



Department of Computer Science & Engineering (CSE)

---

# Course Title: Digital Logic Design

Shahriar Rahman Khan

Lecturer

Dept. of CSE, MIST

Course Code: CSE 103

Credit Hr: 3.00

Contact Hr: 3.00



# Text Books

---

- Digital Fundamentals – T L Floyd
- Digital Logic and Computer Design Fundamentals, M. Morris Mano.
- Digital Principles and Logic Design  
A Saha and N Manna
- Digital Logic Design Brian  
Holdsworth and Clive Woods



# Grading

---

Description	Percentage
Class Performance and Attendance	$5\% + 5\% = 10\%$
Class Test/ Assignment	20%
Mid Term Assessment (Exam / Project)	10%
Final Examination (Section A & B)	60%
<b>Total</b>	<b>100%</b>



# Course Objectives

---

1. To understand the different boolean algebra theorems and apply them for simplifying logic functions.
2. To understand the Karnaugh map and other methods to perform an algorithmic reduction of multivariable logic functions.
3. To understand the usefulness of combinational circuits: adder, subtractor, code converters encoders/decoders, multiplexers, demultiplexers, ROM, RAM, PLAs.
4. To design and analysis of clocked sequential circuits, flip-flops, state diagrams, state tables, different latches.
5. To understand the analysis of various registers, shift-registers, counters, and how more complex systems are constructed.



# Overview

---

- **Information Representation**
- **Number Systems [binary, octal and hexadecimal]**
- **Arithmetic Operations**
- **Base Conversion**
- **Complements**
- **Signed Binary Numbers**
- **Binary Codes**
- **Decimal Codes**
- **Gray Codes**
- **Alphanumeric Codes**
- **Parity Bit**
- **Binary Logic**



# Outline

---

- Introduction to DLD
- Information Presentation
- Binary Numbers
- Arithmetic Operations
- Number Base Conversion
- Octal and Hexadecimal Numbers



# Introduction to Digital Logic Design

---

- Digital logic design means designing digital electronic circuits (computers, video games, watch) & logic gates in an efficient and optimized way.
- A Digital system is a system that includes logic gates, flip-flops, shifts, registers, etc. A computer is the best example of a digital system.
- One of the main characteristics of the digital system is using discrete elements of information to make something meaningful. Like 2, 3, 7 are 3 different discrete elements but using numerical computation, the computer can make it 237. Same as 'd', 'o', 'g'. These are discrete letters, but 'dog' is a meaningful word.



# INFORMATION REPRESENTATION - Signals

---

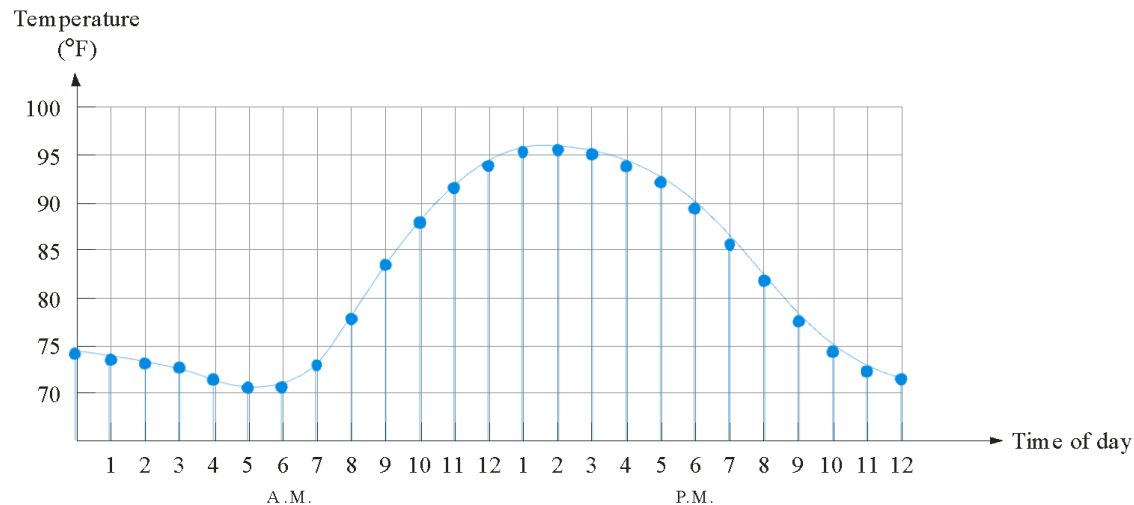
- What is signal?
- It is a function, that represents the variation of a physical quantity with respect to any parameter. Here parameter is independent(time, distance).
- Physical quantity means voltage and current.
- For example,  $f(x) = ax^2 + bx + c$
- Here  $x$  is independent and, this function depends on  $x$ .
- Digital signal is used in digital electronic devices.
- There are 2 types of signals- analog and digital signals.
- Analog signal gives a continuous value and digital signal gives a discrete value.
- Mainly digital signals are the discretization of analog signals.





