Digital Logic Design

Digital Signals

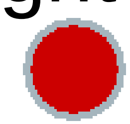
- Digital Signals have two basic states:

  1 (logic “high”, or H, or “on”)

  0 (logic “low”, or L, or “off”)

- Digital values are in a *binary* format. Binary means 2 states.

- A good example of binary is a light (only on or off)



ON

- Strings of binary digits ("bits")
  - One bit can store a number from 0 to 1
  - *n* bits can store numbers from 0 to 2^n
- Positional representation
- Each digit represents a power of 2

So 101 binary is

$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

or

$1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 5$

Converting Binary to Decimal

- multiply digit by power of 2
- Just like a decimal number is represented

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

What is 10011100 in decimal?

1	0	0	1	1	1	0	0

128 + 0 + 0 + 16 + 8 + 4 + 0 + 0 = 156

Binary

In Binary, there are only 0's and 1's. These numbers are called "Base-2" **(Example: 010_2)**

Binary to Decimal

Base 2 = Base 10
000 = 0
001 = 1
010 = 2
011 = 3
100 = 4
101 = 5
110 = 6
111 = 7

We count in "Base-10" (0 to 9)

Binary as a Voltage

Voltages are used to represent logic values:

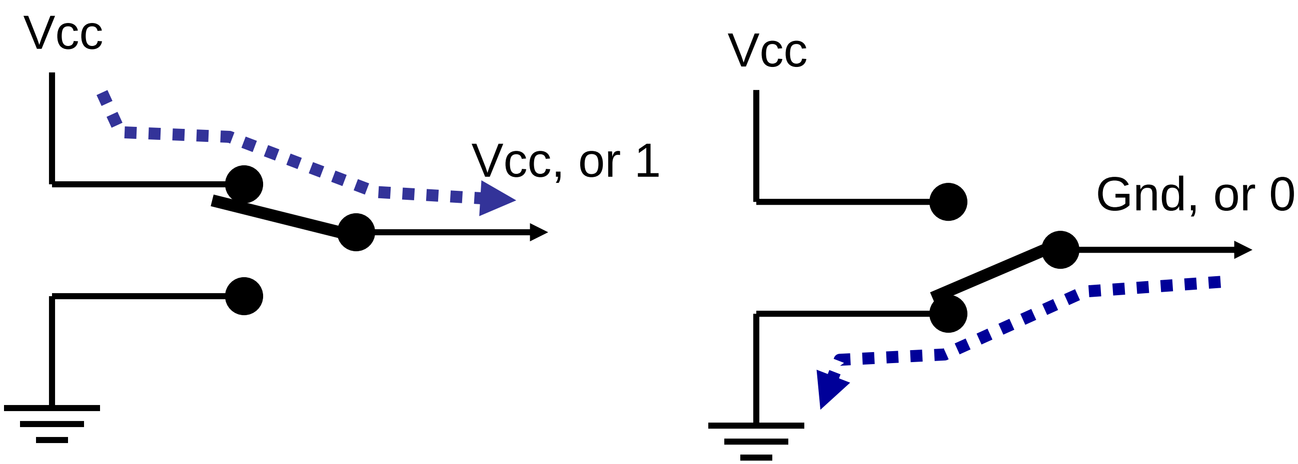
A voltage present (called Vcc or Vdd) = 1

Zero Volts or ground (called gnd or Vss) = 0

A simple switch can provide a logic high or a logic low.

A Simple Switch

- Here is a simple switch used to provide a logic value:



There are other ways to connect a switch.

Digital Logic

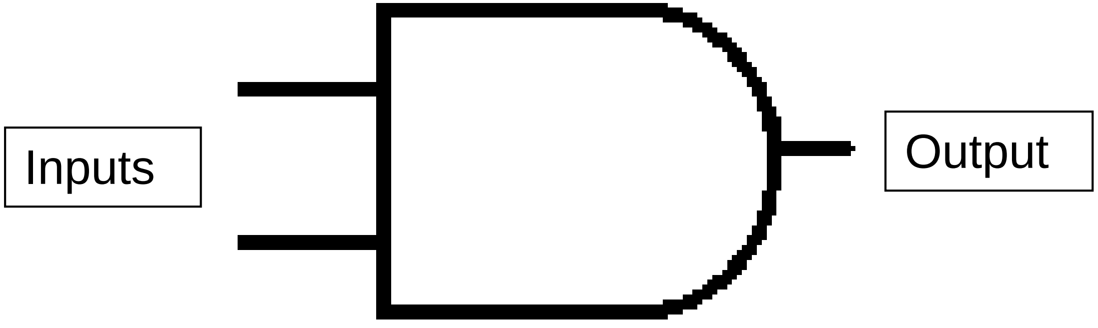
- Basic Digital logic is based on 3 primary functions (the basic gates):
  - AND
  - OR
  - NOT

The AND function

- The AND function:
  - If all the inputs are high the output is high
  - If any input is low, the output is low

- “If this input AND this input are high, the output is high”

AND Logic Symbol



If both inputs are 1, the output is 1

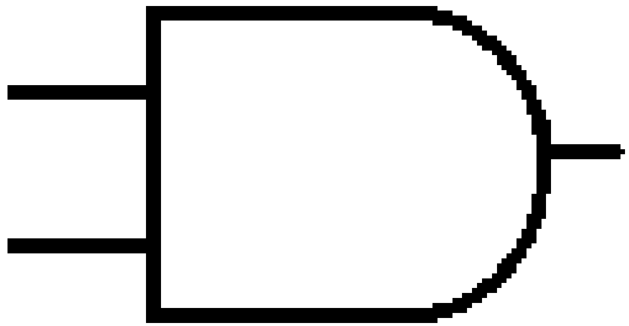
If any input is 0, the output is 0

AND Logic Symbol

Inputs

0

0



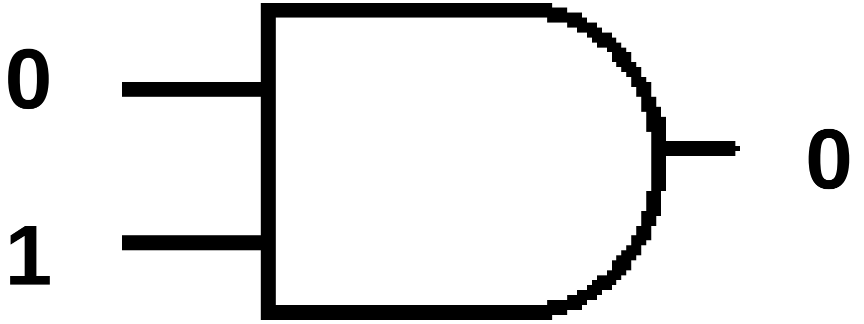
0

Output

Determine the output

AND Logic Symbol

Inputs



Output

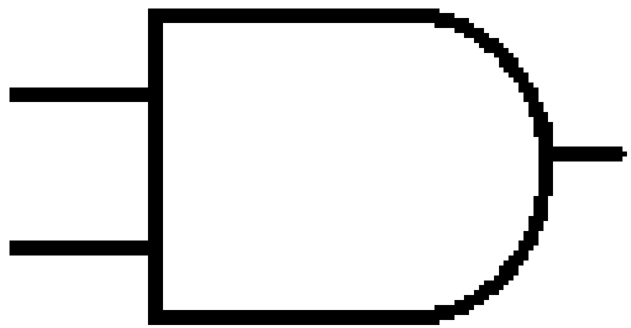
Determine the output

AND Logic Symbol

Inputs

1

1



1

Output

Determine the output

AND Truth Table

- To help understand the function of a digital device, a Truth Table is used:

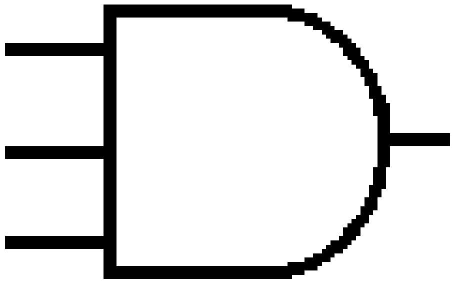
Every possible input combination

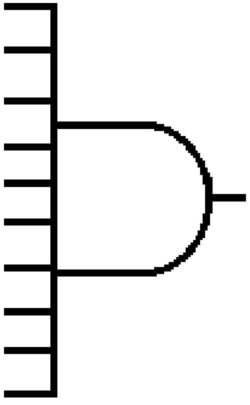
Input		Output
0	0	0
0	1	0
1	0	0
1	1	1

AND Function

AND Gates

- It is possible to have AND gates with more than 2 inputs. The same logic rules apply – “if any input…”

