

# Elementary Logic Gates

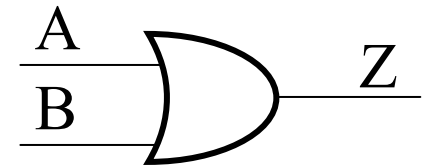
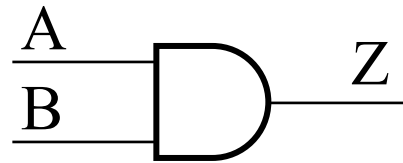
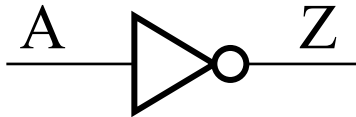
*Name*

Inverter  
(NOT Gate)

AND Gate

OR Gate

*Symbol*



*Truth  
Table*

| A | Z |
|---|---|
| 0 | 1 |
| 1 | 0 |

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

*Logic  
Equation*

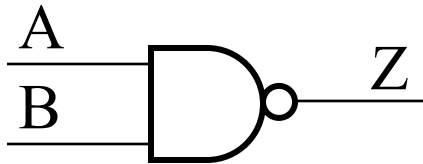
$$Z = A' = \bar{A}$$

$$Z = A \cdot B = AB$$

$$Z = A + B$$

# Other Elementary Logic Gates

NAND Gate  
(NOT AND)



| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

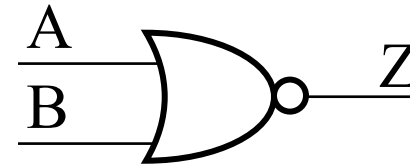
$$Z = (A \cdot B)' = \overline{AB}$$

*Name*

*Symbol*

*Truth Table*

NOR Gate  
(NOT OR)



| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

$$Z = (A + B)' = \overline{A+B}$$

# Using Truth Tables to Prove Theorems

- DeMorgan's Theorems

T8a:  $(X+Y)' = X' \cdot Y'$

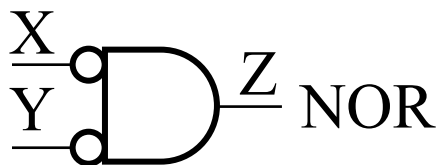
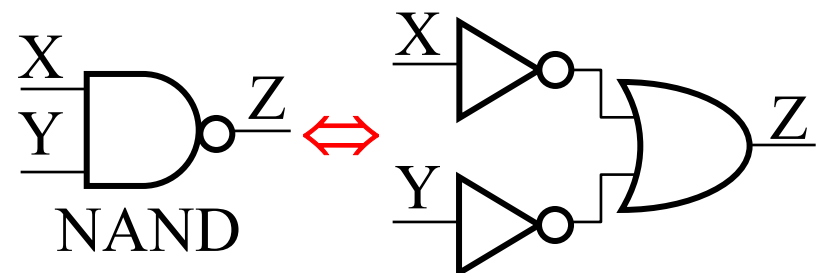
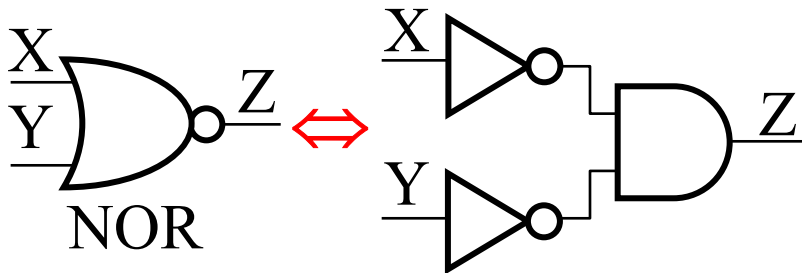
T8b:  $(X \cdot Y)' = X' + Y'$

*a NOR gate is equivalent to an AND gate with inverted inputs*

| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

*a NAND gate is equivalent to an OR gate with inverted inputs*

| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



*alternate logic symbols*



# Other Logic Gates

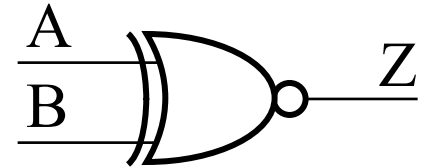
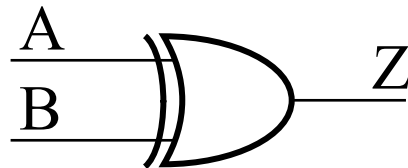
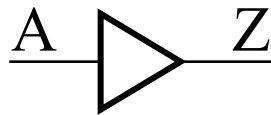
*Name*