# INFORMATION REPRESENTATION - Signals

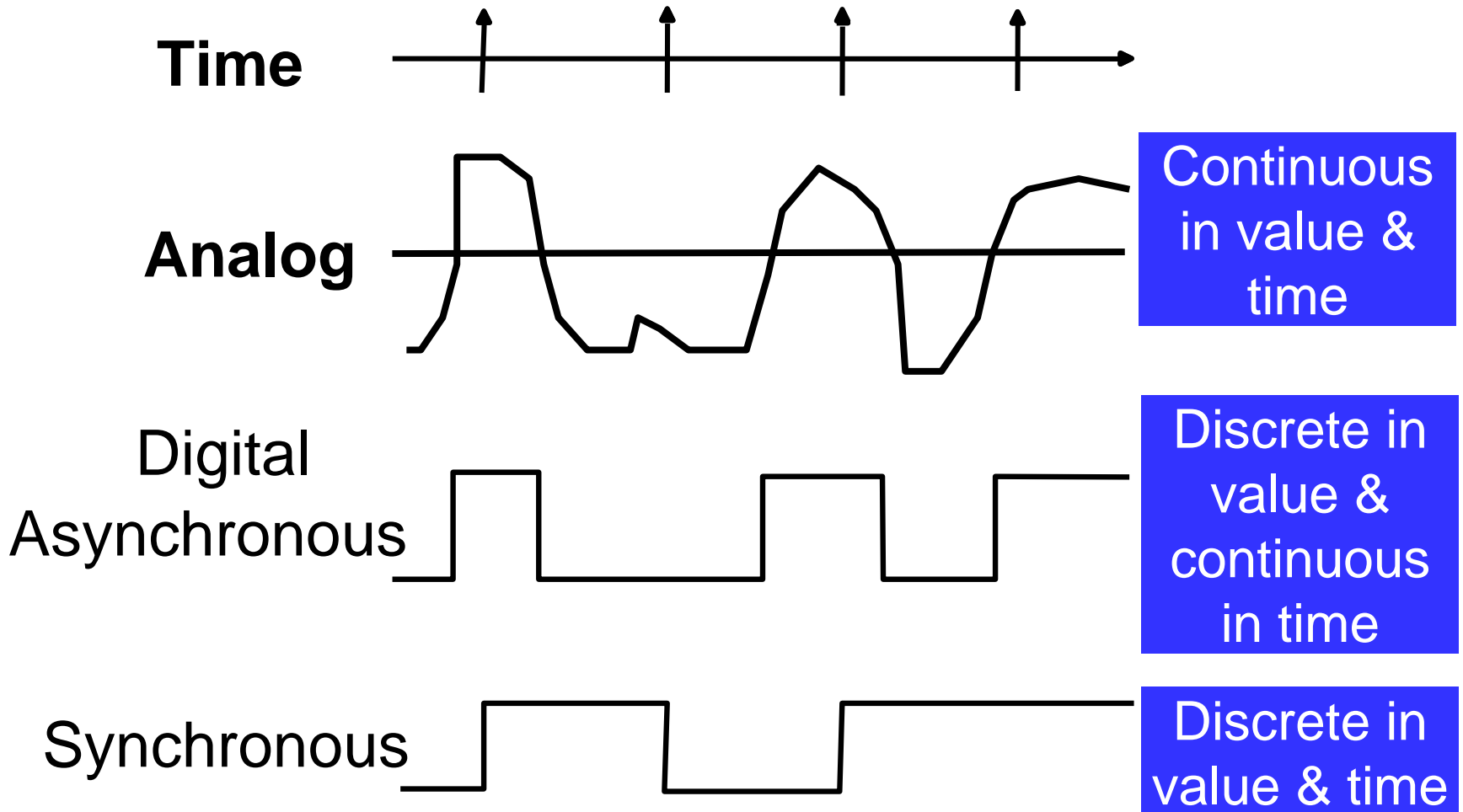
Most natural quantities that we see are **analog** and vary continuously. Analog systems can generally handle higher power than digital systems.