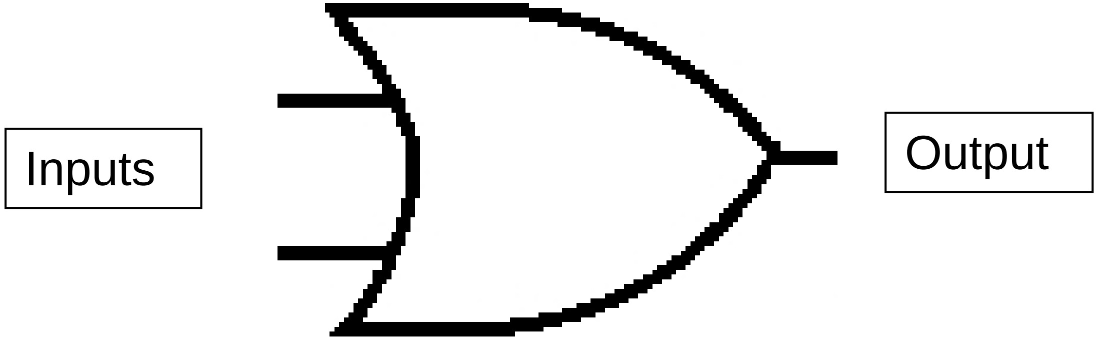




The OR function

- The OR function:
  - if any input is high, the output is high
  - if all inputs are low, the output is low
- “If this input OR this input is high, the output is high”

OR Logic Symbol

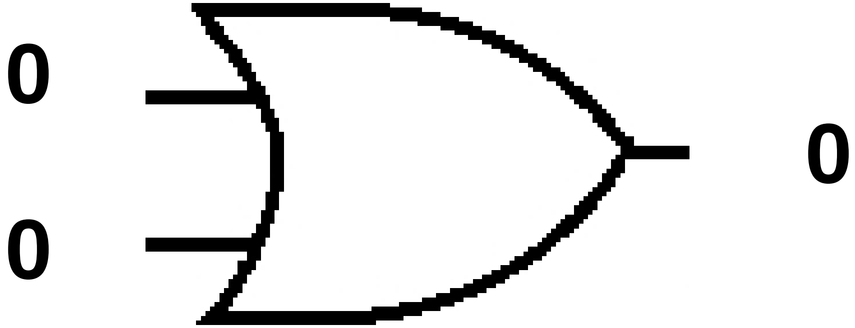


If any input is 1, the output is 1

If all inputs are 0, the output is 0

OR Logic Symbol

Inputs



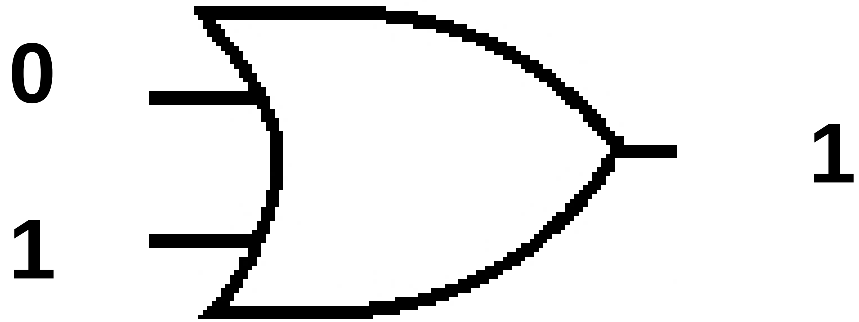
Output

Determine the output

Animated Slide

OR Logic Symbol

Inputs



Output

Determine the output

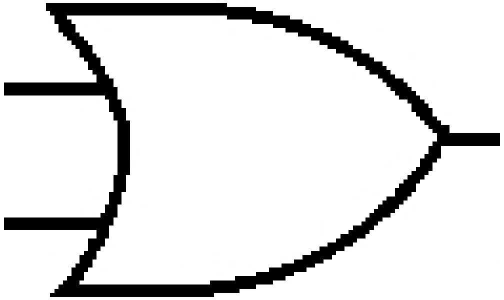
Animated Slide

OR Logic Symbol

Inputs

1

1



1

Output

Determine the output

Animated Slide

OR Truth Table

- Truth Table

Input		Output
0	0	0
0	1	1
1	0	1
1	1	1

OR Function

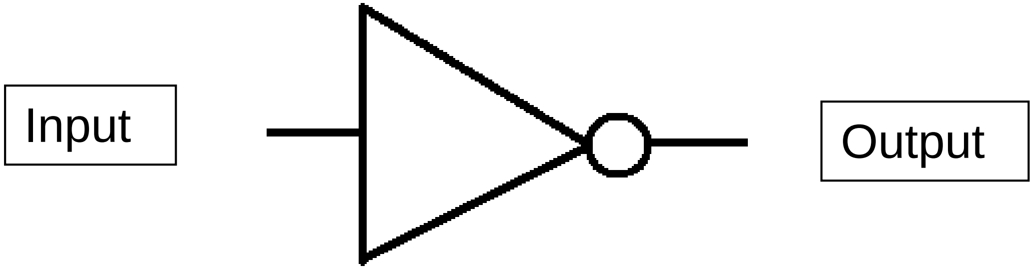
The NOT function

- The NOT function:
  - If any input is high, the output is low
  - If any input is low, the output is high

- “The output is the opposite state of the input”

- The NOT function is often called INVERTER

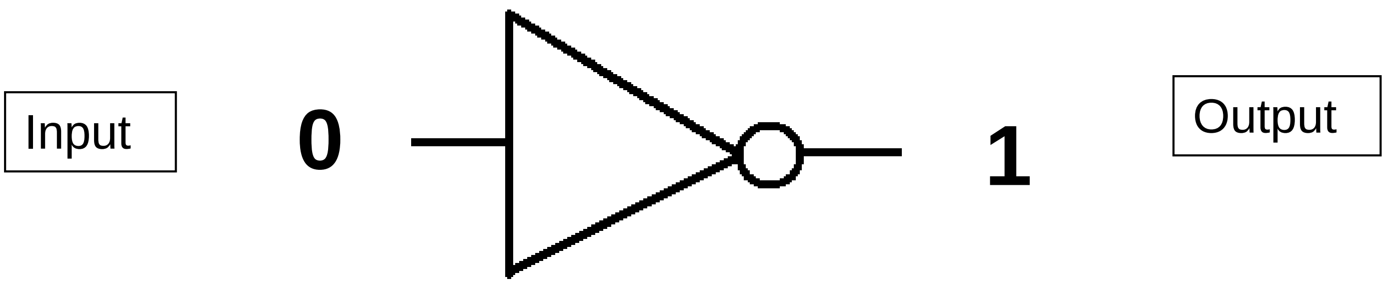
NOT Logic Symbol



If the input is 1, the output is 0

If the input is 0, the output is 1

NOT Logic Symbol

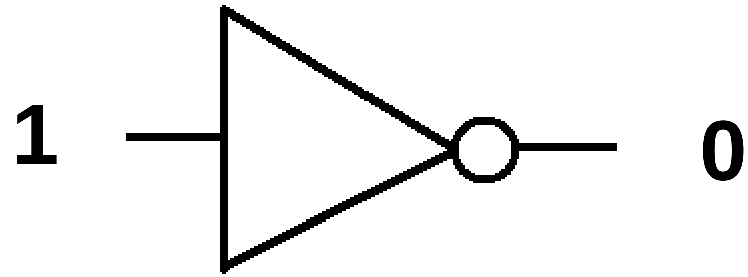


Determine the output

Animated Slide

NOT Logic Symbol

Input



Output

Determine the output

Animated Slide

NOT Truth Table

- Truth Table

Input	Output
0	1
1	0

NOT Function

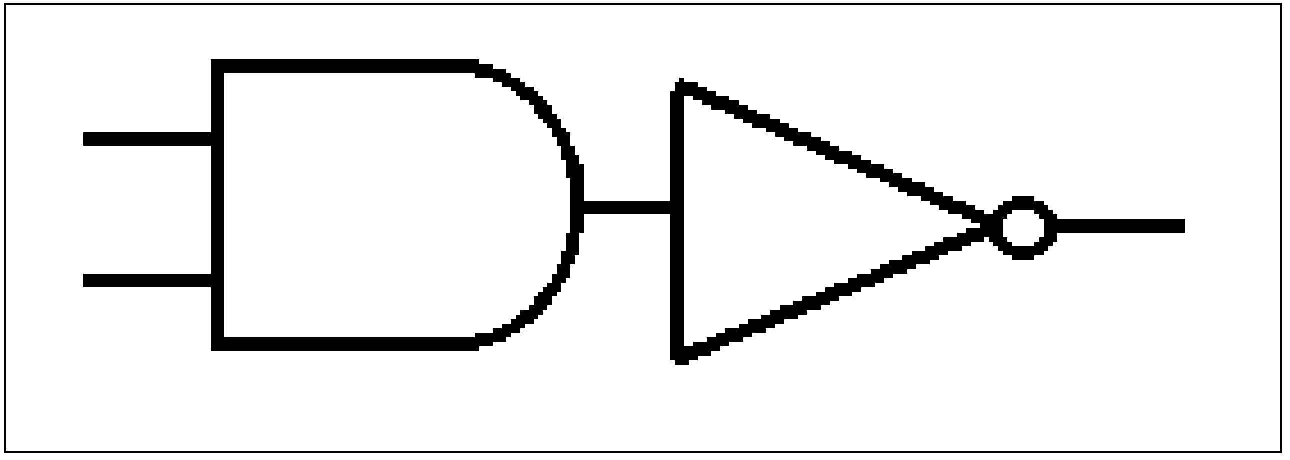
Combinational Logic

- A circuit that utilizes more that 1 logic function has Combinational Logic.