Buffer

Exclusive-OR Gate  
aka XOR Gate

Exclusive-NOR Gate  
aka XNOR or NXOR Gate

*Symbol*



*Truth Table*

| A | Z |
|---|---|
| 0 | 0 |
| 1 | 1 |

| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

*Logic Equation*

$$Z = A$$

$$Z = A \oplus B = \bar{A}B + A\bar{B}$$

$$Z = \overline{A \oplus B} = \bar{A}\bar{B} + AB$$

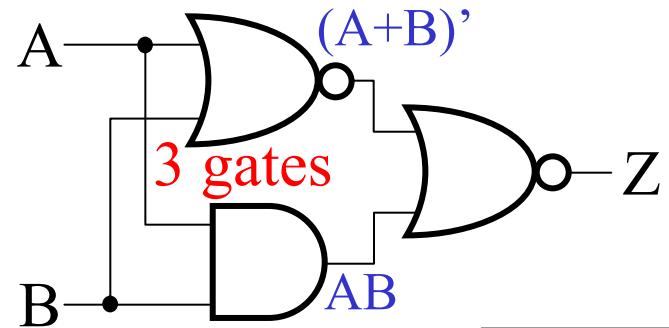
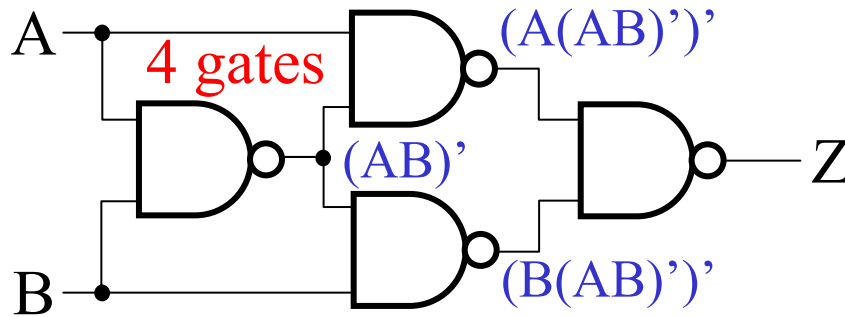
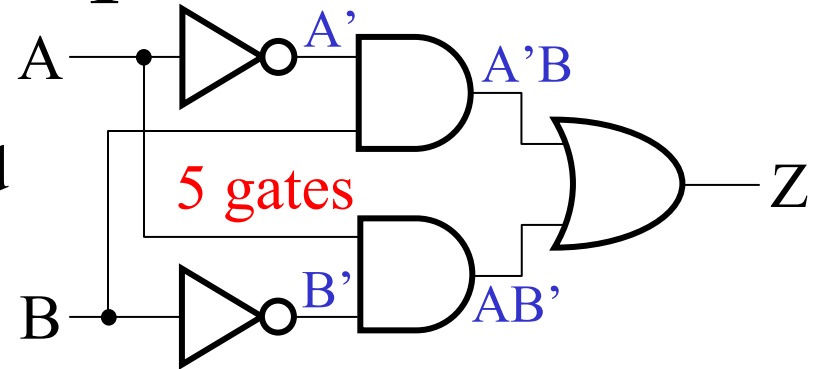
*also denoted  $Z = A \odot B$*

# Interesting Properties of Exclusive-OR

- Controlled inverter
  - $X \oplus 0 = X$
  - $X \oplus 1 = X'$
- XOR with one input inverted = XNOR
  - $X \oplus Y' = X' \oplus Y = (X \oplus Y)'$
- XNOR with one input inverted = XOR
  - $(X \oplus Y')' = (X' \oplus Y)' = X \oplus Y$
- Constant output
  - $X \oplus X = 0$
  - $X \oplus X' = 1$

# Exclusive-OR Implementations

- $Z=A'B+AB'$
- XOR & XNOR not considered elementary logic gates by many designers



