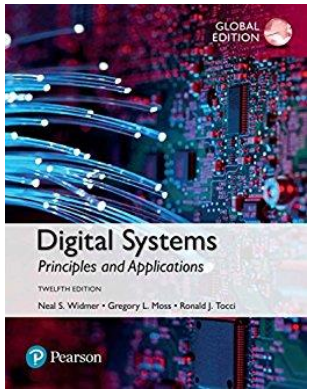
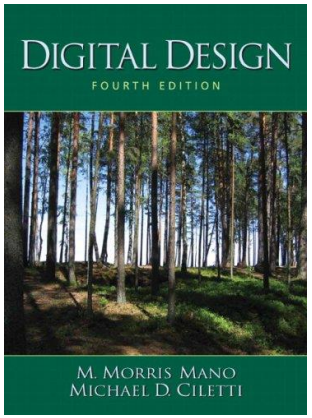
Information

- **Course: Digital Logic Design (EE053IU)**
- **Lecturer: Do Ngoc Hung**
- **Office:** O2.206, School of EE, IU Campus, VNU-HCM
- **Email:** dnhung@hcmiu.edu.vn
- **Number of credits:** 3
- **Textbook:**

1. Neal Widmer, Greg Moss, and Ron Tocci
   "Digital Systems", Pearson;
2. Moriss Mano, Michael D. Ciletti,
   "*Digital Design*", Prentice Hall,





- **Lecture Note & Homework:** Blackboard
- **TA: Trang Kien**, email: tkien@hcmiu.edu.vn
- **Office:** LA2.109

Course Syllabus

Content
Introduction of Digital world and Number Systems
Basic math operations for digital systems and Binary codes
Binary Logic, Logic gates, and Boolean Algebra
Combinational Logic Circuit
Sequential Logic Circuit (part I): Latches and Flip-Flop Devices
Sequential Logic Circuit (part II): Synchronous and Asynchronous Counters Designs
Sequential Logic Circuit (part III): IC Counter and Register.
Memory and Storage in the Computer
Review

Grading

1. Midterm examination: 30%

2. Final examination: 40%

3. Attendance + Quiz + Homework: 30%

(Software is used for circuit simulation: NI Multisim)

Course Policy

1. All assignments need to be submitted on the due date

2. Students are expected to do their own work at all times. Any evidence of plagiarism or cheating will be treated as grounds for failure in the class.

3. Attendance: at least 80%

Chapter 1 Objectives

- *Selected areas covered in this chapter*:
  - Analog & digital representations.
    - How information is represented using two states.
  - Advantages/drawbacks of digital/analog techniques.
    - Analog-to-digital and digital-to-analog converters.
  - Basic characteristics of the binary number system.
    - Convert binary numbers to decimal equivalents.
  - Identify typical digital signals & a timing diagram.
  - Converting between number systems.
    - Decimal, binary, hexadecimal.

Chapter 1 Objectives

Selected areas covered in this chapter:

- Advantages of the hexadecimal number system.
  - Counting in hexadecimal.
- Representing decimal numbers using the BCD code.
  - Pros and cons of using BCD.
  - Differences between BCD and straight binary.
- Purpose of alphanumeric codes such as ASCII code.
- Parity method for error detection.
  - Determine the parity bit to be attached to a digital data string.

Chapter 1

Numerical Representations

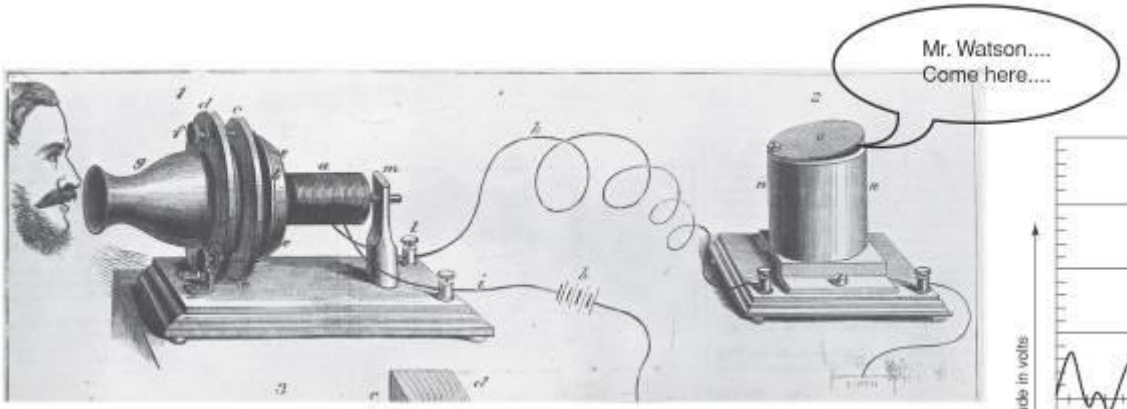
- Physical systems use quantities which must be manipulated arithmetically.
- Quantities may be represented numerically in either analog or digital form.

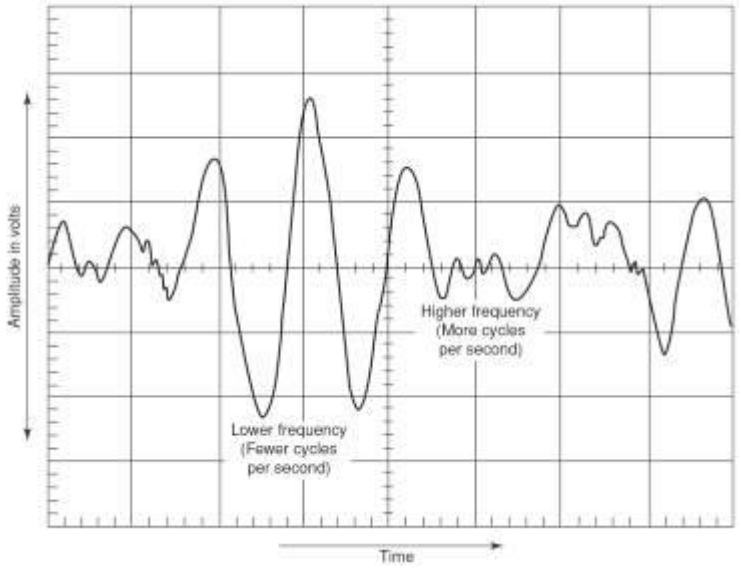
Analog Representation: a continuously variable, proportional indicator.

- Sound through a microphone causes voltage changes.
- Automobile speedometer changes with speed.
- Mercury thermometer varies over a range of values with temperature.

Chapter 1

- In 1875, Alexander Graham Bell figured out how to change his voice into a continuously variable electrical signal, send it through a wire, and change it back to sound energy at the other end.





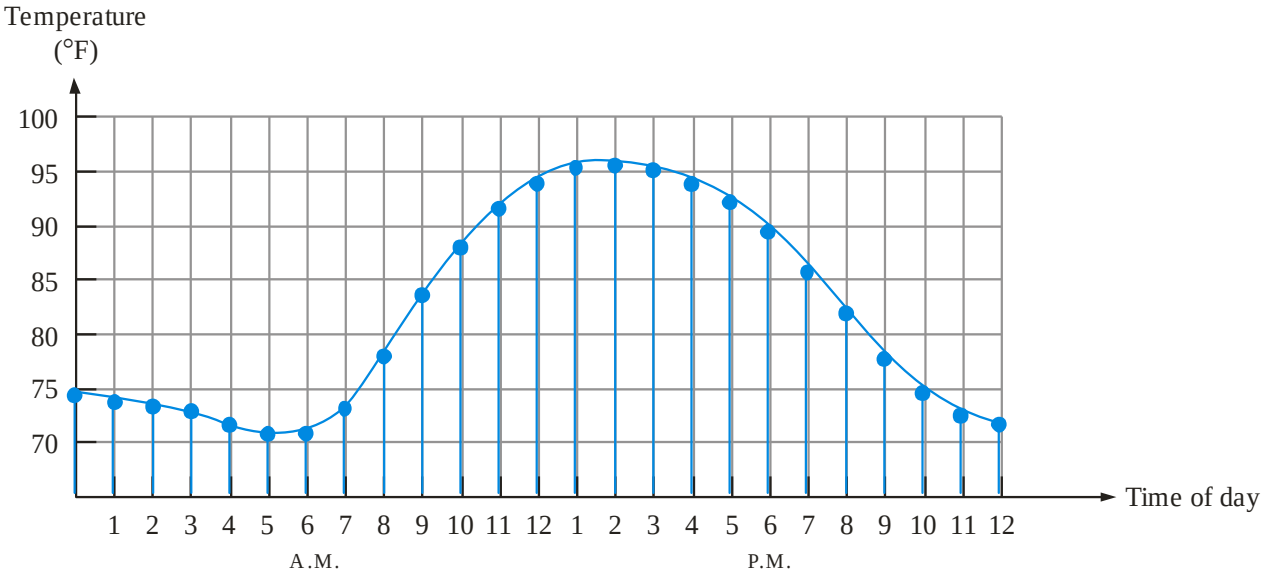
- Today, the device that converts sound energy to an analog voltage signal is known as a microphone.

Chapter 1

- Digital Representation—varies in discrete (separate) steps.
  - Passing time is shown as a change in the display on a digital clock at one minute intervals.
  - A change in temperature is shown on a digital display only when the temperature changes at least one degree.

Chapter 1

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



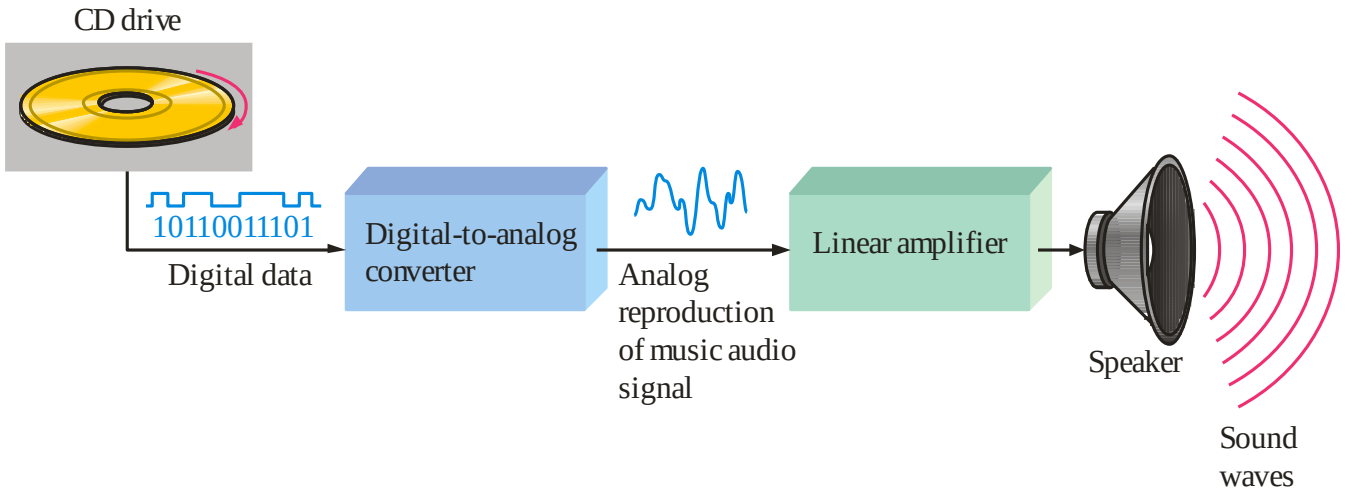
Digital systems can process, store, and transmit data more efficiently but can only assign discrete values to each point.

Chapter 1

- Digital system:
  - A combination of devices that manipulate values represented in digital form.
  - Quantities can take on only discrete values.
- Analog system:
  - A combination of devices that manipulate values represented in analog form.
  - Quantities can vary over a continuous range of values.

Chapter 1

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.

