

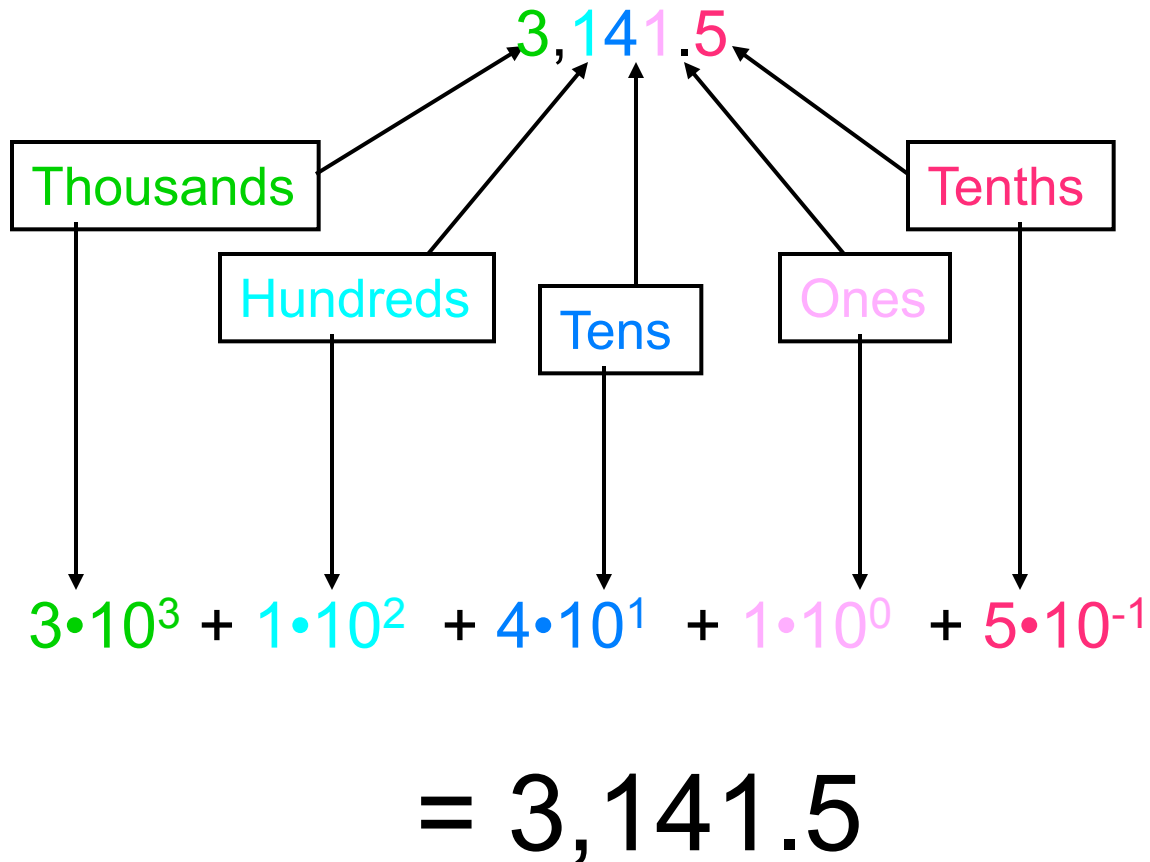
# Today

## **Digital**

- **Numbering systems**
- **Digital circuits**
- **Logic States**
- **Boolean Logic Gates**