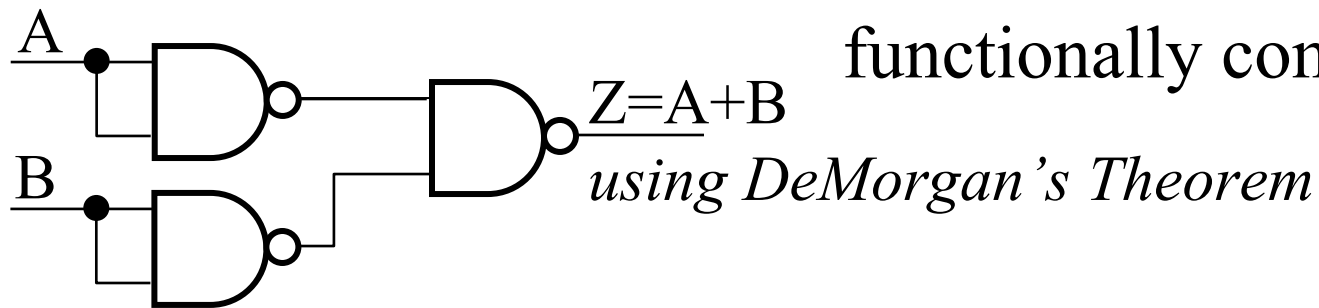
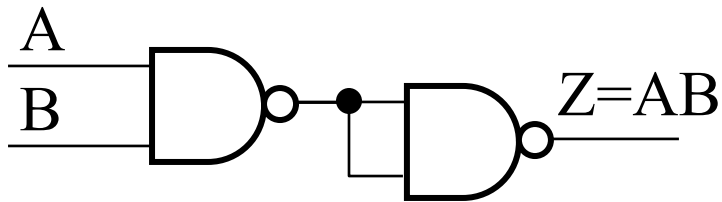
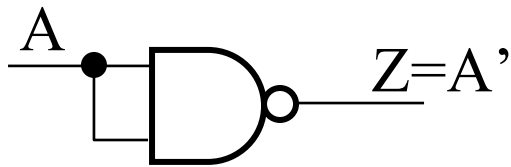
$$\begin{aligned}
 Z &= \overline{\overline{\overline{A(AB)'}} \overline{\overline{B(AB)'}}} \\
 &= \overline{\overline{A} \overline{\overline{AB}} \cdot \overline{\overline{B} \overline{\overline{AB}}}} = \overline{\overline{A} \overline{\overline{AB}} + \overline{\overline{B} \overline{\overline{AB}}}} \\
 &= A(\overline{\overline{A}} + \overline{\overline{B}}) + B(\overline{\overline{A}} + \overline{\overline{B}}) \\
 &= AA' + AB' + A'B + BB' = AB' + A'B
 \end{aligned}$$

$$\begin{aligned}
 Z &= \overline{\overline{\overline{(A+B)'}} + \overline{\overline{AB}}} = \overline{\overline{\overline{A+B}} + \overline{\overline{AB}}} \\
 &= \overline{\overline{\overline{A+B}} \cdot \overline{\overline{AB}}} = \overline{\overline{(A+B)}(\overline{\overline{A}} + \overline{\overline{B}})} \\
 &= AA' + AB' + A'B + BB' \\
 &= 0 + AB' + A'B + 0 = AB' + A'B
 \end{aligned}$$

# Functionally Complete Set of Gates

- If any digital circuit can be built from a set of gates, that set is said to be *functionally complete*
- Functionally complete sets of gates:
  - AND, OR, & NOT
  - NAND
  - NOR
  - Multiplexers
- To show a set of gates is functionally complete, we must show that you can construct AND, OR and NOT functions

# Functionally Complete Set of Gates



- The NAND gate is functionally complete
  - We can build any digital logic circuit out of all NAND gates
- Same holds true for the NOR gate and the multiplexer
- The XOR & XNOR are not functionally complete

