Part 1 – Gate Circuits and Boolean Equations**
 - **Binary Logic and Gates**
 - **Boolean Algebra**
 - **Standard Forms**
- **Part 2 – Circuit Optimization**
 - **Two-Level Optimization**
 - **Map Manipulation**
 - **Practical Optimization (Espresso)**
 - **Multi-Level Circuit Optimization**
- **Part 3 – Additional Gates and Circuits**
 - **Other Gate Types**
 - **Exclusive-OR Operator and Gates**
 - **High-Impedance Outputs**

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

AND, OR, NOT Operations

- They are defined by Boolean functions.
- They represent 3 functions out of 16 possible two variable functions.

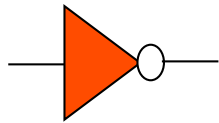
x	y	F ₀	F ₁	F ₂	F ₃	F ₄	F ₅	F ₆	F ₇
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

x	y	F ₈	F ₉	F ₁₀	F ₁₁	F ₁₂	F ₁₃	F ₁₄	F ₁₅
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

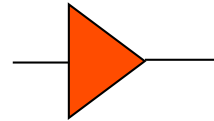
Other Logic Operations

- **Some of the two variable Boolean functions**
 - **Constant functions**: $F_0 = 0$ and $F_{15} = 1$
 - **AND function**: $F_1 = xy$
 - **OR function**: $F_7 = x + y$
 - **ExclusiveOR function (XOR)**:
 - $F_6 = x' y + xy' = x \oplus y$ (x or y, but not both)
 - **Equivalence function (XNOR)**:
 - $F_9 = xy + x' y' = (x \oplus y)'$ (x equals to y)
 - **NOR function**:
 - $F_8 = (x + y)' = (x \downarrow y)$ (Not-OR)
 - **NAND function**:
 - $F_{14} = (x y)' = (x \uparrow y)$ (Not-AND)

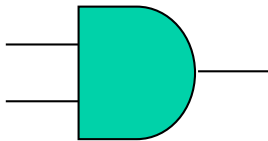
Logic Gate Symbols



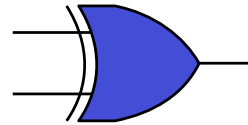
NOT



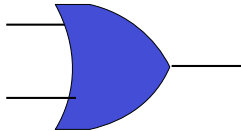
BUFFER



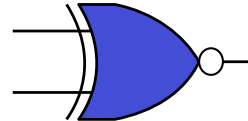
AND



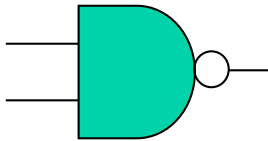
XOR



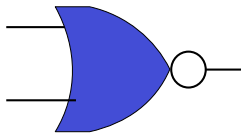
OR



XNOR



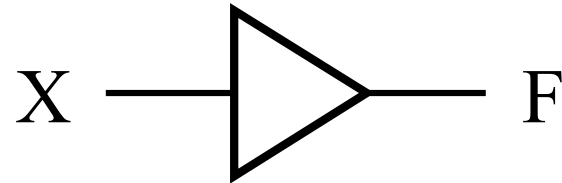
NAND



NOR

Buffer

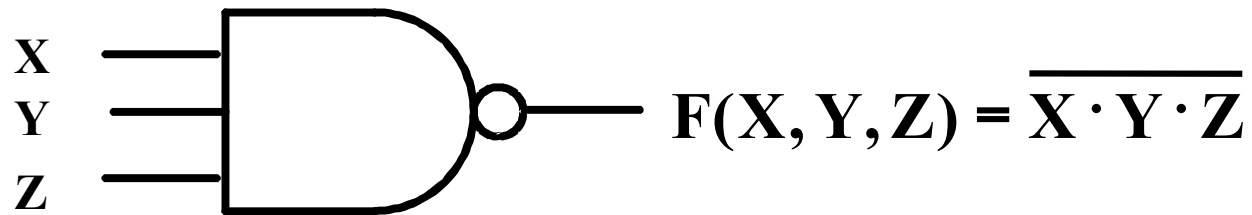
- A buffer is a gate with the function $F = X$:



- In terms of Boolean function, a buffer is the same as a connection!
- So why use it?
 - A buffer is an electronic amplifier used to improve circuit voltage levels and increase the speed of circuit operation.