# Numbering Systems *I*

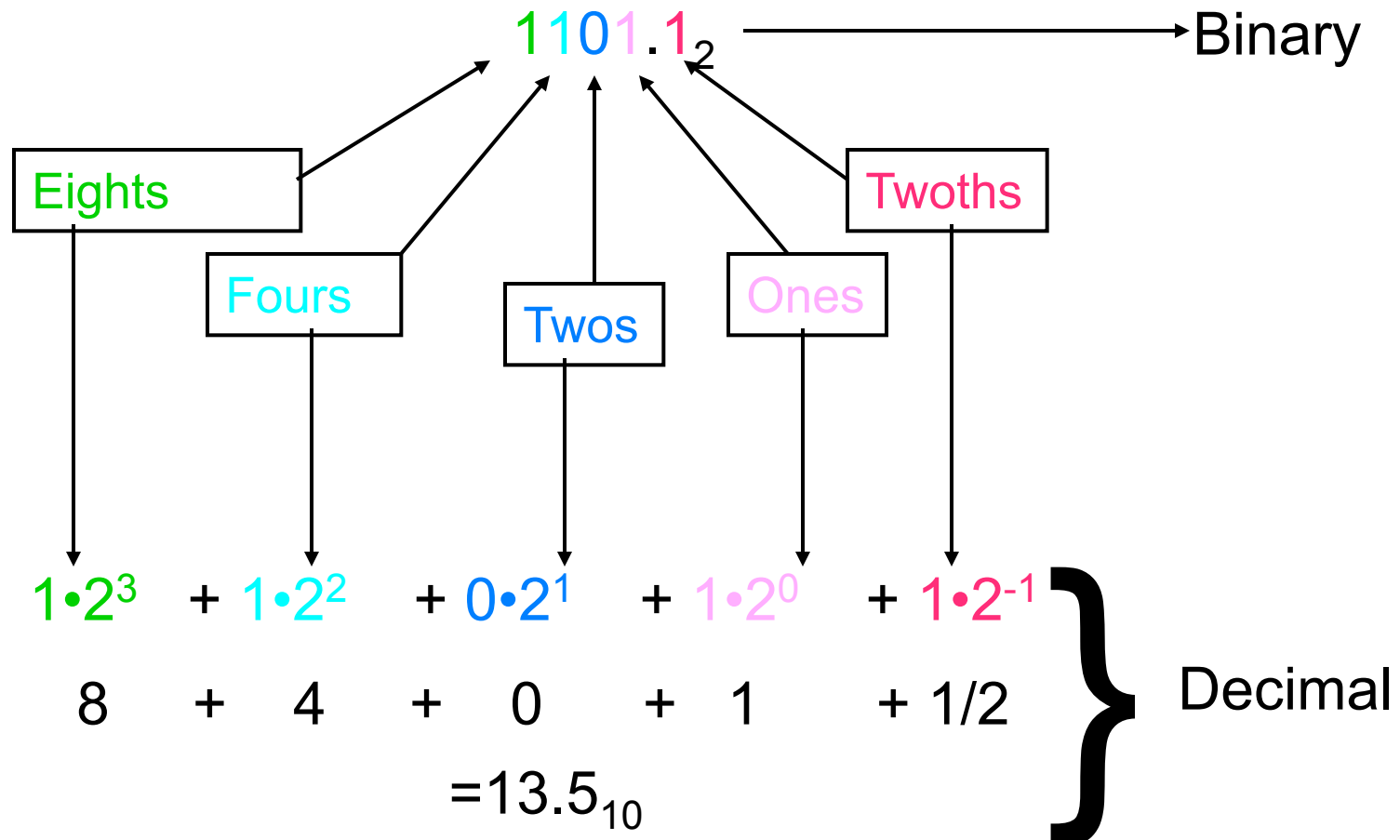
Decimal - Counting in Base 10 and *Converting to Decimal*  
There are a total of 10 unique numbers, 0, 1, . . . , 9



# Numbering Systems *II*

## Binary - Counting in Base 2 *and Converting to Decimal*

There are a total of 2 unique numbers, 0 and 1



# Numbering Systems *III*

## Hexadecimal - Counting in Base 16

and  
Converting to  
Decimal

There are a total of 16 unique numbers:  
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

9E7B.8<sub>16</sub>

$$9 \cdot 16^3 + E \cdot 16^2 + 7 \cdot 16^1 + B \cdot 16^0 + 8 \cdot 16^{-1} = 40,571.5_{10}$$

This method can be applied to any number base to convert to decimal equivalent (base 10).

# Base Conversion

For small numbers, a conversion table is convenient and can be readily developed.

Since most of the symbols look the same, it is critical that the base of the number be explicitly identified.

(Octal is base 8)

Decimal	Binary	Octal	Hex.
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

# Modulo

Converting from Decimal to a different base is most easily accomplished using the modulo operation. For example, convert  $435_{10}$  to hexadecimal.

$$435_{10} / 16_{10} = 27_{10} \text{ rmndr } 3_{10} \quad \xrightarrow{\text{Dec to Hex}} \quad 3_{16} \quad \text{LSD}$$

$$27_{10} / 16_{10} = 1_{10} \text{ rmndr } 11_{10} \quad \xrightarrow{\text{Dec to Hex}} \quad B_{16}$$

$$1 / 16_{10} = 0_{10} \text{ rmndr } 1_{10} \quad \xrightarrow{\text{Dec to Hex}} \quad 1_{16} \quad \text{MSD}$$

$$\text{Therefore } 435_{10} = 1B3_{16}$$

$$\text{Check: } 1 * 16^2 + 11 * 16^1 + 3 * 16^0 = 435_{10}$$

# Decimal to Binary

Convert  $171_{10}$  to binary:

$$171_{10} / 2 = 85_{10} \text{ r}1 \quad 1 \quad \text{LSB}$$

$$85_{10} / 2 = 42_{10} \text{ r}1 \quad 1$$

$$42_{10} / 2 = 21_{10} \text{ r}0 \quad 0$$

$$21_{10} / 2 = 10_{10} \text{ r}1 \quad 1$$

$$10_{10} / 2 = 5_{10} \text{ r}0 \quad 0$$

$$5_{10} / 2 = 2_{10} \text{ r}1 \quad 1$$

$$2_{10} / 2 = 1_{10} \text{ r}0 \quad 0$$

$$1_{10} / 2 = 0_{10} \text{ r}1 \quad 1 \quad \text{MSB}$$

$$171_{10} = 10101011_2$$

# Bits and Computer-ese

**Bit:** 1 or 0

- High or Low; True or False; Binary value

**Byte:**  $11111111_2$ ,  $377_8$ ,  $255_{10}$ ,  $FF_{16}$

- 8 bits, represented by two hexadecimal numbers
- 32 and 64 bit processors are now common
- One byte is 256 units in decimal (0 to 255)
- Storage measure (4GByte RAM, 1TB Hard disk)

**Word:** 2 Bytes typically (sometimes more)

- Word is integer number of bytes (16 or more bits)