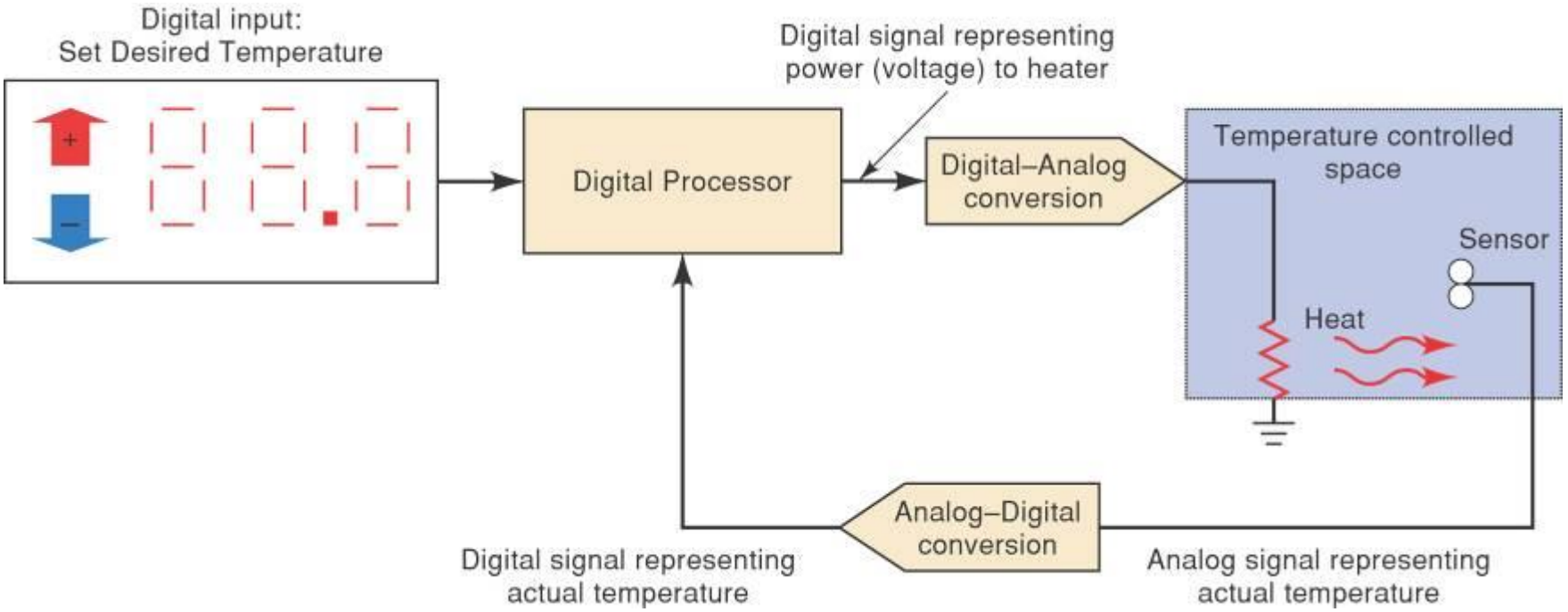


Chapter 1

The cell phone has digital & analog components, and uses *both* types of signals.

Chapter 1

Temperature-regulation system using an analog-to-digital converter.



Chapter 1

- Advantages of digital:
  - Ease of design
  - Well suited for storing information.
  - Accuracy and precision are easier to maintain.
  - Programmable operation.
  - Less affected by noise.
  - Ease of fabrication on IC chips.

Chapter 1

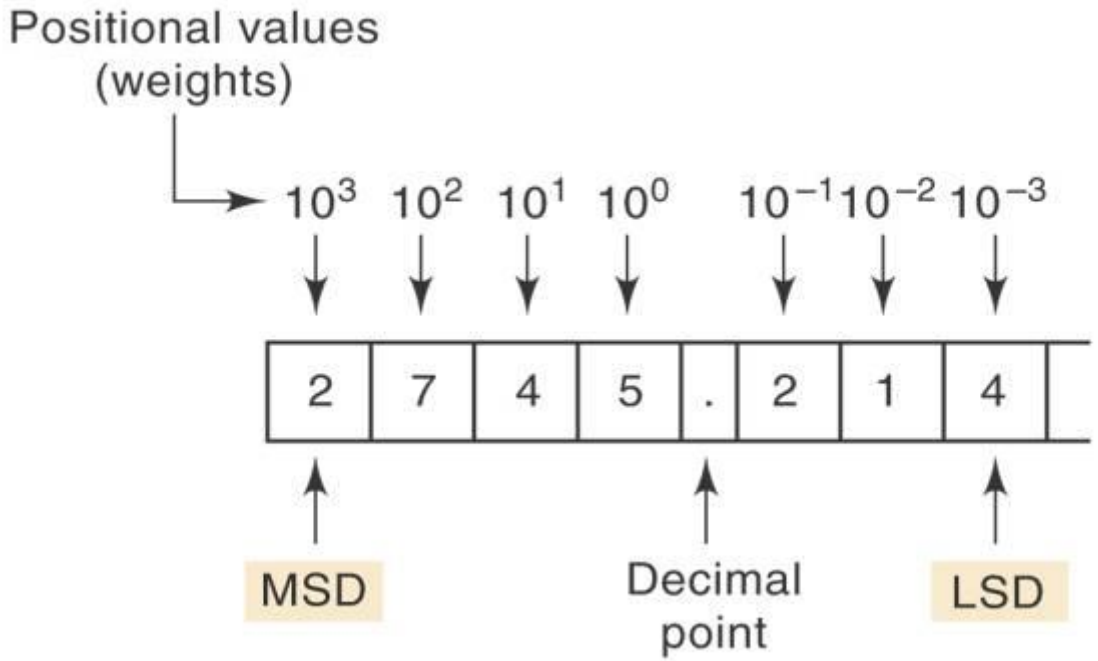
- There are limits to digital techniques:
  - The analog nature of the world requires a time consuming conversion process:
    1. Convert the physical variable to an electrical signal (analog).
    2. Convert the analog signal to digital form.
    3. Process (operate on) the digital information.
    4. Convert the digital output back to real-world analog form.

Chapter 1

- Understanding digital systems requires an understanding of the decimal, binary, octal, and hexadecimal numbering systems.
  - Decimal – 10 symbols (base 10)
  - Hexadecimal – 16 symbols (base 16)
  - Octal – 8 symbols (base 8)
  - Binary – 2 symbols (base 2)

Chapter 1: Digital Number Systems

- The Decimal (base 10) System
  - 10 symbols: 0, 1, 2, 3, 4, 5, 6 , 7, 8, 9.
    - Each number is a *digit* (from Latin for *finger*).

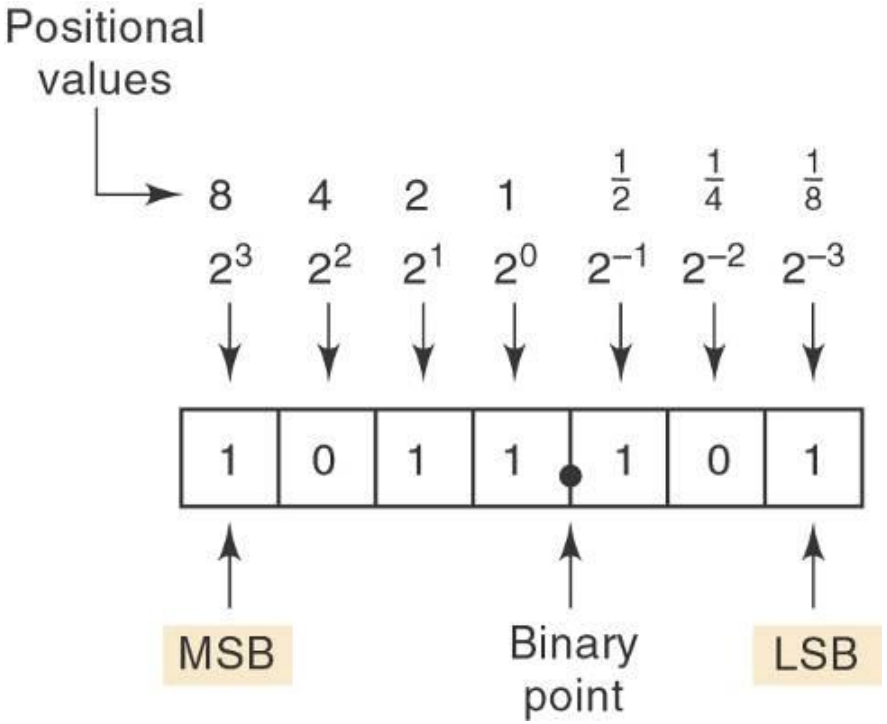


Most significant digit (MSD) & least significant digit (LSD).

Positional value may be stated as a digit multiplied by a power of 10.

Chapter 1: Digital Number Systems

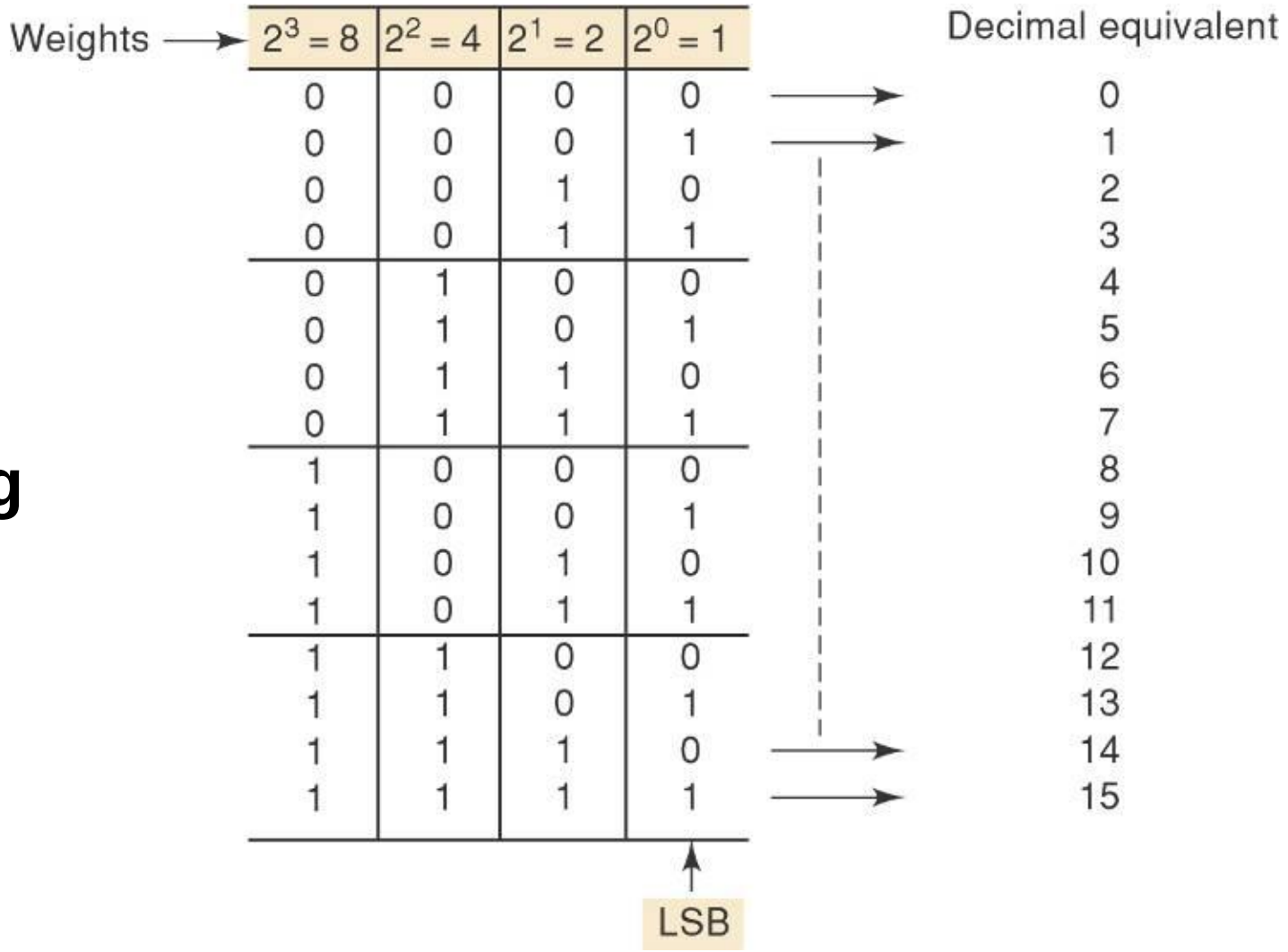
- The Binary (base 2) System
  - 2 symbols: 0,1
    - Lends itself to electronic circuit design since only two different voltage levels are required.



Positional value may be stated as a digit multiplied by a power of 2.

