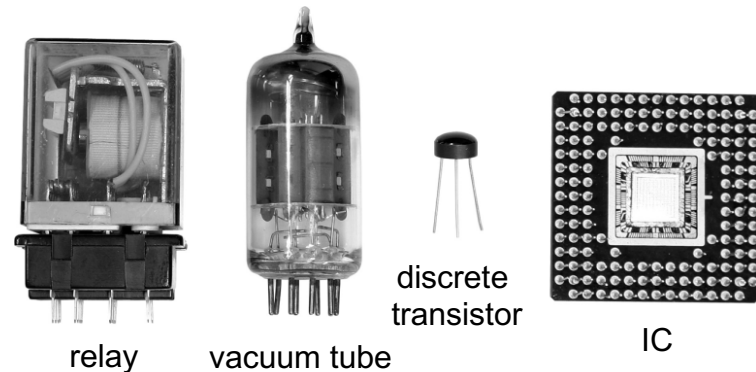
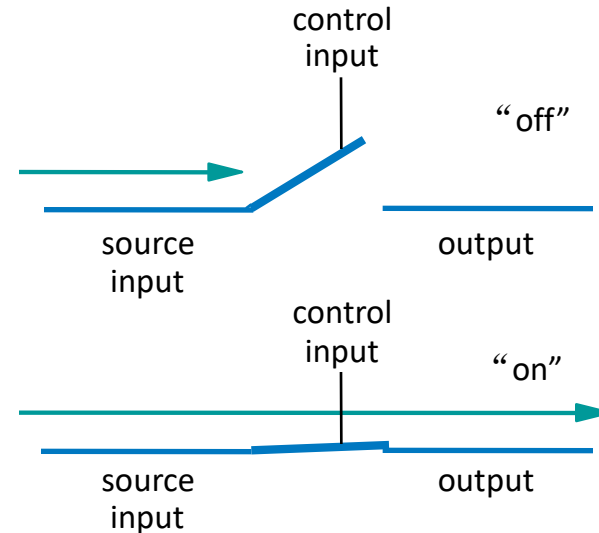


# Topic 2

## Basic Logic Gates

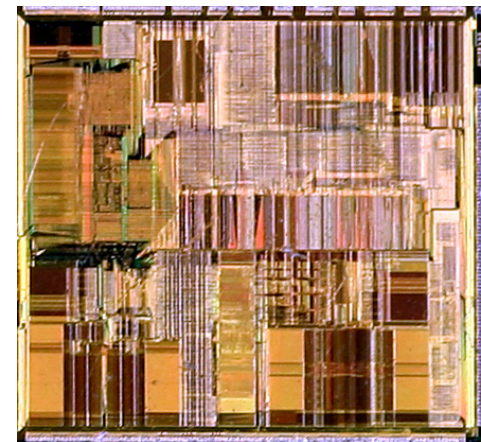
# Electronic Switch – Transistors

- **Transistors are the basis of binary digital circuits**
  - Transistors operate at 2 values  
H / L or  
On / Off or  
1 / 0
- **Evolution of electronic switches**
  - 1930s: Relays
  - 1940s: Vacuum tubes
  - 1950s: Discrete transistor
  - 1960s: Integrated circuits (ICs)
    - Initially just a few transistors on IC
    - Then tens, thousands, millions...



# Moore's Law

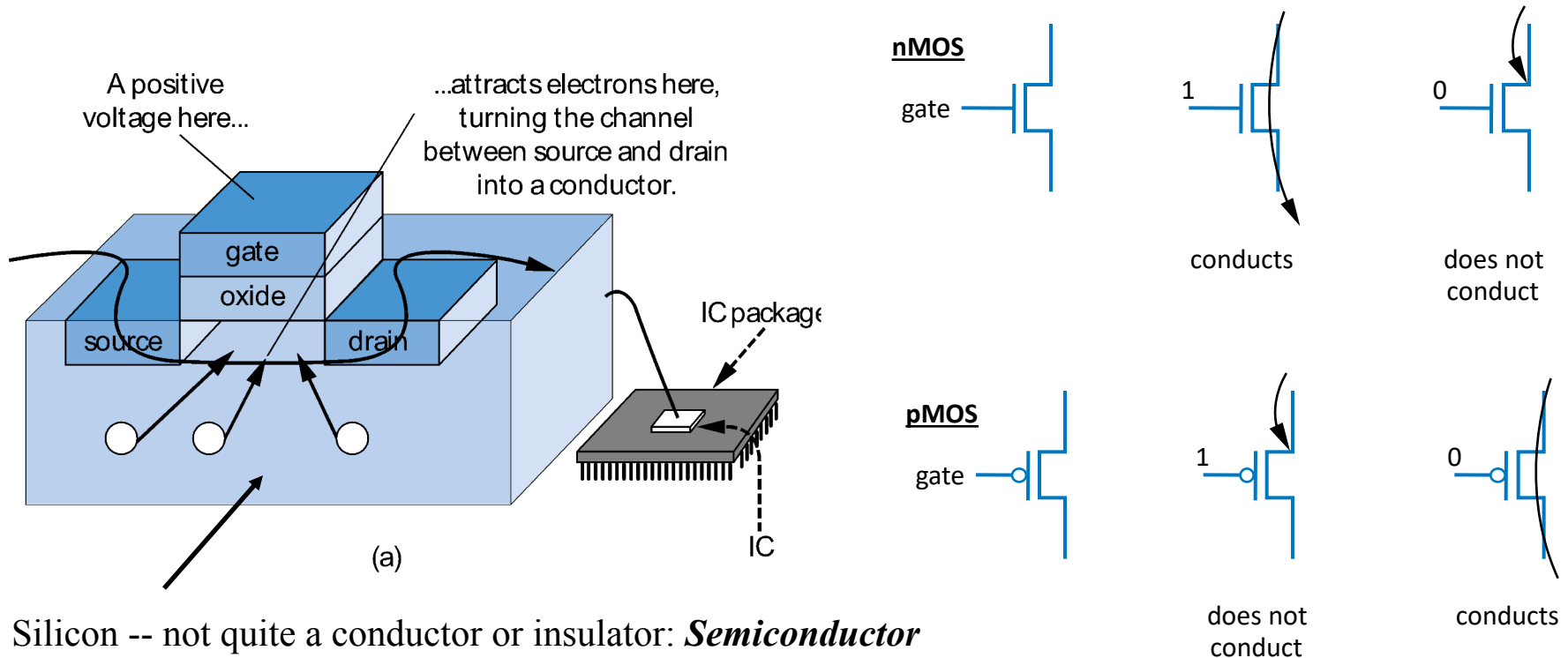
- **IC capacity doubling about every 18 months for several decades**
  - Known as “Moore’s Law” after Gordon Moore, co-founder of Intel
    - Predicted in 1965 predicted that components per IC would double roughly every 18 months or so
  - For a particular number of transistors, the IC shrinks by half every 18 months
    - Enables incredibly powerful computation in incredibly tiny devices
  - Today’s ICs hold *billions* of transistors
    - The first Pentium processor (early 1990s) had only 3 million