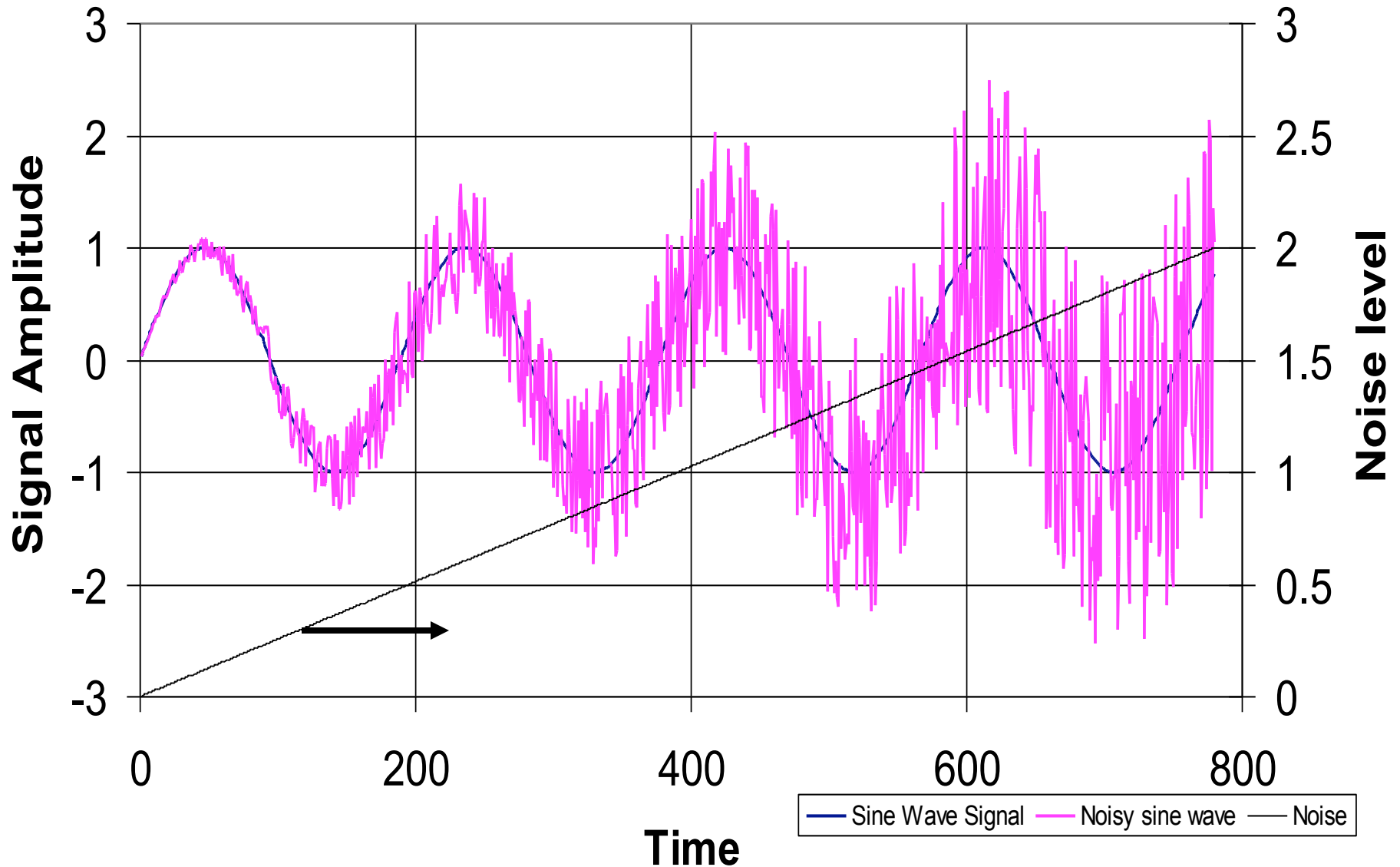


# Digital Signals and Noise

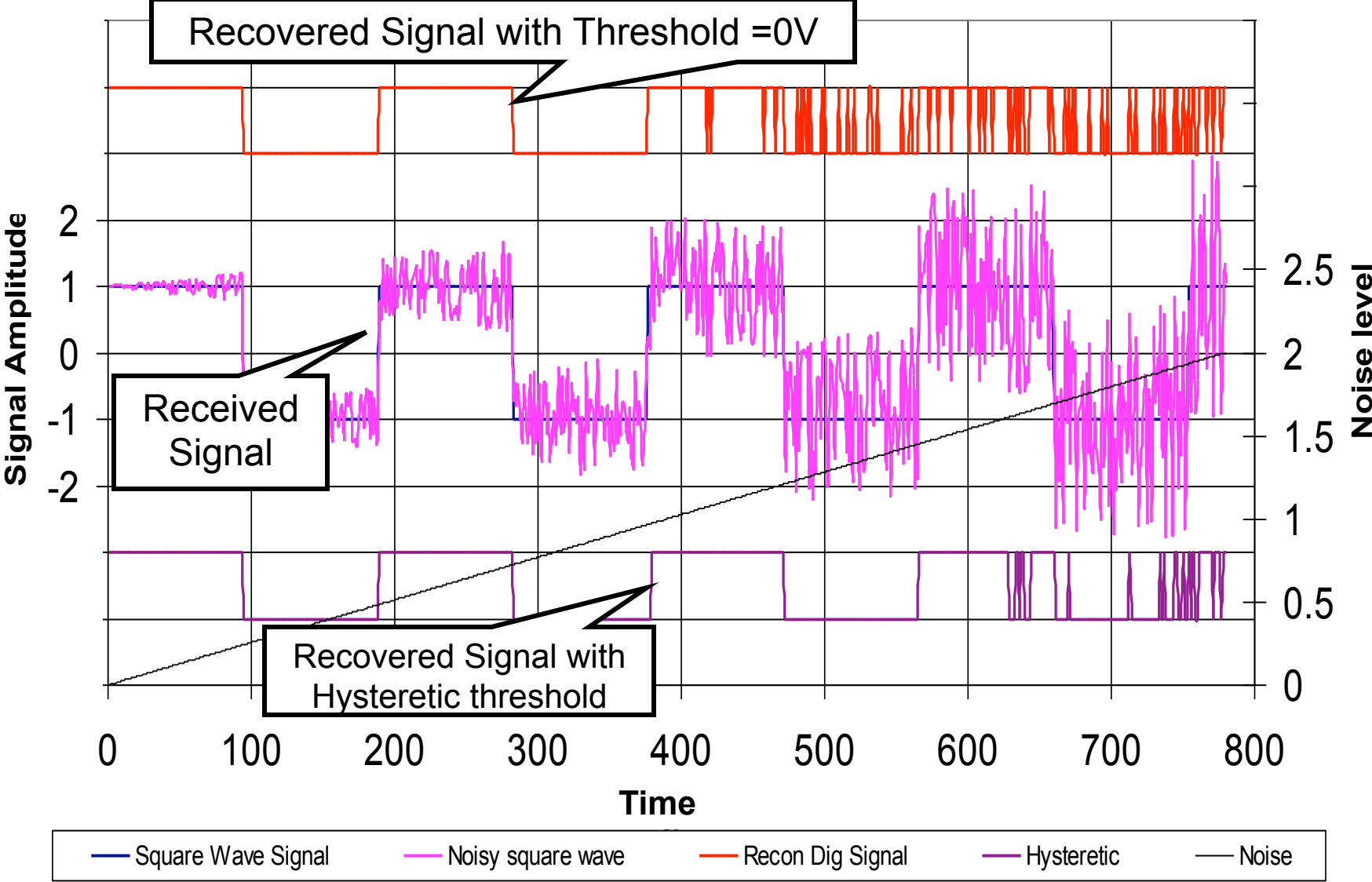
## “Immune” to noise

- All electronic signals are susceptible to noise, but digital signals are more tolerant to low “signal to noise ratios”.
- Information (a value) transmitted via an analog signal can lose accuracy due to additive noise. Accuracy = SNR.
- Digital signals, if no bits are lost, transfer is 100% accurate up the number of bits (information) transmitted.
- Threshold restoration of digital data
  - Example:  $>0.5V \Rightarrow 1V$  (high)       $<0.5V \Rightarrow 0V$  (low)
- Hysteretic processing decreases susceptibility to noise  
Example for -1V to 1V binary signal:
  - If current state is “1”, requires signals to drop to “-0.5” to switch to “-1”
  - If current state is “-1”, requires signals to rise to “0.5” to switch to “1”

