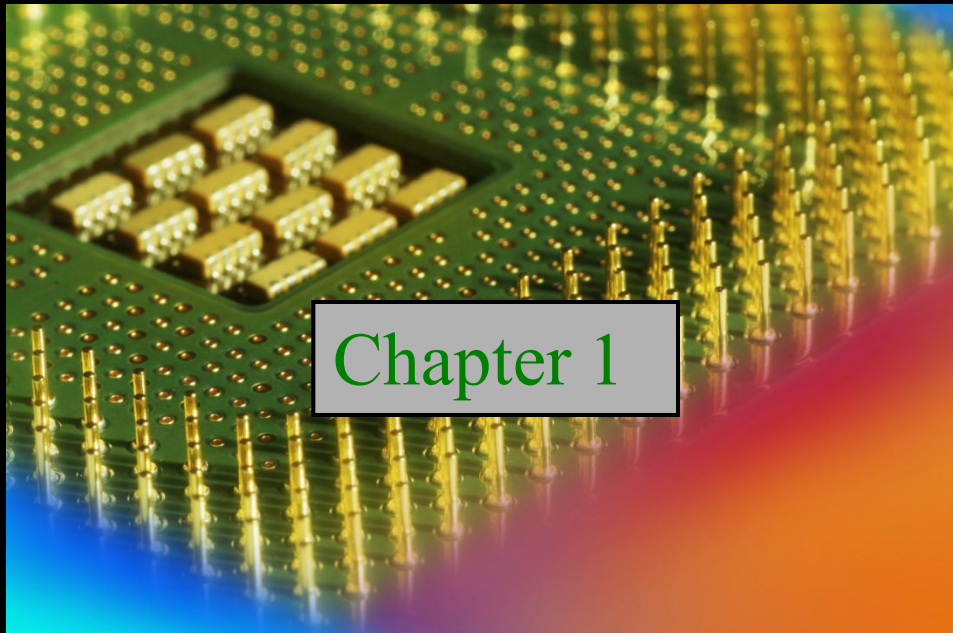


# Digital Fundamentals

Tenth Edition

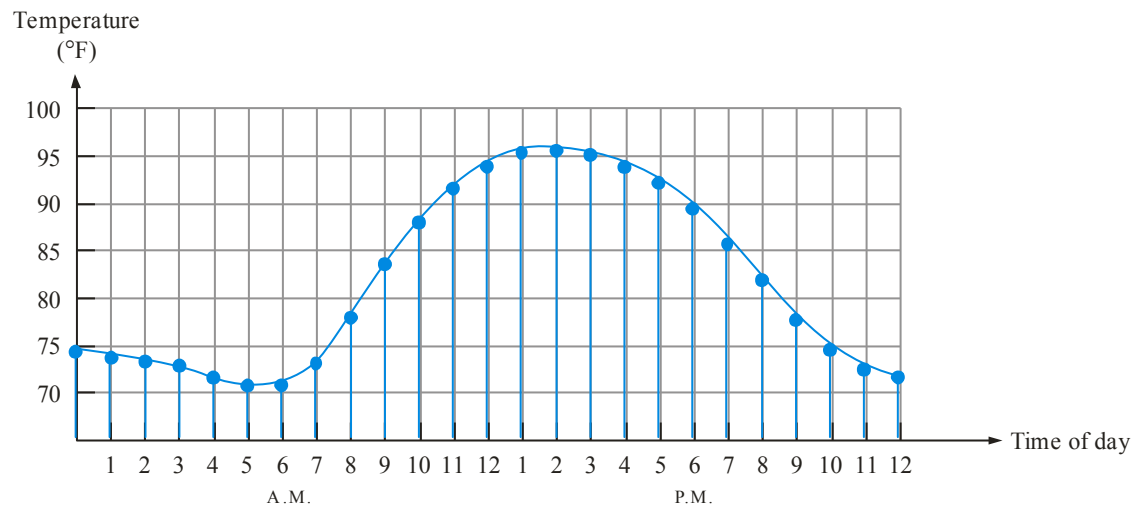
Floyd



# Summary

## Analog Quantities

Most natural quantities that we see are **analog** and vary continuously.

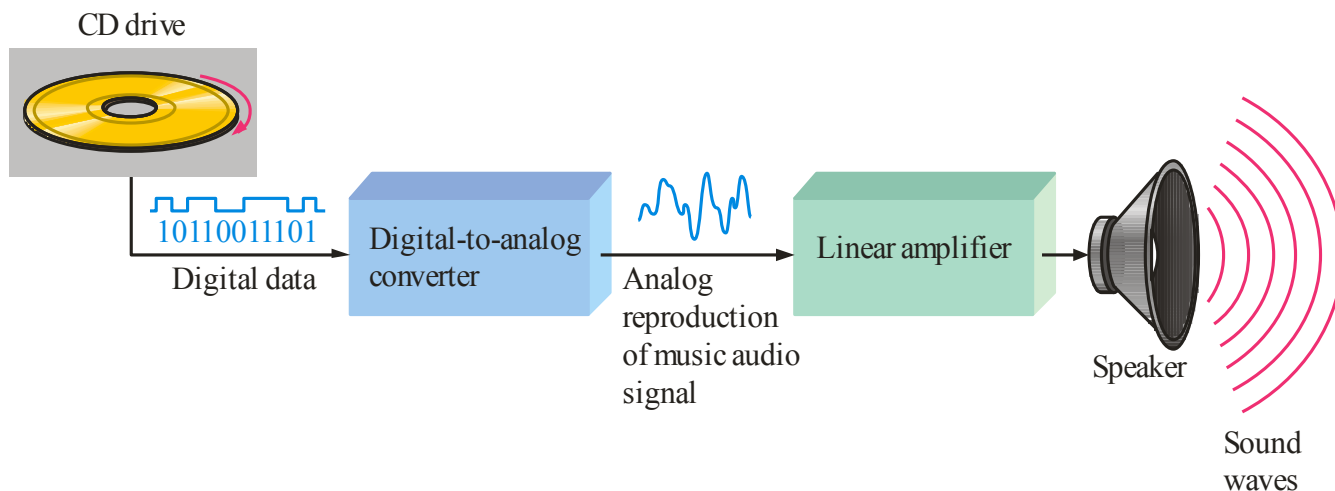


Digital systems can process, store, and transmit data more efficiently but can only assign discrete values to each point.

# Summary

## Analog and Digital Systems

Many systems use a mix of analog and digital electronics to take advantage of each technology. A typical CD player accepts digital data from the CD drive and converts it to an analog signal for amplification.