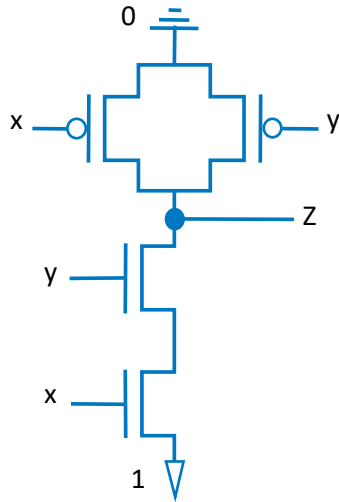
*An Intel Pentium processor IC having millions of transistors*

# CMOS Transistor

- **CMOS – Complementary Metal-Oxide-Semiconductor**
- **Transistors with CMOS technology**



# AND Logic



X	Y	Z = XY
0	0	0
0	1	0
1	0	0
1	1	1

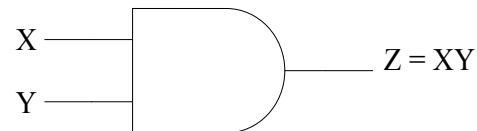
Truth Table

- Definition of AND operation**

$Z = X \bullet Y$  means  $Z = 1$  if and only if both  $X = 1$  and  $Y = 1$ ;

Variable

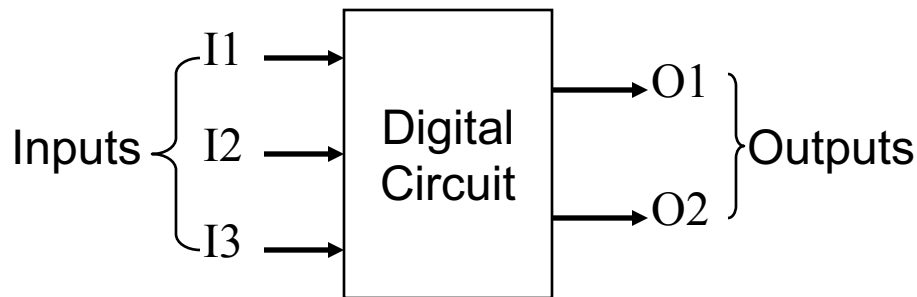
AND operator



2-input AND gate

# Truth Table

- **Truth table creates the relationship between the inputs and outputs**
  - Must include all the inputs to the device in the left columns
  - Must include all the outputs of the device in the right columns
  - The behavior of the circuit is implied by the table



Number of combinations is  $2^N$ ; N is the number of the inputs

Inputs			Outputs	
I1	I2	I3	O1	O2
0	0	0	?	?
0	0	1	?	?
0	1	0	?	?
0	1	1	?	?
1	0	0	?	?
1	0	1	?	?
1	1	0	?	?
1	1	1	?	?

# Example of Truth Table

a	b	F
0	0	
0	1	
1	0	
1	1	

(a)

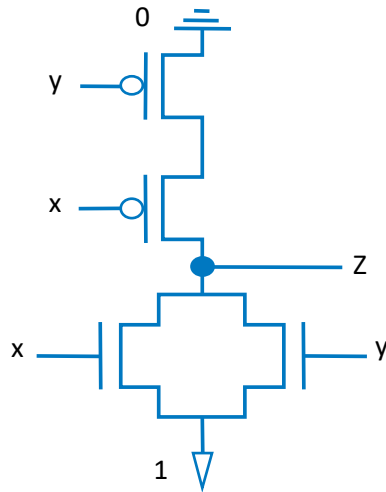
a	b	c	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

(b)

a	b	c	d	F
0	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	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

(c)

# OR Logic



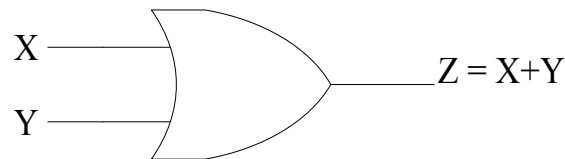
X	Y	Z = X + Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

- Definition of OR operation**

$Z = X + Y$  means  $Z = 1$  if either  $X=1$  or  $Y=1$ , or both;