# Noisy Analog Signal



# Noisy Digital Signal and Hysteresis



Based with permission on lectures by John Getty

# Digital Signals

## Advantages

- Improved storage density
- More powerful/accurate signal processing

## Disadvantages

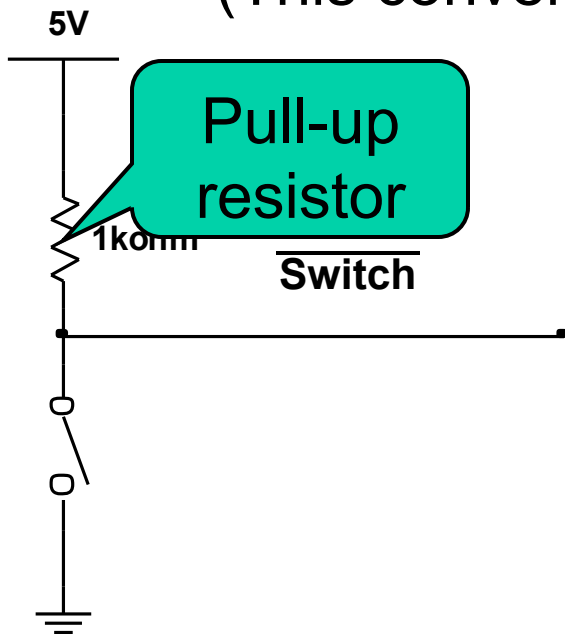
- We live in analog world
- A/D and D/A conversion introduces noise or distortion
  - Accurate to within  $\pm 1$ LSB (least significant bit)
  - What is the difference between Noise and Distortion?

# Logic States

The on-off behavior of digital signals is used to represent individual bits. :

- 0 = False, binary zero
- 1 = True, binary one

(This convention is referred to as *positive logic*.)



“Switch” is read as switch closed. “ $\overline{\text{Switch}}$ ” is read as “not” switch. So when the switch is open, “ $\overline{\text{Switch}}$ ” is true.

# Logic Thresholds and Levels

**Input Threshold** - the boundary voltage at which a logic device changes its interpretation of an input signal either as a logical false or logical true.

**Output Level** - the output voltage from a logic device that can be expected for logic false and logic true values.

→ = typical value

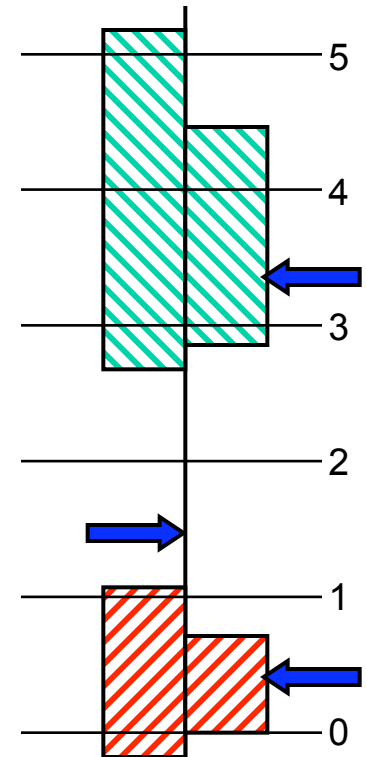


High or True




Low or False


CMOS  
Input Output

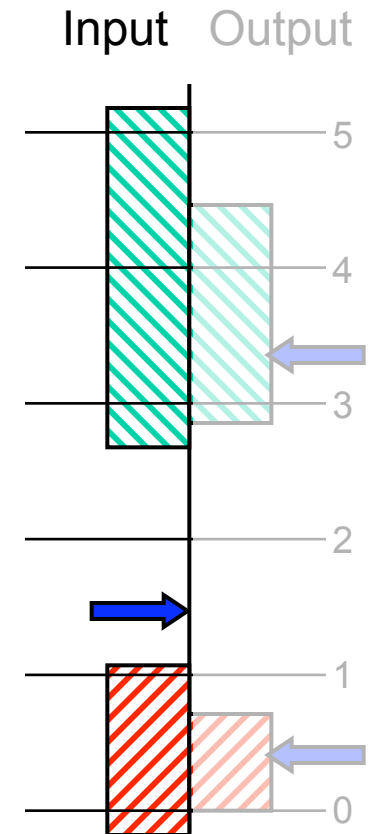


# Input Thresholds

→ For input signals, the arrow represents the **typical decision threshold**.


 The upper block is the range of input voltages where it is **guaranteed** that the device recognizes the signal as a logical true.


 The lower block is the **guaranteed** range for logical false inputs.