Chapter 1: Digital Number Systems

Binary Counting



Weights →	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	Decimal equivalent
	0	0	0	0	0
	0	0	0	1	1
	0	0	1	0	2
	0	0	1	1	3
	0	1	0	0	4
	0	1	0	1	5
	0	1	1	0	6
	0	1	1	1	7
	1	0	0	0	8
	1	0	0	1	9
	1	0	1	0	10
	1	0	1	1	11
	1	1	0	0	12
	1	1	0	1	13
	1	1	1	0	14
	1	1	1	1	15
				LSB	

Chapter 1: Digital Number Systems

Counting range

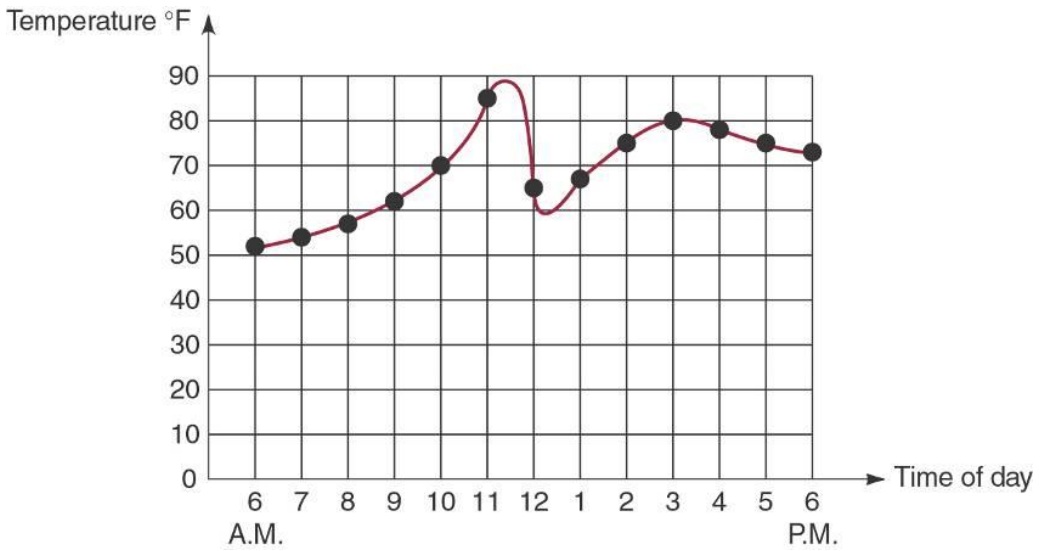
Using *N bits, we can represent decimal numbers ranging from 0 to* $2^N - 1$

A total of 2^N different numbers.

(a) What is the total range of decimal values that can be represented in eight bits?
(b) How many bits are needed to represent decimal values ranging from 0 to 12,500?

Chapter 1:Representing Binary Quantities

- Analog signals can be converted to digital by taking measurements or "samples" of the continuously varying signal at regular intervals.
- Appropriate time between samples depends on the maximum rate of change of the analog signal.



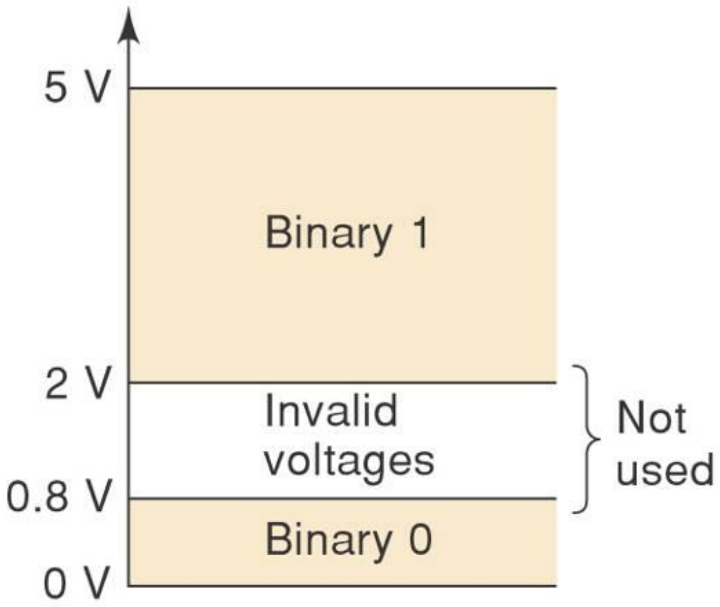
- Air temperature is an analog quantity.
- Recorded samples are discrete integer data.

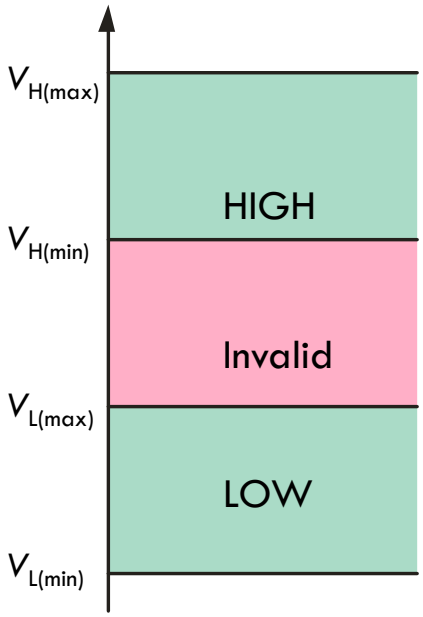
Chapter 1:Representing Binary Quantities

Typical representation of the two states of a digital signal.

A *higher* range of voltages represent a valid 1 and a *lower* range of voltages represent a valid 0.

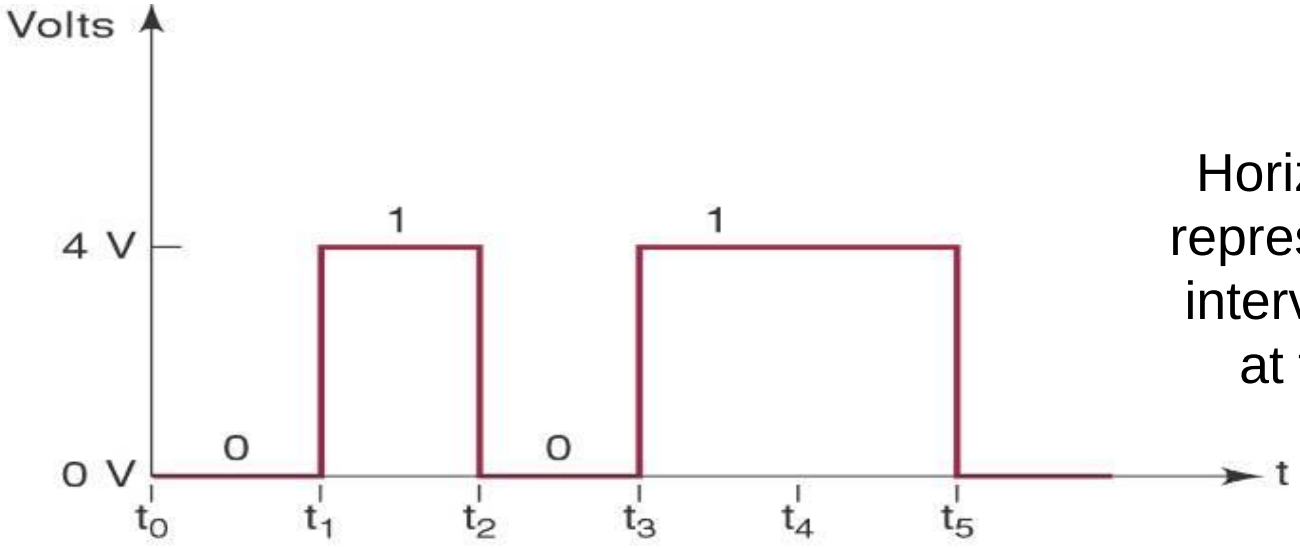
HIGH and LOW are often used to describe the states of a digital system—instead of "1" and "0"





Chapter 1:Representing Binary Quantities

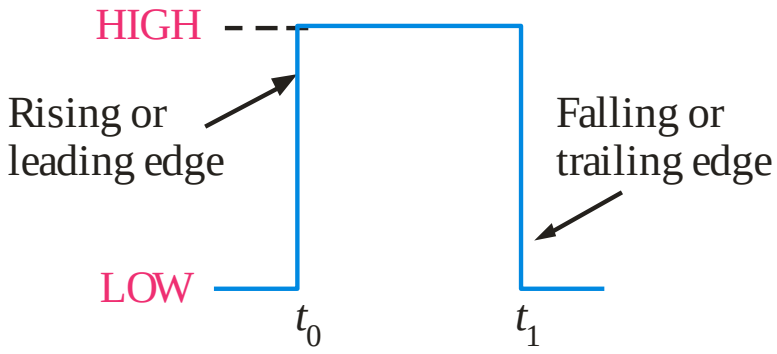
- The oscilloscope and logic analyzer are used to produce timing diagrams.
- Timing diagrams show voltage versus time.
- Used to show how digital signals change with time, or to compare two or more digital signals.



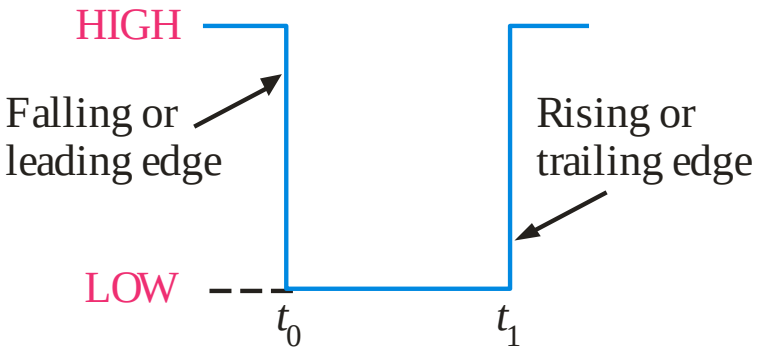
Horizontal scale represents regular intervals, starting at time zero.

Chapter 1:Representing Binary Quantities

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

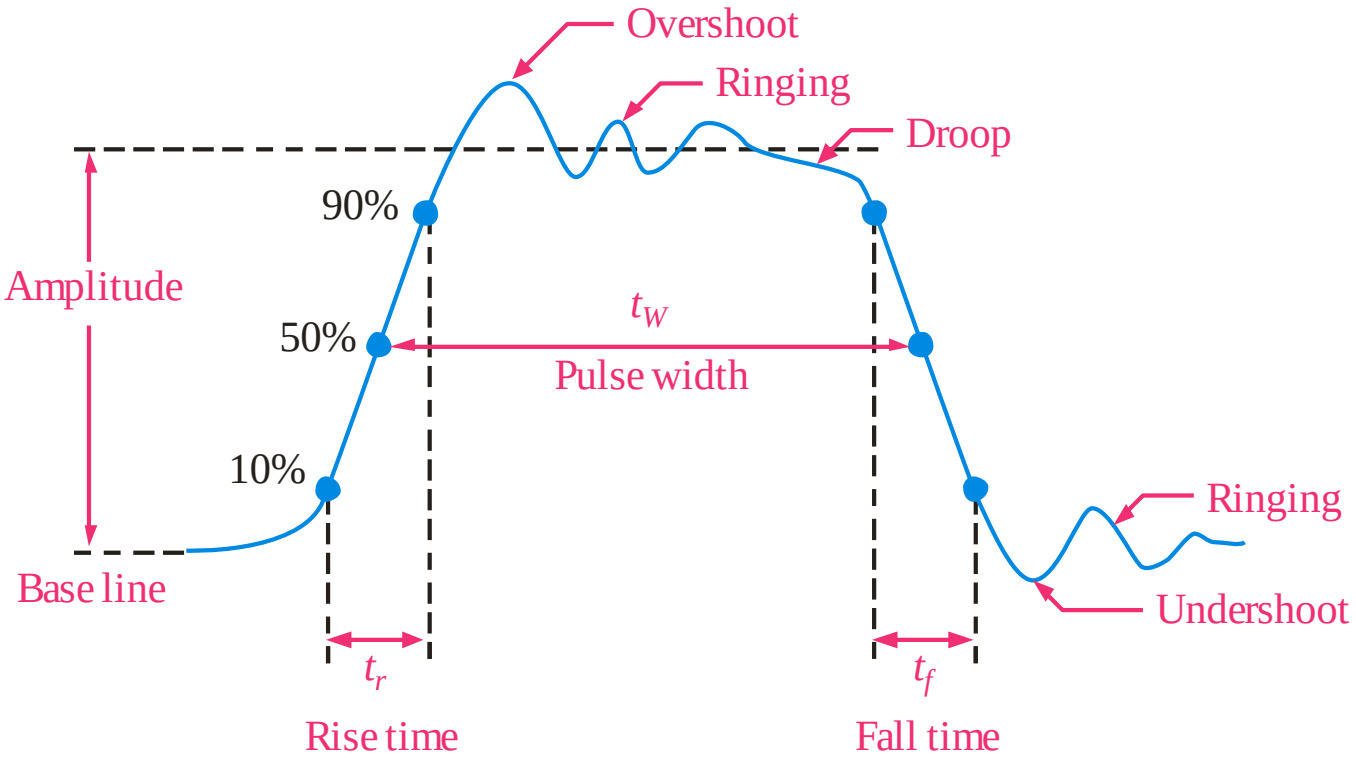


(b) Negative–going pulse

Chapter 1:Representing Binary Quantities

Pulse Definitions

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



Chapter 1:Representing Binary Quantities

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.