- As an example, if a circuit has an AND gate connected to an Inverter gate, this circuit has combinational logic.

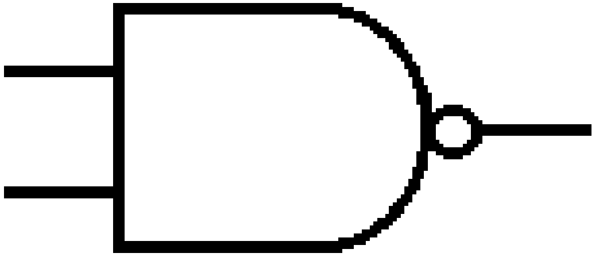
Combinational logic

- How would your describe the output of this combinational logic circuit?



NAND Gate

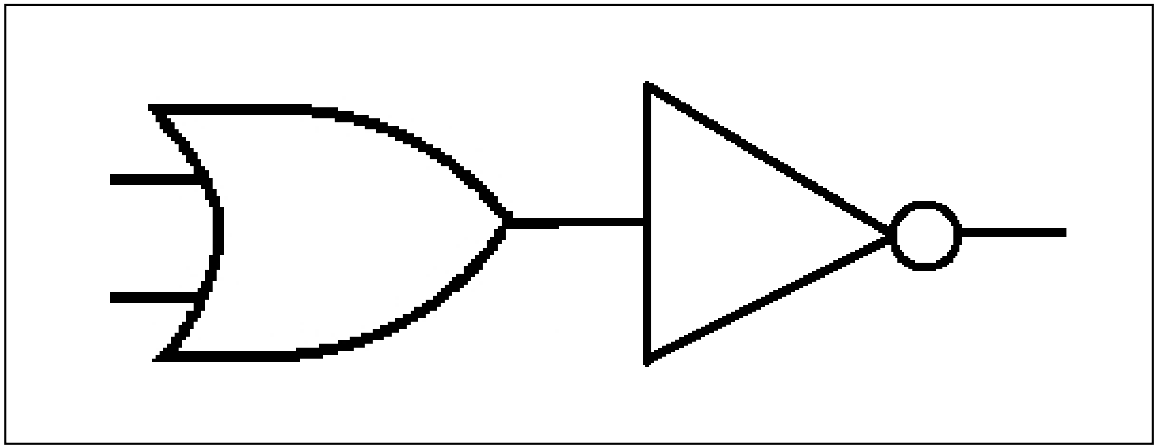
- The NAND gate is the combination of an NOT gate with an AND gate.



The Bubble in front of the gate is an inverter.

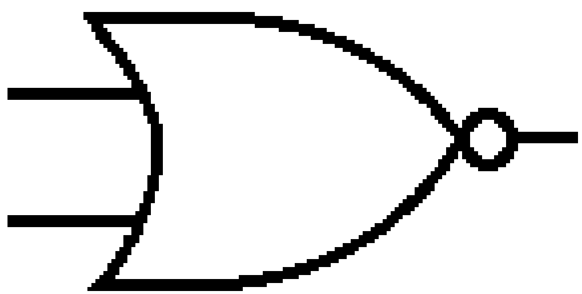
Combinational logic

- How would your describe the output of this combinational logic circuit?



NOR gate

- The NOR gate is the combination of the NOT gate with the OR gate.



The Bubble in front of the gate is an inverter.

NAND and NOR gates

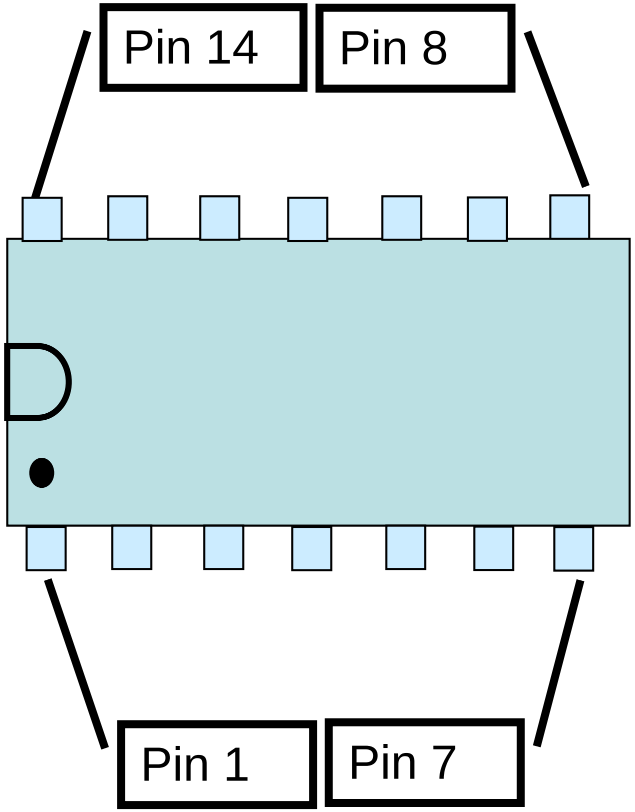
- The NAND and NOR gates are very popular as they can be connected in more ways that the simple AND and OR gates.

Basic Digital Chips

- Digital Electronics devices are usually in a chip format.
- The chip is identified with a part number or a model number.
- A standard series starts with numbers 74, 4, or 14.
  - 7404 is an inverter
  - 7408 is an AND
  - 7432 is an OR
  - 4011B is a NAND

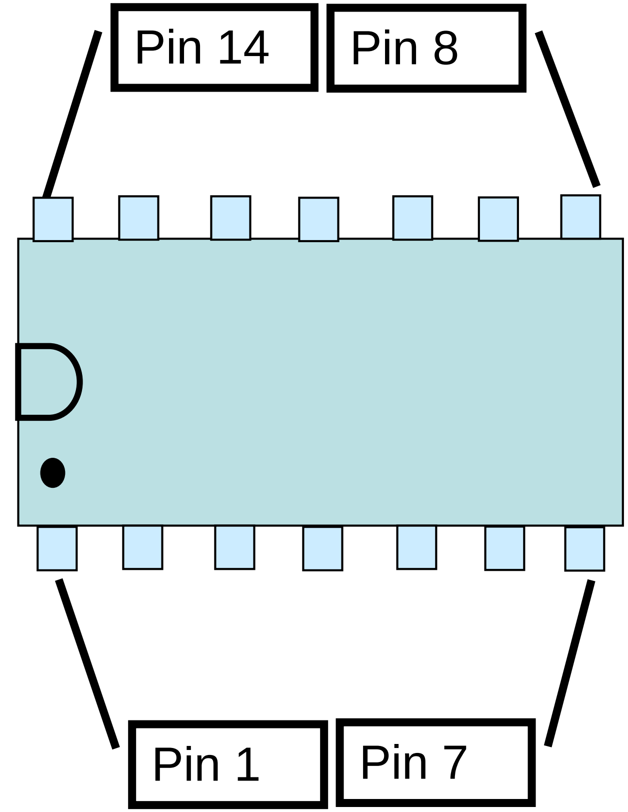
Chips

- Basic logic chips often come in 14-pin packages.
- Package sizes and styles vary.
- Pin 1 is indicated with a dot or half-circle
- Numbers are read counter-clockwise from pin 1 (viewed from the top)