NAND Gate

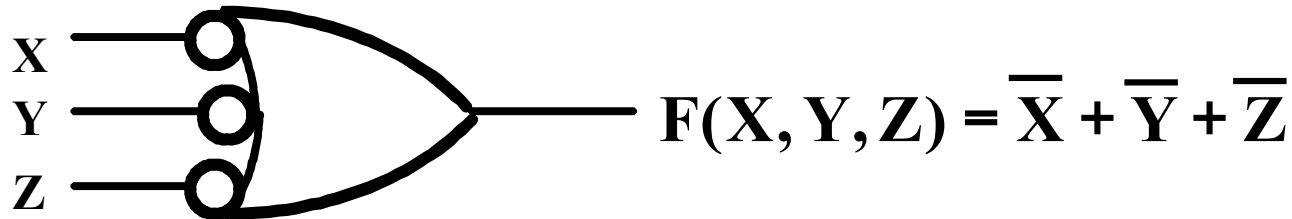
- The basic NAND gate has the following symbol, illustrated for three inputs:
 - **AND-Invert (NAND)**



- 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)



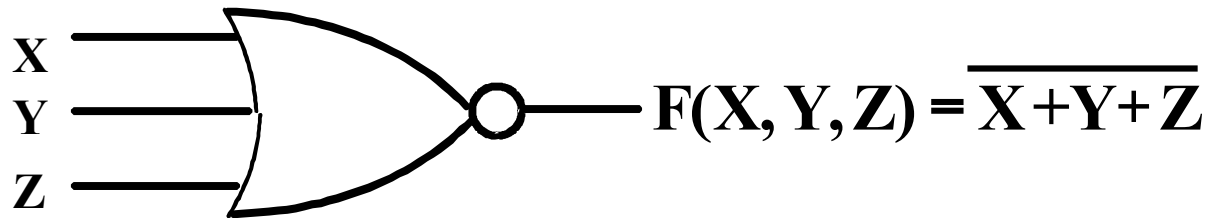
- This NAND symbol is called Invert-OR, since inputs are inverted and then ORed together.
- AND-Invert and Invert-OR both represent the NAND gate. Having both makes visualization of circuit function easier.
- A NAND gate with one input degenerates to an inverter.

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

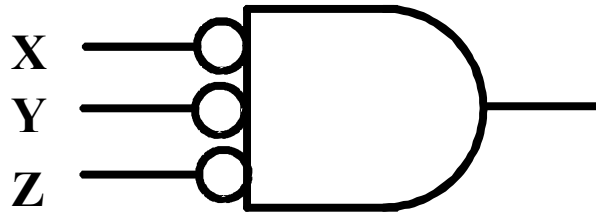
- The basic NOR gate has the following symbol, illustrated for three inputs:
 - OR-Invert (NOR)



- 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!**

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:

$$X \oplus 0 = X$$

$$X \oplus X = 0$$

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

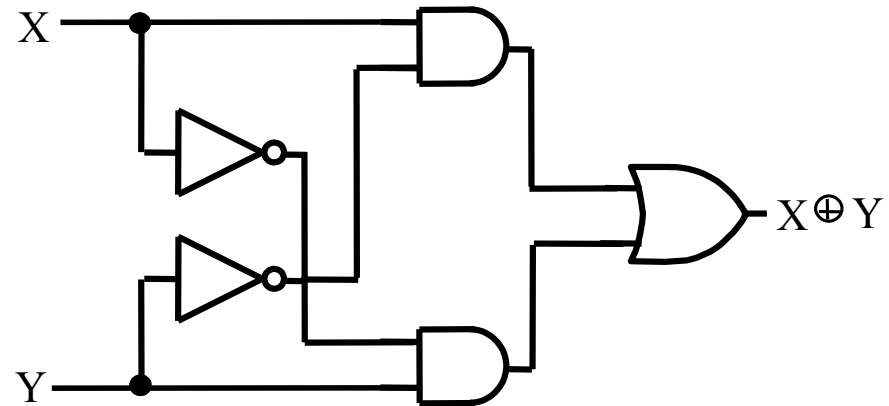
$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