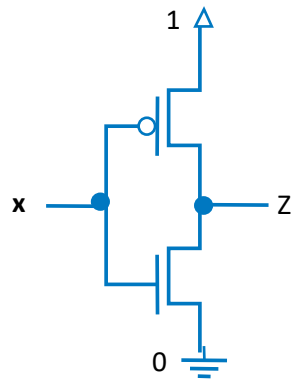
OR operator



2-input OR gate



# NOT Logic



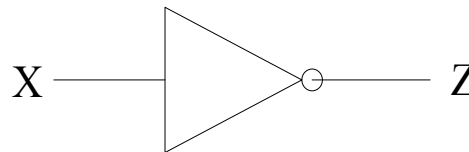
X	Z = X'
0	1
1	0

Truth Table

- Definition of NOT operation**

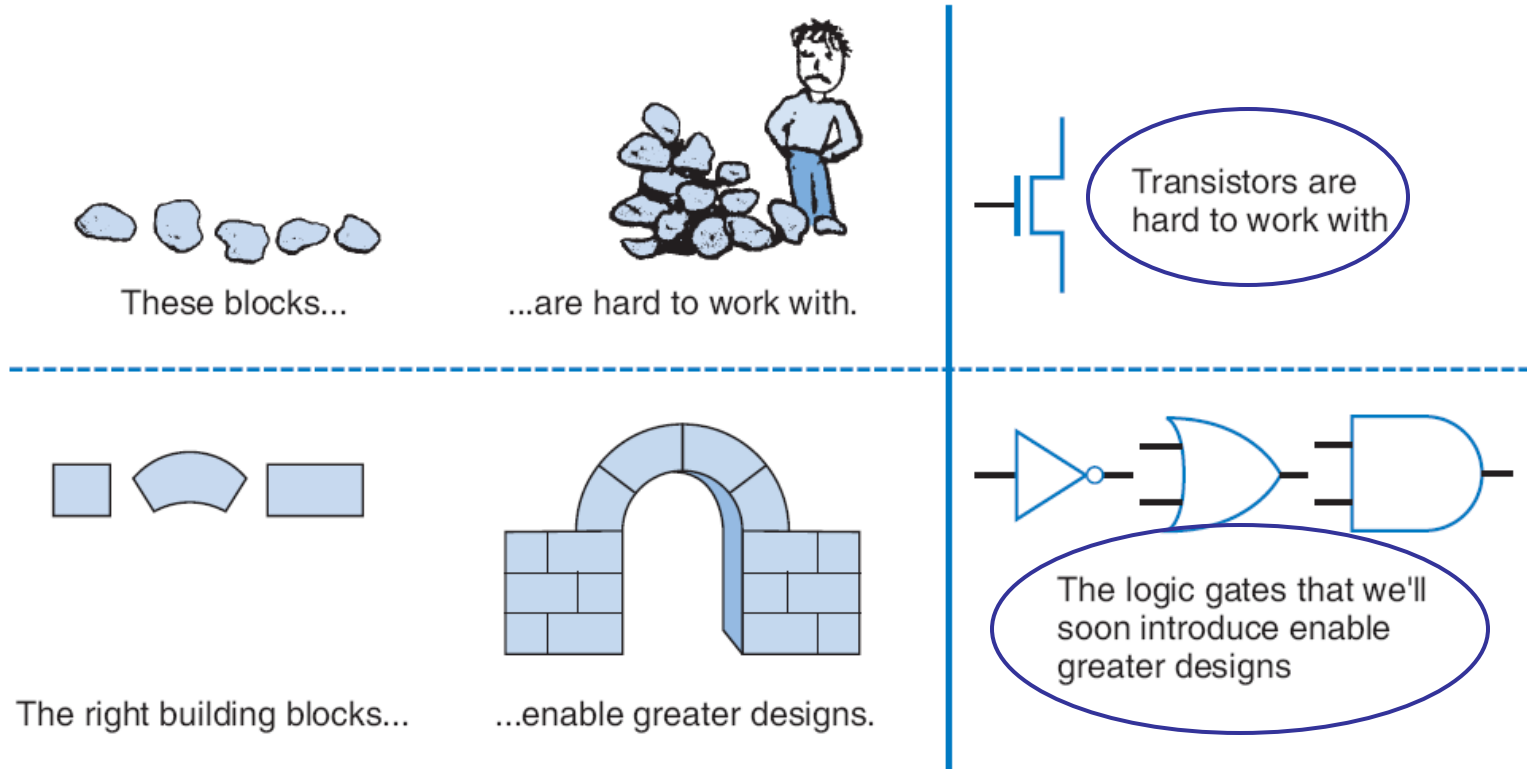
$Z = X'$  or  $Z = \overline{X}$  means  $Z = 0$  if  $X = 1$ ;  $Z = 1$  if  $X = 0$ ;  $Z$  is the complement of  $X$

NOT operator



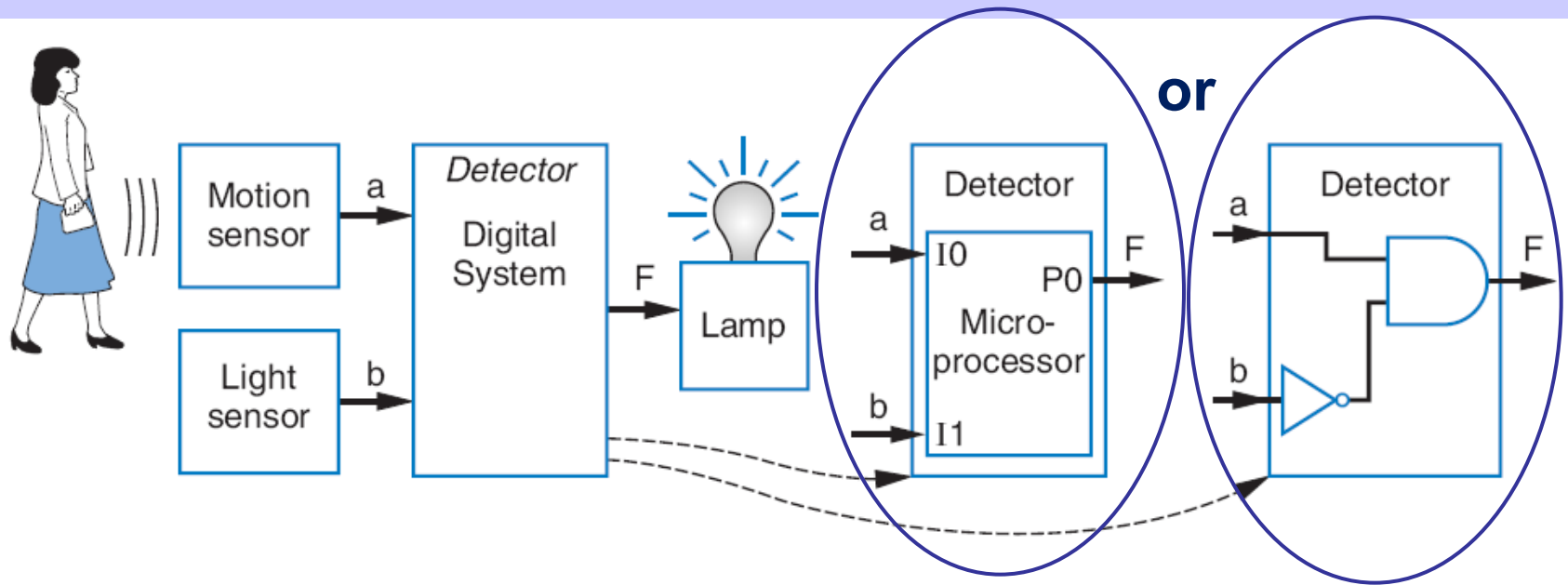
NOT gate/Inverter

# Logic Gates



- **“Logic gates” are better digital circuit building blocks than switches (transistors)**