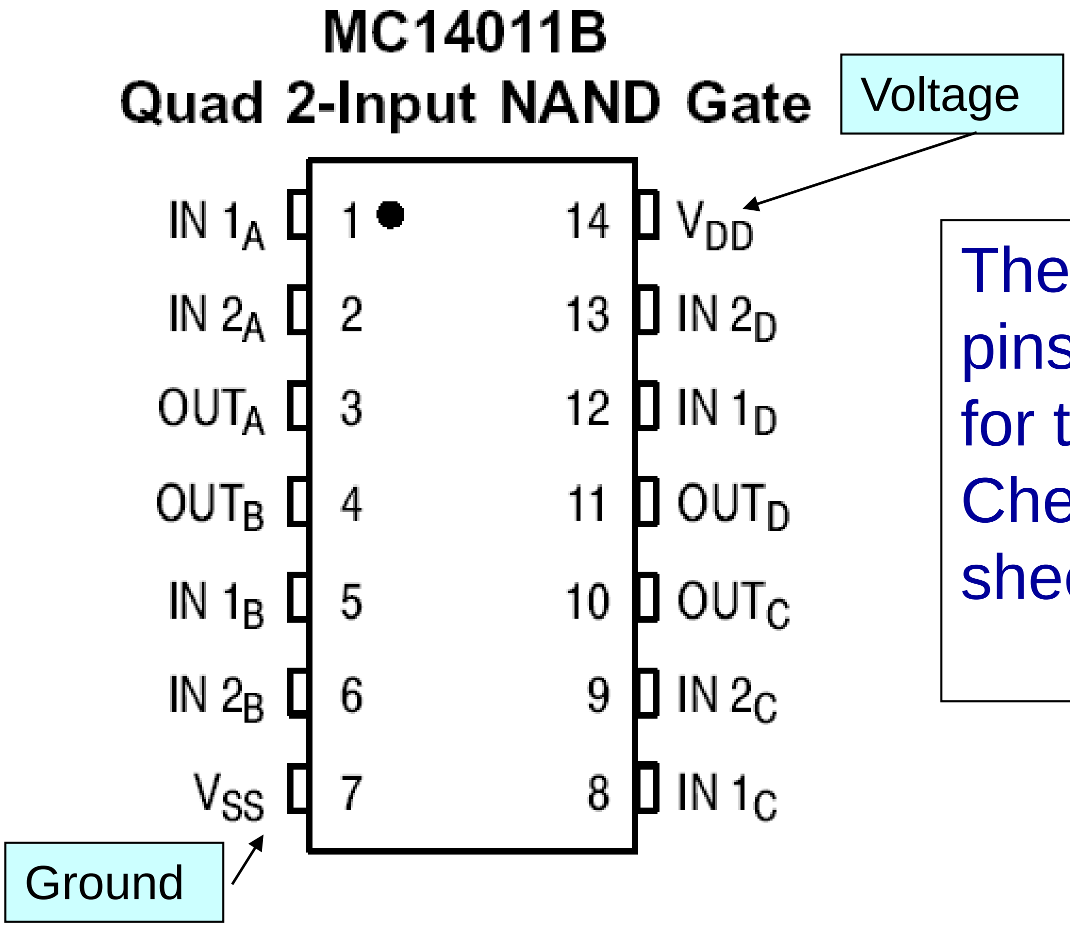


Chips

- Chips require a voltage to function
- Vcc is equal to 5 volts and is typically pin 14
- Ground is typically pin 7

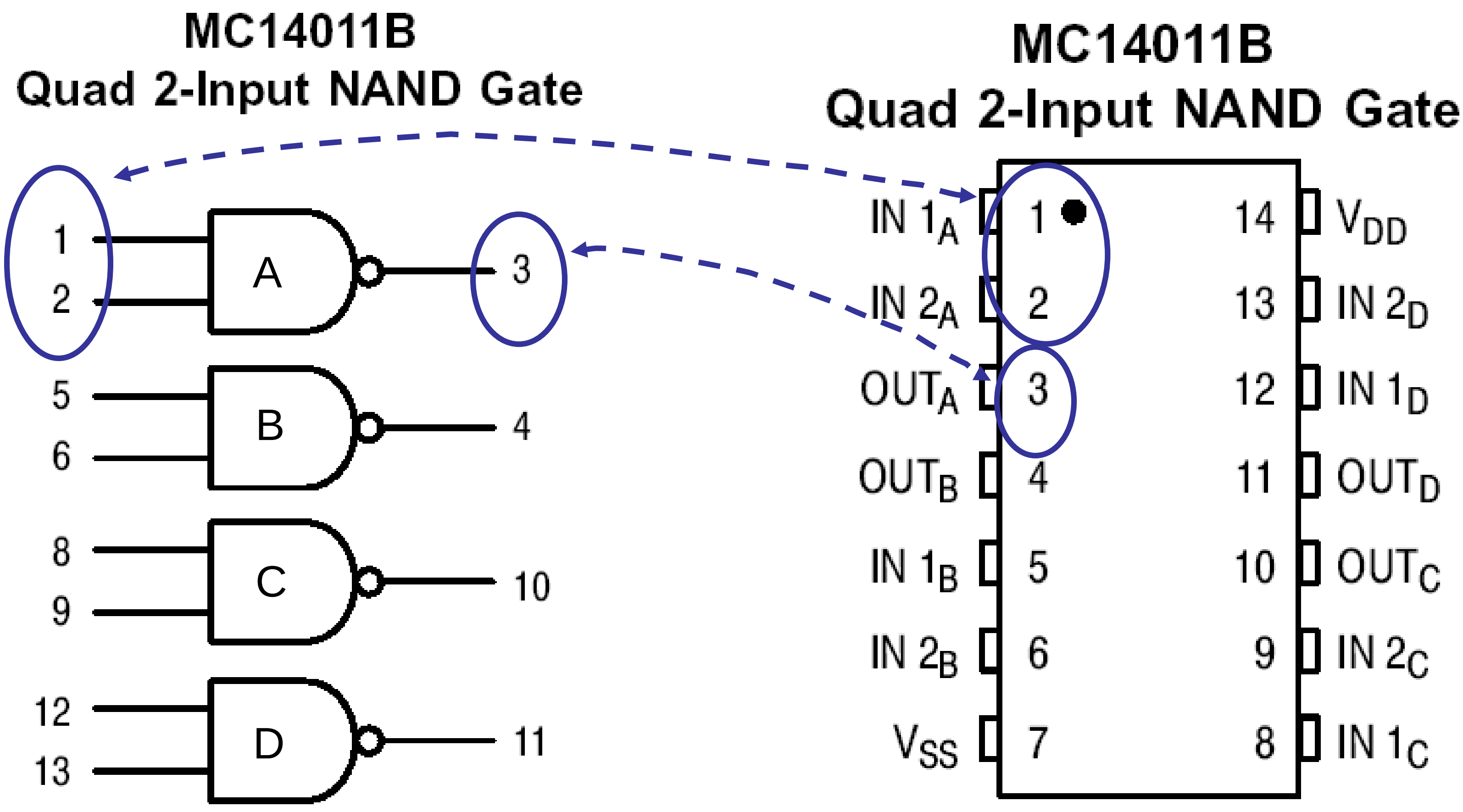


Chips – Specification Sheet

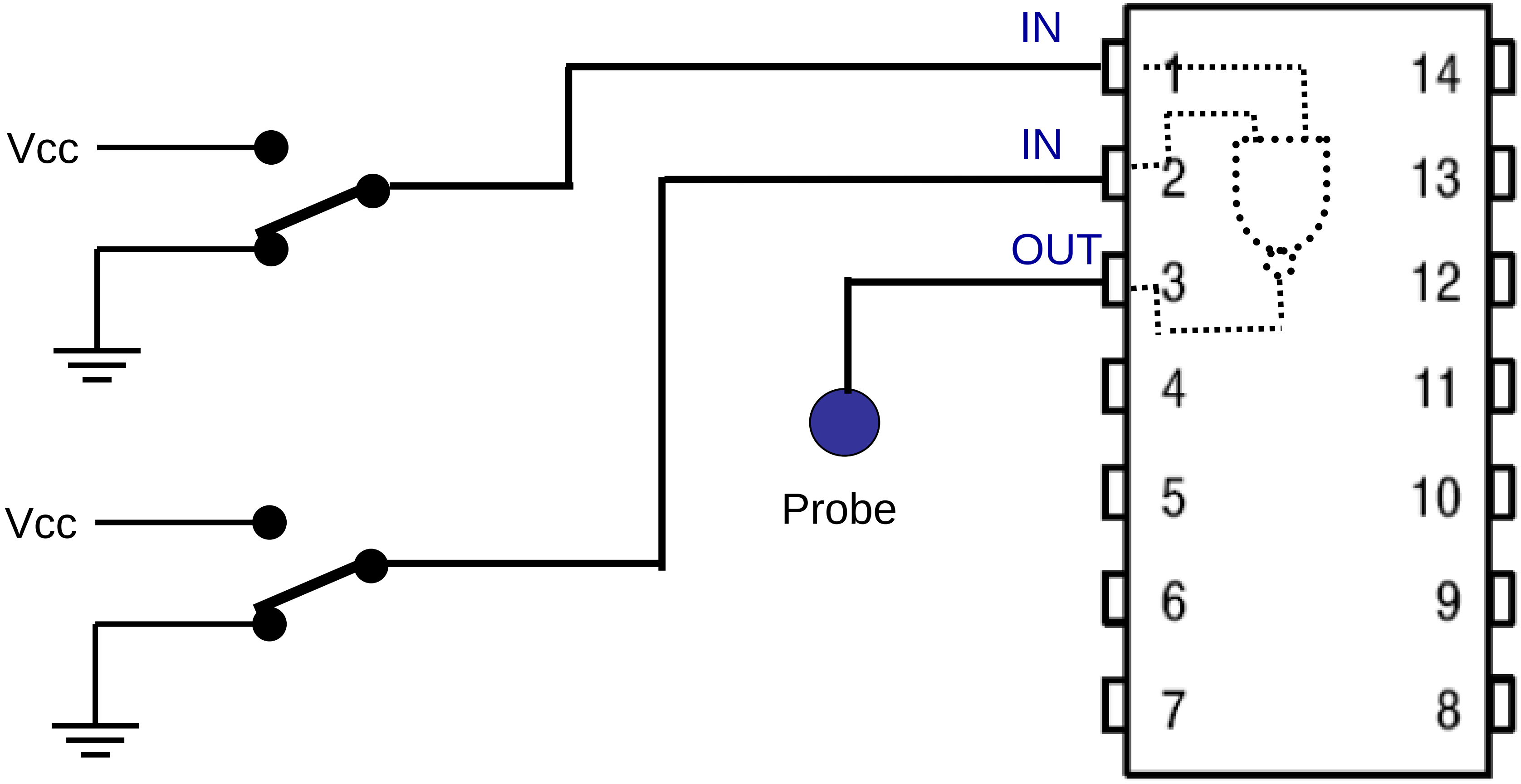


The voltage and ground pins must be connected for the device to function. Check the specification sheet to make sure.