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



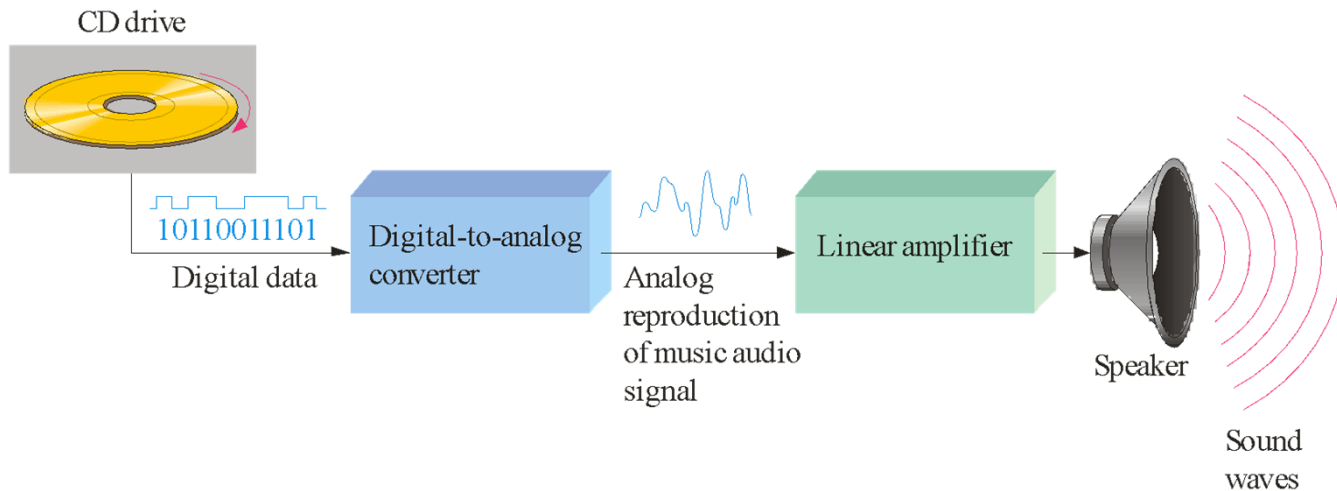
# Signal Examples Over Time





# INFORMATION REPRESENTATION - Signals

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.

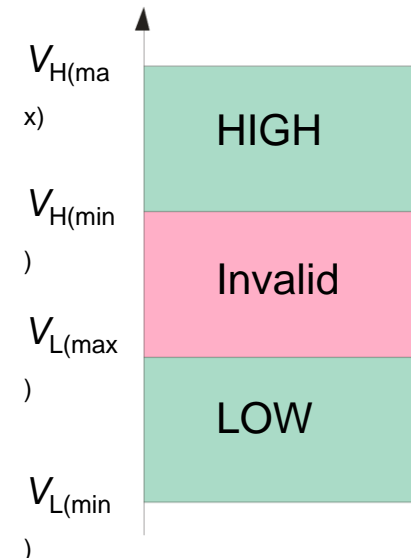




# INFORMATION REPRESENTATION - Signals

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. Variables are used to store the digital signals. Normally there are 2 discrete values in a digital signal. That is why it is called binary value.

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.