# Logics in Human Language

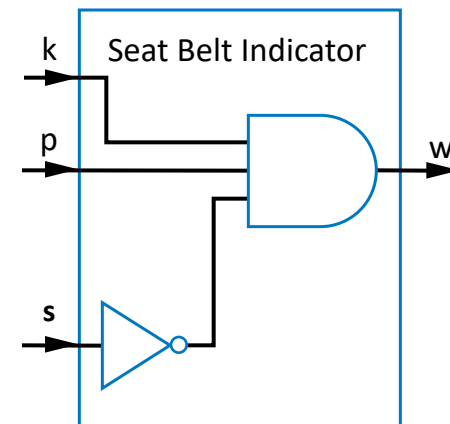


- **Motion-in-dark example**

- Human language: Turn on lamp ( $F=1$ ) when motion sensed ( $a$ ) and no light (not  $b$ )
- *Logic Equation*:  $F = a \text{ AND NOT}(b) = ab'$
- *Logic circuit*: implementation using logic gates, AND and NOT, as shown

# Example: Seat Belt Warning Light System

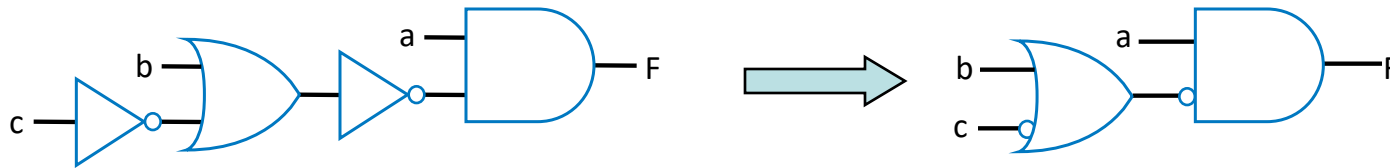
- **Design circuit for warning light**
- **Sensors**
  - $s=1$ : seat belt fastened
  - $k=1$ : key inserted
  - $p=1$ : person in seat
- **Function description**
  - Light on if person in seat, and seat belt not fastened, and key inserted
- **Logic equation**  
$$w = p \text{ AND } \text{NOT}(s) \text{ AND } k$$



# Example: Represent Logic Equation with Logic Gates

- Q: Convert the following equation to logic gates:

$$F = a \text{ AND NOT}( b \text{ OR NOT}(c) )$$

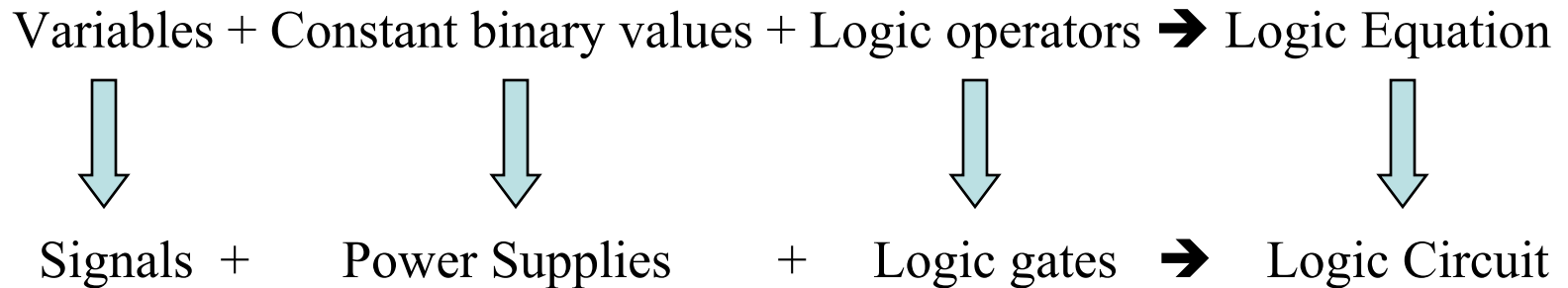


- Precedence of Logic Operations

**NOT > AND > OR**

# From Logic Equation to Logic Circuit

- There exists a correspondence between a Logic Equation and its logic circuit.

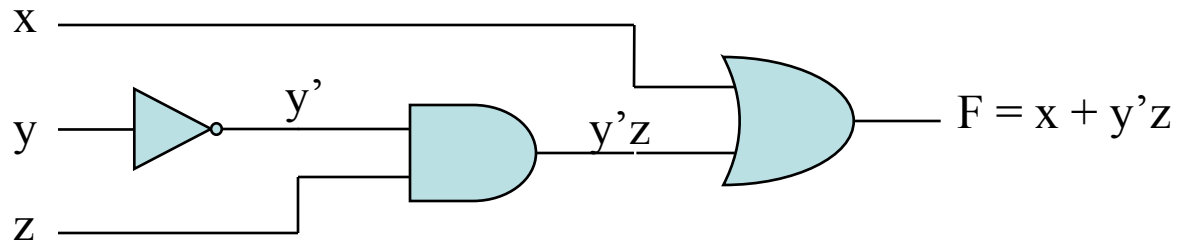
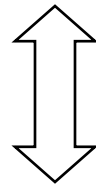
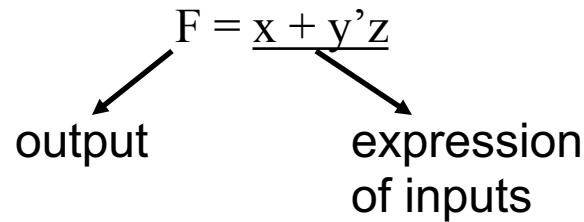


- **Logic Circuit:**  
A net of logic gates.

# Logic Equation and Logic Circuit

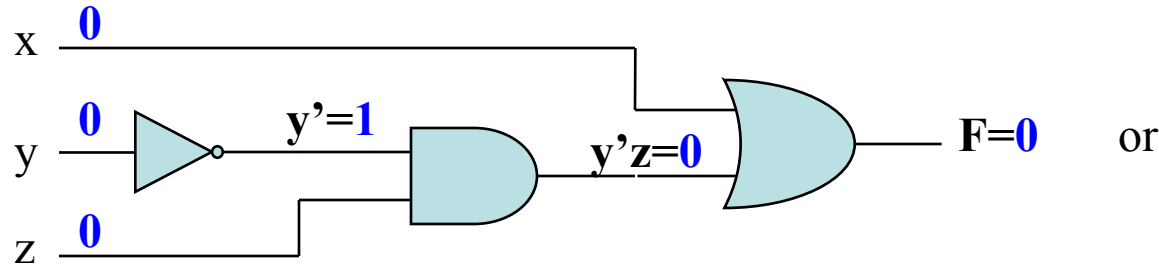
- **$F = x \text{ OR } (\text{NOT } y \text{ AND } z)$**

$$= x + y'z$$



# Build Truth Table

- $F = x + y'z$



$$\begin{aligned}
 F &= 0 + 0' \cdot 0 \\
 &= 0 + 1 \cdot 0 \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