Example What is the period of a repetitive wave if f = 3.2 GHz?

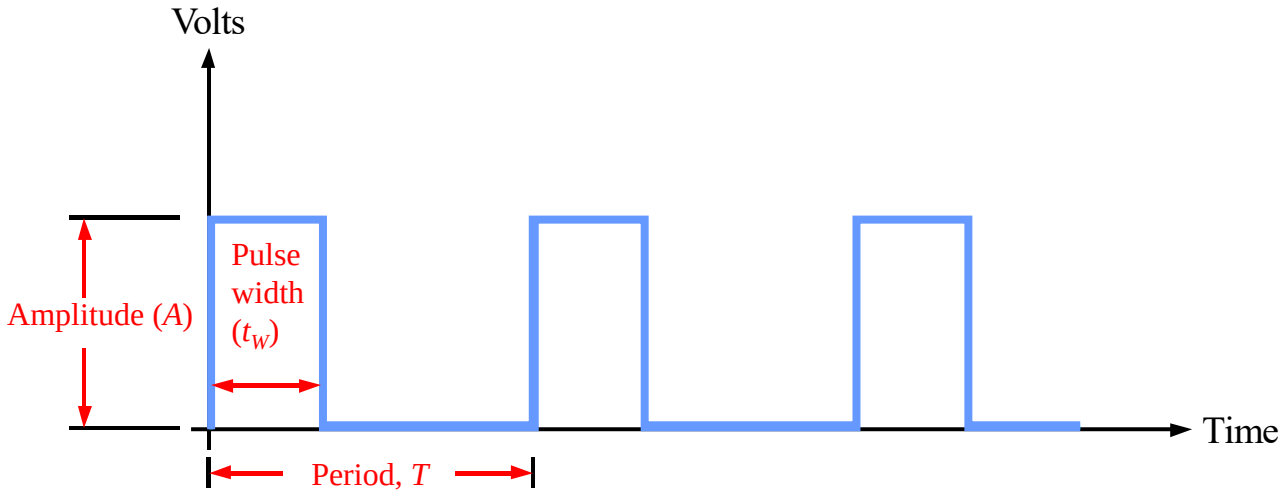
Solution

$$T = \frac{1}{f} = \frac{1}{3.2\,\text{GHz}} = 313\ \text{ps}$$

Chapter 1:Representing Binary Quantities

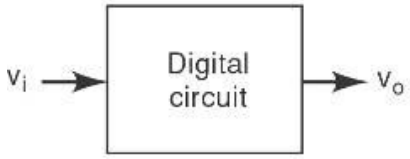
Pulse Definitions

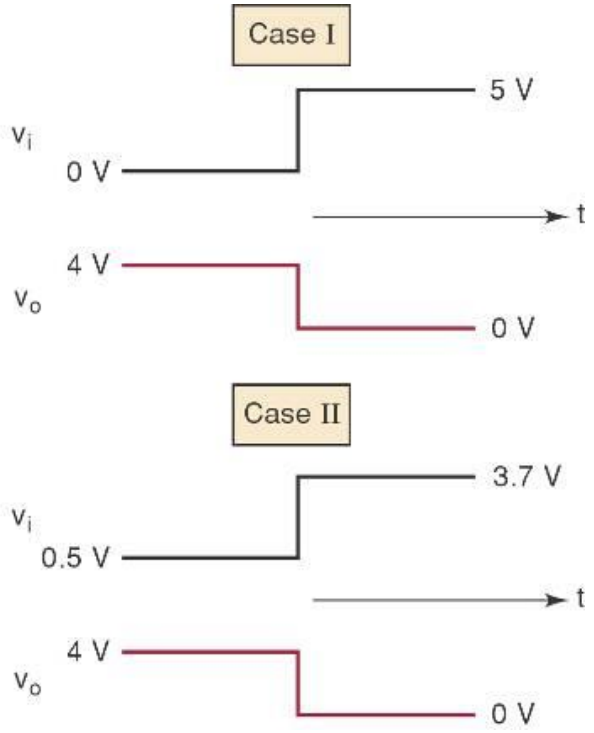
In addition to frequency and period, repetitive pulse waveforms are described by the amplitude (A), pulse width (t_W) and duty cycle. Duty cycle is the ratio of t_W to T.



Chapter 1:Digital Circuits/Logic Circuits

- Digital circuits - produce & respond to predefined voltage ranges.
  - The term *logic circuits* is used interchangeably.
  - A digital circuit responds to an input's binary level of 0 or 1 not to its actual voltage.

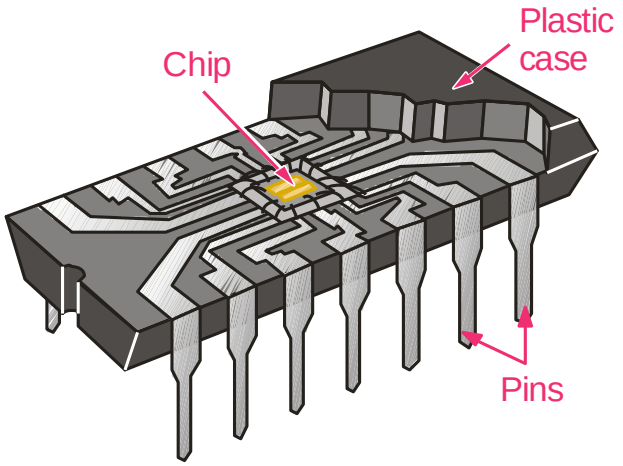




Chapter 1: Integrated Circuits

A monolithic integrated circuit (IC) is an electronic circuit that is constructed entirely on a single small chip of silicon.

All the components that make up the circuit: transistors, diodes, resistors, and capacitors



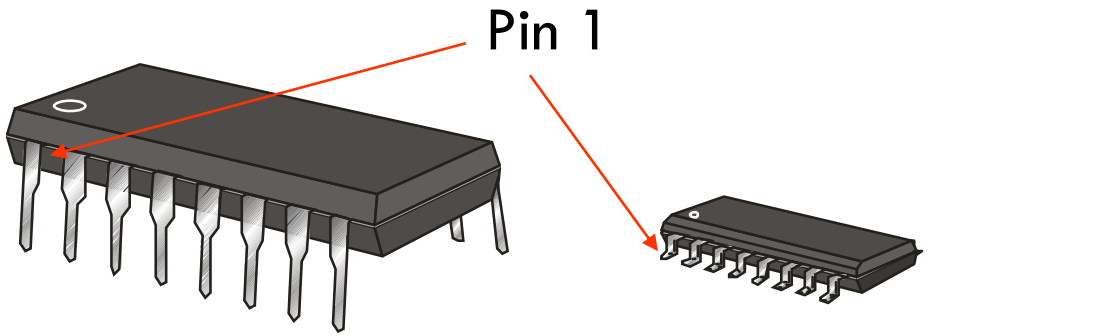
Cutaway view of one type of fixed-function IC package

Chapter 1: Integrated Circuits

Integrated circuit (IC) packages are classified according to the way they are mounted on printed circuit boards (PCBs) as either through-hole mounted or surface mounted

The through-hole type packages have pins (leads) that are inserted through holes in the PCB and can be soldered to conductors on the opposite side

DIP chips and surface mount chips



Dual in-line package

Small outline IC (SOIC)

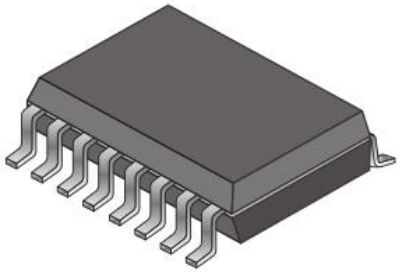
Chapter 1: Integrated Circuits

Surface-mount technology (SMT) is a space-saving alternative to through-hole mounting. The holes through the PCB are unnecessary for SMT. The pins of surface-mounted packages are soldered directly to conductors, on one side of the board, leaving the other side free for additional circuits.

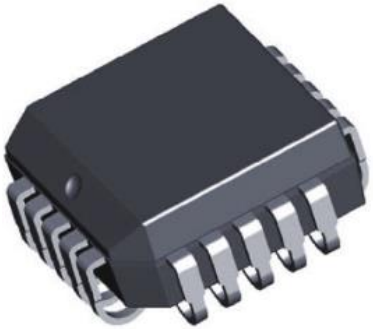
- SOIC (small-outline integrated circuit)
- SSOP (shrink small-outline package)
- PLCC (plastic-leaded chip carrier)
- LCC (leadless ceramic chip)
- LQFP (low-profile quad flat package)
- FBGA (fine-pitch ball grid array)

Chapter 1: Integrated Circuits

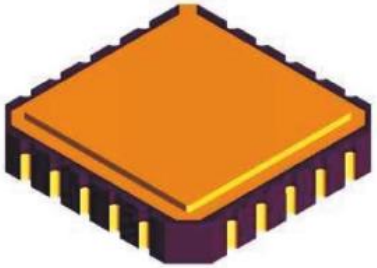
Other surface mount packages:



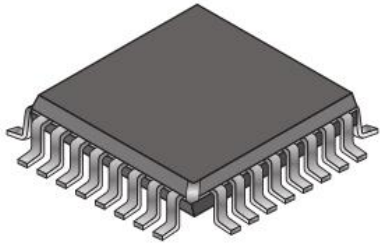
(a) SSOP (153 × 193 mils)



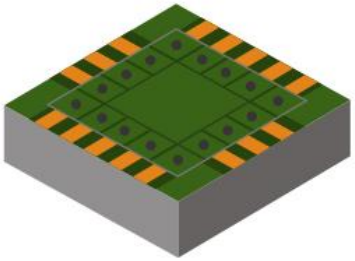
(b) PLCC (350 × 350 mils)



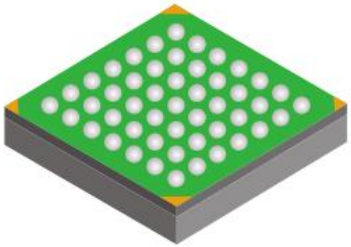
(c) LCC (350 × 350 mils)



(d) LQFP (7 × 7 mm)



(e) Laminate CSP bottom view (3.5 × 3.5 mm)



(f) FBGA bottom view (4 × 4 mm)

Chapter 1: Integrated Circuits

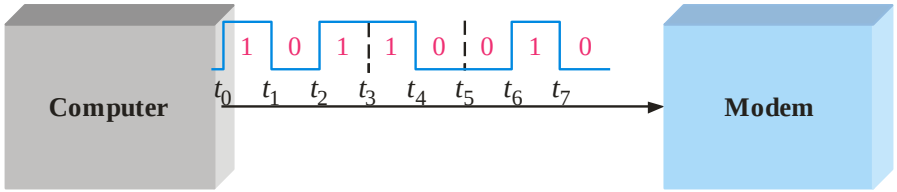
Fixed-function digital ICs are classified according to their complexity.