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

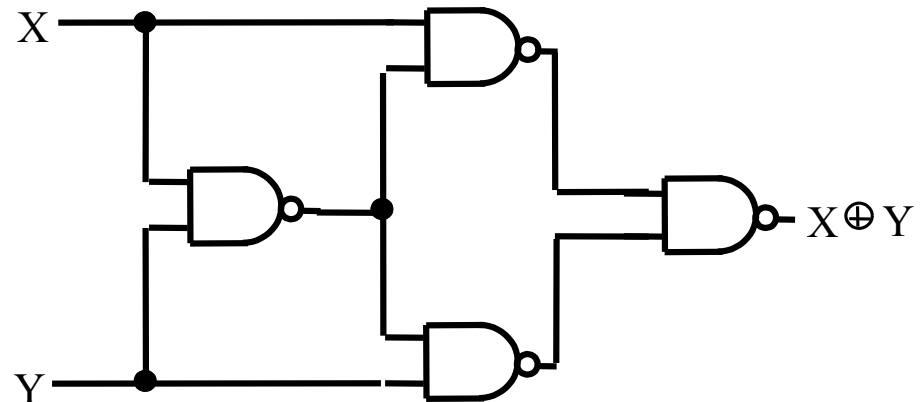
$$X \oplus \bar{X} = 1$$

XOR Implementations

- The simple SOP implementation uses the following structure:



- A NAND only implementation is:

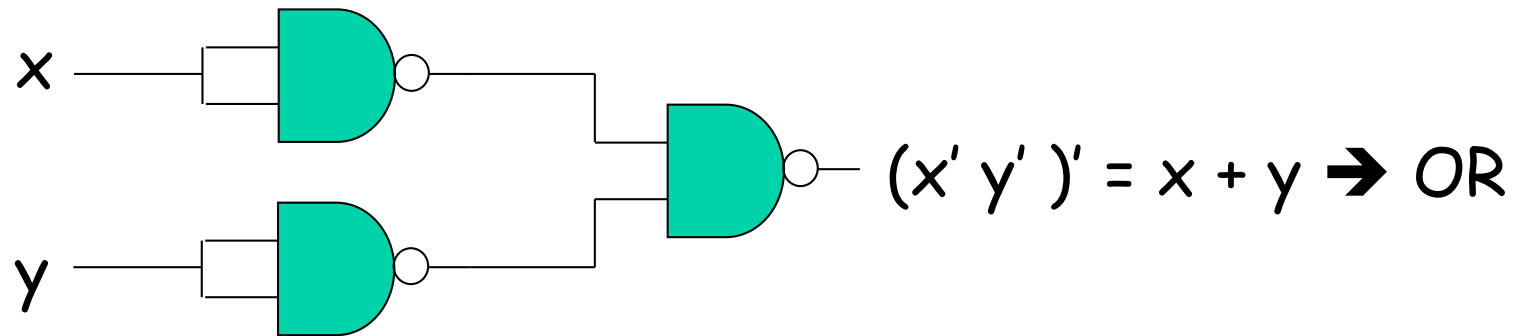
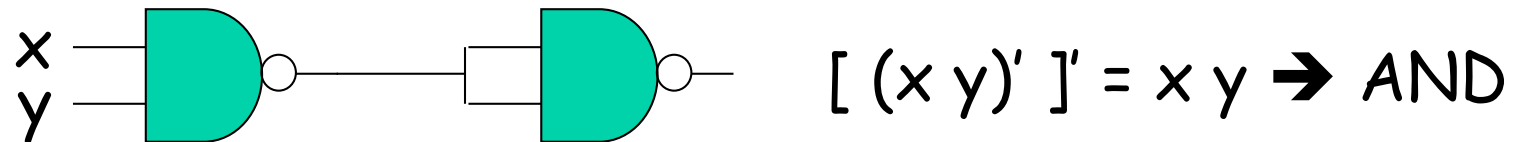
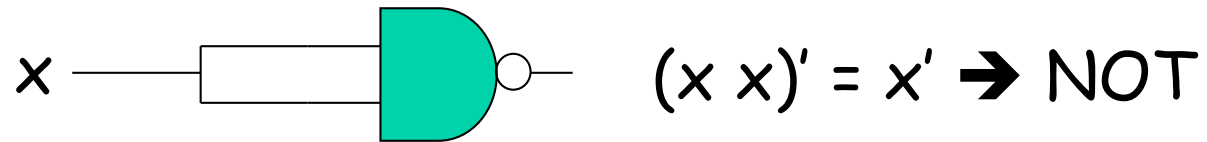