# Gate-level Representations

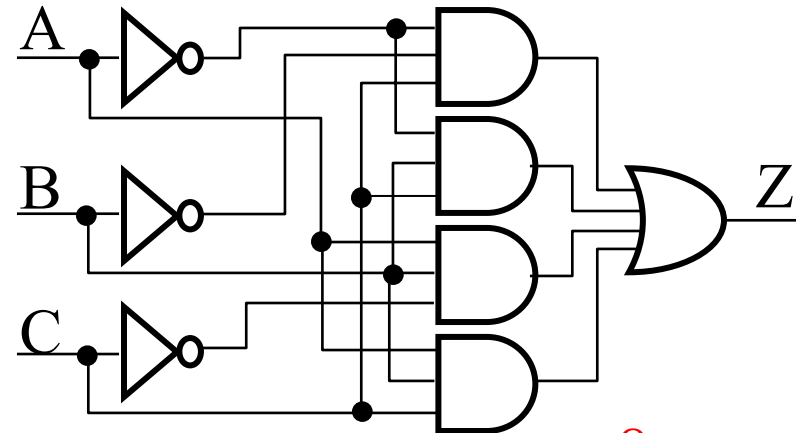
- SOP expressions

- AND-OR

- With inverters for complemented literals

$$Z = A'B'C + A'BC + ABC' + ABC$$

- aka 2-level AND-OR logic representation



8 gates

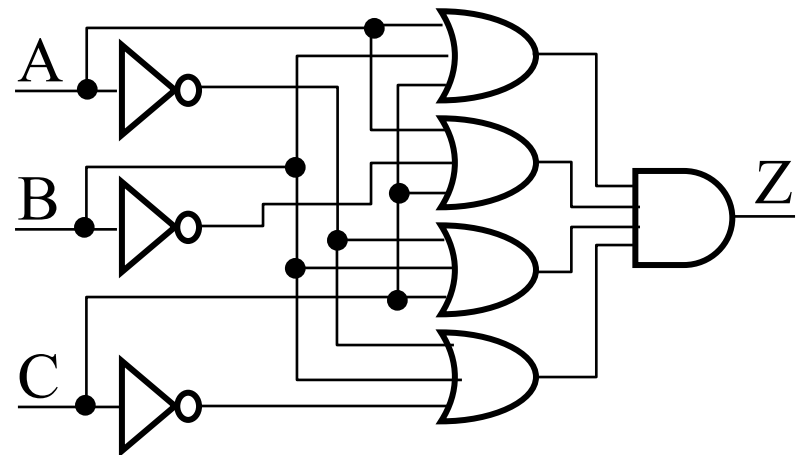
- POS expressions

- OR-AND

- With inverters for complemented literals

$$Z = (A+B+C) \cdot (A+B'+C) \cdot (A'+B+C) \cdot (A'+B+C')$$

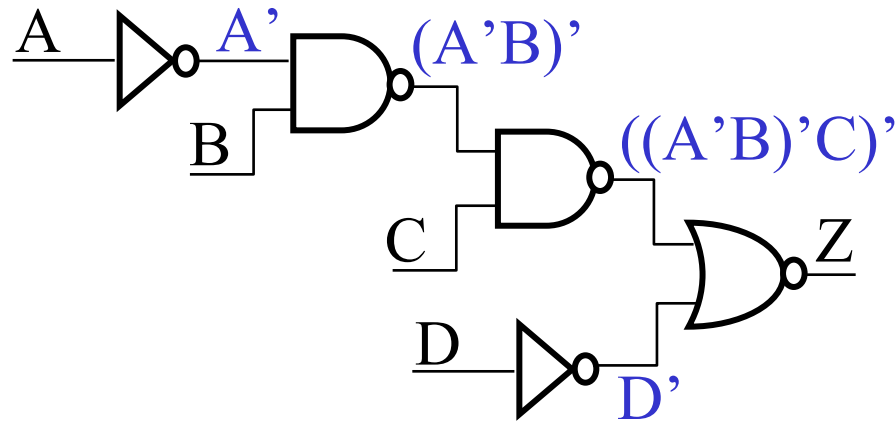
- aka 2-level OR-AND logic representation