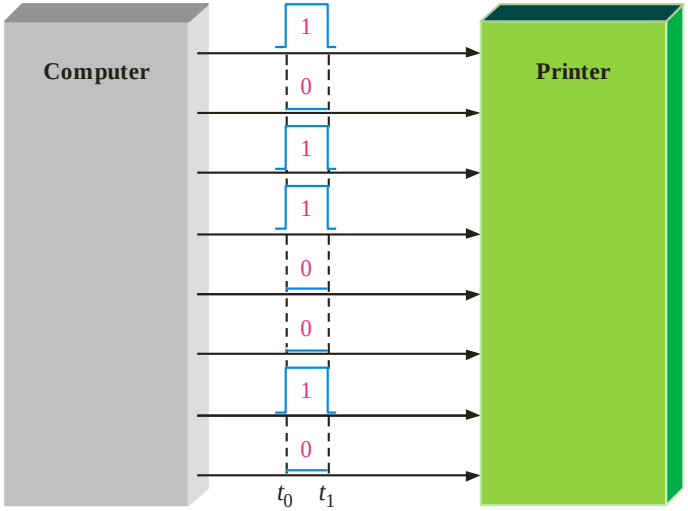
- **Small-scale integration (SSI)** describes fixed-function ICs that have up to ten equivalent gate circuits on a single chip, and they include basic gates and flip-flops.
- **Medium-scale integration (MSI)** describes integrated circuits that have from 10 to 100 equivalent gates on a chip. They include logic functions such as encoders, decoders, counters, registers, multiplexers, arithmetic circuits, small memories, and others.
- **Large-scale integration (LSI)** is a classification of ICs with complexities of from more than 100 to 10,000 equivalent gates per chip, including memories.
- **Very large-scale integration (VLSI)** describes integrated circuits with complexities of from more than 10,000 to 100,000 equivalent gates per chip.
- **Ultra large-scale integration (ULSI)** describes very large memories, larger microprocessors. Complexities of more than 100,000 equivalent gates per chip

Chapter 1:Serial and Parallel Data

Serial and Parallel Data

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



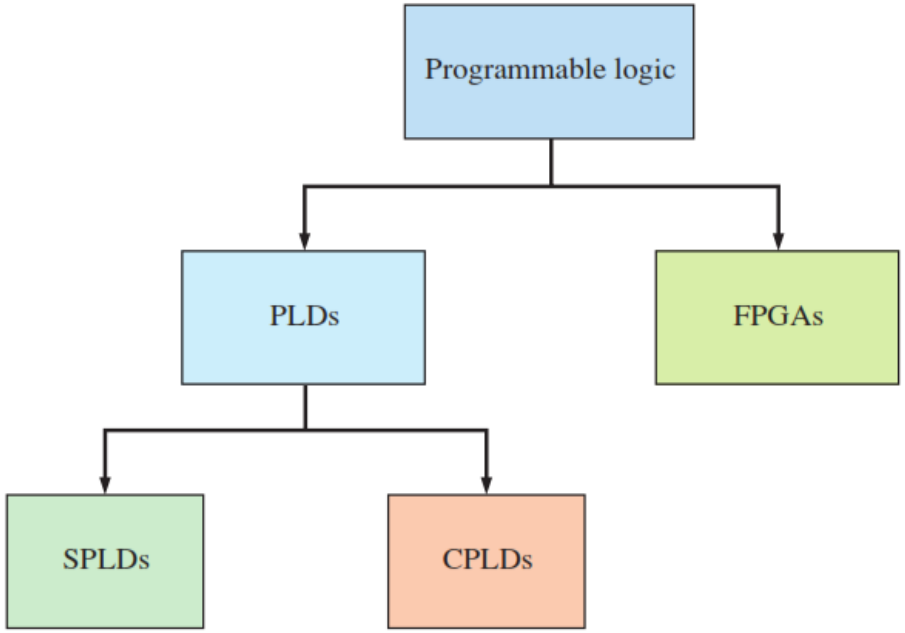


Chapter 1:Programmable Logic

Programmable logic devices (PLDs) are an alternative to fixed function devices. The logic can be programmed for a specific purpose. In general, they cost less and use less board space that fixed function devices.

Two major categories of user-programmable logic are

- **PLD (programmable logic device)**
- **FPGA (field-programmable gate array)**

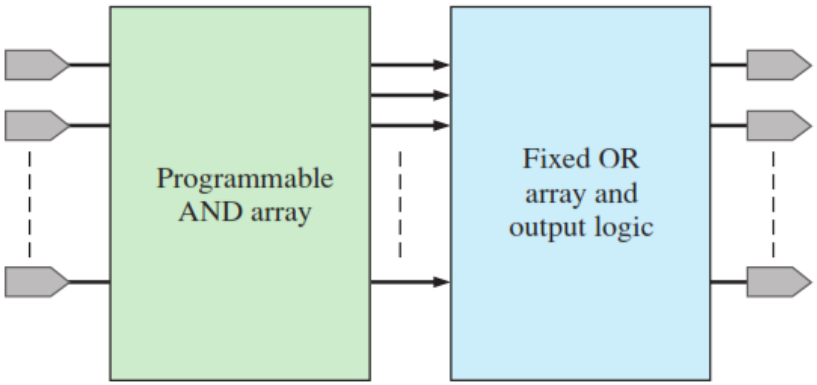


Chapter 1:Programmable Logic

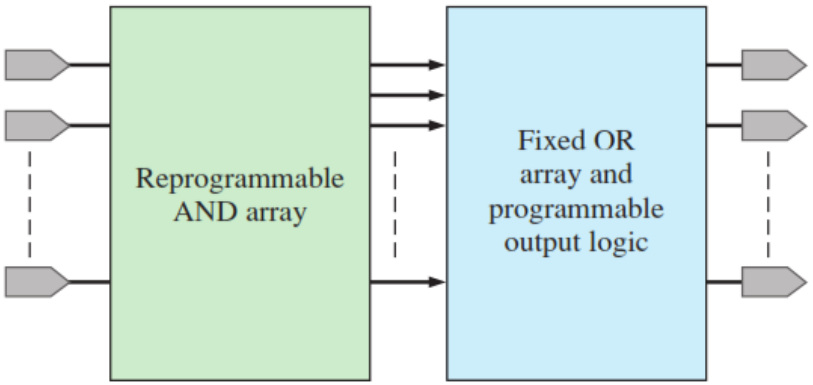
PLDs are either SPLDs (simple PLDs) or CPLDs (complex PLDs)

PAL device is a form of PLD that uses a combination of a programmable AND array and a fixed OR array (can be **reprogrammed 1 time**)

GAL device is a form of PLD that uses a combination of a programmable AND array and a fixed OR array and programmable output logic (can be **reprogrammed many times**)



(a) PAL



(b) GAL

Chapter 1:Programmable Logic

The SPLD was the original PLD and is still available for small-scale applications. Generally, an SPLD can replace up to ten fixed-function ICs and their interconnections.

The CPLD is a device containing multiple SPLDs and can replace many fixed-function ICs

An FPGA is generally more complex and has a much higher density than a CPLD, although their applications can sometimes overlap

Chapter 1: Binary to Decimal Conversion

- Convert binary to decimal by summing the positions that contain a 1:

$$\begin{array}{ccccc} 1 & 1 & 0 & 1 & 1_2 \\ 2^4 + 2^3 + 0 + 2^1 + 2^0 & = & 16 + 8 + 2 + 1 & & \\ & = & 27_{10} & & \end{array}$$

- An example with a greater number of bits:

$$\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1_2 = \\ 2^7 + 0 + 2^5 + 2^4 + 0 + 2^2 + 0 + 2^0 & = & 181_{10} & & & & & \end{array}$$

Chapter 1:Decimal to Binary Conversion

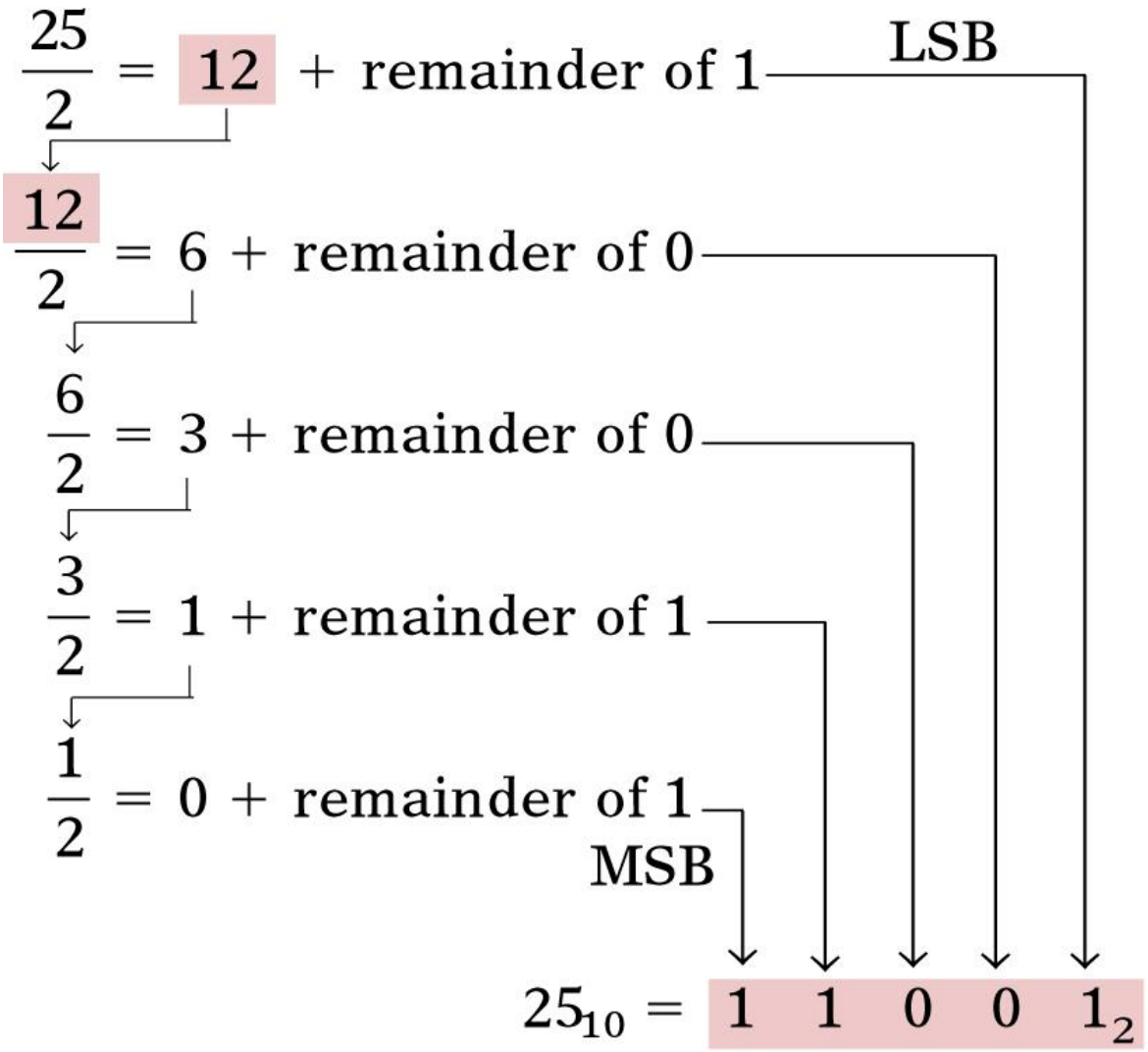
Repeated Division-by-2 Method

Divide the decimal number by 2.

Write the remainder after each division until a quotient of zero is obtained.

The first remainder is the LSB.

The last is the MSB.



Chapter 1:Decimal to Binary Conversion

Converting Decimal Fractions to Binary

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.

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

0.188 x 2 = 0.376 carry = 0
0.376 x 2 = 0.752 carry = 0
0.752 x 2 = 1.504 carry = 1
0.504 x 2 = 1.008 carry = 1
0.008 x 2 = 0.016 carry = 0

MSB

Answer = .00110 (for five significant digits)

Chapter 1:Binary Addition

The rules for binary addition are

0 + 0 = 0 Sum = 0, carry = 0

0 + 1 = 1 Sum = 1, carry = 0

1 + 0 = 1 Sum = 1, carry = 0

1 + 1 = 0 Sum = 0, carry = 1

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

$$\begin{array}{rr} {\scriptstyle 0\ 1\ 1\ 1} & \\ 00111 & 7 \\ 10101 & 21 \\ \hline 11100 & = 28 \end{array}$$

Chapter 1:Binary Subtraction

The rules for binary subtraction are

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$0 - 1 = 1$ with a borrow of 1

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

Solution

$$\begin{array}{rr} {\scriptstyle 1\ 1\ 1} & \\ 10101 & 21 \\ 00111 & 7 \\ \hline 01110 & = 14 \end{array}$$

Chapter 1:Binary Multiplication

The rules for binary multiplication are

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Multiplication is performed with binary numbers in the same manner as with decimal numbers.