Chips – Specification Sheet

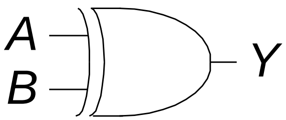


Wiring a chip



More Two-Input Logic Gates

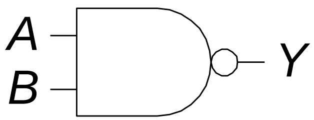
XOR



$$Y = A \oplus B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

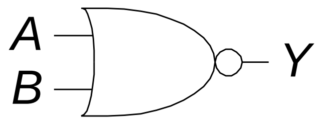
NAND



$$Y = \overline{AB}$$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

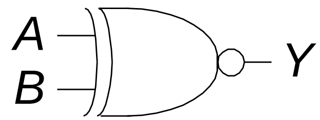
NOR



$$Y = \overline{A + B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

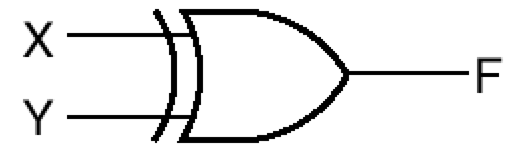


$$Y = \overline{A \oplus B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Exclusive OR

- Exclusive OR
- Symbol is ⊕
  - Plus in a circle

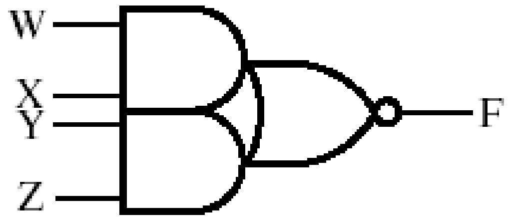


$$F = X\overline{Y} + \overline{X}Y$$
$$= X \oplus Y$$

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

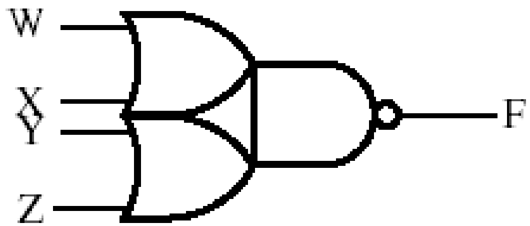
Others

AND-OR-INVERT
(AOI)



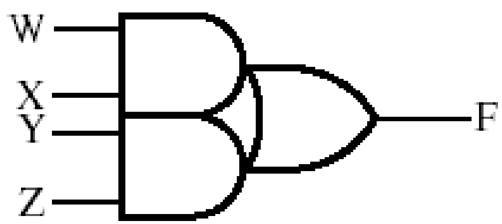
$F = \overline{WX + YZ}$

OR-AND -INVERT
(OAI)



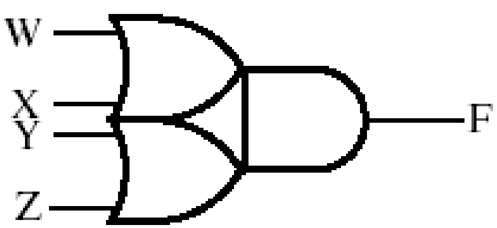
$F = \overline{(W + X)(Y + Z)}$

AND-OR
(AO)



$F = WX + YZ$

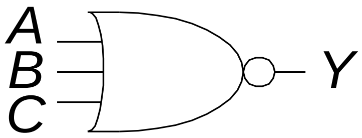
OR-AND
(OA)



