Other Gates

Circuit Optimization

- **Goal: To obtain the simplest implementation for a given function**
- **Optimization is a more formal approach to simplification that is performed using a specific procedure or algorithm**
- **Optimization requires a cost criterion to measure the simplicity of a circuit**
- **Distinct cost criteria we will use:**
  - **Literal cost (L)**
  - **Gate input cost (G)**
  - **Gate input cost with NOTs (GN)**

Literal Cost

- **Literal – a variable or its complement**
- **Literal cost – the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram**
- **Examples:**
  - $F = BD + A\bar{B}C + A\bar{C}\bar{D}$ **L = 8**
  - $F = BD + A\bar{B}C + A\bar{B}\bar{D} + AB\bar{C}$ **L = 11**
  - $F = (A + B)(A + D)(B + C + \bar{D})(\bar{B} + \bar{C} + D)$ **L = 10**
  - **Which solution is best? The first solution**

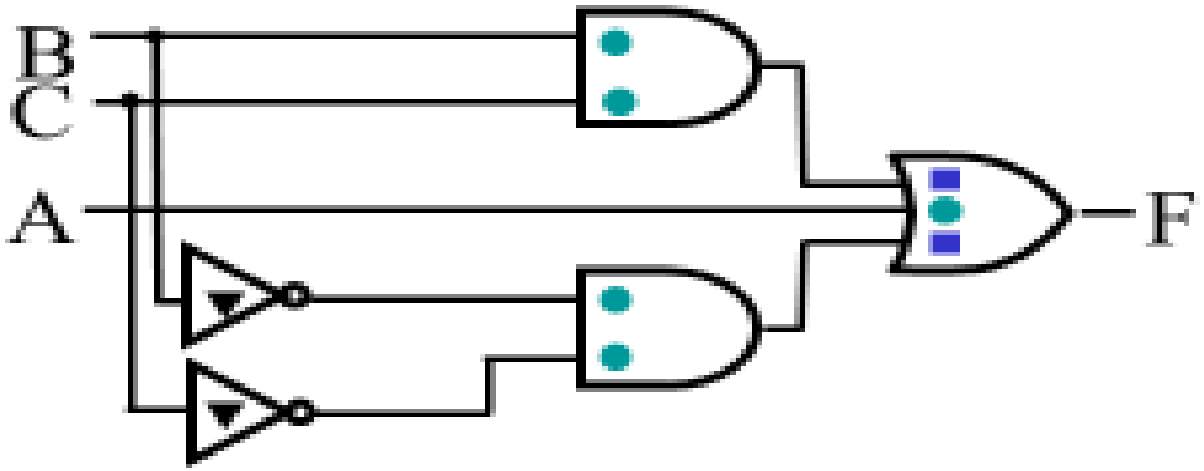
Cost Criteria (continued)

- Example 1:
- $F = A + B\,C + \overline{B}\,\overline{C}$

$GN = G + 2 = 9$

$L = 5$

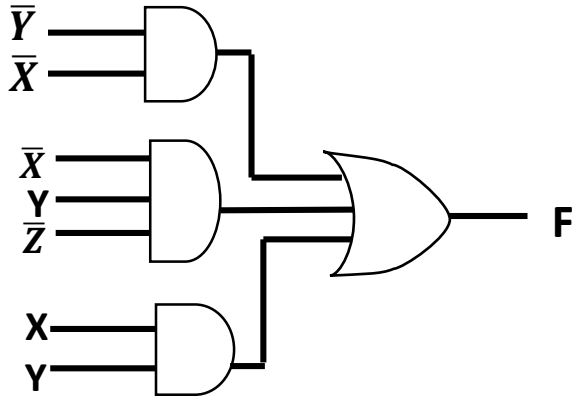
$G = L + 2 = 7$



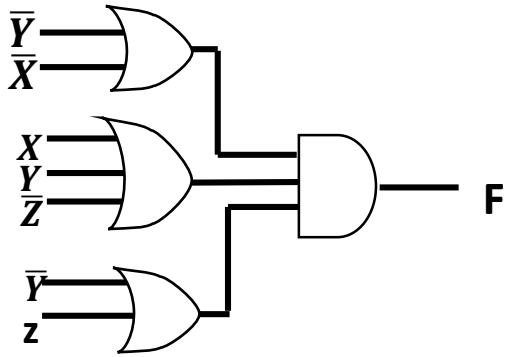
- L (literal count) counts the AND inputs and the single literal OR input.
- G (gate input count) adds the remaining OR gate inputs
- GN(gate input count with NOTs) adds the inverter inputs