Perform the following binary multiplications:

(a) 11×11 **(b)** 101×111

Solution

(a)

$$\begin{array}{r} 11 \\ \times\ 11 \\ \hline 11 \\ +11 \\ \hline \mathbf{1001} \end{array}$$

Partial products

$$\begin{array}{r} 3 \\ \times\ 3 \\ \hline 9 \end{array}$$

(b)

$$\begin{array}{r} 111 \\ \times\ 101 \\ \hline 111 \\ 000 \\ +111 \\ \hline \mathbf{100011} \end{array}$$

Partial products

$$\begin{array}{r} 7 \\ \times\ 5 \\ \hline 35 \end{array}$$

Chapter 1: Binary Division

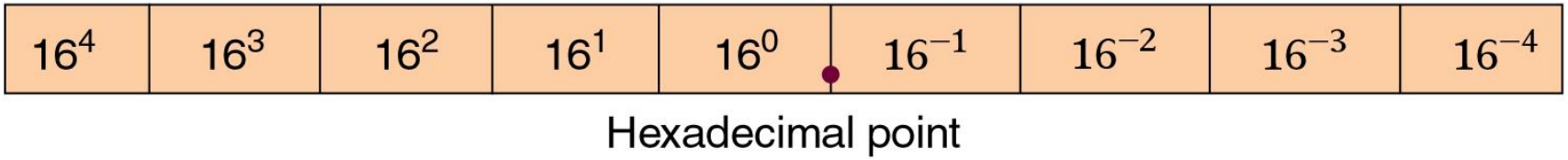
Division is performed with binary numbers in the same manner as with decimal numbers.

Perform the following binary divisions:

(a) $110 \div 11$ **(b)** $110 \div 10$

Chapter 1:Hexadecimal Number System

- Hexadecimal allows convenient handling of long binary strings, using groups of 4 bits—Base 16
  - 16 possible symbols: 0-9 and A,B,C,D,E,F
- The digit positions are weighted as powers of 16 as shown below, rather than as powers of 10 as in the decimal system.



Chapter 1:Hexadecimal Number System

Relationships between hexadecimal, decimal, and binary numbers.

Hexadecimal	Decimal	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
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Chapter 1:Hexadecimal Number System

- **Convert from hex to decimal** by multiplying each hex digit by its positional weight.

$$\begin{aligned} 356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10} \end{aligned}$$

- In a 2nd example, the value 10 was substituted for A and 15 substituted for F.

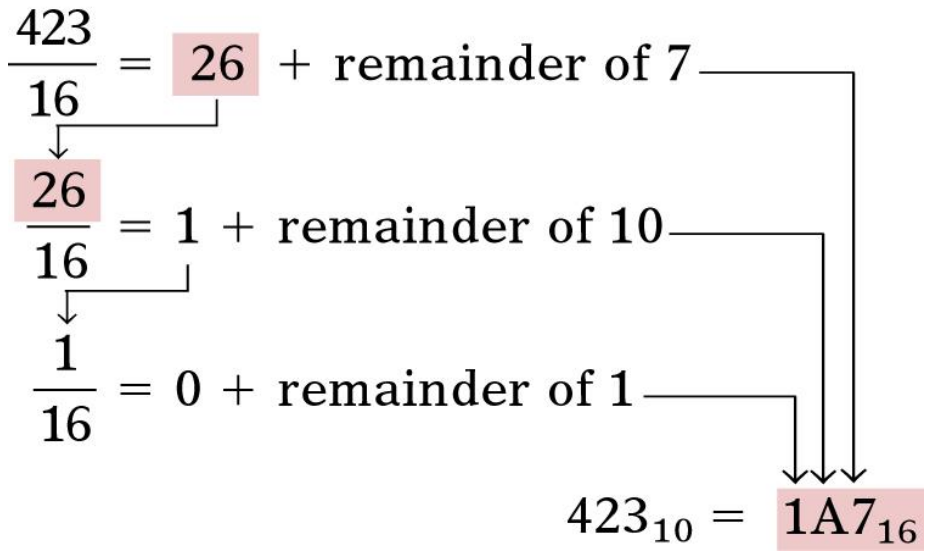
$$\begin{aligned} 2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10} \end{aligned}$$

For practice, verify that $1BC2_{16}$ is equal to 7106_{10}

Chapter 1:Hexadecimal Number System

- **Convert from decimal to hex** by using the repeated division method used for decimal to binary conversion.
- Divide the decimal number by 16
  - The first remainder is the LSB—the last is the MSB.

Convert 423_{10} to hex:



Chapter 1:Hexadecimal Number System

Hex to Binary

- Leading zeros can be added to the left of the MSB to fill out the last group.

$$
\begin{aligned}
9F2_{16} &= \quad 9 \qquad\qquad F \qquad\qquad 2 \\
&\quad\;\; \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \\
&= 1\;\;0\;\;0\;\;1 \quad 1\;\;1\;\;1\;\;1 \quad 0\;\;0\;\;1\;\;0 \\
&= 100111110010_2
\end{aligned}
$$

For practice, verify that $BA6_{16} = 101110100110_2$

Chapter 1:Hexadecimal Number System

Hex to Binary

- Convert from binary to hex by grouping bits in four starting with the LSB.

  Each group is then converted to the hex equivalent

- The binary number is grouped into groups of four bits & each is converted to its equivalent hex digit.

$$1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 = \underbrace{0\,0\,1\,1}_{3}\ \underbrace{1\,0\,1\,0}_{A}\ \underbrace{0\,1\,1\,0}_{6}$$

$$= 3A6_{16}$$

For practice, verify that $101011111_2 = 15F_{16}$

Chapter 1:Hexadecimal Number System

Decimal to Hex to Binary

- Convert decimal 378 to a 16-bit binary number by first converting to hexadecimal.

$$\frac{378}{16} = 23 + \text{remainder of } 10_{10} = A_{16}$$

$$\frac{23}{16} = 1 + \text{remainder of } 7$$

$$\frac{1}{16} = 0 + \text{remainder of } 1$$

Chapter 1:Hexadecimal Number System

Counting in Hex

- When counting in hex, each digit position can be incremented (increased by 1) from 0 to F.
  - On reaching value F, it is reset to 0, and the next digit position is incremented.

Example:

`38,39,3A,3B,3C,3D,3E,3F,40,41,42`

When there is a 9 in a digit position, it becomes an A when it is incremented.

With three hex digits, we can count from 000_{16} to FFF_{16} which is 0_{10} to 4095_{10} — a total of $4096 = 16^3$ values.

Chapter 1:Hexadecimal Number System

Question

1. Convert $24CE_{16}$ to decimal.
2. Convert 311_{10} to hex, then from hex to binary.
3. Convert 1001011110110101_{2} to hex.
4. Write the next four numbers in this hex counting sequence: E9A, E9B, E9C, E9D, , , , .
5. Convert 3527_{16} to binary.
6. What range of decimal values can be represented by a four-digit hex number?

Chapter 1: Octal Number System

The octal number system is composed of eight digits: 0, 1, 2, 3, 4, 5, 6, 7

Counting in octal is similar to counting in decimal, except that the digits 8 and 9 are not used

Octal-to-Decimal Conversion

Decimal-to-Octal Conversion

} Similar to Hexadecimal – Decimal

Octal-to-Binary Conversion

Binary-to-Octal Conversion

} Similar to Hexadecimal – Binary (but 1 octal digit ⟺ 3bits)

Convert each of the following octal numbers to binary:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

Solution

(a) 1 3 → 001011

(b) 2 5 → 010101

(c) 1 4 0 → 001100000

(d) 7 5 2 6 → 111101010110

Chapter 1: Octal Number System

Convert each of the following binary numbers to octal:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solution

(a) $\underbrace{110}_{6}\underbrace{101}_{5} = \mathbf{65}_8$

(b) $\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{001}_{1} = \mathbf{571}_8$

(c) $\underbrace{100}_{4}\underbrace{110}_{6}\underbrace{011}_{3}\underbrace{010}_{2} = \mathbf{4632}_8$

(d) $\underbrace{011}_{3}\underbrace{010}_{2}\underbrace{000}_{0}\underbrace{100}_{4} = \mathbf{3204}_8$

Chapter 1: BCD Code

- Binary Coded Decimal (BCD) is a widely used way to present decimal numbers in binary form.
  - Combines features of both decimal and binary systems.
    - Each digit is converted to a binary equivalent.
- BCD is *not* a number system.
  - It is a decimal number with each digit encoded to its binary equivalent.
- A BCD number is *not* the same as a straight binary number.
  - The primary advantage of BCD is the relative ease of converting to and from decimal.

Chapter 1: BCD Code

- Convert the number 874_{10} to BCD:
  - Each decimal digit is represented using 4 bits.
    - Each 4-bit group can never be greater than 9.

8	7	4	(decimal)
↓	↓	↓	
1000	0111	0100	(BCD)

- Reverse the process to convert BCD to decimal.

9	4	3	(decimal)
↓	↓	↓	
1001	0100	0011	(BCD)

Chapter 1: BCD Code

- Convert 0110100000111001 (BCD) to its decimal equivalent.

$$\underbrace{0110}_{6}\ \underbrace{1000}_{8}\ \underbrace{0011}_{3}\ \underbrace{1001}_{9}$$

Divide the BCD number into four-bit groups and convert each to decimal.

- Convert BCD 011111000001 to its decimal equivalent.

$$\underbrace{0111}_{7}\ \underbrace{1100}_{\downarrow}\ \underbrace{0001}_{1}$$

The forbidden group represents an error in the BCD number.

Chapter 1: BCD Code

BCD Addition

Step 1: Add the two BCD numbers
Step 2: If a 4-bit sum is equal to or less than 9, it is a valid BCD number.
Step 3: If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is an invalid result. Add 6 (0110) to the 4-bit sum in order to skip the six. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

(a)

0011	3
+ 0100	+ 4
0111	7

(b)

0010	0011	23
+ 0001	0101	+ 15
0011	**1000**	38

(c)

0001	0110	16
+ 0001	0101	+ 15
0010	1011	31
		Right group is invalid (>9), left group is valid.
	+ 0110	Add 6 to invalid code. Add carry, 0001, to next group.
0011	**0001**	Valid BCD number
↓	↓	
3	1	

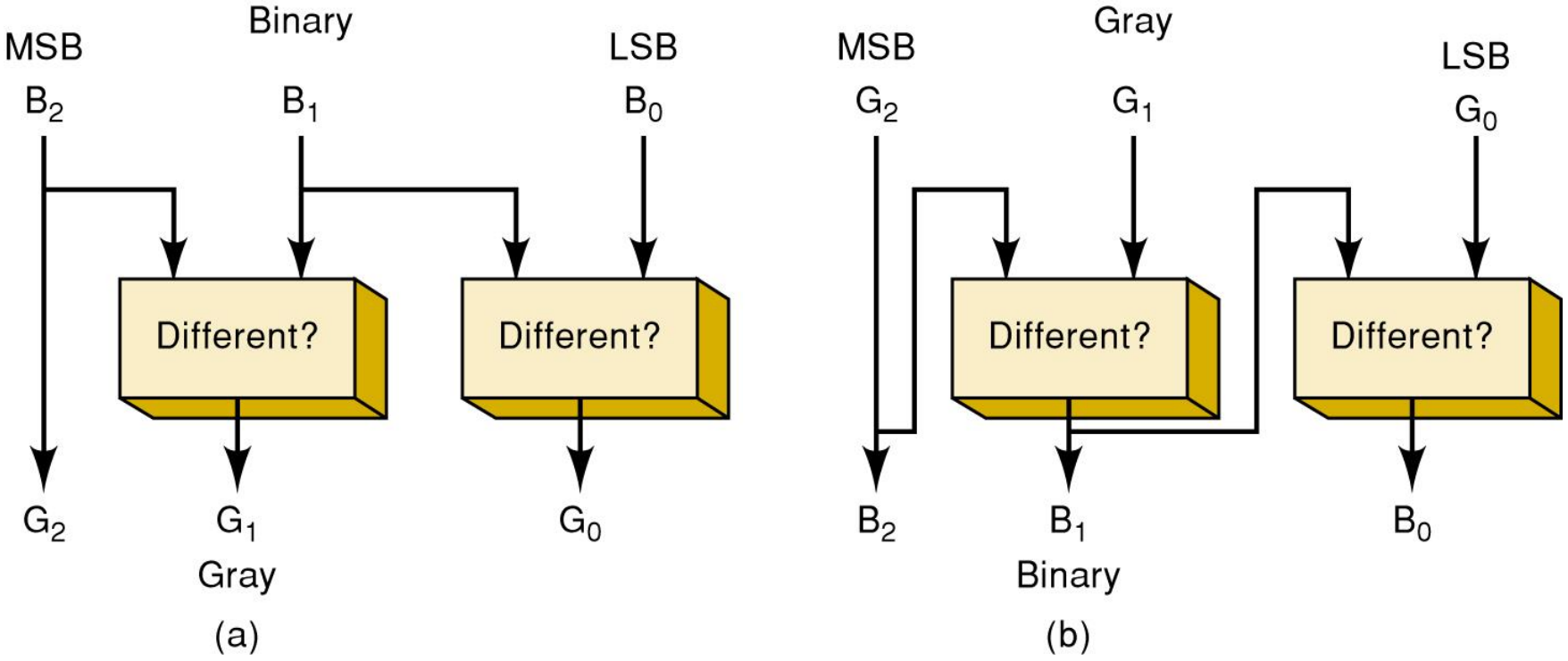
Chapter 1: Gray Code

- The Gray code is used in applications where numbers change rapidly.
  - Only one bit changes from each value to the next.
  - Gray code is used to avoid problems in systems where an error can occur if more than one bit changes at a time.