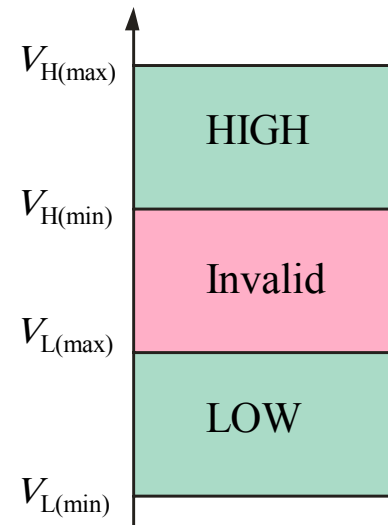


# Summary

## Binary Digits and Logic Levels

Digital electronics uses circuits that have two states, which are represented by two different voltage levels called HIGH and LOW. The voltages represent numbers in the binary system.

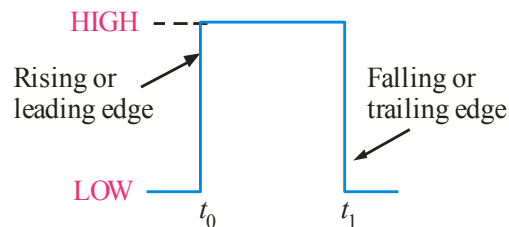
In binary, a single number is called a *bit* (for *binary digit*). A bit can have the value of either a 0 or a 1, depending on if the voltage is HIGH or LOW.



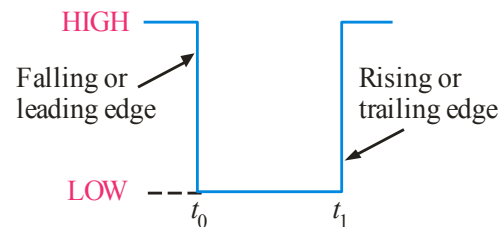
# Summary

## Digital Waveforms

Digital waveforms change between the LOW and HIGH levels. A positive going pulse is one that goes from a normally LOW logic level to a HIGH level and then back again. Digital waveforms are made up of a series of pulses.



(a) Positive-going pulse

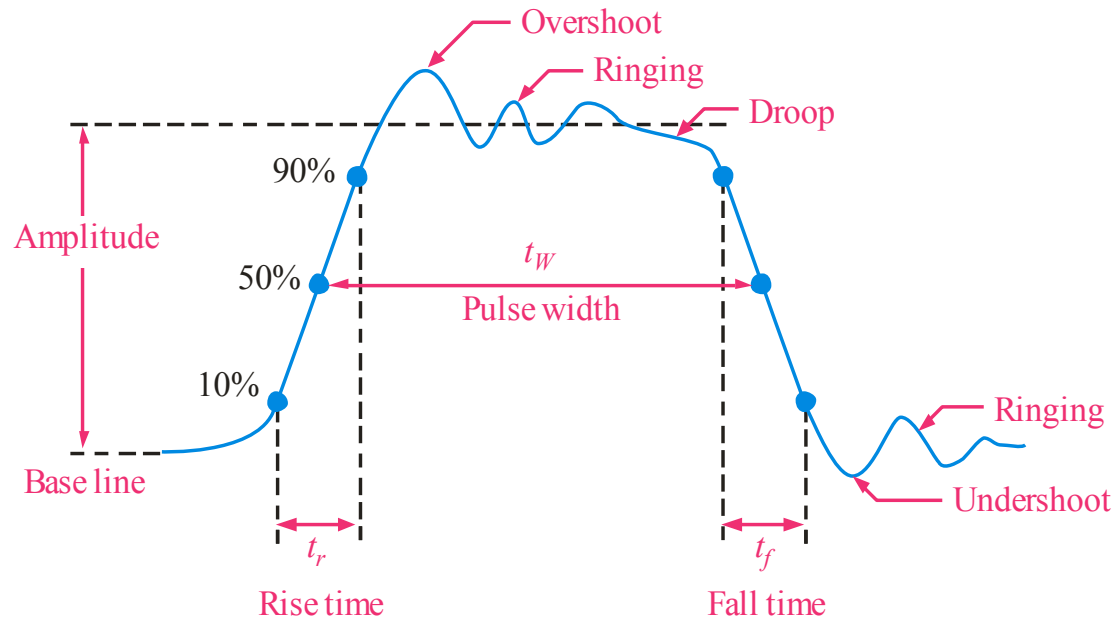


(b) Negative-going pulse

# Summary

## Pulse Definitions

Actual pulses are not ideal but are described by the rise time, fall time, amplitude, and other characteristics.



# Summary

## Periodic Pulse Waveforms

Periodic pulse waveforms are composed of pulses that repeats in a fixed interval called the **period**. The **frequency** is the rate it repeats and is measured in hertz.

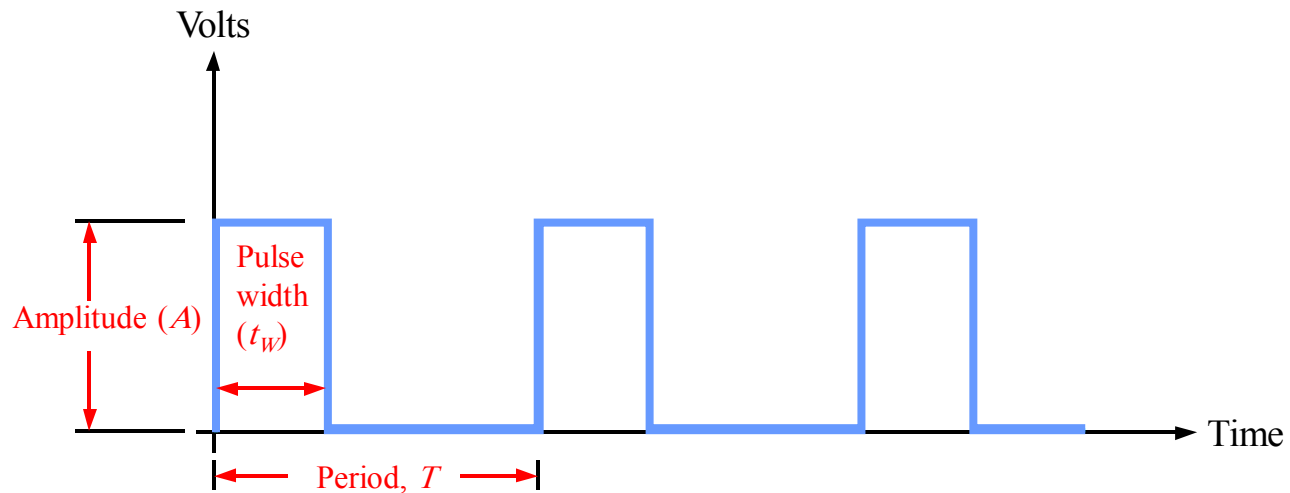
$$f = \frac{1}{T} \qquad T = \frac{1}{f}$$

The **clock** is a basic timing signal that is an example of a periodic wave.