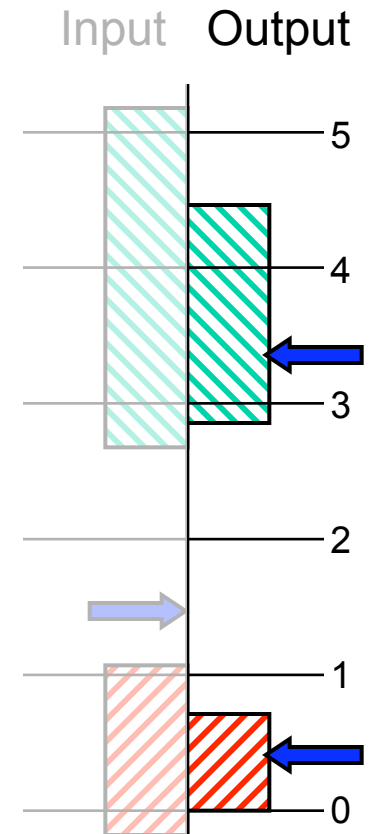


# Output Levels

← For output signals, the arrow represents the **typical** voltages that are output for true and false signals.

 Manufacturers **guarantee** that the output voltage representing a logical true will fall within the range indicated by the upper block.\*

 The lower block is the guaranteed range for logical false output signals.



\* - assuming all other specifications are satisfied



# Noise Margin

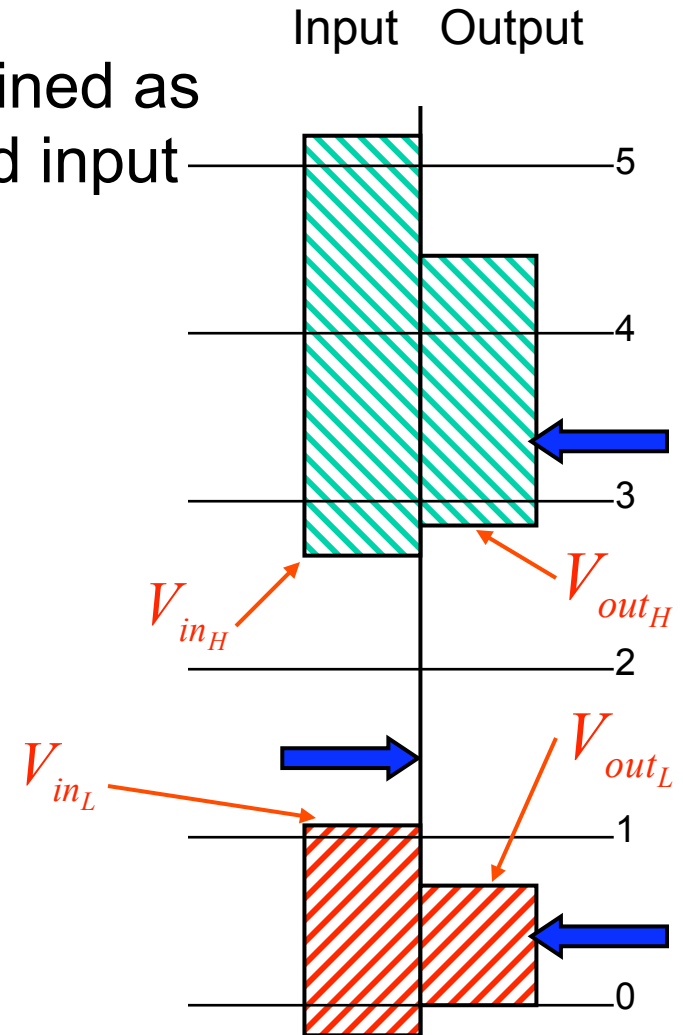
*Noise Margin* or *noise immunity* is defined as the difference between the guaranteed input threshold and output levels.