Three bit binary and Gray code equivalents.

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Chapter 1: Gray Code



Binary to Gray **Gray to Binary**

Compare the MSB binary with the next binary bit (B1). If they are the same, then G1 = 0. If they are different, then G1 = 1. G0 can be found by comparing B1 with B0.

Chapter 1: Gray Code

Decimal numbers 1 – 15 in binary, hex, BCD, Gray

Decimal	Binary	Hexadecimal	BCD	GRAY
0	0	0	0000	0000
1	1	1	0001	0001
2	10	2	0010	0011
3	11	3	0011	0010
4	100	4	0100	0110
5	101	5	0101	0111
6	110	6	0110	0101
7	111	7	0111	0100
8	1000	8	1000	1100
9	1001	9	1001	1101
10	1010	A	0001 0000	1111
11	1011	B	0001 0001	1110
12	1100	C	0001 0010	1010
13	1101	D	0001 0011	1011
14	1110	E	0001 0100	1001
15	1111	F	0001 0101	1000

Chapter 1: The Byte, Nibble, and Word

- Most microcomputers handle and store binary data and information in groups of eight bits.
  - 8 bits = 1 byte.
    - A byte can represent numerous types of data/information.
- Binary numbers are often broken into groups of four bits.
  - Because a group of four bits is half as big as a byte, it was named a **nibble**.
- A **word** is a group of bits that represents a certain unit of information.
  - **Word size** can be defined as the number of bits in the binary word a digital system operates on.
    - PC word size is eight bytes (64 bits).

Chapter 1: ASCII

ASCII – American Standard Code for Information Interchange.

Seven bit code: $2^7 = 128$ possible code groups

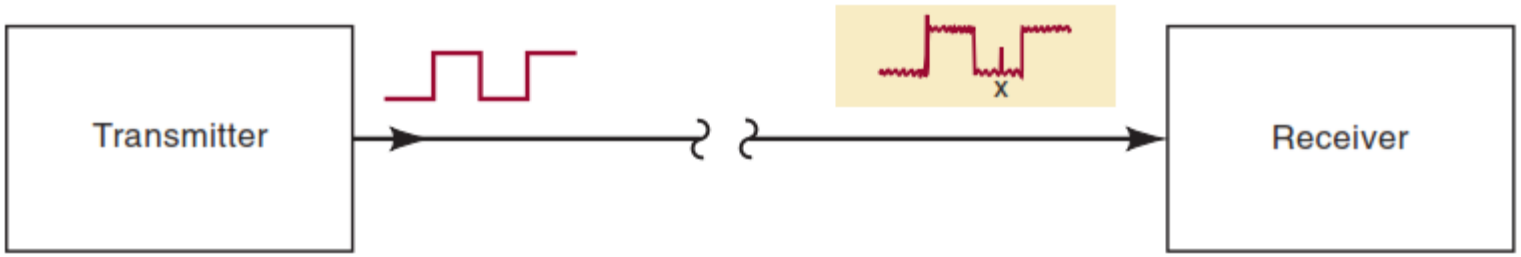
Examples of use: transfer information between computers; computers & printers; internal storage.

Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal
NUL (null)	0	0	Space	20	32	@	40	64	.	60	96
Start Heading	1	1	!	21	33	A	41	65	a	61	97
Start Text	2	2	“	22	34	B	42	66	b	62	98
End Text	3	3	#	23	35	C	43	67	c	63	99
End Transmit.	4	4	$	24	36	D	44	68	d	64	100
Enquiry	5	5	%	25	37	E	45	69	e	65	101
Acknowlege	6	6	&	26	38	F	46	70	f	66	102
Bell	7	7	`	27	39	G	47	71	g	67	103
Backspace	8	8	(	28	40	H	48	72	h	68	104
Horiz. Tab	9	9	)	29	41	I	49	73	i	69	105
Line Feed	A	10	*	2A	42	J	4A	74	j	6A	106
Vert. Tab	B	11	+	2B	43	K	4B	75	k	6B	107
Form Feed	C	12	,	2C	44	L	4C	76	l	6C	108
Carriage Return	D	13	-	2D	45	M	4D	77	m	6D	109
Shift Out	E	14	.	2E	46	N	4E	78	n	6E	110

Chapter 1: Parity Method For Error Detection

The movement of binary data and codes from one location to another is the most frequent operation performed in digital systems.

There is a possibility that errors can occur such that the receiver does not receive the identical information that was sent by the transmitter.



One of the simplest and most widely used schemes for error detection is the **parity method**

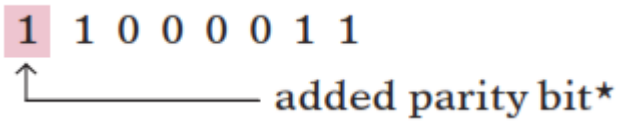
Chapter 1: Parity Method For Error Detection

Parity bit

A parity bit is an extra bit that is attached to a code group that is being transferred from one location to another.

The parity bit is made either 0 or 1, depending on the number of 1s that are contained in the code group.

The **even-parity method**, the value of the parity bit is chosen so that the total number of 1s in the code group (including the parity bit) is an even number.



The **odd-parity method** is used in exactly the same way except that the parity bit is chosen so the total number of 1s (including the parity bit) is an odd number.

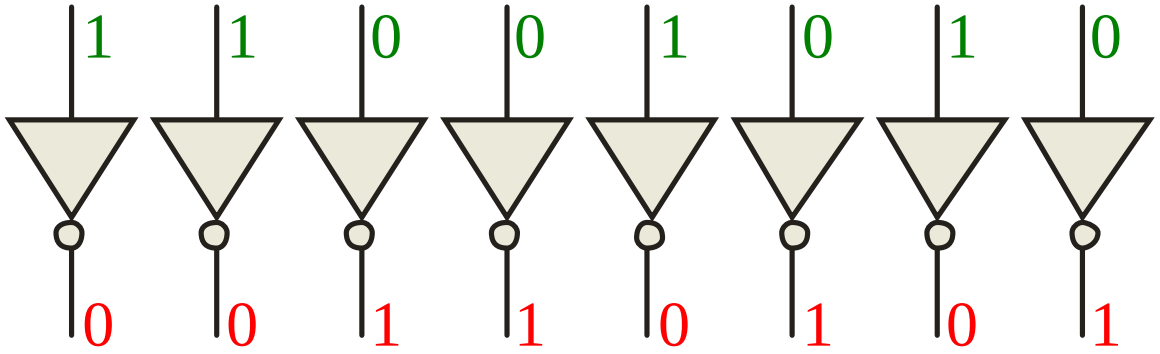
The parity bit is issued to detect any *single-bit errors*

Chapter 1: 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:



Chapter 1: 2's complement

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

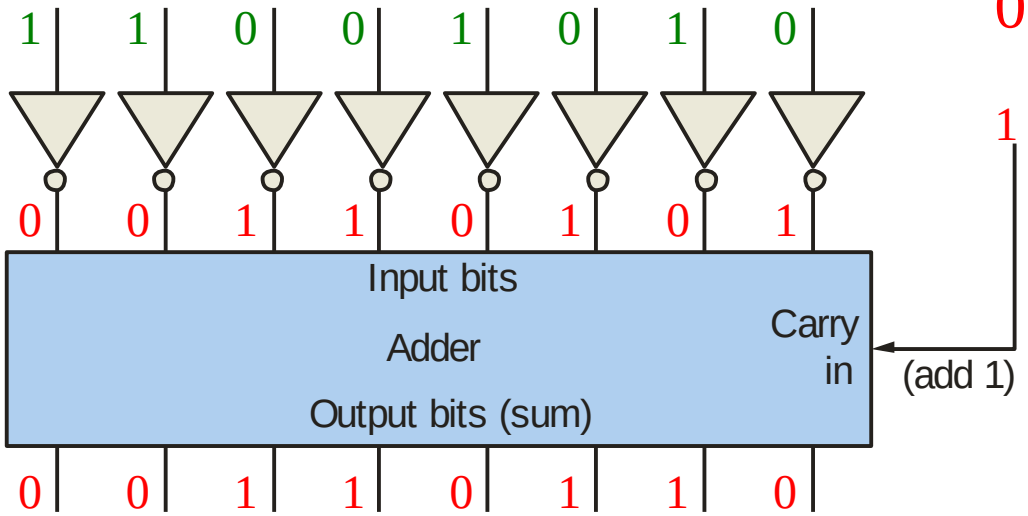
Recall that the 1's complement of 11001010 is

00110101 (1's complement)

To form the 2's complement, add 1:

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

(2's complement)



Chapter 1: 2's complement

An alternative method of finding the 2's complement of a binary number is as follows:

1. Start at the right with the LSB and write the bits as they are up to and including the first 1.
2. Take the 1's complements of the remaining bits.

Find the 2's complement of 10111000 using the alternative method.

Solution

10111000 Binary number

01001000 2's complement

1's complements of original bits

These bits stay the same.

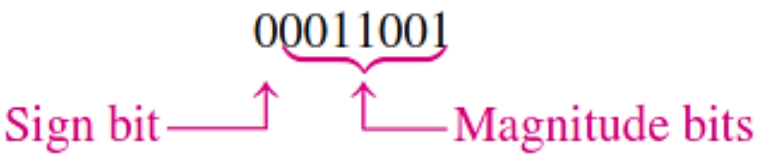
Chapter 1: Signed Binary Numbers

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.

Sign-Magnitude Form

When a signed binary number is represented in sign-magnitude, the left-most bit is the sign bit and the remaining bits are the magnitude bits.

+25 is expressed as an 8-bit signed binary number using the sign-magnitude form as



The decimal number −25 is expressed as

10011001

In the sign-magnitude form, a negative number has the same magnitude bits as the corresponding positive number but the sign bit is a 1 rather than a zero.

Chapter 1: Signed Binary Numbers

1's Complement Form

Positive numbers in 1's complement form are represented the same way as the positive sign-magnitude numbers.

Negative numbers, however, are the 1's complements of the corresponding Positive numbers.

In the 1's complement form, a negative number is the 1's complement of the corresponding positive number.

For example, using eight bits, the decimal number -25 is expressed as the 1's complement of +25 (00011001) as 11100110

Chapter 1: Signed Binary Numbers

2's Complement Form

Positive numbers in 2's complement form are represented the same way as in the sign magnitude and 1's complement forms.
Negative numbers are the 2's complements of the corresponding Positive numbers.

In the 2's complement form, a negative number is the 2's complement of the corresponding positive number.

Example: Using eight bits, let's take decimal number -25 and express it as the 2's complement of +25 (00011001).
Inverting each bit and adding 1, you get -25 = 11100111

For 2's complement signed numbers, **the range of values** for n-bit numbers is

$$\text{Range} = -(2^{n-1}) \text{ to } +(2^{n-1} - 1)$$

Chapter 1: Signed Binary Numbers

Express the decimal number −39 as an 8-bit number in the sign-magnitude, 1's complement, and 2's complement forms.