AND/OR Two-level Circuit optimization

- **The two implementations for F are shown below – it is quite apparent which is simpler!**



$F = \bar{Y} + \bar{X}Y\bar{Z} + XY$



$F = X(\bar{Y} + Z)(X + Y + \bar{Z})$

Multiple-Level Optimization

- Multiple-level circuits - circuits that are not two-level (with or without input and/or output inverters)
- Multiple-level circuits can have reduced gate input cost compared to two-level (SOP and POS) circuits
- Multiple-level optimization is performed by applying transformations to circuits represented by equations while evaluating cost

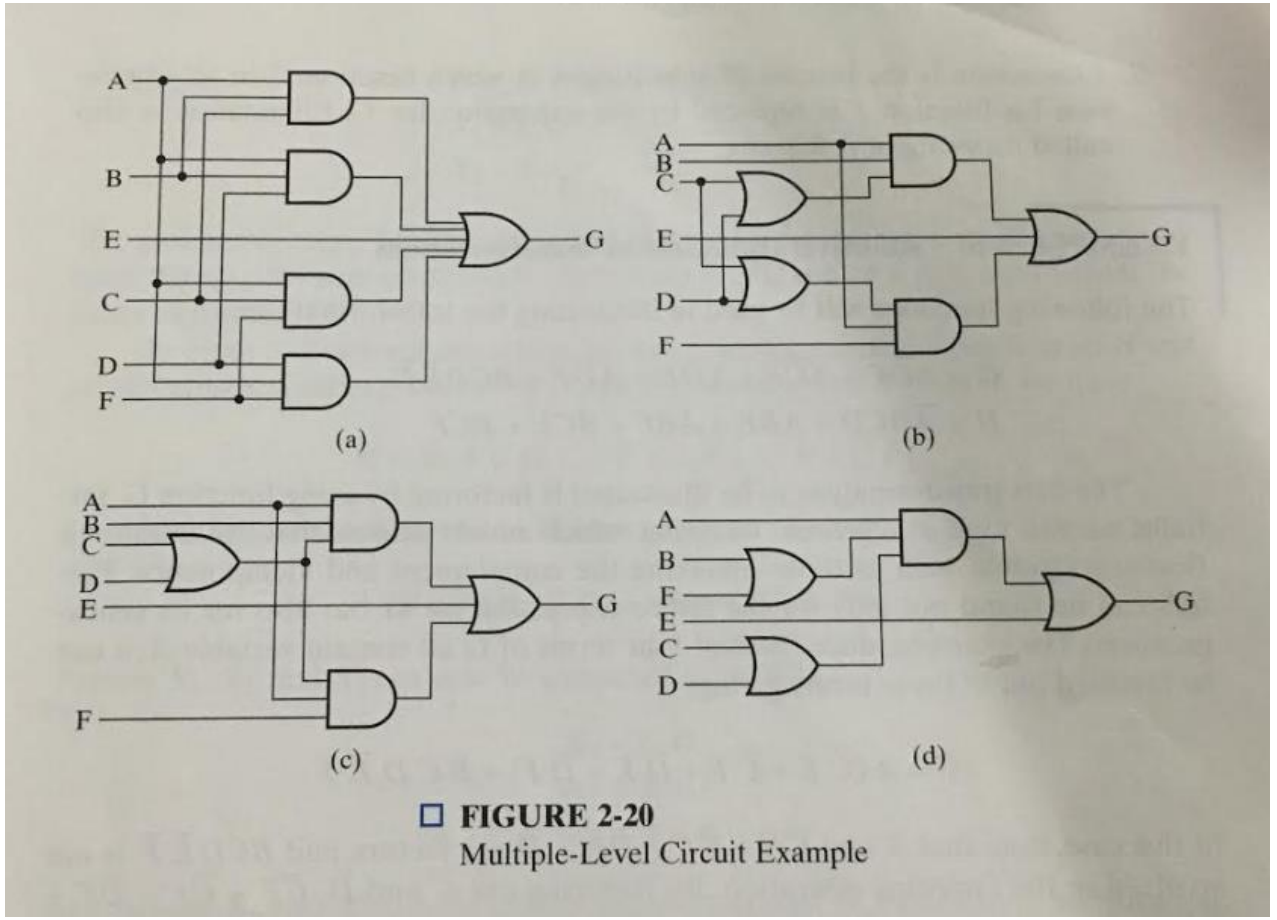