$F = (W + X)(Y + Z)$

Multiple-Input Logic Gates

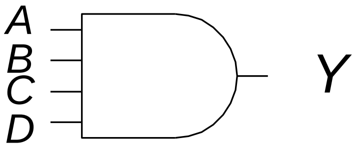
NOR3



$$Y = \overline{A+B+C}$$

A	B	C	Y
0	0	0	1
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	0

AND4

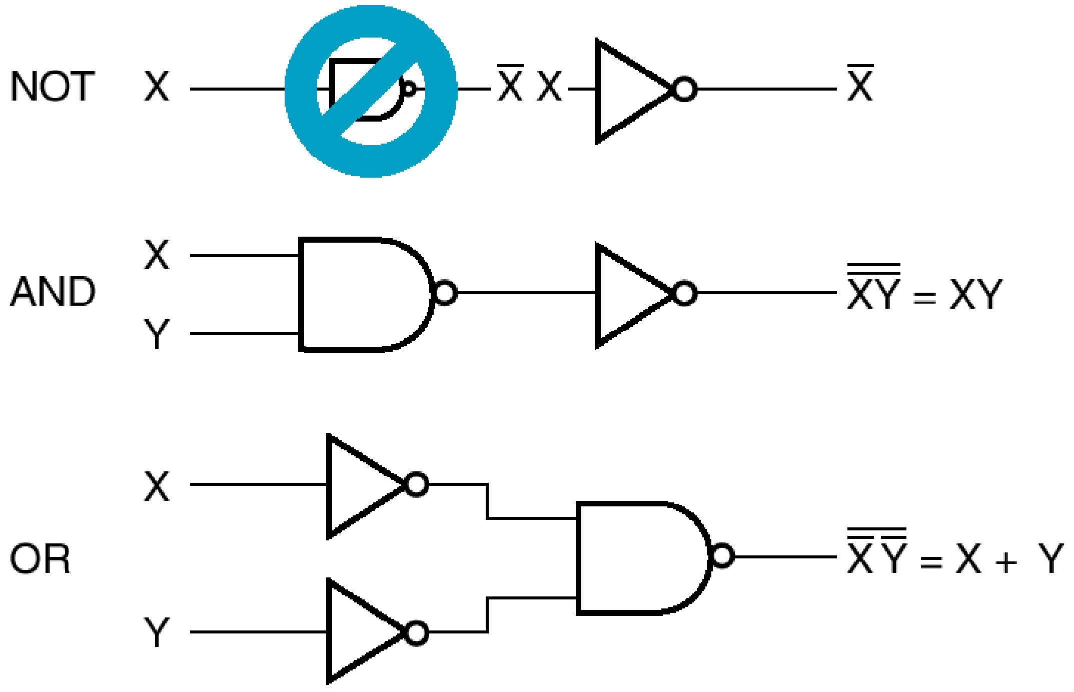


$$Y = ABCD$$

A	B	C	Y
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

NAND is Universal

* Can express any Boolean Function
* Equivalents below



Using NAND as Invert-OR

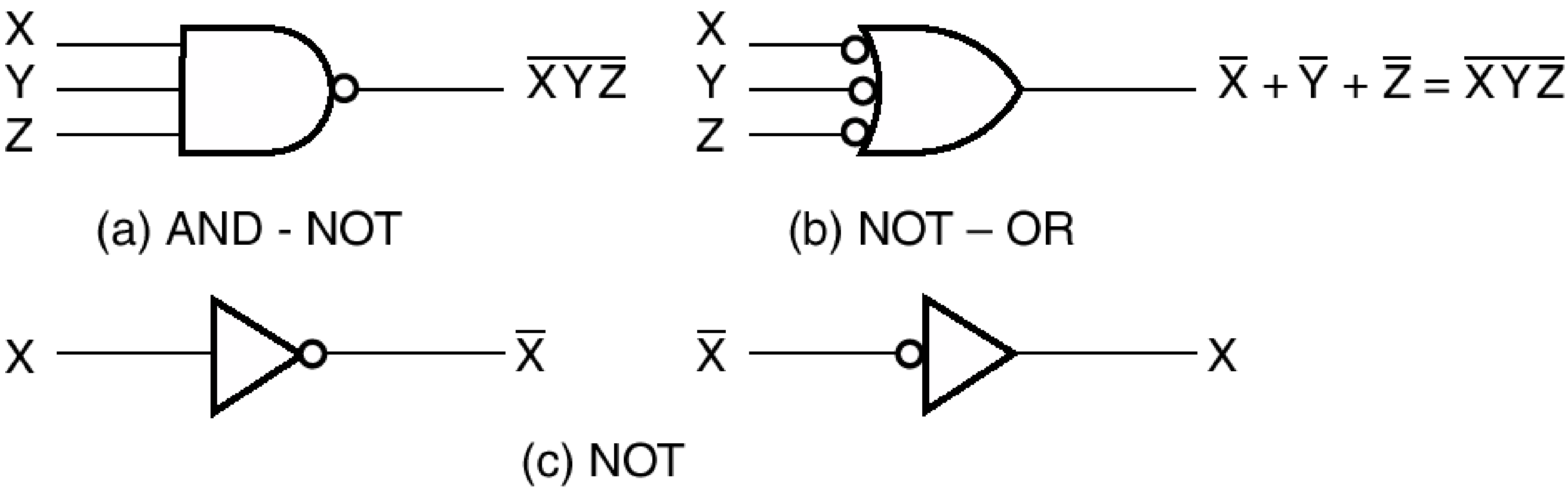


Fig. 2-28 Alternative Graphics Symbols for NAND and NOT Gates

- Also reverse inverter diagram for clarity

NOR Also Universal

- Dual of NAND

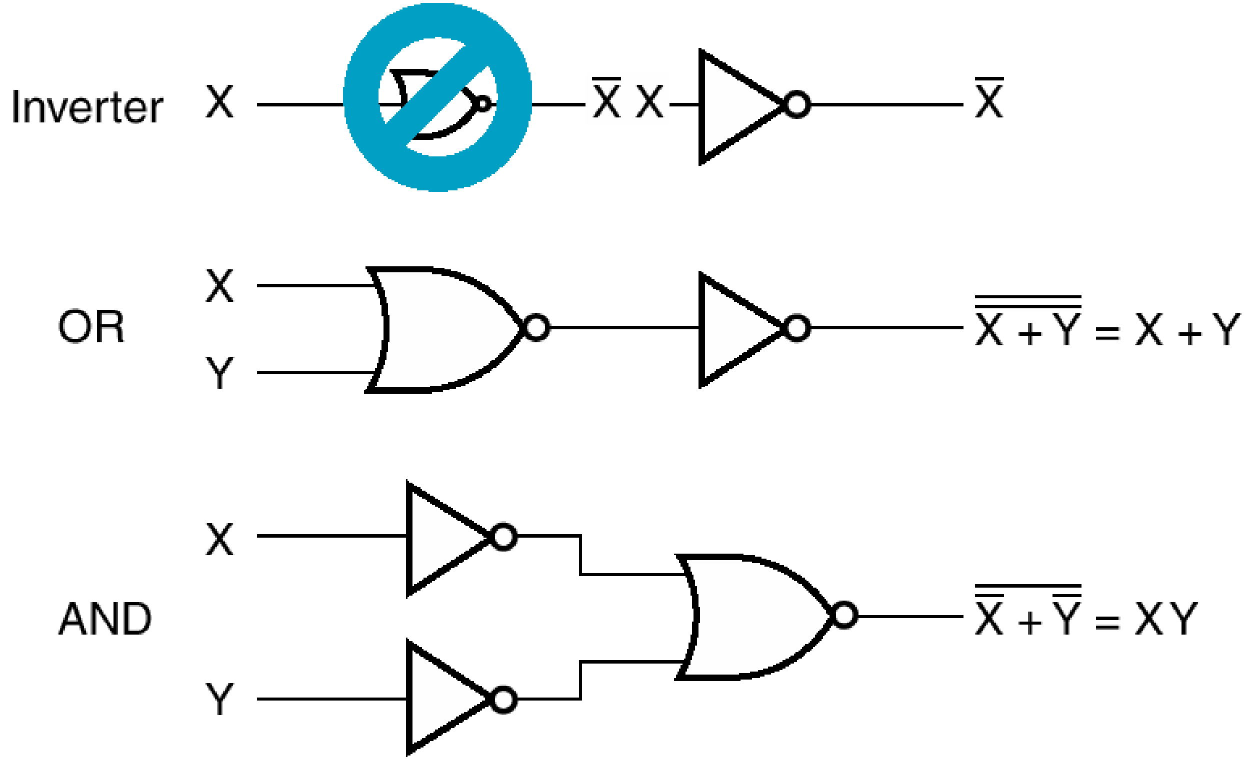


Fig. 2-33 Logic Operations with NOR Gates

Representation: Schematic

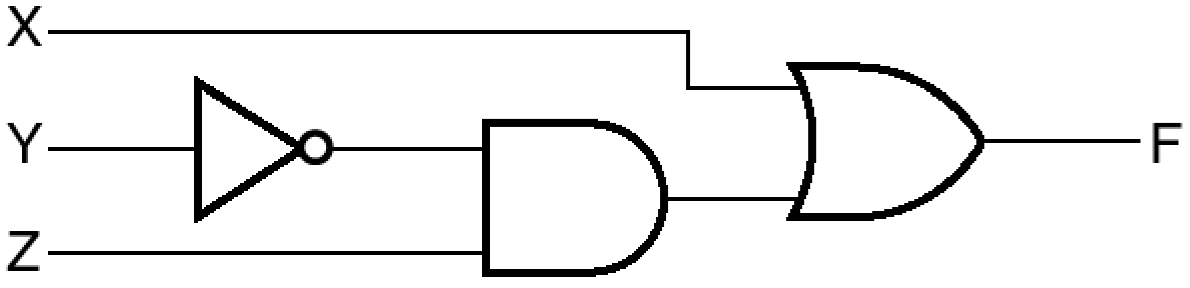


Fig. 2-3 Logic Circuit Diagram for $F = X + \overline{Y}Z$

Representation: Boolean Algebra

$$F = X + \overline{Y}Z$$

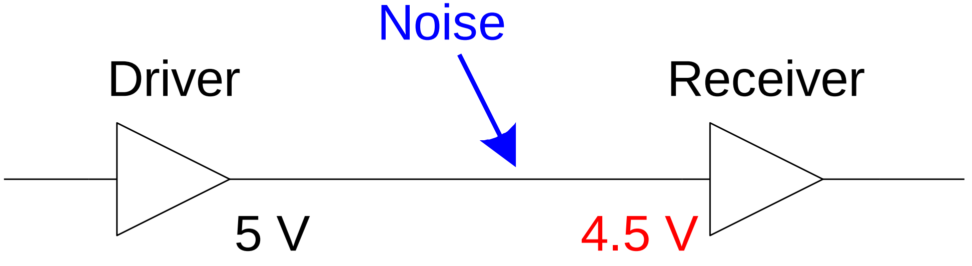
- 2^n rows:
- where n # of variables

TABLE 2-2
Truth Table
for the Function $F = X + \overline{Y}Z$

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

What is Noise?

- Anything that degrades the signal
  - E.g., resistance, power supply noise, coupling to neighboring wires, etc.
- Example: a gate (driver) could output a 5 volt signal but, because of resistance in a long wire, the signal could arrive at the receiver with a degraded value, for example, 4.5 volts



The Static Discipline

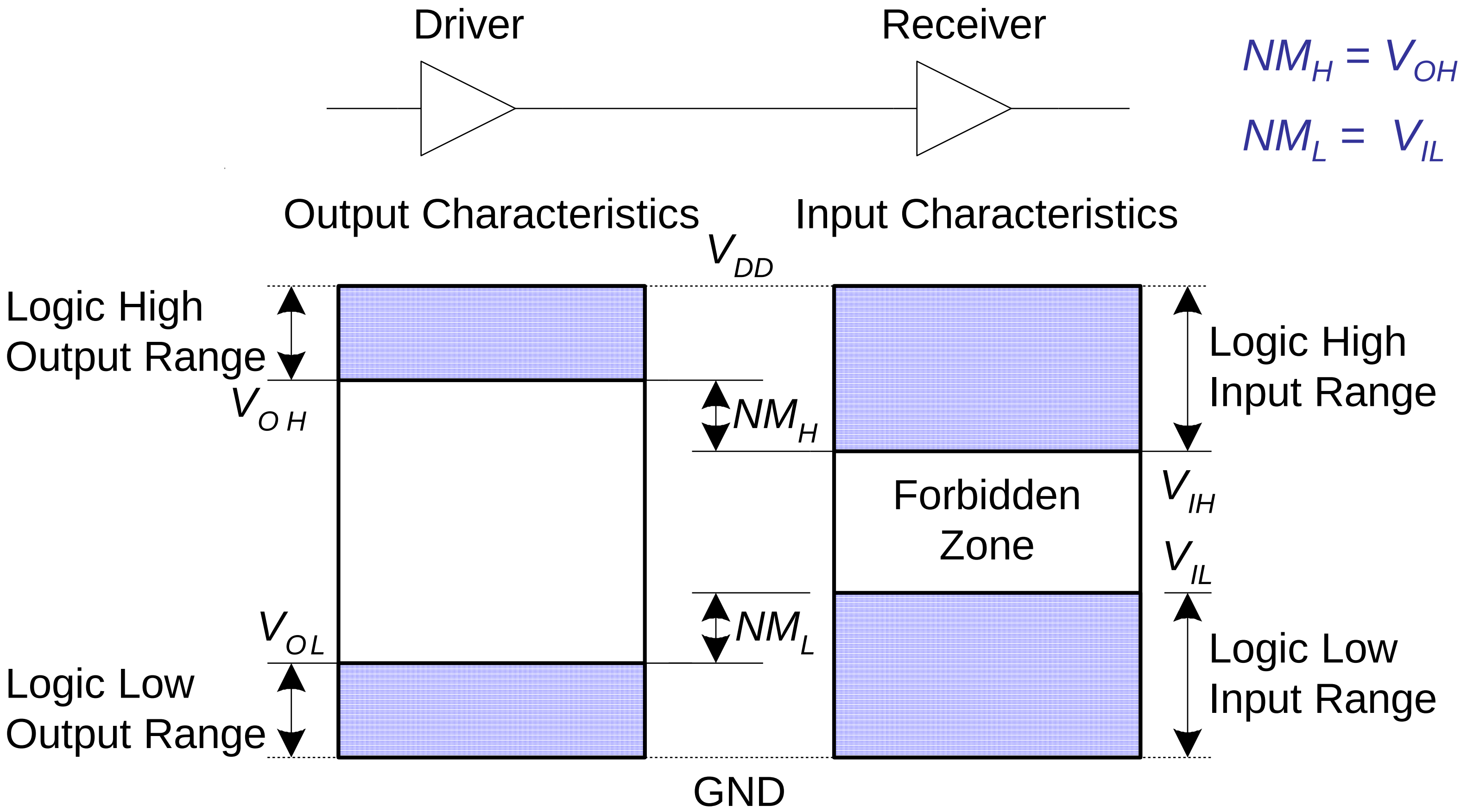
* Given logically valid inputs, every circuit element must produce logically valid outputs