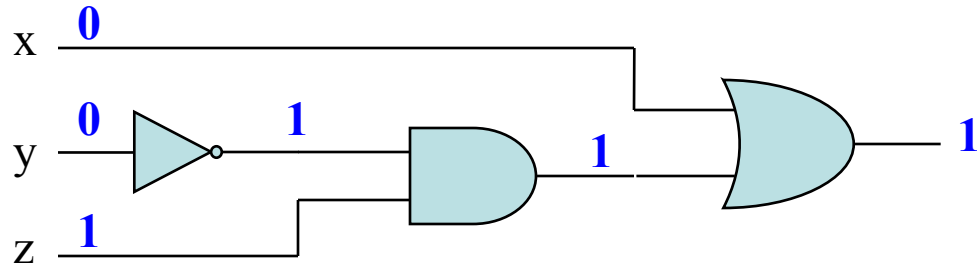
- Truth table

x	y	z	y'	y'z	F
0	0	0	1	0	0
0	0	1	?	?	?
0	1	0	?	?	?
0	1	1	?	?	?
1	0	0	?	?	?
1	0	1	?	?	?
1	1	0	?	?	?
1	1	1	?	?	?



# Build Truth Table

- $F = x + y'z$



or

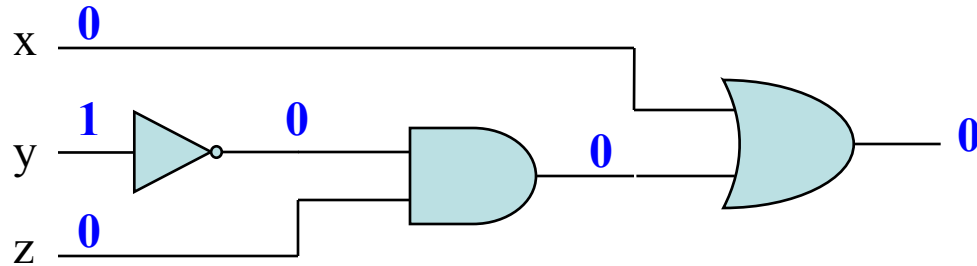
$$\begin{aligned} F &= 0 + 0' \cdot 1 \\ &= 0 + 1 \cdot 1 \\ &= 0 + 1 \\ &= 1 \end{aligned}$$

- Truth table

x	y	z	$y'$	$y'z$	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	?	?	?
0	1	1	?	?	?
1	0	0	?	?	?
1	0	1	?	?	?
1	1	0	?	?	?
1	1	1	?	?	?

# Build Truth Table

- $F = x + y'z$



or

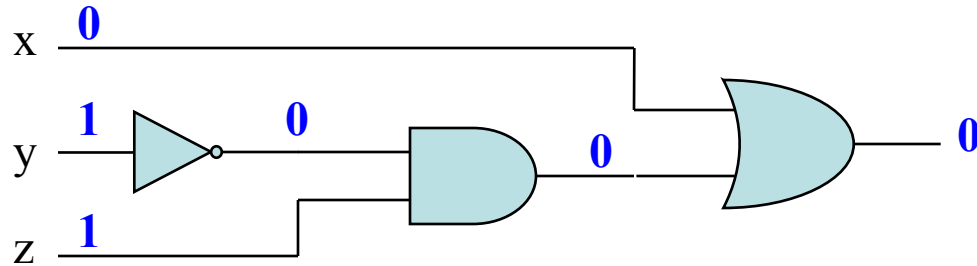
$$\begin{aligned} F &= 0 + 1' \cdot 0 \\ &= 0 + 0 \cdot 0 \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

- Truth table

x	y	z	$y'$	$y'z$	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	?	?	?
1	0	0	?	?	?
1	0	1	?	?	?
1	1	0	?	?	?
1	1	1	?	?	?

# Build Truth Table

- $F = x + y'z$



or

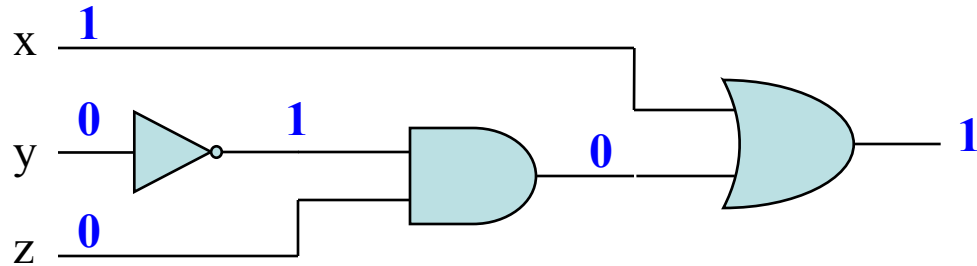
$$\begin{aligned} F &= 0 + 1' \cdot 1 \\ &= 0 + 0 \cdot 1 \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

- Truth table

x	y	z	y'	y'z	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	?	?	?
1	0	1	?	?	?
1	1	0	?	?	?
1	1	1	?	?	?

# Build Truth Table

- $F = x + y'z$



or

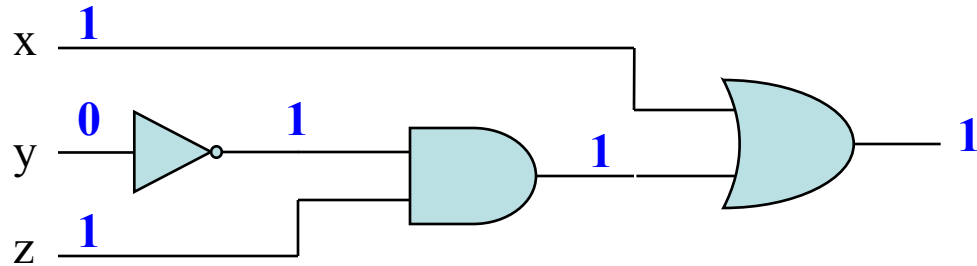
$$\begin{aligned} F &= 1 + 0' \cdot 0 \\ &= 1 + 1 \cdot 0 \\ &= 1 + 0 \\ &= 1 \end{aligned}$$

- Truth table

x	y	z	y'	y'z	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	?	?	?
1	1	0	?	?	?
1	1	1	?	?	?