# Gate Level Representation

- from Boolean equation

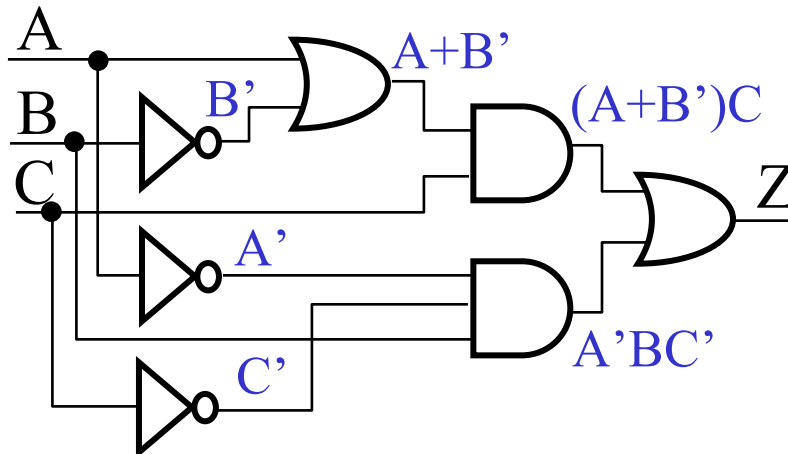
$$Z = \overline{\overline{\overline{((A'B)'C)} + D'}} = \overline{\overline{(\overline{A} \cdot B) \cdot C} + \overline{D}}$$



# Circuit Analysis

- Going from gate-level to
    - truth table
      - Apply 0s & 1s to inputs to get outputs
    - Boolean equation
      - Move equations to output
- $$Z = (A + B')C + A'BC' = AC + B'C + A'BC'$$

| A | B | C | Z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |



# Circuit Analysis

- We can implement different circuits for same logic function that are *functionally equivalent* (produce the correct output response for all input values)
  - Which implementation is the best?
    - Depends on design goals and criteria
- Area analysis
  - Number of gates,  $G$  (most commonly used)
  - Number of gate inputs and outputs,  $G_{IO}$  (more accurate)
    - Bigger gates take up more area
- Performance analysis (*worst case path from inputs to outputs*)
  - Number of gates in worst case path from input to output,  $G_{del}$
  - More accurate delay measurement per gate
    - Propagation delay = intrinsic (*internal*) delay + extrinsic (*external*) delay
    - Relative prop delay,  $P_{del} = \# \text{ inputs to gate (intrinsic)} + \# \text{ loads (extrinsic)}$

# Circuit Analysis Example

- From previous example:

$$Z = (A + B')C + A'BC'$$

➤ # gates:  $G = 7$

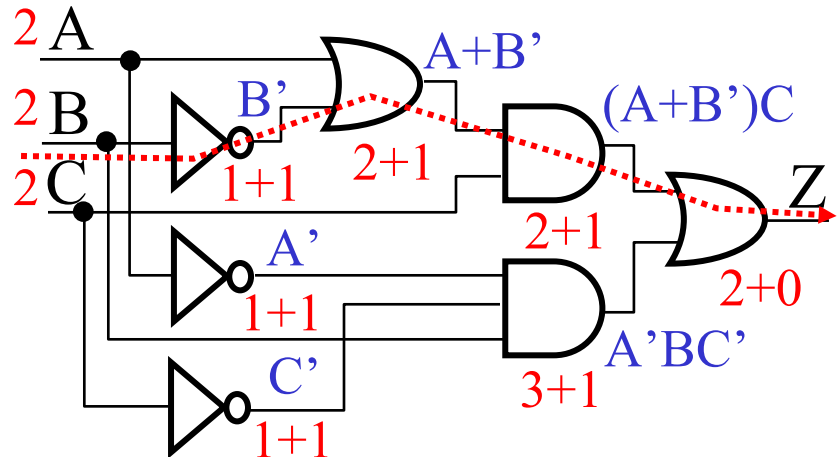
➤ # gate I/O:  $G_{IO} = 19$

➤ Gate delay:  $G_{del} = 4$

- worst case path:  $B \rightarrow Z$

➤ Prop delay:  $P_{del} = 12$

- worst case path:  $B \rightarrow Z$



# Circuit Optimization

- Obviously we want smallest, fastest circuit
- Some Basic Goals:
  - Minimizing # product terms minimizes # of AND gates and # inputs to OR gate in a 2-level SOP (AND-OR) representation
  - Minimizing # literals in each product term minimizes # inputs to its AND gate
- We can use postulates & theorems, but...
  - It would be nice to find a more reliable procedure