* Discipline ourselves to use limited ranges of voltages to represent discrete values

Noise Margins

$$NM_H = V_{OH} - V_{IH}$$

$$NM_L = V_{IL} - V_{OL}$$



Logic Family Examples

Logic Family	V_{DD}	V_{IL}	V_{IH}	V_{OL}	V_{OH}
TTL	5 (4.75 - 5.25)	0.8	2.0	0.4	2.4
CMOS	5 (4.5 - 6)	1.35	3.15	0.33	3.84
LVTTL	3.3 (3 - 3.6)	0.8	2.0	0.4	2.4
LVCMOS	3.3 (3 - 3.6)	0.9	1.8	0.36	2.7

Identities

- Use identities to manipulate functions
- You can use distributive law …

$$X + YZ = (X + Y)(X + Z)$$

… to transform from $F = X + \bar{Y}Z$

to

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

Table of Identities

TABLE 2-3
Basic Identities of Boolean Algebra

1.	$X+0=X$	2.	$X \cdot 1 = X$	
3.	$X+1=1$	4.	$X \cdot 0 = 0$	
5.	$X+X=X$	6.	$X \cdot X = X$	
7.	$X+\overline{X}=1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}}=X$			
10.	$X+Y=Y+X$	11.	$XY=YX$	Commutative
12.	$X+(Y+Z)=(X+Y)+Z$	13.	$X(YZ)=(XY)Z$	Associative
14.	$X(Y+Z)=XY+XZ$	15.	$X+YZ=(X+Y)(X+Z)$	Distributive
16.	$\overline{X+Y}=\overline{X}\cdot\overline{Y}$	17.	$\overline{X \cdot Y}=\overline{X}+\overline{Y}$	DeMorgan's

Duals

- Left and right columns are *duals*
- Replace AND and OR, 0s and 1s

TABLE 2-3
Basic Identities of Boolean Algebra

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

Single Variable Identities

1. $X + 0 = X$
2. $X \cdot 1 = X$
3. $X + 1 = 1$
4. $X \cdot 0 = 0$
5. $X + X = X$
6. $X \cdot X = X$
7. $X + \overline{X} = 1$
8. $X \cdot \overline{X} = 0$
9. $\overline{\overline{X}} = X$

Commutativity

- Operation is independent of order of variables

$$10. \quad X + Y = Y + X$$

$$11. \quad XY = YX$$

Associativity

- Independent of order in which we group

$$12. \qquad X + (Y + Z) = (X + Y) + Z$$

$$13. \qquad X(YZ) = (XY)Z$$

- So can also be written as $X + Y + Z$ and XYZ

Distributivity

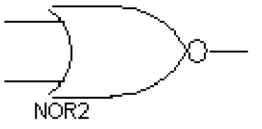
$$14. \quad X(Y+Z) = XY + XZ$$

$$15. \quad X + YZ = (X+Y)(X+Z)$$

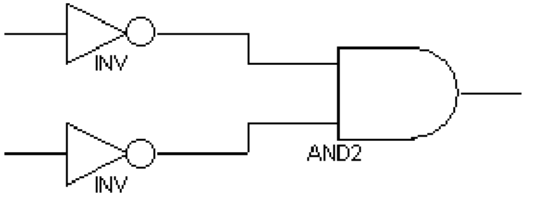
- Can substitute arbitrarily large algebraic expressions for the variables
  - Distribute an operation over the entire expression

DeMorgan’s Theorem

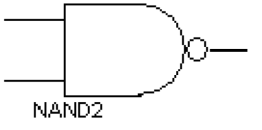
- Used a lot
- NOR ➔ invert, then AND



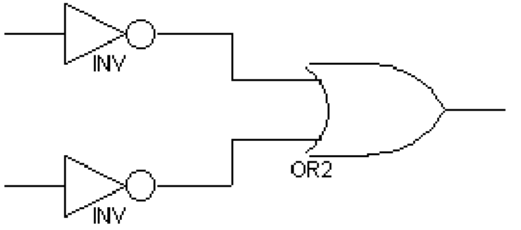
$$16. \qquad \overline{X+Y} = \overline{X} \cdot \overline{Y}$$



- NAND ➔ invert, then OR



$$17. \qquad \overline{X \cdot Y} = \overline{X} + \overline{Y}$$



Truth Tables for DeMorgan's

A)	X	Y	$X + Y$	$\overline{X+Y}$
	0	0	0	1
	0	1	1	0
	1	0	1	0
	1	1	1	0

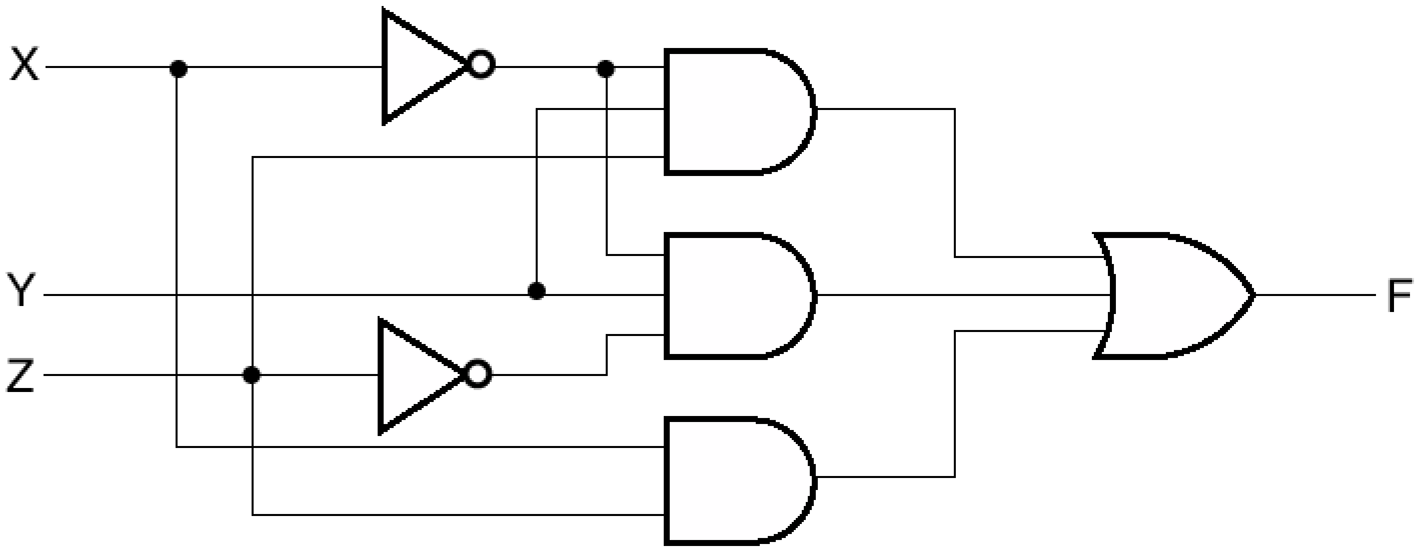
B)	X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \cdot \overline{Y}$
	0	0	1	1	1
	0	1	1	0	0
	1	0	0	1	0
	1	1	0	0	0