# Summary

## Pulse Definitions

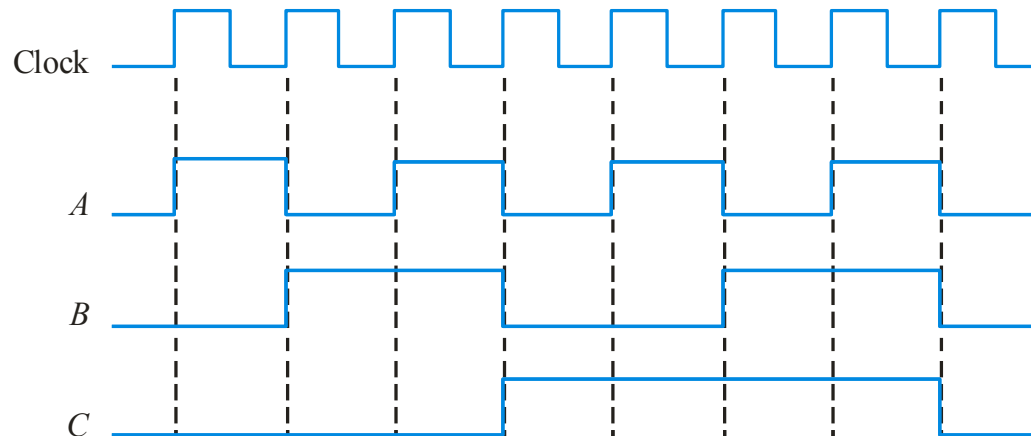
In addition to frequency and period, repetitive pulse waveforms are described by the amplitude ( $A$ ), pulse width ( $t_w$ ) and duty cycle. Duty cycle is the ratio of  $t_w$  to  $T$ .



# Summary

## Timing Diagrams

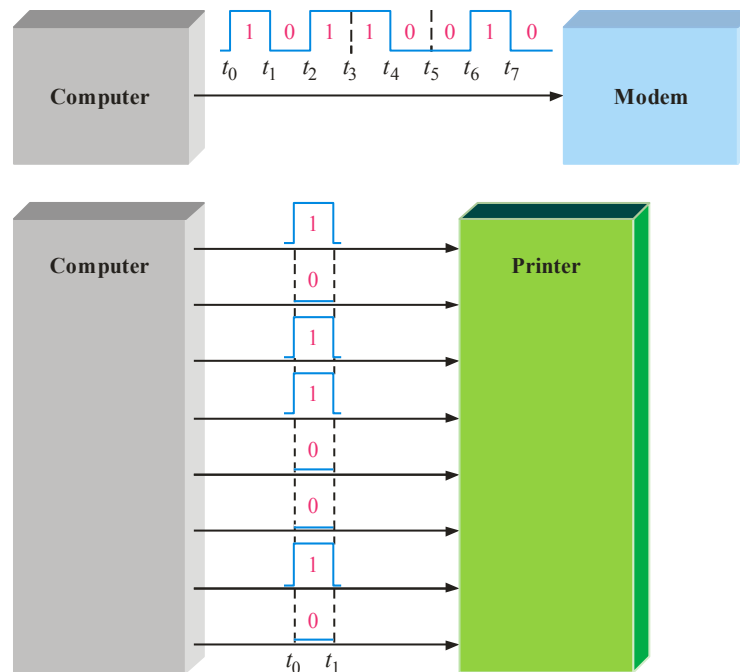
A timing diagram is used to show the relationship between two or more digital waveforms,



# Summary

## Serial and Parallel Data

Data can be transmitted by either serial transfer or parallel transfer.

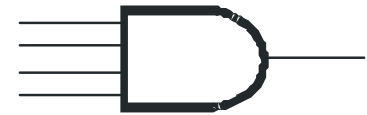


# Summary

## Basic Logic Functions

**AND**

True only if *all* input conditions are true.



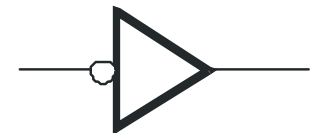
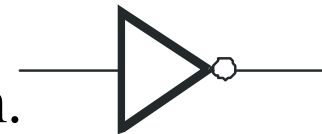
**OR**

True only if *one or more* input conditions are true.



**NOT**

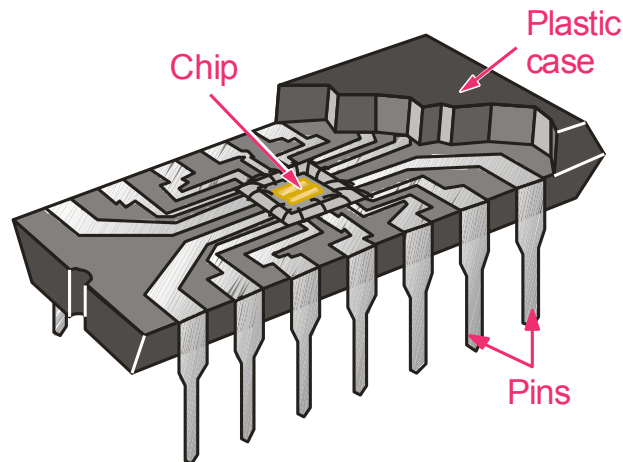
Indicates the *opposite* condition.



# Summary

## Integrated Circuits

Cutaway view of DIP (Dual-In-line Pins) chip:



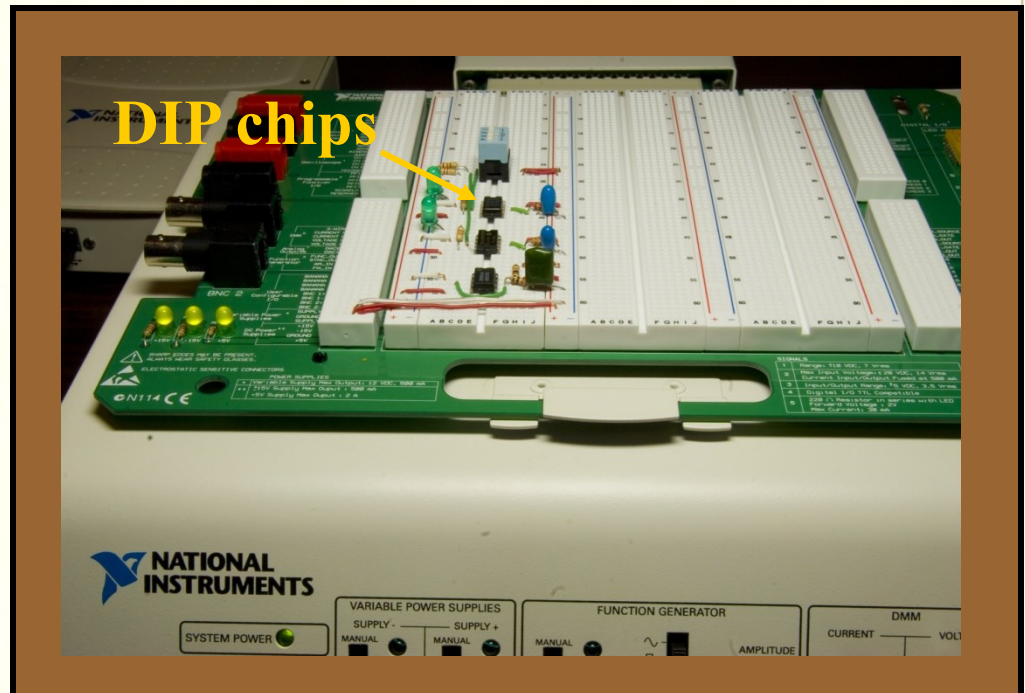
The TTL series, available as DIPs are popular for laboratory experiments with logic.

# Summary

## Integrated Circuits

An example of laboratory prototyping is shown. The circuit is wired using DIP chips and tested.

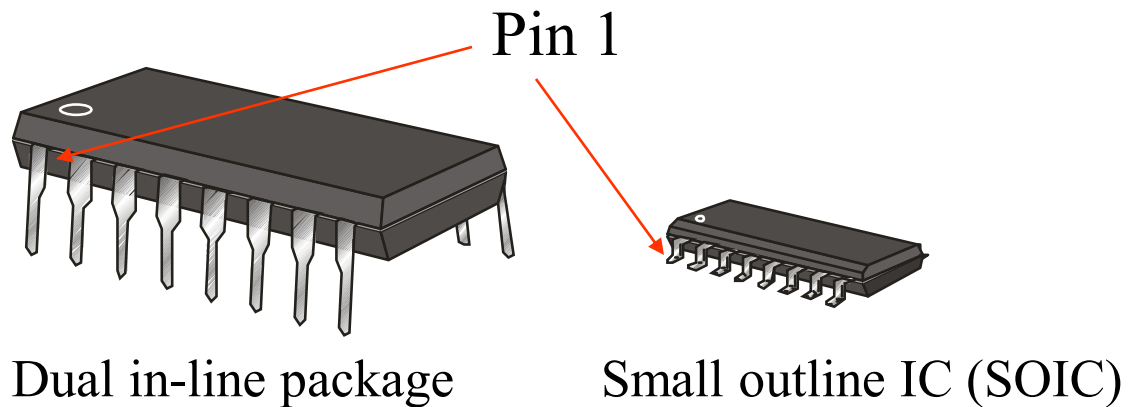
In this case, testing can be done by a computer connected to the system.



# Summary

## Integrated Circuits

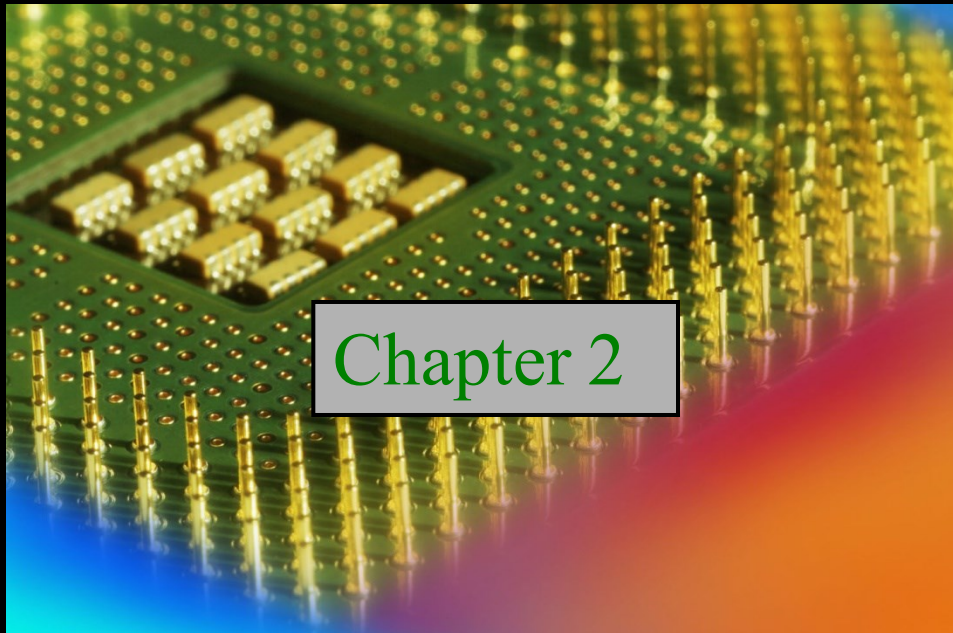
DIP chips and surface mount chips



# Digital Fundamentals

Tenth Edition

Floyd



# Summary

## Decimal Numbers

The radix of decimal numbers is ten, because only ten symbols (0 through 9) are used to represent any number.

The column weights of decimal numbers are powers of ten that increase from right to left beginning with  $10^0 = 1$ :

$\dots 10^5 \ 10^4 \ 10^3 \ 10^2 \ 10^1 \ 10^0$ .

For fractional decimal numbers, the column weights are negative powers of ten that decrease from left to right:

$10^2 \ 10^1 \ 10^0 \ . \ 10^{-1} \ 10^{-2} \ 10^{-3} \ 10^{-4} \ \dots$

# Summary

## Decimal Numbers

Decimal numbers can be expressed as the sum of the products of each digit times the column value for that digit. Thus, the number 9240 can be expressed as

$$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$$

or

$$9 \times 1,000 + 2 \times 100 + 4 \times 10 + 0 \times 1$$

### Example

Express the number 480.52 as the sum of values of each digit.

### Solution

$$480.52 = (4 \times 10^2) + (8 \times 10^1) + (0 \times 10^0) + (5 \times 10^{-1}) + (2 \times 10^{-2})$$

# Summary

## Binary Numbers

For digital systems, the binary number system is used. Binary has a radix of two and uses the digits 0 and 1 to represent quantities.