Solution

First, write the 8-bit number for +39.

00100111

In the *sign-magnitude form,* −39 is produced by changing the sign bit to a 1 and leaving the magnitude bits as they are. The number is

10100111

In the *1's complement form,* −39 is produced by taking the 1's complement of +39 (00100111).

11011000

In the *2's complement form,* −39 is produced by taking the 2's complement of +39 (00100111) as follows:

11011000	1's complement
+ 1	
11011001	2's complement

Chapter 1: 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 (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.

1 1 0 0 0 1 1 0

–128 +64 +4 +2 = –58

Chapter 1: Signed Binary Numbers

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	00001110 = +14	11111111 = –1
00001111 = +15	11101111 = –17	11111000 = –8
00101101 = +45	11111101 = –3	1 11110111 = –9

Discard carry

Chapter 1: Signed Binary Numbers

Both numbers positive:

$$\begin{array}{rr} 00000111 & 7 \\ +\ 00000100 & +\ 4 \\ \hline 00001011 & 11 \end{array}$$

The sum is positive and is therefore in true (uncomplemented) binary.

Positive number with magnitude larger than negative number:

$$\begin{array}{rrr} & 00001111 & 15 \\ & +\ 11111010 & +\ -6 \\ \hline \text{Discard carry} \longrightarrow 1 & 00001001 & 9 \end{array}$$

The final carry bit is discarded. The sum is positive and therefore in true (uncomplemented) binary.

Negative number with magnitude larger than positive number:

$$\begin{array}{rr} 00010000 & 16 \\ +\ 11101000 & +\ -24 \\ \hline 11111000 & -8 \end{array}$$

The sum is negative and therefore in 2's complement form.

Both numbers negative:

$$\begin{array}{rrr} & 11111011 & -5 \\ & +\ 11110111 & +\ -9 \\ \hline \text{Discard carry} \longrightarrow 1 & 11110010 & -14 \end{array}$$

The final carry bit is discarded. The sum is negative and therefore in 2's complement form.

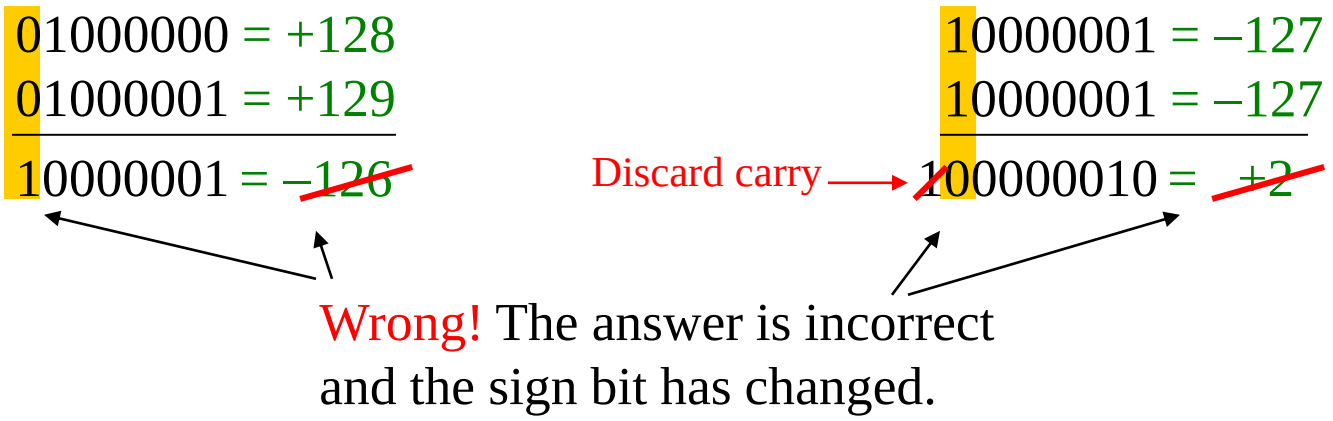
Chapter 1: Signed Binary Numbers

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:



Chapter 1: Signed Binary Numbers

Numbers added two at a time

Add the signed numbers: 01000100, 00011011, 00001110, and 00010010.

Solution

The equivalent decimal additions are given for reference.

68	01000100	
+ 27	+ 00011011	Add 1st two numbers
95	01011111	1st sum
+ 14	+ 00001110	Add 3rd number
109	01101101	2nd sum
+ 18	+ 00010010	Add 4th number
127	**01111111**	Final sum

Chapter 1: Signed Binary Numbers

Subtraction

Subtraction is a special case of addition. The subtraction operation changes the sign of the subtrahend and adds it to the minuend. The result of subtraction is called the difference.

Example: subtracting +6 (the subtrahend) from +9 (the minuend) is equivalent to adding -6 to +9.

The sign of a positive or negative binary number is changed by taking its 2's complement.

To subtract two signed numbers, take the 2's complement of the subtrahend and add. Discard any final carry bit.

Chapter 1: Signed Binary Numbers

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:

	00011110	(+30)		00001110	(+14)		11111111	(–1)
–	00000111	–(+15)	–	11101111	–(–17)	–	11111000	–(–8)

2's complement subtrahend and add:

	00011110	= +30		00001110	= +14		11111111	= –1
	11110001	= –15		00010001	= +17		00001000	= +8
	100001111	= +15		00011111	= +31		100000111	= +7

Discard carry

Discard carry

Chapter 1: Signed Binary Numbers

Perform each of the following subtractions of the signed numbers:

(a) 00001000 − 00000011 **(b)** 00001100 − 11110111

(c) 11100111 − 00010011 **(d)** 10001000 − 11100010

(a) In this case, $8 - 3 = 8 + (-3) = 5$.

	00001000	Minuend (+8)
	+ 11111101	2's complement of subtrahend (−3)
Discard carry ⟶	**1 00000101**	Difference (+5)

(b) In this case, $12 - (-9) = 12 + 9 = 21$.

00001100	Minuend (+12)
+ 00001001	2's complement of subtrahend (+9)
00010101	Difference (+21)

Chapter 1: Signed Binary Numbers

(c) In this case, $-25 - (+19) = -25 + (-19) = -44$.

	11100111	Minuend (−25)
	+ 11101101	2's complement of subtrahend (−19)
Discard carry	**1 11010100**	Difference (−44)

(d) In this case, $-120 - (-30) = -120 + 30 = -90$.

10001000	Minuend (−120)
+ 00011110	2's complement of subtrahend (+30)
10100110	Difference (−90)

Chapter 1: Signed Binary Numbers

Multiplication

The numbers in a multiplication are the multiplicand, the multiplier, and the product

8	Multiplicand
× 3	Multiplier
24	Product

The sign of the product of a multiplication depends on the signs of the multiplicand and the multiplier according to the following two rules:

- **If the signs are the same, the product is positive.**
- **If the signs are different, the product is negative.**

Two basic methods: ***Direct addition*** and ***partial products***

Chapter 1: Signed Binary Numbers

Multiplication

In the ***direct addition*** method, you add the multiplicand a number of times equal to the multiplier.

Multiply the signed binary numbers: 01001101 (multiplicand) and 00000100 (multiplier) using the direct addition method.

Solution

Since both numbers are positive, they are in true form, and the product will be positive. The decimal value of the multiplier is 4, so the multiplicand is added to itself four times as follows:

01001101	1st time
+ 01001101	2nd time
10011010	Partial sum
+ 01001101	3rd time
11100111	Partial sum
+ 01001101	4th time
100110100	Product

Chapter 1: Signed Binary Numbers

Multiplication

The ***partial products*** method is perhaps the more common one because it reflects the way you multiply longhand. The multiplicand is multiplied by each multiplier digit beginning with the least significant digit. The result of the multiplication of the multiplicand by a multiplier digit is called a partial product.

239	Multiplicand
× 123	Multiplier
717	1st partial product (3 × 239)
478	2nd partial product (2 × 239)
+ 239	3rd partial product (1 × 239)
29,397	Final product

Chapter 1: Signed Binary Numbers

The **basic steps** in the partial products method of binary multiplication:
Step 1: Determine if the signs of the multiplicand and multiplier are the same or different. This determines what the sign of the product will be.
Step 2: Change any negative number to true (uncomplemented) form. Because most computers store negative numbers in 2's complement, a 2's complement operation is required to get the negative number into true form.
Step 3: Starting with the least significant multiplier bit, generate the partial products. When the multiplier bit is 1, the partial product is the same as the multiplicand. When the multiplier bit is 0, the partial product is zero. Shift each successive partial product one bit to the left.
Step 4: Add each successive partial product to the sum of the previous partial products to get the final product.
Step 5: If the sign bit that was determined in step 1 is negative, take the 2's complement of the product. If positive, leave the product in true form. Attach the sign bit to the product.

Chapter 1: Signed Binary Numbers

Multiply the signed binary numbers: 01010011 (multiplicand) and 11000101 (multiplier).

Solution

Step 1: The sign bit of the multiplicand is 0 and the sign bit of the multiplier is 1. The sign bit of the product will be 1 (negative).

Step 2: Take the 2's complement of the multiplier to put it in true form.

$$11000101 \longrightarrow 00111011$$

Chapter 1: Signed Binary Numbers

Step 3 and 4: The multiplication proceeds as follows. Notice that only the magnitude bits are used in these steps.

1010011	Multiplicand
× 0111011	Multiplier
1010011	1st partial product
+ 1010011	2nd partial product
11111001	Sum of 1st and 2nd
+ 0000000	3rd partial product
011111001	Sum
+ 1010011	4th partial product
1110010001	Sum
+ 1010011	5th partial product
100011000001	Sum
+ 1010011	6th partial product
1001100100001	Sum
+ 0000000	7th partial product
1001100100001	Final product

Chapter 1: Signed Binary Numbers

Step 5: Since the sign of the product is a 1 as determined in step 1, take the 2's complement of the product.

$$1001100100001 \longrightarrow 0110011011111$$

Attach the sign bit →

1 0110011011111

Chapter 1: Signed Binary Numbers

Division

The numbers in a division are the **dividend, the divisor, and the quotient.** These are illustrated in the following standard division format.

$$\frac{\text{dividend}}{\text{divisor}} = \text{quotient}$$

The sign of the quotient depends on the signs of the dividend and the divisor according to the following two rules:

- If the signs are the same, the quotient is positive.
- If the signs are different, the quotient is negative.

Chapter 1: Signed Binary Numbers

Division

The basic steps in a division process are as follows:

Step 1: Determine if the signs of the dividend and divisor are the same or different. This determines what the sign of the quotient will be. Initialize the quotient to zero.

Step 2: Subtract the divisor from the dividend using 2's complement addition to get the first partial remainder and add 1 to the quotient. If this partial remainder is positive, go to step 3. If the partial remainder is zero or negative, the division is complete.

Step 3: Subtract the divisor from the partial remainder and add 1 to the quotient. If the result is positive, repeat for the next partial remainder. If the result is zero or negative, the division is complete.

Continue to subtract the divisor from the dividend and the partial remainders until there is a zero or a negative result. Count the number of times that the divisor is subtracted and you have the quotient.

Chapter 1: Signed Binary Numbers

Divide 01100100 by 00011001.

Solution

Step 1: The signs of both numbers are positive, so the quotient will be positive. The quotient is initially zero: 00000000.

Step 2: Subtract the divisor from the dividend using 2's complement addition (remember that final carries are discarded).

01100100	Dividend
+ 11100111	2's complement of divisor
01001011	Positive 1st partial remainder

Add 1 to quotient: 00000000 + 00000001 = 00000001.

Step 3: Subtract the divisor from the 1st partial remainder using 2's complement addition.

01001011	1st partial remainder
+ 11100111	2's complement of divisor
00110010	Positive 2nd partial remainder

Add 1 to quotient: 00000001 + 00000001 = 00000010.

Chapter 1: Signed Binary Numbers

Step 4: Subtract the divisor from the 2nd partial remainder using 2's complement addition.

00110010	2nd partial remainder
+ 11100111	2's complement of divisor
00011001	Positive 3rd partial remainder

Add 1 to quotient: 00000010 + 00000001 = 00000011.

Step 5: Subtract the divisor from the 3rd partial remainder using 2's complement addition.

00011001	3rd partial remainder
+ 11100111	2's complement of divisor
00000000	Zero remainder

Add 1 to quotient: 00000011 + 00000001 = **00000100** (final quotient). The process is complete.