16. $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

Algebraic Manipulation

* Consider function

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

Simplify Function

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

Apply $\boxed{14. \quad X(Y+Z) = XY + XZ}$

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

Apply $\boxed{7. \quad X + \overline{X} = 1}$

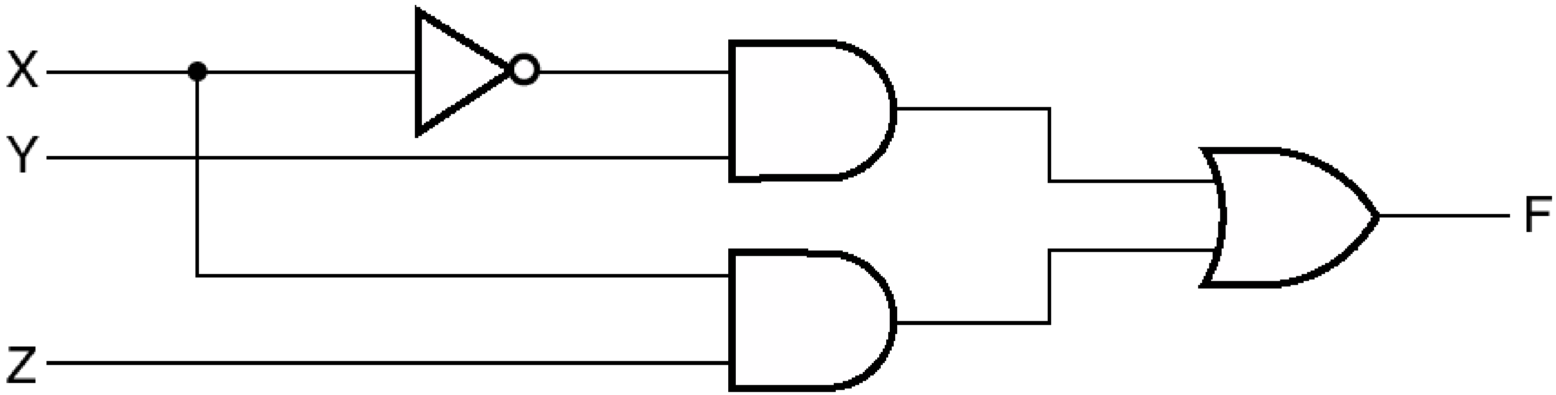
$$F = \overline{X}Y \bullet 1 + XZ$$

Apply $\boxed{2. \quad X \cdot 1 = X}$

$$F = \overline{X}Y + XZ$$

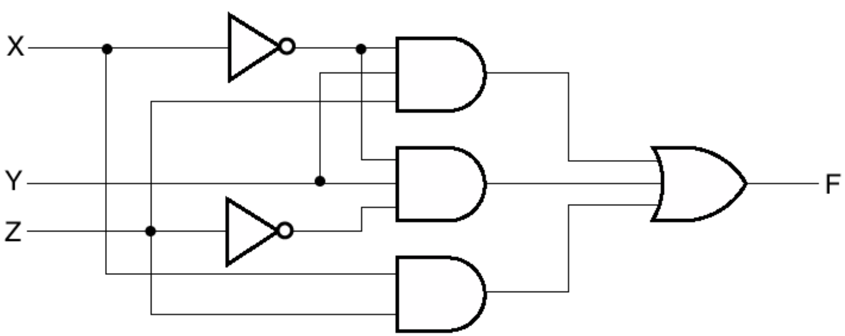
Fewer Gates

$$F = \overline{X}Y + XZ$$



(b) $F = \overline{X}Y + XZ$

Fig. 2-4 Implementation of Boolean Function with Gates



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

Consensus Theorem

$$XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

- The third term is redundant
  - Can just drop
- Proof summary:
  - For third term to be true, Y & Z both must be 1
  - Then one of the first two terms is already 1!

Standard Forms

- Definitions:
  - Product terms – AND $\rightarrow$ $\bar{A}BZ$
  - Sum terms – OR $\rightarrow$ $X + \bar{A}$
  - This is logical product and sum, not arithmetic

Definition: Minterm

- Product term in which all variables appear once (complemented or not)

X	Y	Z	Product Term	Symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\,\overline{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\,\overline{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

Number of Minterms

- For *n* variables, there will be 2^n minterms
- Like binary numbers from 0 to 2^n-1
- Often numbered same way (with decimal conversion)

Maxterms

- Sum term in which all variables appear once (complemented or not)

X	Y	Z	Sum Term	Symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X+Y+Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X+Y+\overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M_7	1	1	1	1	1	1	1	0

Minterm related to Maxterm

- Minterm and maxterm with same subscripts are complements

$$\overline{m}_j = Mj$$

- Example

$$\overline{m}_3 = \overline{\overline{X}YZ} = X + \overline{Y} + \overline{Z} = M_3$$

Sum of Minterms

- OR all of the minterms of truth table row with a 1
  - “ON-set minterms”

X	Y	Z	F	$\overline{F}$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

$$F = \overline{X}\,\overline{Y}Z + X\overline{Y}\,\overline{Z} + X\overline{Y}Z + XY\overline{Z} + XYZ$$

Sum of Products

- Simplifying sum-of-minterms can yield a sum of products
- Difference is each term need not be a minterm
  – i.e., terms do not need to have all variables
- A bunch of ANDs and one OR

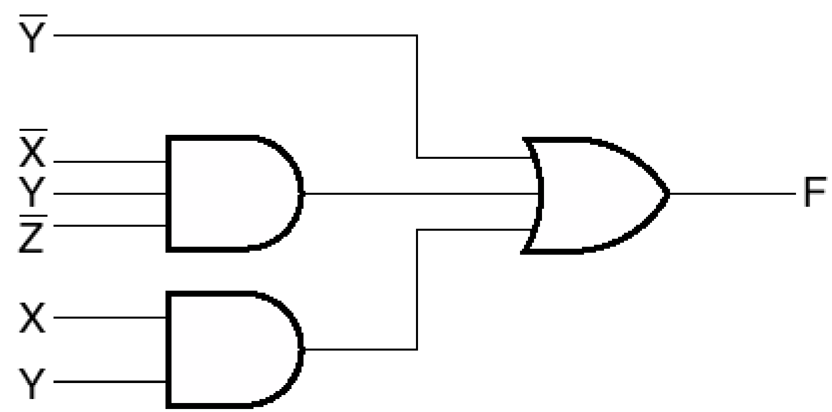
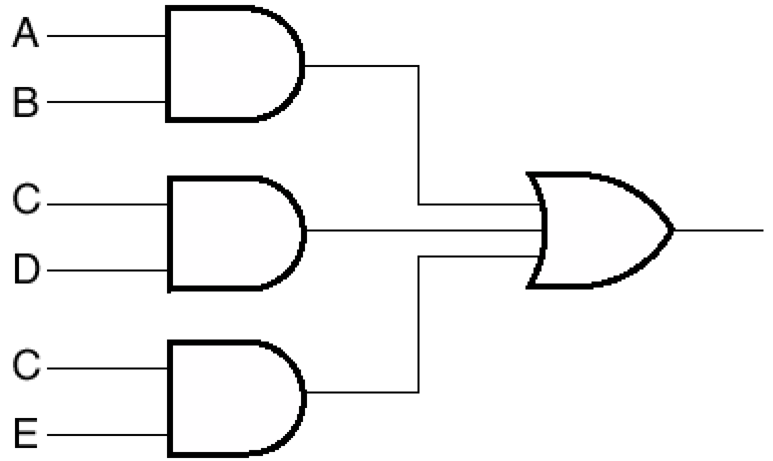


Fig. 2-5 Sum-of-Products Implementation

Two-Level Implementation

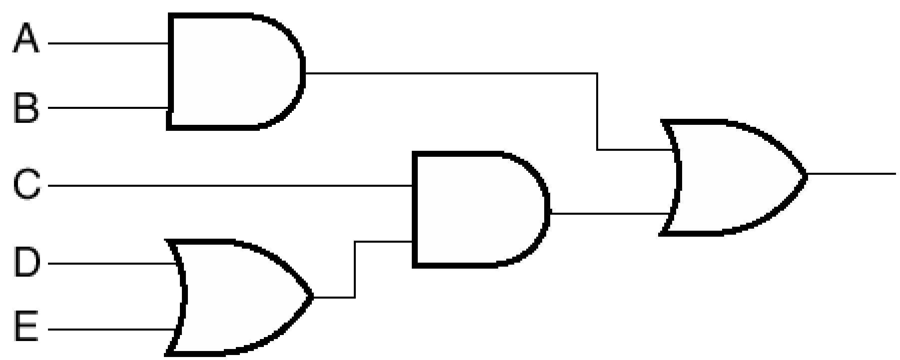
- Sum of products has 2 levels of gates



(b) AB + CD + CE

More Levels of Gates?

- What's best?
  - Hard to answer
  - More gate delays
  - But maybe we only have 2-input gates
    - So multi-input ANDs and ORs have to be decomposed



(a) AB + C (D + E)

Complement of a Function

- Definition: 1s & 0s swapped in truth table
- Mechanical way to derive algebraic form
  - Take the dual
    - Recall: Interchange AND and OR, and 1s & 0s
  - Complement each literal

Complement of F

- Not surprisingly, just sum of the other minterms
  - "OFF-set minterms"
- In this case

$m_1 + m_3 + m_4 + m_6$

X	Y	Z	F	$\overline{F}$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

Product of Maxterms

- Recall that maxterm is true except for its own case
- So M1 is only false for 001

X	Y	Z	Sum Term	Symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X+Y+Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X+Y+\overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M_7	1	1	1	1	1	1	1	0

Product of Maxterms

- Can express F as AND of all rows that should evaluate to 0

$$F = M_1 \bullet M_3 \bullet M_4 \bullet M_6$$

or

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

X	Y	Z	F	$\overline{F}$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

Product of Sums

- Result: another standard form
- ORs followed by AND
  - Terms do not have to be maxterms

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

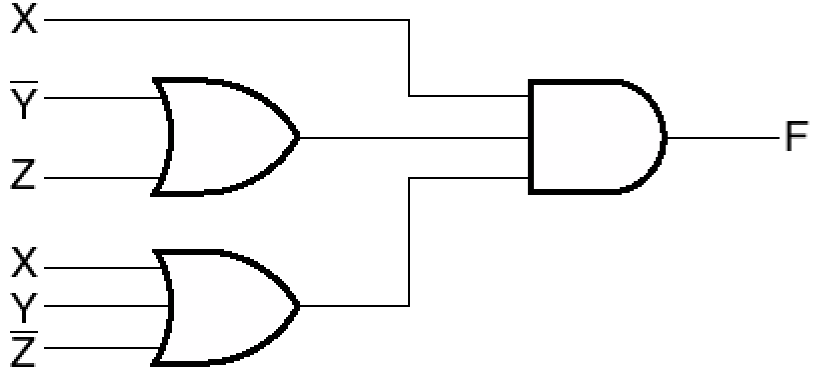


Fig. 2-7 Product-of-Sums Implementation