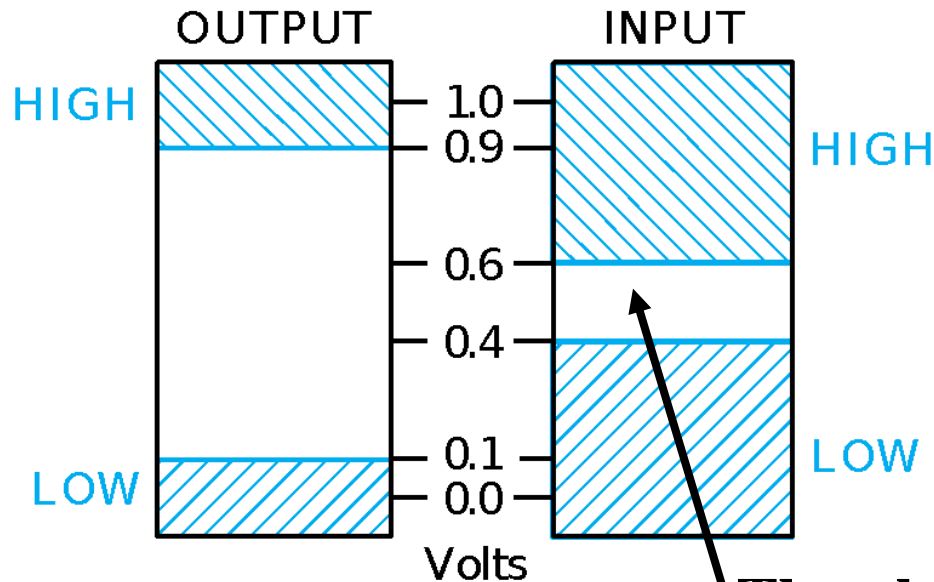
# INFORMATION REPRESENTATION - Signals

---

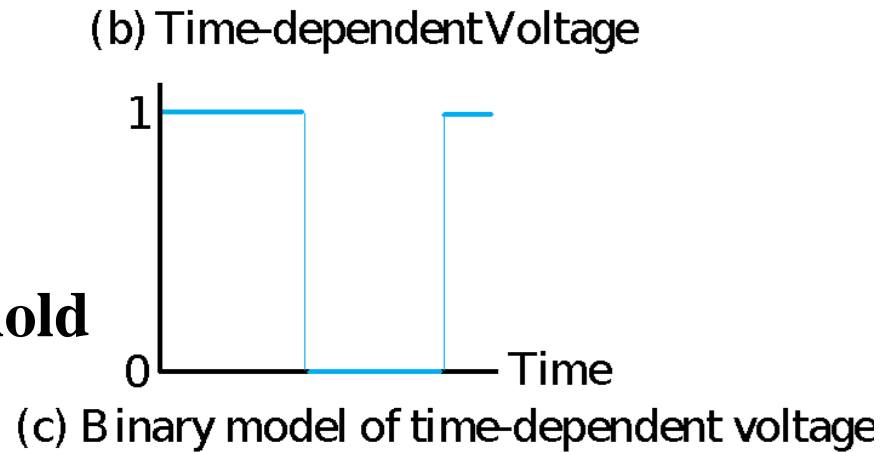
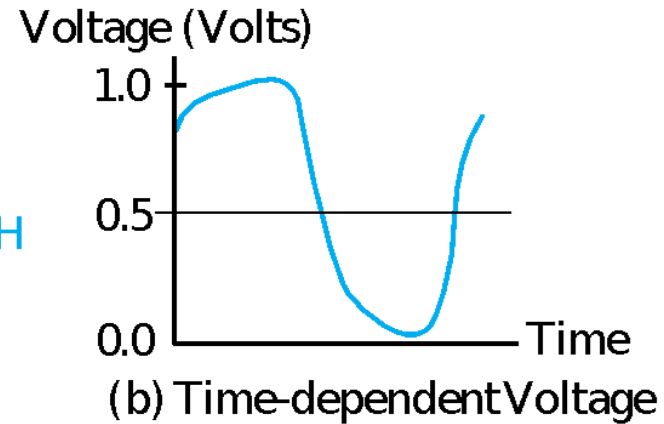
- In a binary system, we have 2 digits- 0 & 1.
- To represent the state of these 2 bits, we normally use 1 for high voltage and 0 for low voltage.
- It is called positive logic.



# Signal Example – Physical Quantity: Voltage



(a) Example voltage ranges



**Threshold  
Region**



# INFORMATION REPRESENTATION – Why Use Digital Signal?

---

- Analog signals can be highly affected by noise, but digital signals are noise immune.
- Analog signals continuously changes with respect to time, so which part of the signal is infected by noise, really difficult to identify.
- Digital systems are more accurate, and the probability of error occurrence can be reduced by employing error detection and correction codes.
- Digital signal processing is more secure because digital information can be easily encrypted and compressed.
- Digital signals can be transmitted over long distances.