High noise margin

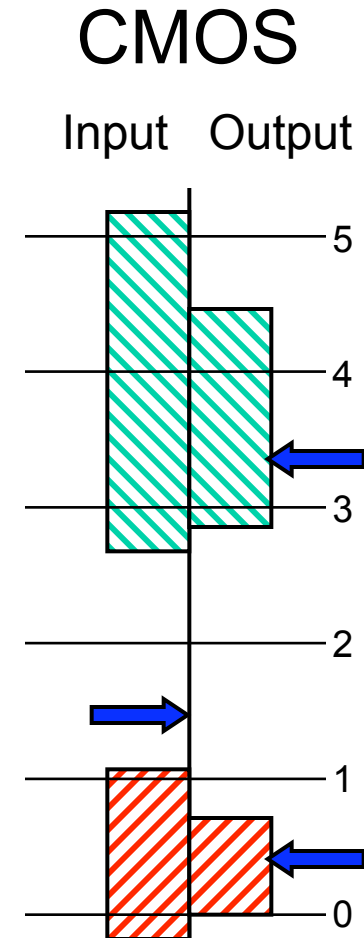
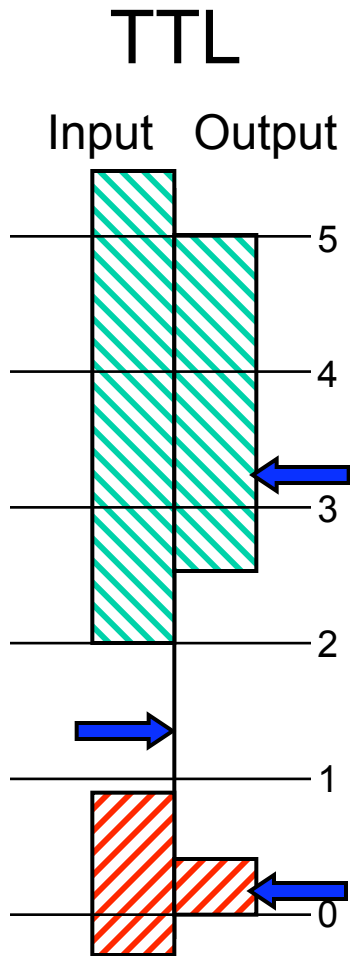
$$N_{M_H} = V_{out_H} - V_{in_H}$$

Low noise margin

$$N_{M_L} = V_{in_L} - V_{out_L}$$



# Some Standard Thresholds



# Logic Functions

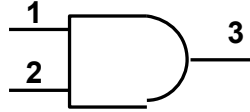
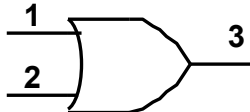
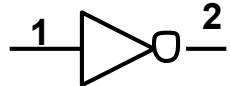


George Boole  
(1815-1864)

## Boolean Operators

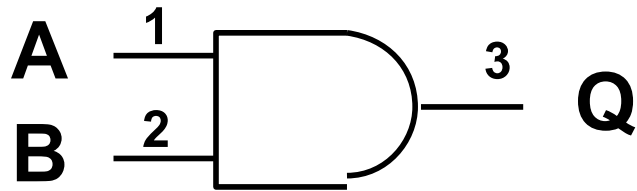
### Symbol

---

Name	Algebraic	Schematic	Example
AND	$\cdot, \cap, \wedge$		$1 \cdot 0 = 0$
OR	$+, \cup, \vee$		$1 + 0 = 1$
NOT	$\bar{A}, ', \neg$		$1' = 0$

# AND Gate

## Schematic symbol



## *Truth Table*

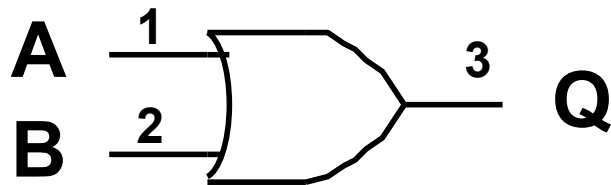
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