The column weights of binary numbers are powers of two that increase from right to left beginning with  $2^0 = 1$ :

$$\dots 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0.$$

For fractional binary numbers, the column weights are negative powers of two that decrease from left to right:

$$2^2 \ 2^1 \ 2^0. \ 2^{-1} \ 2^{-2} \ 2^{-3} \ 2^{-4} \dots$$

# Summary

## Binary Numbers

A binary counting sequence for numbers from zero to fifteen is shown.

Notice the pattern of zeros and ones in each column.

| Decimal Number | Binary Number |
|----------------|---------------|
| 0              | 0 0 0 0       |
| 1              | 0 0 0 1       |
| 2              | 0 0 1 0       |
| 3              | 0 0 1 1       |
| 4              | 0 1 0 0       |
| 5              | 0 1 0 1       |
| 6              | 0 1 1 0       |
| 7              | 0 1 1 1       |
| 8              | 1 0 0 0       |
| 9              | 1 0 0 1       |
| 10             | 1 0 1 0       |
| 11             | 1 0 1 1       |
| 12             | 1 1 0 0       |
| 13             | 1 1 0 1       |
| 14             | 1 1 1 0       |
| 15             | 1 1 1 1       |

# Summary

## Binary Conversions

The decimal equivalent of a binary number can be determined by adding the column values of all of the bits that are 1 and discarding all of the bits that are 0.

### Example

Convert the binary number 100101.01 to decimal.

### Solution

Start by writing the column weights; then add the weights that correspond to each 1 in the number.

|       |       |       |       |       |       |               |                                |
|-------|-------|-------|-------|-------|-------|---------------|--------------------------------|
| $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$      | $2^{-2}$                       |
| 32    | 16    | 8     | 4     | 2     | 1     | $\frac{1}{2}$ | $\frac{1}{4}$                  |
| 1     | 0     | 0     | 1     | 0     | 1     | 0             | 1                              |
| 32    |       |       | +4    |       | +1    |               | $+\frac{1}{4} = 37\frac{1}{4}$ |

# Summary

## Binary Conversions

You can convert a decimal whole number to binary by reversing the procedure. Write the decimal weight of each column and place 1's in the columns that sum to the decimal number.

### Example

Convert the decimal number 49 to binary.

### Solution

The column weights double in each position to the right. Write down column weights until the last number is larger than the one you want to convert.

|       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|
| $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| 64    | 32    | 16    | 8     | 4     | 2     | 1     |
| 0     | 1     | 1     | 0     | 0     | 0     | 1     |

# Summary

## Binary Conversions

You can convert a **decimal fraction** to binary by repeatedly multiplying the fractional results of successive multiplications by 2. **The carries form the binary number.**

**Example** Convert the decimal fraction 0.188 to binary by repeatedly multiplying the fractional results by 2.

**Solution**

|                          |           |          |
|--------------------------|-----------|----------|
| $0.188 \times 2 = 0.376$ | carry = 0 | MSB<br>↓ |
| $0.376 \times 2 = 0.752$ | carry = 0 |          |
| $0.752 \times 2 = 1.504$ | carry = 1 |          |
| $0.504 \times 2 = 1.008$ | carry = 1 |          |
| $0.008 \times 2 = 0.016$ | carry = 0 |          |

Answer = **.00110** (for five significant digits)

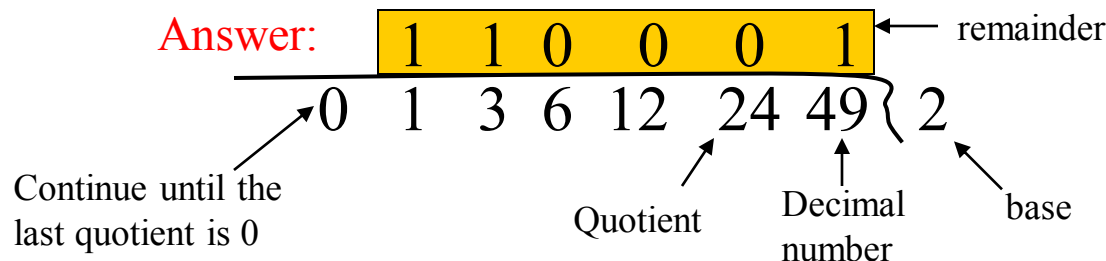
# Summary

## Binary Conversions

You can convert decimal to any other base by repeatedly dividing by the base. For binary, repeatedly divide by 2:

**Example** Convert the decimal number 49 to binary by repeatedly dividing by 2.

**Solution** You can do this by “reverse division” and the answer will read from left to right. Put quotients to the left and remainders on top.



# Summary

## Binary Addition

The rules for binary addition are

|              |                    |
|--------------|--------------------|
| $0 + 0 = 0$  | Sum = 0, carry = 0 |
| $0 + 1 = 0$  | Sum = 1, carry = 0 |
| $1 + 0 = 0$  | Sum = 1, carry = 0 |
| $1 + 1 = 10$ | Sum = 0, carry = 1 |

When an input carry = 1 due to a previous result, the rules are

|                  |                    |
|------------------|--------------------|
| $1 + 0 + 0 = 01$ | Sum = 1, carry = 0 |
| $1 + 0 + 1 = 10$ | Sum = 0, carry = 1 |
| $1 + 1 + 0 = 10$ | Sum = 0, carry = 1 |
| $1 + 1 + 1 = 11$ | Sum = 1, carry = 1 |

# Summary

## Binary Addition

**Example** Add the binary numbers 00111 and 10101 and show the equivalent decimal addition.

**Solution**

$$\begin{array}{r} \textcolor{red}{0\ 1\ 1\ 1} \\ 00111 \quad 7 \\ 10101 \quad 21 \\ \hline 11100 \textcolor{red}{=} 28 \end{array}$$

# Summary

## Binary Subtraction

The rules for binary subtraction are

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$10 - 1 = 1, 0-1 \text{ with a borrow of } 1$$

**Example** Subtract the binary number 00111 from 10101 and show the equivalent decimal subtraction.

**Solution**

$$\begin{array}{r} \phantom{0}1\phantom{0}1\phantom{0}1 \\ \underline{00111} \\ 01110 \end{array} \quad \begin{array}{r} 21 \\ \underline{7} \\ 14 \end{array}$$