# NUMBER SYSTEMS – Representation

- Positive radix, positional number systems
- A number with *radix*  $r$  is represented by a string of digits:

$A_{n-1}A_{n-2} \dots A_1A_0 \cdot A_{-1}A_{-2} \dots A_{-m+1}A_{-m}$   
in which  $0 \leq A_i < r$  and  $.$  is the *radix point*.

- The string of digits represents the power series:

$$\begin{aligned} (\text{Number})_r &= \left( \sum_{i=0}^{i=n-1} A_i \cdot r^i \right) + \left( \sum_{j=-1}^{j=-m} A_j \cdot r^j \right) \\ &\quad (\text{Integer Portion}) + (\text{Fraction Portion}) \end{aligned}$$





# Commonly Occurring Bases

---

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- The six letters (in addition to the 10 integers) in hexadecimal represent:



# Decimal Number System

---

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

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

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

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



# Binary Number System

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:

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



# Why Octal and Hexadecimal Numbers

---

- Why octal number system is used?

It uses fewer digits than the decimal and Hexa number system. So it has lower computations. Besides, it takes only 3 digits to convert from binary to octal and, it is easy to convert from octal to binary. The use of octal numbers **has declined as most modern computers no longer base their word length on multiples of three bits**, (they are based on multiples of four bits, so hexadecimal is more widely used). Decimal number system has only ten (10) digits from 0 to 9.

- Why Hexa number system is used?

Computers use binary numbering system while humans use hexadecimal numbering system to shorten binary and make it easier to understand. It takes only 4 digits to convert from binary to hexa which means we need only 2 hex digits to represent an 8-bit binary number. We need hexadecimal number system in microprocessor coding.



# ARITHMETIC OPERATIONS - Binary Arithmetic

---

- Single Bit Addition with Carry
- Multiple Bit Addition
- Single Bit Subtraction with Borrow
- Multiple Bit Subtraction
- Multiplication
- BCD Addition



# Binary Numbers

---

- Two possible values: 0 or 1
- Base 2
- Bit= Binary digits (0 or 1)
- 1 Byte=8 bits



# Single Bit Binary Addition with Carry

---

The rules for binary addition are

$$0 + 0 = \text{Sum} = 0, \text{carry} = 0$$

$$0 + 1 = \text{Sum} = 1, \text{carry} = 0$$

$$1 + 0 = \text{Sum} = 1, \text{carry} = 0$$

$$1 + 1 = \text{Sum} = 0, \text{carry} = 1$$

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

$$1 + 0 + 0 = 01 \quad \text{Sum} = 1, \text{carry} = 0$$

$$1 + 0 + 1 = 10 \quad \text{Sum} = 0, \text{carry} = 1$$

$$1 + 1 + 0 = 10 \quad \text{Sum} = 0, \text{carry} = 1$$

$$1 + 1 + 1 = 11 \quad \text{Sum} = 1, \text{carry} = 1$$



# Multiple Bit Binary Addition

- Extending this to two multiple bit examples:

Carries	<u>0</u>	<u>0</u>
Augend	01100	10110
Addend	<u>+10001</u>	<u>+10111</u>
Sum		

- Note: The 0 is the default Carry-In to the least significant bit.





# Single Bit Binary Subtraction with Borrow

The rules for binary subtraction are

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

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

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

$$\begin{array}{r} \overset{1}{\cancel{1}}\overset{1}{\cancel{0}}\overset{1}{\cancel{1}}01 \quad 21 \\ \underline{00111} \quad \underline{7} \\ 01110 = 14 \end{array}$$



# Multiple Bit Binary Subtraction

- Extending this to two multiple bit examples:

Borrows	<u>0</u>	<u>0</u>
Minuend	10110	10110
Subtrahend	<u>- 10010</u>	<u>- 10011</u>
Difference		

- Notes: The 0 is a Borrow-In to the least significant bit. If the Subtrahend > the Minuend, interchange and append a – to the result.



# Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form  $\Sigma$  (digit  $\times$  respective power of 2).
- Example: Convert  $11010_2$  to  $N_{10}$ :

Powers of 2:

$$11010_2 \Rightarrow 1 \times 2^4 = 16$$

$$0.101 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$2 + 1 \times 2^{-3}$$

$$+ 1 \times 2^3 = 8$$

$$+ 0 \times 2^2 = 0$$

$$+ 1 \times 2^1 = 2$$

$$+ 0 \times 2^0 = \underline{0}$$

$$26_{10}$$



# Converting Decimal to Binary

- Method 1

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.