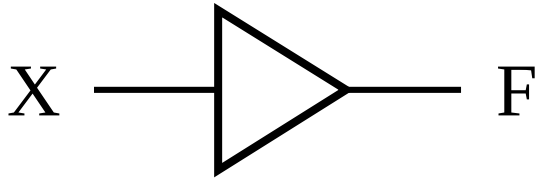
FIGURE 2-20
Multiple-Level Circuit Example

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

Buffer

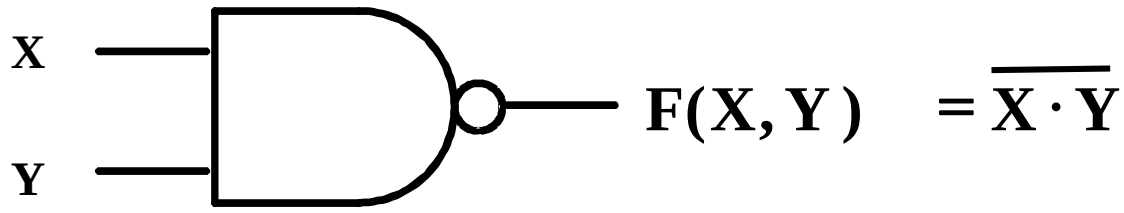
- A buffer is a gate with the function F = X:



- 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)

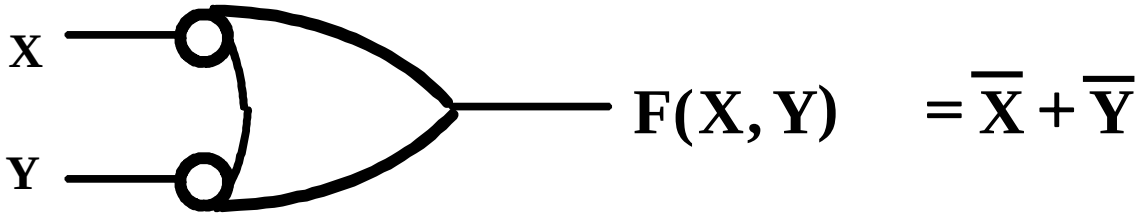


X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

- 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)

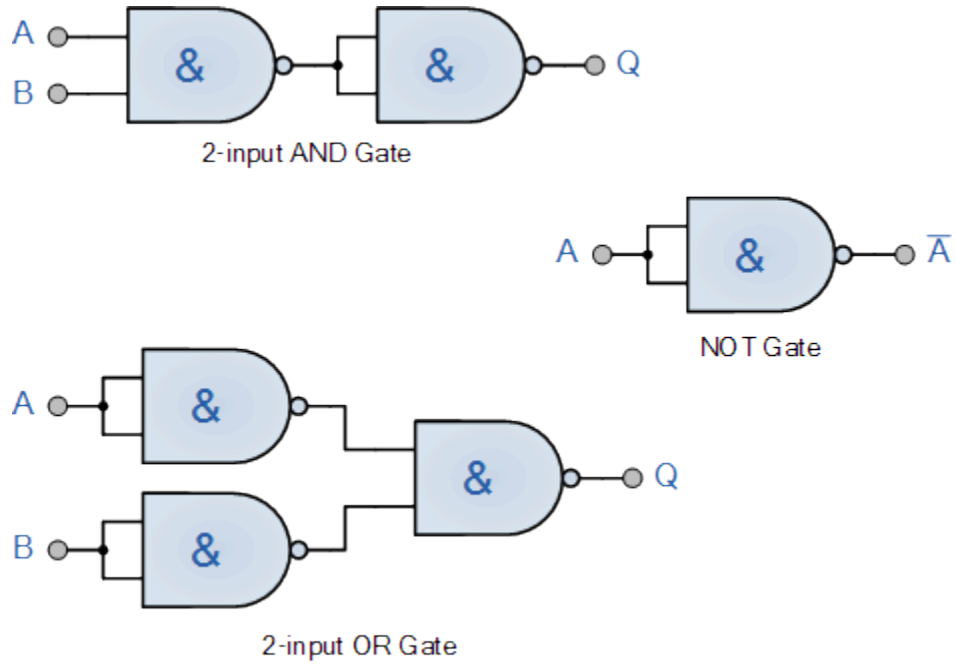


X	Y	$\bar{X}$	$\bar{Y}$	F=$\bar{X}$+$\bar{Y}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

- 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.