## Algebraic example

$$Q = A \cdot B$$

# OR Gate

## Schematic symbol



## Algebraic example

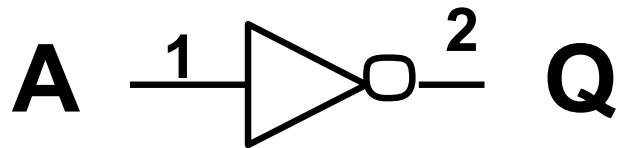
$$Q = A + B$$

## Truth Table

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

# NOT Gate

## Schematic symbol



## Truth Table

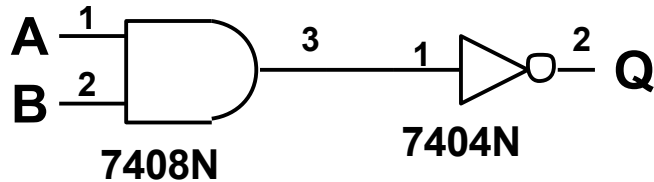
A	Q
0	1
1	0

## Algebraic example

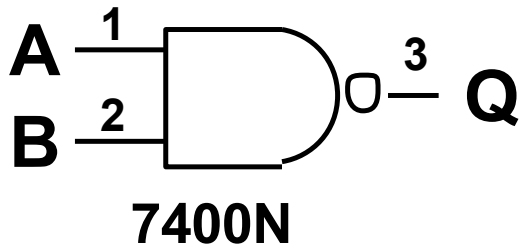
$$Q = \overline{A}$$

# NAND Gate

NOT AND is created by inverting an AND gate



Schematic symbol



Algebraic example

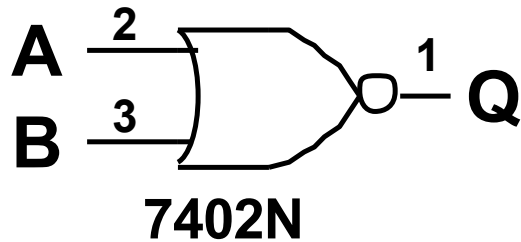
$$\overline{A \cdot B} = Q$$

Truth Table

A	B	A•B	Q
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

# NOR Gate

## Schematic symbol



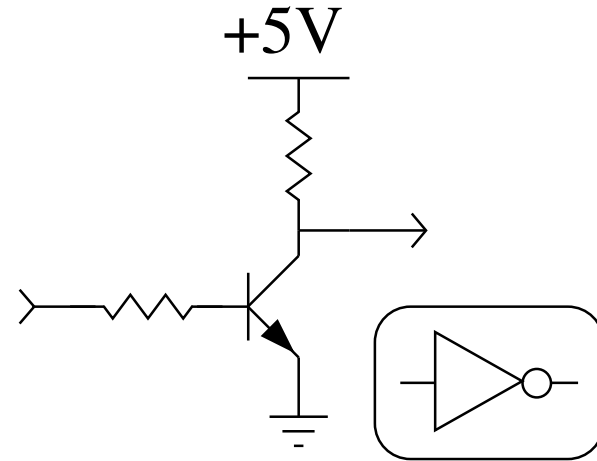
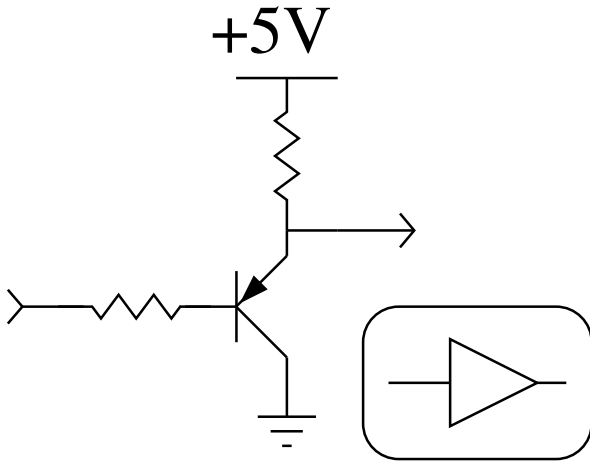
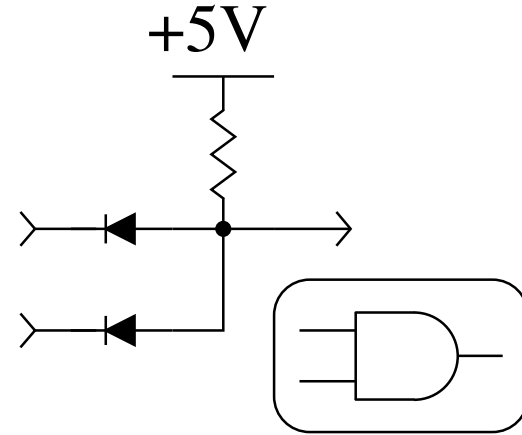
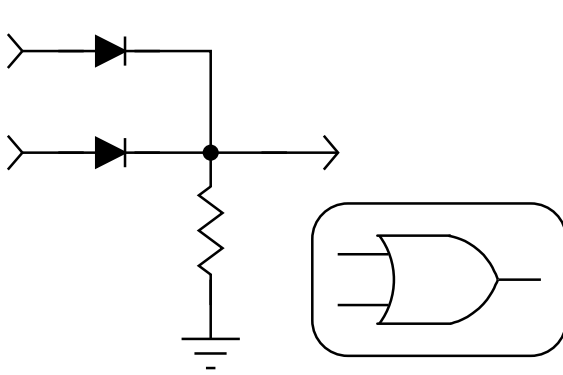
$$\overline{A + B} = Q$$

## Truth Table

A	B	A+B	Q
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



# Example gate circuits



good resource: <http://www.hanssummers.com/gates.html>

# Basic Boolean Identities

For a more complete list, see Table 8.3 in H&H page 491

Identity and Zero  $A \cdot 1 = A, A \cdot 0 = 0, A \cdot A = A$   
 $A + 1 = 1, A + 0 = A, A + A = A$

$$\bar{1} = 0, \quad \bar{0} = 1$$

$A \cdot \bar{A} = 0, A + \bar{A} = 1, \overline{(\bar{A})} = A$  “Not” (inversion)

Associative Properties  $A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$   
 $A + B + C = (A + B) + C = A + (B + C)$

$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$  Distributive Property

Absorption Theorems  $A + A \cdot B = A, A \cdot (A + B) = A$

$\overline{A + B} = \bar{A} \cdot \bar{B}$   $\overline{A \cdot B} = \bar{A} + \bar{B}$  DeMorgan's Theorems