Universal Gate

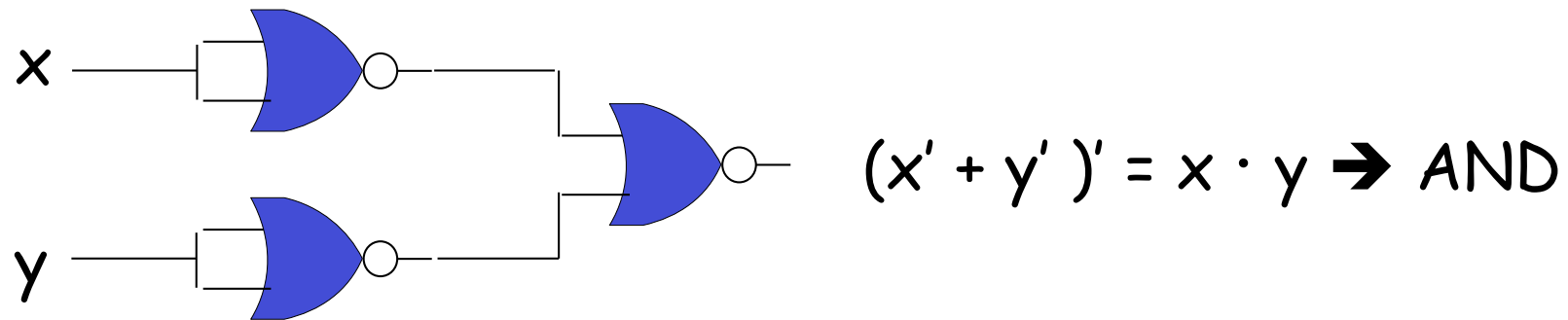
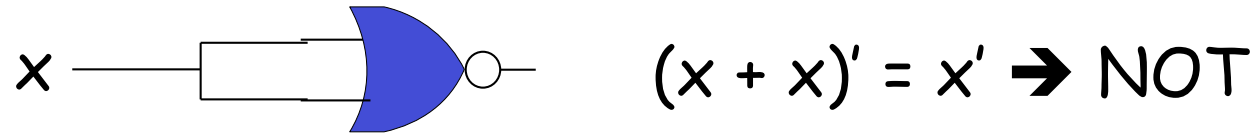
- **NAND and NOR gates are universal.**
- **All Boolean functions can be implemented by AND, OR and NOT gates.**
- **These operations can be implemented by NAND and NOR gates.**

x	y	$(xy)'$	x'	y'	$(x' y')'$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	1