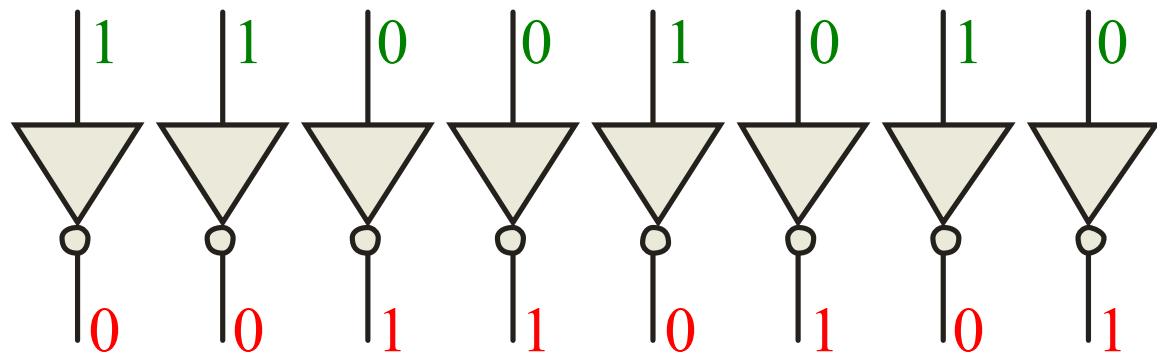
# Summary

## 1's Complement

The 1's complement of a binary number is just the inverse of the digits. To form the 1's complement, change all 0's to 1's and all 1's to 0's.

For example, the 1's complement of **11001010** is  
**00110101**

In digital circuits, the 1's complement is formed by using inverters:



# Summary

## 2's Complement

The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

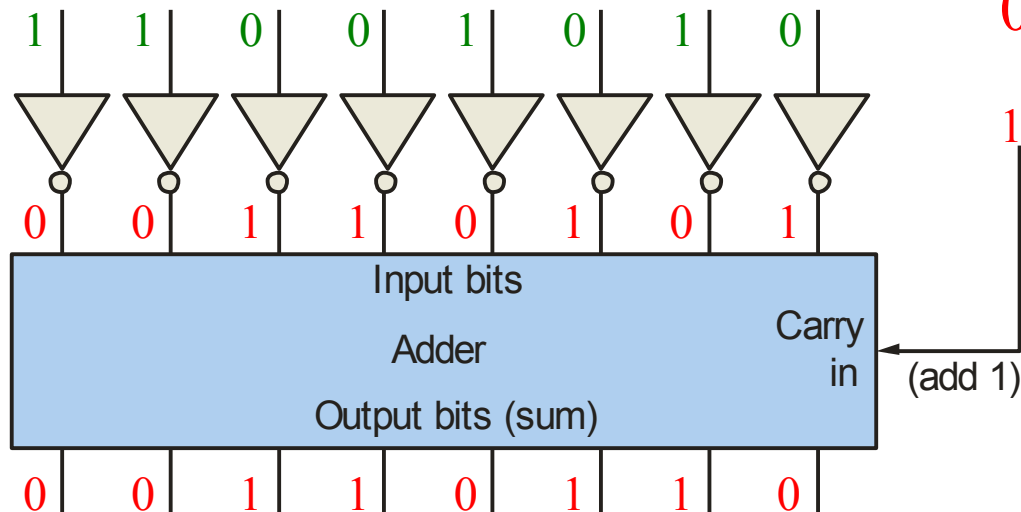
Recall that the 1's complement of **11001010** is

**00110101** (1's complement)

To form the 2's complement, add 1:

$$\begin{array}{r} 00110101 \\ +1 \\ \hline 00110110 \end{array}$$

(2's complement)



# Summary

## Signed Binary Numbers

There are several ways to represent signed binary numbers. In all cases, **the MSB in a signed number is the sign bit**, that tells you if the number is positive or negative.

Computers use a modified 2's complement for signed numbers. **Positive numbers** are stored in *true* form (with **a 0 for the sign bit**) and negative numbers are stored in *complement* form (with a 1 for the sign bit).

For example, the positive number 58 is written using 8-bits as

**00111010** (true form).

Sign bit

Magnitude bits

# Summary

## Signed Binary Numbers

- **Negative** numbers are written as the **2's complement** of the corresponding positive number.

The negative number  $-58$  is written as:

$$-58 = 11000110 \text{ (complement form)}$$

Sign bit

Magnitude bits

- An easy way to read a signed number that uses this notation is to assign the sign bit a column weight of  $-128$  (for an 8-bit number).
- Then add the column weights for the 1's.

**Example**

Assuming that the sign bit =  $-128$ , show that  $11000110 = -58$  as a 2's complement signed number:

**Solution**

Column weights:  $-128$  64 32 16 8 4 2 1.

$$\begin{array}{cccccccc} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ -128 & +64 & & & & +4 & +2 & = -58 \end{array}$$

# Summary

## Arithmetic Operations with Signed Numbers

- Using the signed number notation with negative numbers in 2's complement form simplifies addition and subtraction of signed numbers.
- Rules for **addition**:
  - ✓ Add the two signed numbers.
  - ✓ **Discard** any final carries.
  - ✓ The result is in signed form.
- Examples:

$$00011110 = +30$$

$$00001111 = +15$$

$$00101101 = +45$$

$$00001110 = +14$$

$$11101111 = -17$$

$$11111101 = -3$$

$$11111111 = -1$$

$$11111000 = -8$$

$$11110111 = -9$$

Discard carry

# Summary

## Arithmetic Operations with Signed Numbers

- Note that if the number of bits required for the answer is exceeded, overflow will occur.
- This occurs only if both numbers have the same sign. The overflow will be indicated by an incorrect sign bit.

Two examples are:

$$01000000 = +128$$

$$01000001 = +129$$

$$10000001 = -126$$

$$10000001 = -127$$

$$10000001 = -127$$

Discard carry →  $100000010 = +2$

**Wrong!** The answer is incorrect  
and the sign bit has changed.

# Summary

## Rules for subtraction

- 2's complement the subtrahend and add the numbers.
- Discard any final carries.
- The result is in signed form.

Repeat the examples done previously, but subtract:

$$\begin{array}{r} 00011110 \quad (+30) \\ - 00001111 \quad -(+15) \\ \hline \end{array} \quad \begin{array}{r} 00001110 \quad (+14) \\ - 11101111 \quad -(-17) \\ \hline \end{array} \quad \begin{array}{r} 11111111 \quad (-1) \\ - 11111000 \quad -(-8) \\ \hline \end{array}$$

2's complement subtrahend and add:

$$\begin{array}{r} 00011110 = +30 \\ 11110001 = -15 \\ \hline 100001111 = +15 \end{array} \quad \begin{array}{r} 00001110 = +14 \\ 00010001 = +17 \\ \hline 00011111 = +31 \end{array} \quad \begin{array}{r} 11111111 = -1 \\ 00001000 = +8 \\ \hline 100000111 = +7 \end{array}$$

Discard carry

Discard carry

## Hexadecimal Numbers

- Hexadecimal uses sixteen characters to represent numbers: the numbers 0 through 9 and the alphabetic characters A through F.
- Large **binary number** can easily be converted to hexadecimal by grouping bits **4 at a time** and writing the equivalent hexadecimal character.