# Build Truth Table

- $F = x + y'z$



or

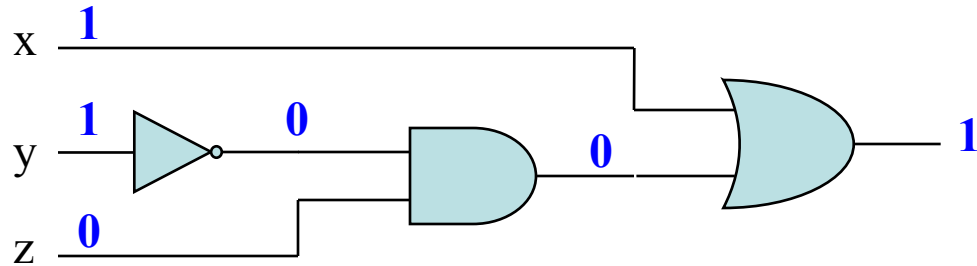
$$\begin{aligned}
 F &= 1 + 0' \cdot 1 \\
 &= 1 + 1 \cdot 1 \\
 &= 1 + 1 \\
 &= 1
 \end{aligned}$$

- Truth table

x	y	z	y'	y'z	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	?	?	?
1	1	1	?	?	?

# Build Truth Table

- $F = x + y'z$



or

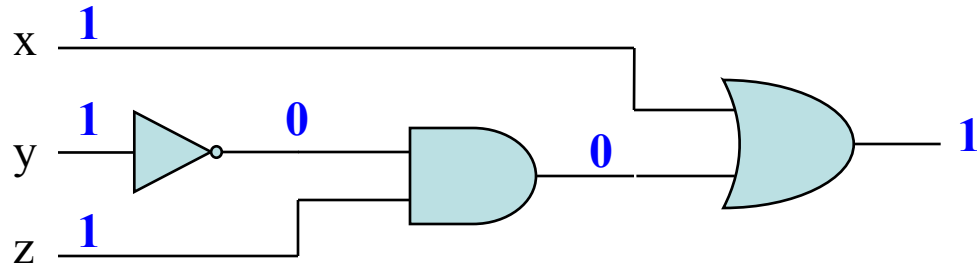
$$\begin{aligned}
 F &= 1 + 1' \cdot 0 \\
 &= 1 + 0 \cdot 0 \\
 &= 1 + 0 \\
 &= 1
 \end{aligned}$$

- Truth table

x	y	z	y'	y'z	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	?	?	?

# Build Truth Table

- $F = x + y'z$



or

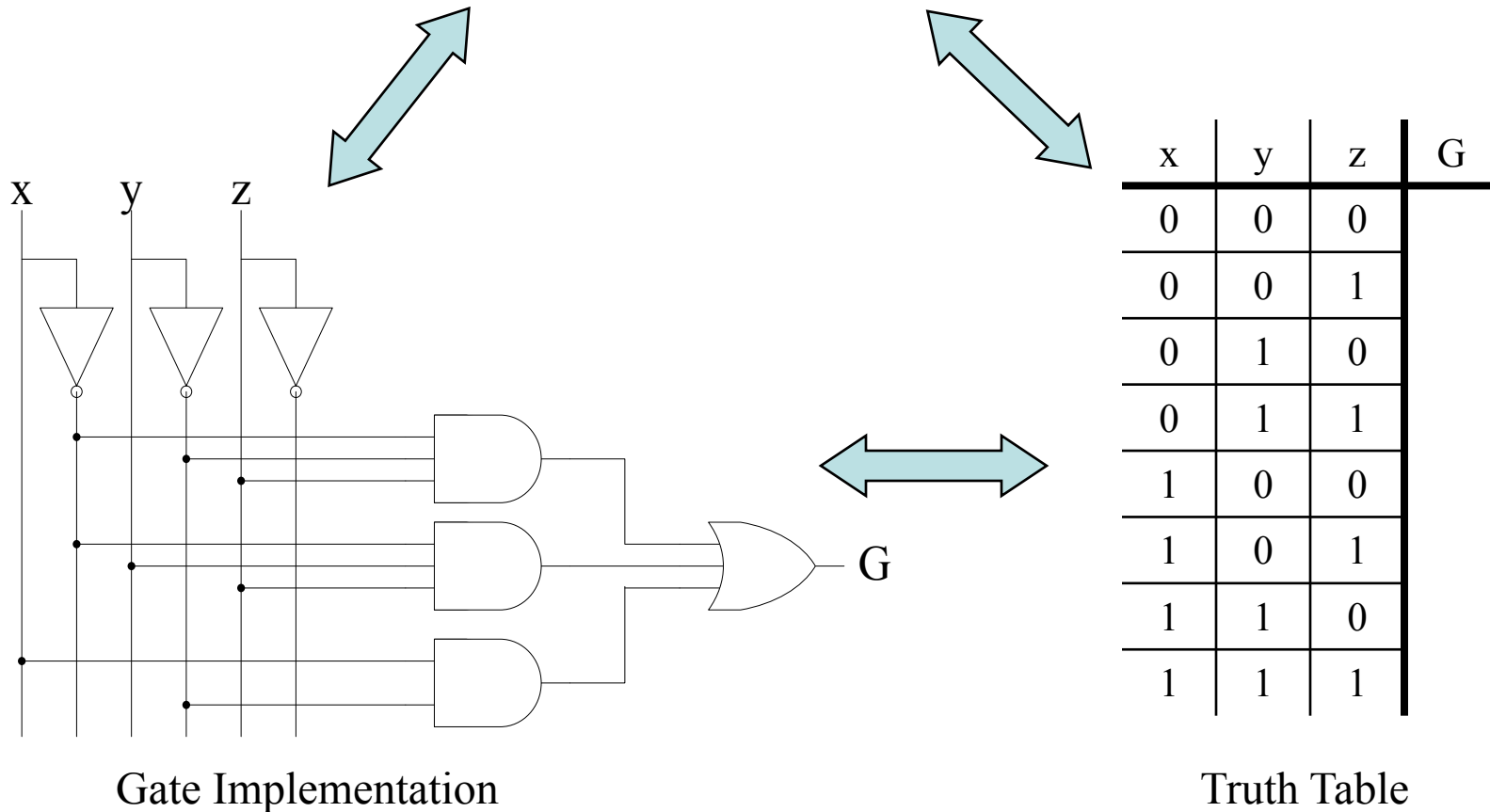
$$\begin{aligned} F &= 1 + 1' \cdot 1 \\ &= 1 + 0 \cdot 1 \\ &= 1 + 0 \\ &= 1 \end{aligned}$$