NAND Gates

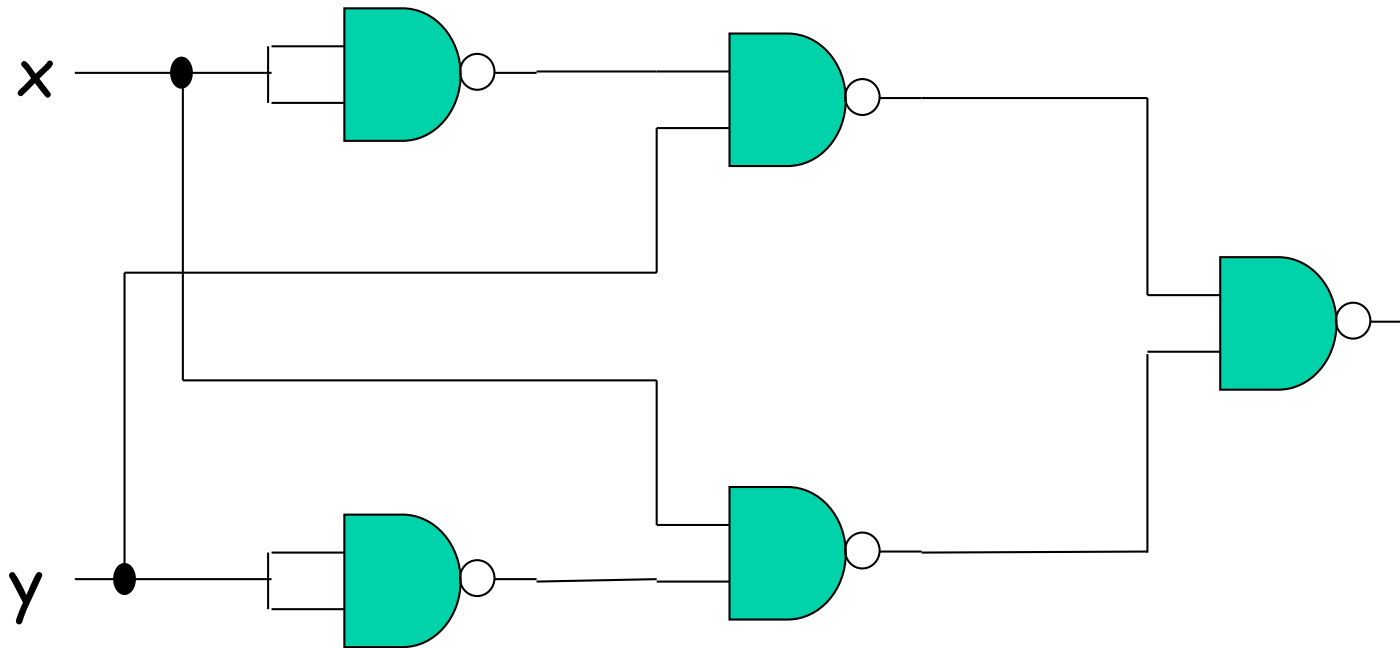


NOR Gate



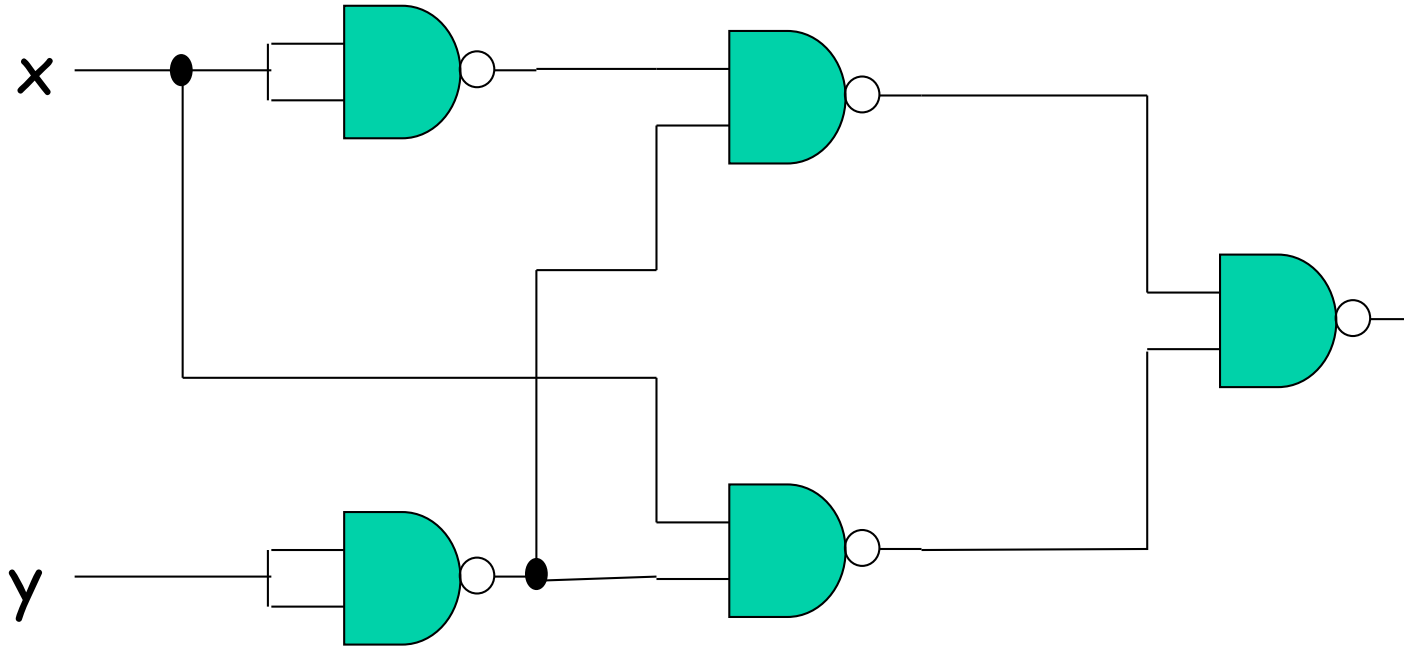
Example 1

- $F_1 = x' y + x y'$



Example 2

- $F_2 = x' y' + xy'$



Multiple Input Gates

- **AND and OR gates:**

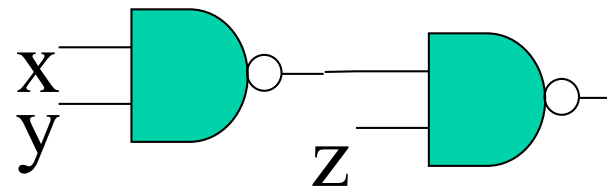
- Commutative and associative properties exist.
- There is no problem to increase the number of inputs.

- **NAND and NOR gates:**

- They have commutative, but not associative property.
- It is not easy to increase the number of inputs.

- **Example: NAND gates**

- $((x \ y)'z)' \neq (x(yz)')'$
- $((xy)'z)' = ((x'+y')z)' = xy+yz'$



- $(x \ (yz)')' = x'+yz$