**Example** Express 1001 0110 0000 1110<sub>2</sub> in hexadecimal:

**Solution** Group the binary number by 4-bits starting from the right. Thus, **960E**

| Decimal | Hexadecimal | Binary |
|---------|-------------|--------|
| 0       | 0           | 0000   |
| 1       | 1           | 0001   |
| 2       | 2           | 0010   |
| 3       | 3           | 0011   |
| 4       | 4           | 0100   |
| 5       | 5           | 0101   |
| 6       | 6           | 0110   |
| 7       | 7           | 0111   |
| 8       | 8           | 1000   |
| 9       | 9           | 1001   |
| 10      | A           | 1010   |
| 11      | B           | 1011   |
| 12      | C           | 1100   |
| 13      | D           | 1101   |
| 14      | E           | 1110   |
| 15      | F           | 1111   |

## Hexadecimal Numbers

- Hexadecimal is a weighted number system.
- The column weights are powers of 16, which increase from right to left.

$$\text{Column weights } \begin{cases} 16^3 & 16^2 & 16^1 & 16^0 \\ 4096 & 256 & 16 & 1 \end{cases}$$

**Example** Express  $1A2F_{16}$  in decimal.

**Solution** Start by writing the column weights:

$$\begin{array}{cccc} 4096 & 256 & 16 & 1 \\ 1 & A & 2 & F_{16} \end{array}$$

$$1(4096) + 10(256) + 2(16) + 15(1) = 6703_{10}$$

## Octal Numbers

- Octal uses eight characters the numbers 0 through 7 to represent numbers.
- There is no 8 or 9 character in octal.
- Binary number can easily be converted to octal by grouping bits **3 at a time** and writing the equivalent octal character for each group.

**Example** Express 1 001 011 000 001 110<sub>2</sub> in octal:

**Solution** Group the binary number by 3-bits starting from the right. Thus, **113016<sub>8</sub>**

| Decimal | Octal | Binary |
|---------|-------|--------|
| 0       | 0     | 0000   |
| 1       | 1     | 0001   |
| 2       | 2     | 0010   |
| 3       | 3     | 0011   |
| 4       | 4     | 0100   |
| 5       | 5     | 0101   |
| 6       | 6     | 0110   |
| 7       | 7     | 0111   |
| 8       | 10    | 1000   |
| 9       | 11    | 1001   |
| 10      | 12    | 1010   |
| 11      | 13    | 1011   |
| 12      | 14    | 1100   |
| 13      | 15    | 1101   |
| 14      | 16    | 1110   |
| 15      | 17    | 1111   |

## Octal Numbers

- Octal is also a weighted number system.
- The column weights are powers of 8, which increase from right to left.

$$\text{Column weights } \begin{cases} 8^3 & 8^2 & 8^1 & 8^0 \\ 512 & 64 & 8 & 1 \end{cases}$$

**Example** Express  $3702_8$  in decimal.

**Solution** Start by writing the column weights:

$$\begin{array}{cccc} 512 & 64 & 8 & 1 \\ 3 & 7 & 0 & 2_8 \end{array}$$

$$3(512) + 7(64) + 0(8) + 2(1) = 1986_{10}$$

## BCD

- Binary coded decimal (BCD) is a weighted code that is commonly used in digital systems when it is necessary to show decimal numbers such as in clock displays.
- The table illustrates the difference between straight binary and BCD. BCD represents each decimal digit with a 4-bit code.
- Notice that the codes 1010 through 1111 are not used in BCD.

| Decimal | Binary | BCD       |
|---------|--------|-----------|
| 0       | 0000   | 0000      |
| 1       | 0001   | 0001      |
| 2       | 0010   | 0010      |
| 3       | 0011   | 0011      |
| 4       | 0100   | 0100      |
| 5       | 0101   | 0101      |
| 6       | 0110   | 0110      |
| 7       | 0111   | 0111      |
| 8       | 1000   | 1000      |
| 9       | 1001   | 1001      |
| 10      | 1010   | 0001 0000 |
| 11      | 1011   | 0001 0001 |
| 12      | 1100   | 0001 0010 |
| 13      | 1101   | 0001 0011 |
| 14      | 1110   | 0001 0100 |
| 15      | 1111   | 0001 0101 |

## BCD

- You can think of BCD in terms of column weights in groups of four bits.
- For an 8-bit BCD number, the column weights are:

80 40 20 10 8 4 2 1.

**Question:** What are the column weights for the BCD number 1000 0011 0101 1001?

**Answer:**

8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1

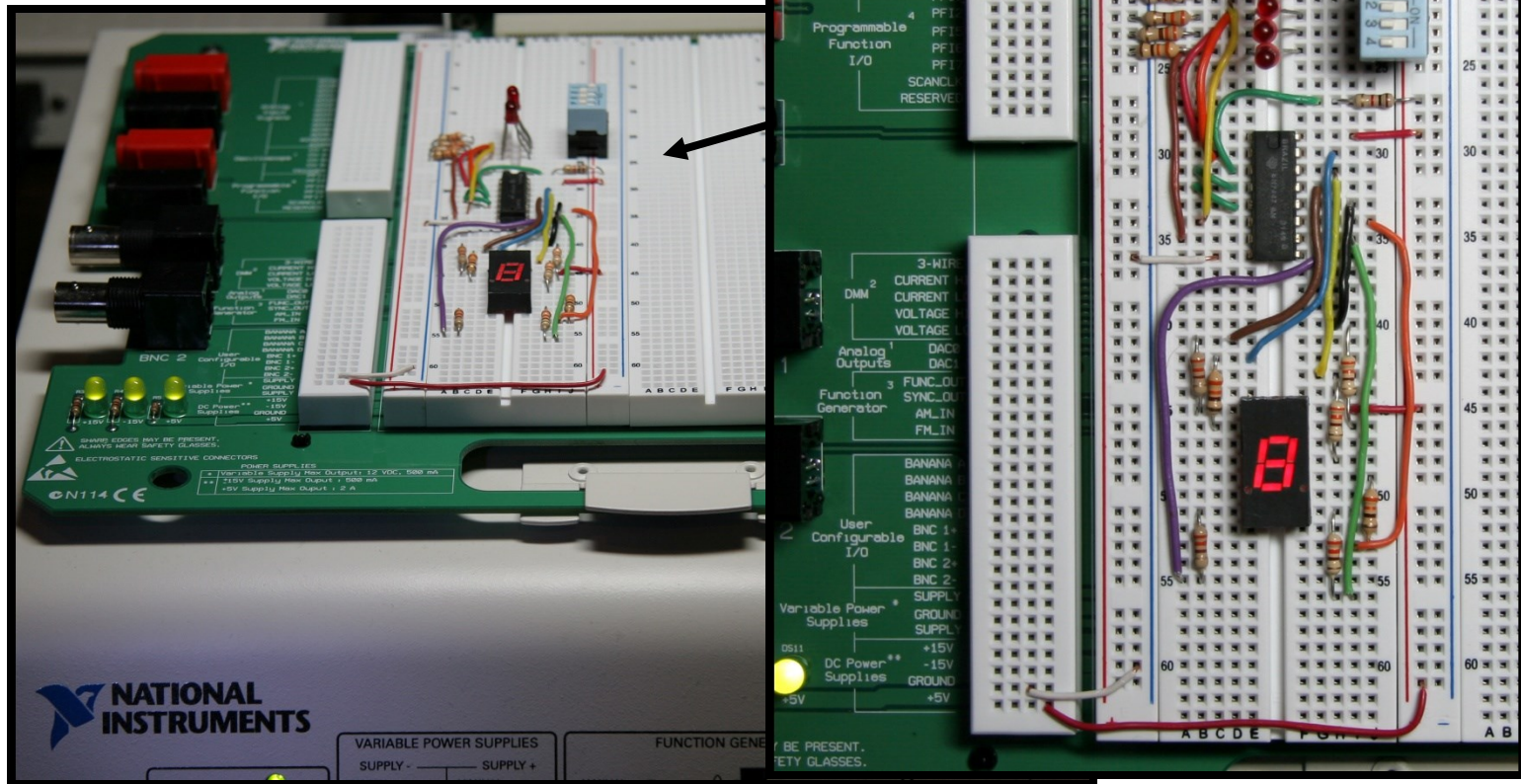
Note that you could add the column weights where there is a 1 to obtain the decimal number. For this case:

$$8000 + 200 + 100 + 40 + 10 + 8 + 1 = 8359_{10}$$

# Summary

## BCD

A lab experiment in which BCD is converted to decimal is shown.



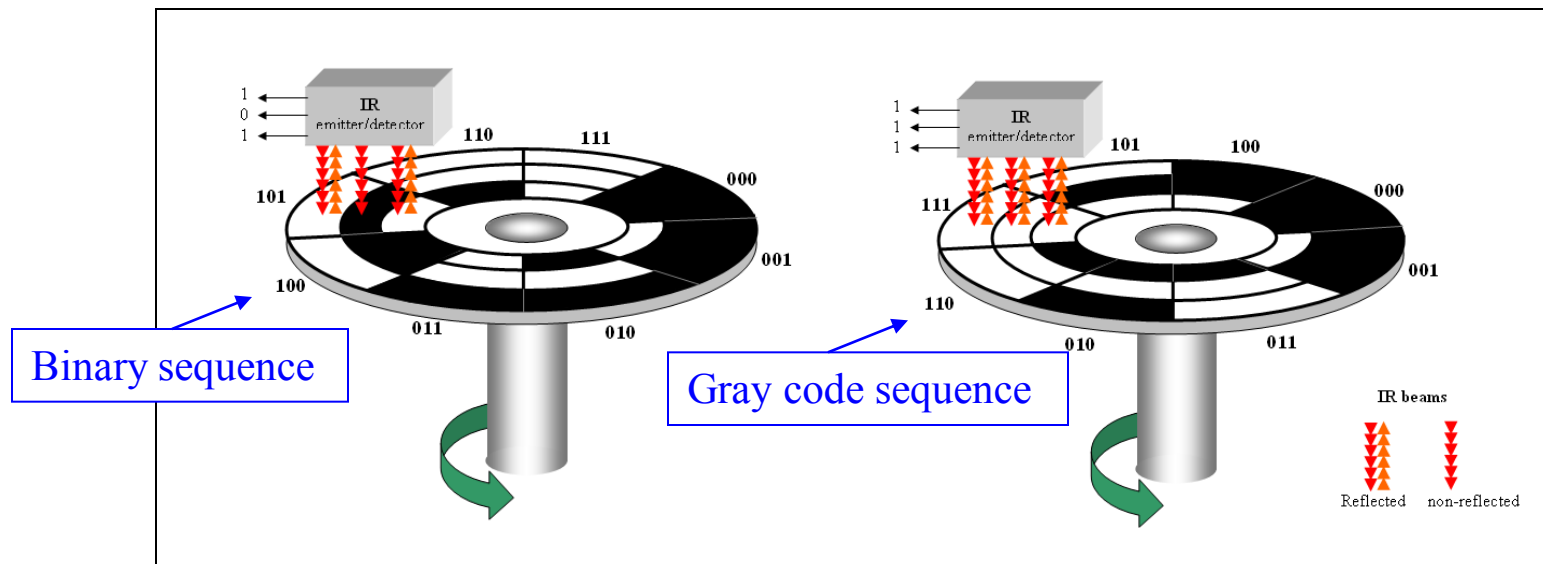
## Gray code

- Gray code is an **unweighted** code that has a single bit change between one code word and the next in a sequence.
- Gray code is used to avoid problems in systems where an error can occur if more than one bit changes at a time.

| Decimal | Binary | Gray code |
|---------|--------|-----------|
| 0       | 0000   | 0000      |
| 1       | 0001   | 0001      |
| 2       | 0010   | 0011      |
| 3       | 0011   | 0010      |
| 4       | 0100   | 0110      |
| 5       | 0101   | 0111      |
| 6       | 0110   | 0101      |
| 7       | 0111   | 0100      |
| 8       | 1000   | 1100      |
| 9       | 1001   | 1101      |
| 10      | 1010   | 1111      |
| 11      | 1011   | 1110      |
| 12      | 1100   | 1010      |
| 13      | 1101   | 1011      |
| 14      | 1110   | 1001      |
| 15      | 1111   | 1000      |

## Gray code

- A shaft encoder is a typical application.
- Three IR emitter/detectors are used to encode the position of the shaft.
- The encoder on the left uses binary and can have three bits change together, creating a potential error.
- The encoder on the right uses gray code and only 1-bit changes, eliminating potential errors.



## ASCII

- ASCII is a code for alphanumeric characters and control characters.
- In its **original form**, ASCII encoded 128 characters and symbols using 7-bits. The first 32 characters are control characters, that are based on obsolete teletype requirements, so these characters are generally assigned to other functions in modern usage.
- In 1981, IBM introduced **extended ASCII**, which is an 8-bit code and increased the character set to 256. Other extended sets (such as Unicode) have been introduced to handle characters in languages other than English.