- Truth table

x	y	z	$y'$	$y'z$	F
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

# Logic Equation, Truth Table, & Logic Circuit

- Another example:  $G = x'y'z + x'yz + xy'$





# Create Truth Table

a	b	F
0	0	
0	1	
1	0	
1	1	

(a)

a	b	c	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

(b)

a	b	c	d	F
0	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	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

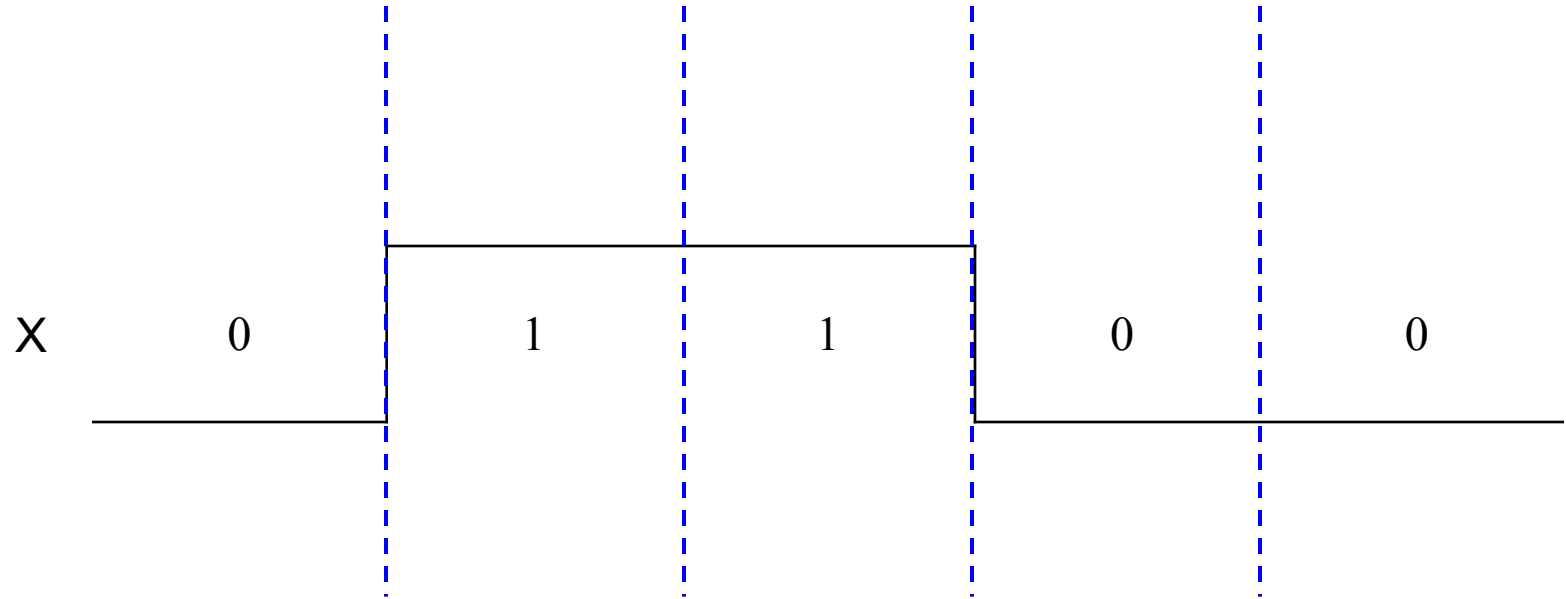
(c)

- Q: Use truth table to define function  $F(a,b,c)$  that is 1 only when  $abc$  is 5 or greater in binary**

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

# Timing Diagram

- **Timing diagrams show the response to changes on a signal in voltage levels with time**