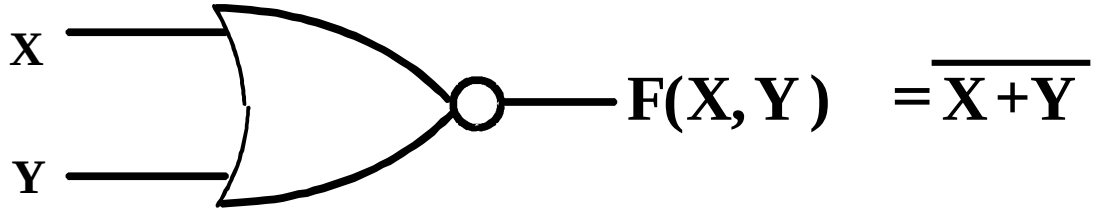
NAND Gates (universal gate)

- *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.



NOR Gate

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

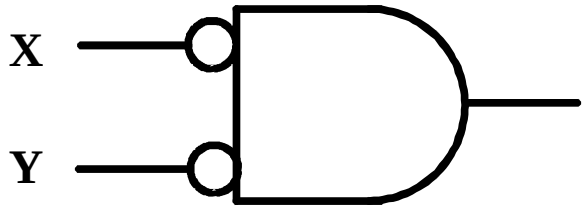


X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

- 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)



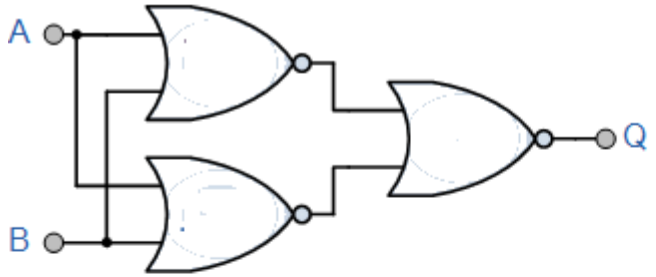
$$F(X,Y) = \bar{X}.\bar{Y}$$

X	Y	$\bar{X}$	$\bar{Y}$	F=$\bar{X}\bar{Y}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

- 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.

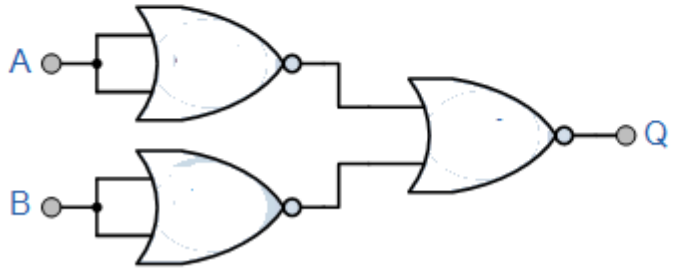
NOR Gate (universal gate)

- **NAND, NOR universal gates: can accomplish any of the basic operations and produce inverter, OR gate or AND gate**
- **The non-inverting gates aren’t: they can't produce an invert.**



2-input OR Gate

NOT Gate



2-input AND Gate

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: $$X \oplus Y = X\overline{Y} + \overline{X}Y$$
  - The eXclusive NOR (XNOR) function, otherwise known as *equivalence* is: $$\overline{X \oplus Y} = XY + \overline{X}\,\overline{Y}$$
- Strictly speaking, XOR and XNOR gates do no exist for more that 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

- the XNOR function is also known as the *equivalence* function, because

  It is equals to 1 if and only if X is equivalent to Y

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:

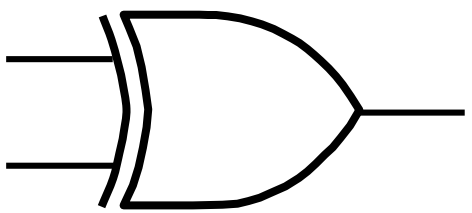
$$X \oplus Y \oplus Z = \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,\overline{Z} + X\,\overline{Y}\,\overline{Z} + X\,Y\,Z$$

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

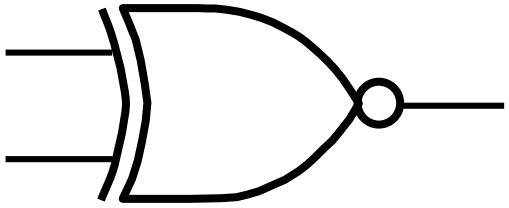
$$X \oplus 0 = X \qquad X \oplus 1 = \overline{X}$$
$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1$$
$$X \oplus Y = Y \oplus X$$
$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

Symbols For XOR and XNOR

- XOR symbol:



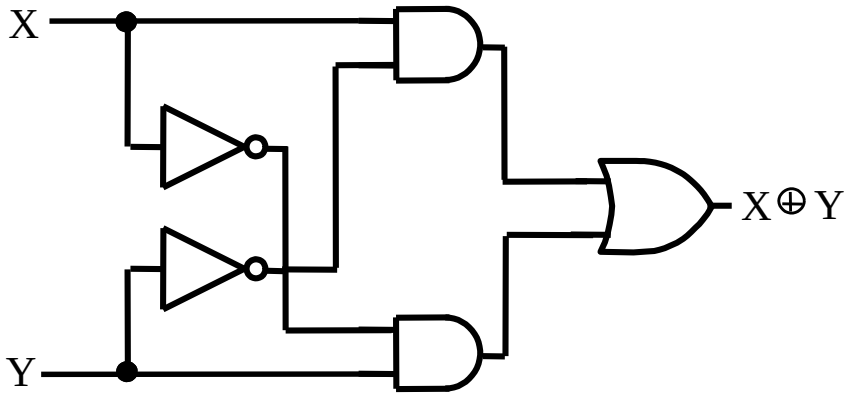
- XNOR symbol:



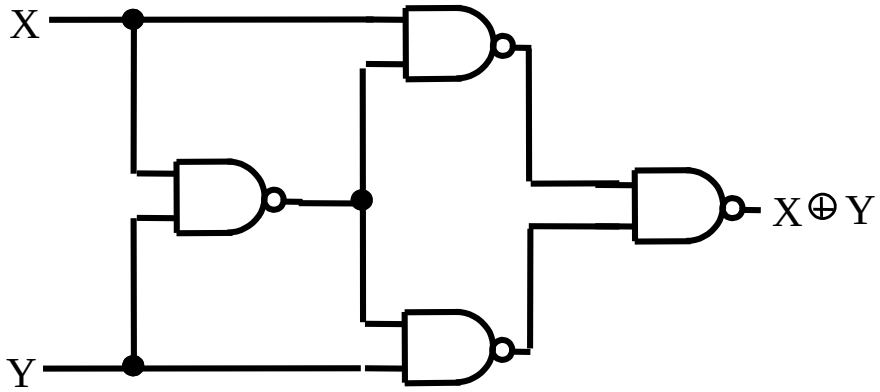
- Shaped symbols exist only for two inputs

XOR Implementations

- The simple SOP implementation uses the following structure:

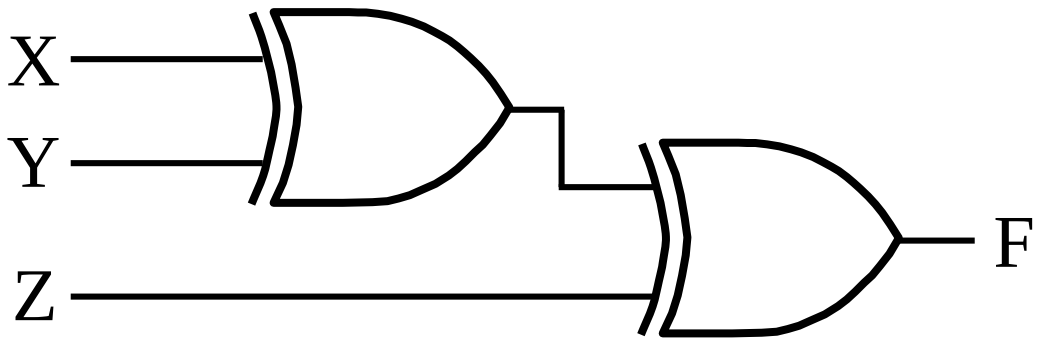


- A NAND only implementation is:



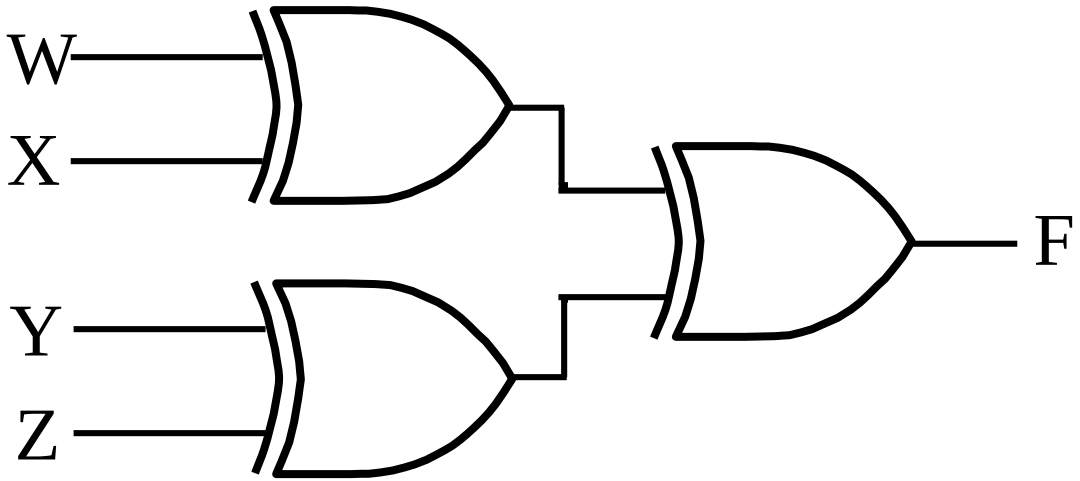
Example: Odd Function Implementation

- 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 = W⊕X⊕ Y⊕Z with 2-input XOR and XNOR gates
- Factoring, F = (W⊕X)⊕(Y⊕Z)
- The circuit:



Hi-Impedance Outputs

- Logic gates introduced thus far
  - have 1 and 0 output values,
  - cannot have their outputs connected together, and
  - transmit signals on connections in only one direction.
- Three-state logic adds a third logic value, Hi-Impedance (Hi-Z), giving three states: 0, 1, and Hi-Z on the outputs.

Hi-Impedance Outputs (continued)

- What is a Hi-Z value?
  - The Hi-Z value behaves as an open circuit
  - This means that, looking back into the circuit, the output appears to be disconnected.
  - It is as if a switch between the internal circuitry and the output has been opened.
- Hi-Z may appear on the output of any gate, but we restrict gates to:
  - a 3-state buffer, or
  - Optional: a transmission gate (See Reading Supplement: More on CMOS Circuit-Level Design),

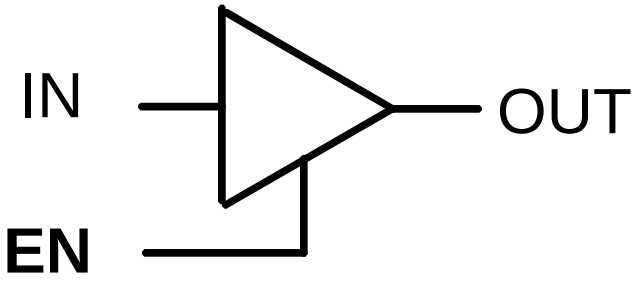
  each of which has one data input and one control input.

The 3-State Buffer

- For the symbol and truth table, IN is the data input, and EN, the control input.
- For EN = 0, regardless of the value on IN (denoted by X), the output value is Hi-Z.
- For EN = 1, the output value follows the input value.
- Variations:
  - Data input, IN, can be inverted
  - Control input, EN, can be inverted

  by addition of “bubbles” to signals.

Symbol



Truth Table

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1