Convert the decimal number 49 to binary.

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



# Conversion Between Bases

---

- Method 2
- To convert from one base to another:
  - 1) Convert the Integer Part
  - 2) Convert the Fraction Part
  - 3) Join the two results with a radix point



# Conversion Details

---

- To Convert the Integral Part:

Repeatedly divide the number by the new radix and save the remainders. The digits for the new radix are the remainders in *reverse order* of their computation. If the new radix is  $> 10$ , then convert all remainders  $> 10$  to digits A, B, ...

- To Convert the Fractional Part:

Repeatedly multiply the fraction by the new radix and save the integer digits that result. The digits for the new radix are the integer digits in *order* of their computation. If the new radix is  $> 10$ , then convert all integers  $> 10$  to digits A, B, ...



# Example: Convert $46.6875_{10}$ To Base 2

---

- Convert 46 to Base 2
- Convert 0.6875 to Base 2:
- Join the results together with the radix point:



# Example: Convert $46.6875_{10}$ To Base 2

Answer 1: Converting 46 as integral part:  
fractional part:

$$46/2 = 23 \text{ rem} = 0$$

$$23/2 = 11 \text{ rem} = 1$$

$$11/2 = 5 \text{ remainder} = 1$$

$$5/2 = 2 \text{ remainder} = 1$$

$$2/2 = 1 \text{ remainder} = 0$$

$$1/2 = 0 \text{ remainder} = 1$$

Reading off in the  
reverse direction:  $101110_2$

Answer 2: Converting 0.6875 as

$$0.6875 * 2 = 1.3750 \text{ int} = 1$$

$$0.3750 * 2 = 0.7500 \text{ int} = 0$$

$$0.7500 * 2 = 1.5000 \text{ int} = 1$$

$$0.5000 * 2 = 1.0000 \text{ int} = 1$$

$$0.0000$$

Reading off in the forward  
direction:  $0.1011_2$

Answer 3: Combining Integral and Fractional Parts:  
 $101110.1011_2$





# Additional Issue - Fractional Part

---

- Note that in this conversion, the fractional part can become 0 as a result of the repeated multiplications.
- In general, it may take many bits to get this to happen or it may never happen.
- Example Problem: Convert  $0.65_{10}$  to  $N_2$ 
  - $0.65 = 0.1010011001001 \dots$
  - The fractional part begins repeating every 4 steps yielding repeating 1001 forever!
- Solution: Specify number of bits to right of radix point and round or truncate to this number.



# Checking the Conversion

- To convert back, sum the digits times their respective powers of  $r$ .
- From the prior conversion of  $46.6875_{10}$   
 $101110_2 = 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1$   
 $= 32 + 8 + 4 + 2$   
 $= 46$   
 $0.1011_2 = 1/2 + 1/8 + 1/16$   
 $= 0.5000 + 0.1250 + 0.0625$   
 $= 0.6875$



# Octal (Hexadecimal) to Binary and Back

---

- Octal (Hexadecimal) to Binary:
  - Restate the octal (hexadecimal) as three (four) binary digits starting at the radix point and going both ways.
- Binary to Octal (Hexadecimal):
  - Group the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part.
  - Convert each group of three bits to an octal (hexadecimal) digit.



# Octal to Hexadecimal via Binary

- Convert octal to binary.
- Use groups of four bits and convert as above to hexadecimal digits.
- Example: Octal to Binary to Hexadecimal

2    3    5    .    1    7    6<sub>8</sub>  
010 011 101 . 001111110<sub>2</sub>

Regroup:

(000)0|1001|1101 . 0011|1111|0(000)<sub>2</sub>

Convert:


0    9    D    .    3    F    0<sub>16</sub>

Answer 2: Marking off in groups of three (four) bits corresponds to dividing or multiplying by  $2^3 = 8$  ( $2^4 = 16$ ) in the binary system.



# Conversion Summary

	To				
		Dec	Bin	Octal	Hex
From	Dec	-	Repeated / or *	Thru Bin	Thru Bin
	Bin	Add weights of 1s	-	Convert every 3 digits	Convert every 4 digits
	Octal	Add digit*weight	Split every digit to 3	-	Thru Bin
	Hex	Add digit*weights	Split every digit to 4	Thru Bin	-



GOOD NEWS!  
THE CLASS IS  
OVER...  
THANK YOU!