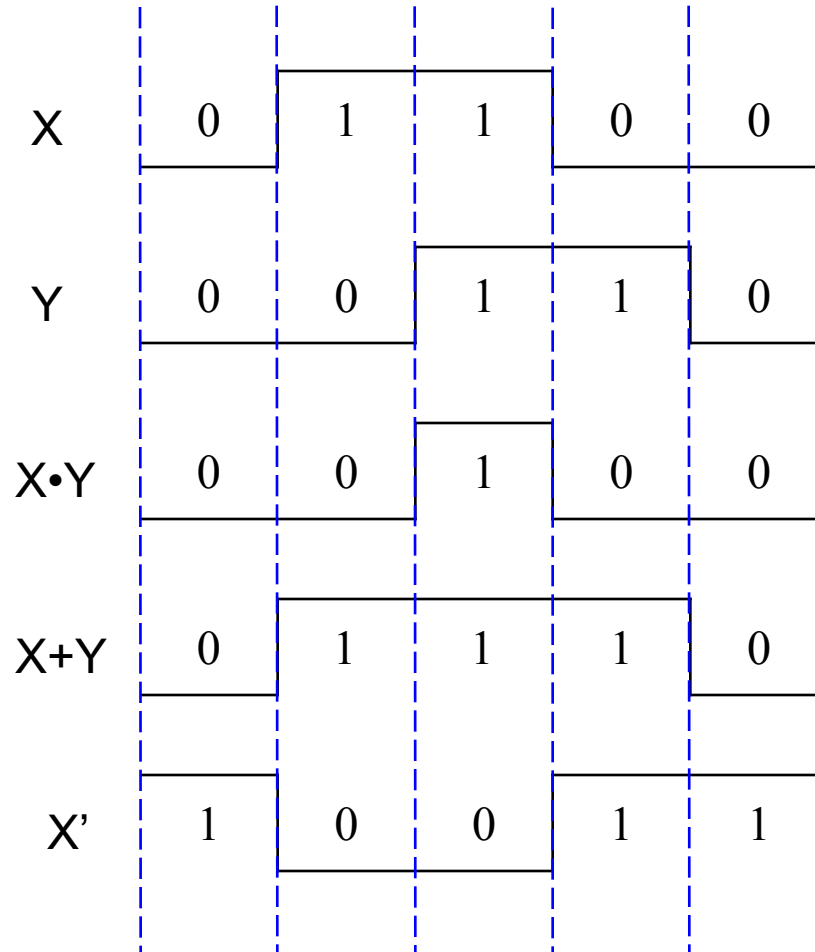


# Timing Diagrams for Gates

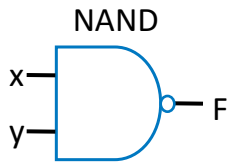
xy	F
0 0	0
0 1	0
1 0	0
1 1	1

x+y	F
0 0	0
0 1	1
1 0	1
1 1	1

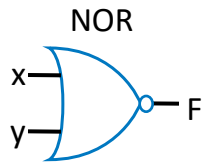
x	F
0	1
1	0



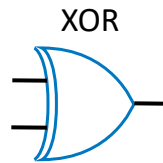
# More Gates



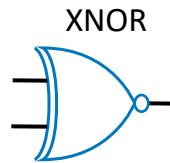
x	y	F
0	0	1
0	1	1
1	0	1
1	1	0



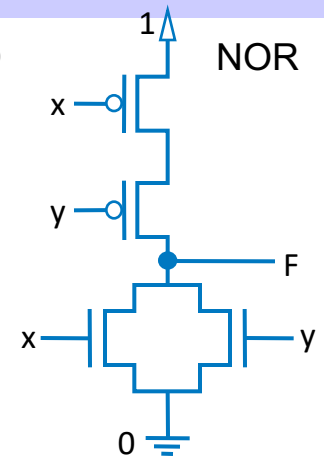
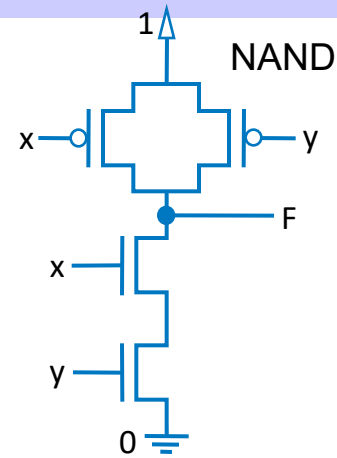
x	y	F
0	0	1
0	1	0
1	0	0
1	1	0



x	y	F
0	0	0
0	1	1
1	0	1
1	1	0



x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

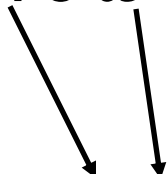


- NAND: Opposite of AND (“NOT AND”)
- NOR: Opposite of OR (“NOT OR”)
- XOR: outputs 1 when inputs have odd number of 1’s
- XNOR: Opposite of XOR (“NOT XOR”)

- AND in CMOS: NAND with NOT
- OR in CMOS: NOR with NOT
- So NAND/NOR more common

# Detecting Overflow: Method 2

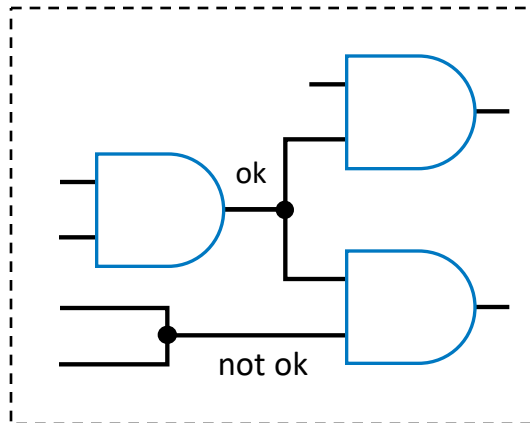
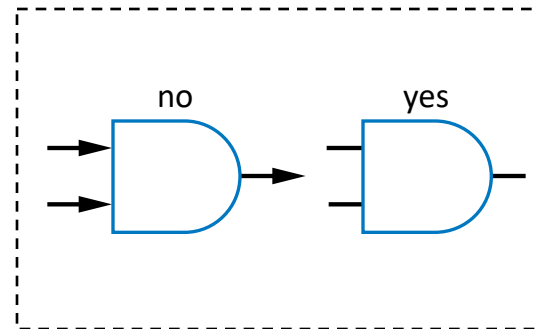
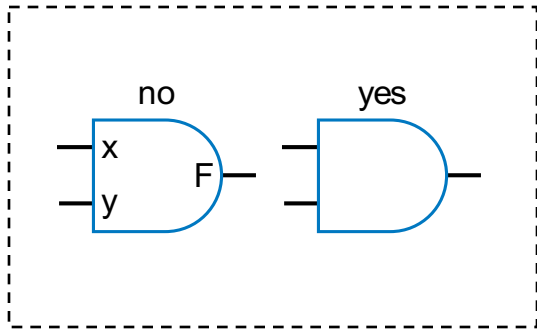
- **Simpler method: Detect difference between carry-in to sign bit and carry-out from sign bit**
- **$\text{overflow} = c4 \oplus c3$**



0	1	1	1		1	0	0	0		0	0	0	0	
	0	1	1	1		1	1	1	1		1	0	0	0
+ 0	0	0	0	1	+ 1	0	0	0	0	+ 0	1	1	1	1
<hr/>					<hr/>					<hr/>				
1	0	0	0	0	0	1	1	1	1	1	1	1	1	1
overflow					overflow					no overflow				
(a)					(b)					(c)				

**If the carry into the sign bit column differs from the carry out of that column, overflow has occurred.**

# Some Circuit Drawing Conventions

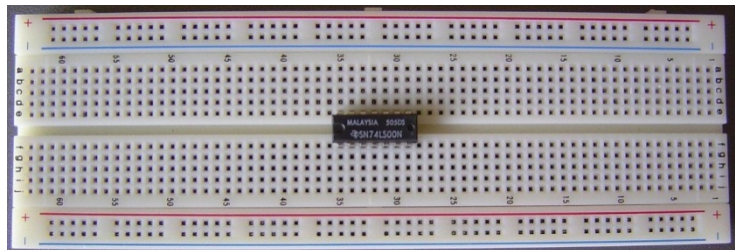


# Integrated Circuit

- **Integrated Circuit (IC) – chip**
  - Contains logic components and/or devices for constructing digital circuits
- **Integration Levels**
  - Small-Scale Integration (SSI)
    - Fewer than 10 gates
  - Medium-Scale Integration (MSI)
    - 10 to 1000 gates
  - Large-Scale Integration (LSI)
    - Thousands of gates
  - Very Large-Scale Integration (VLSI)
    - Millions of gates
  - Ultra Large-Scale Integration (ULSI)
    - Billions of gates
  - ...

# Integrated Circuit

- **TTL** – Transistor-Transistor Logic
- **ECL** – Emitter-Coupled Logic
- **MOS** – Metal-Oxide Semiconductor
- **CMOS** – Complementary MOS



Chip placed on breadboard

