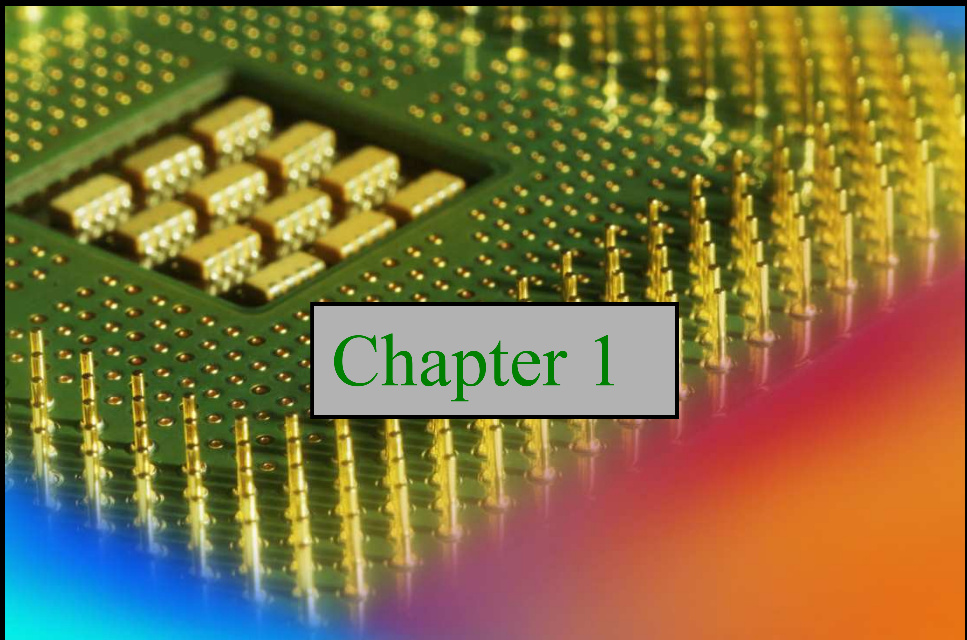
Digital Fundamentals

Tenth Edition

Floyd

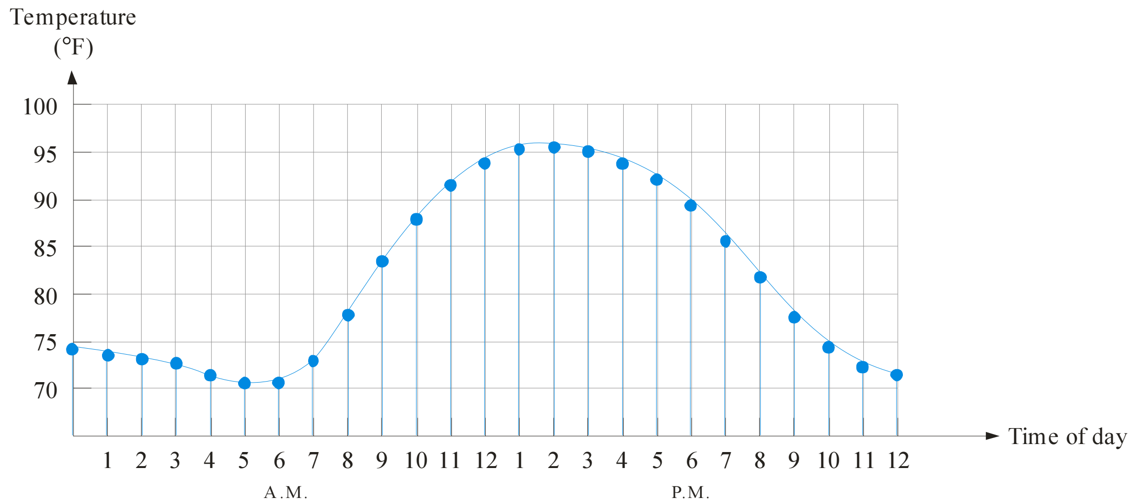


Chapter 1

Summary

Analog Quantities

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

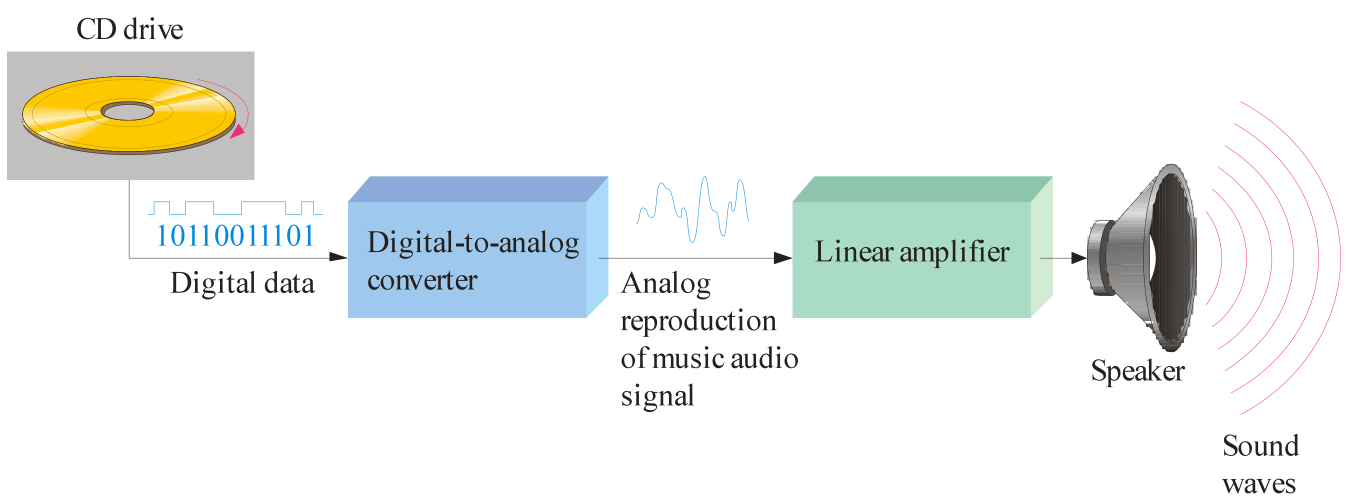


Digital systems can process, store, and transmit data more efficiently but can only assign discrete values to each point.

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.

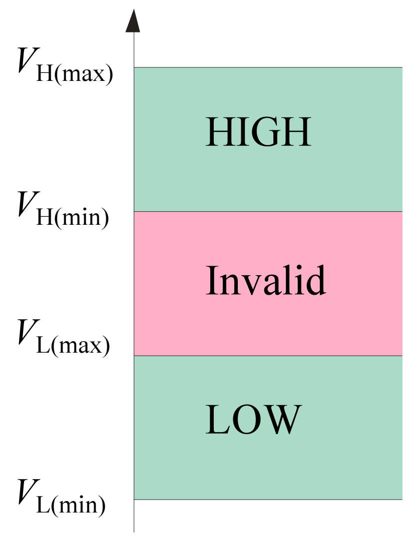


Summary

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.

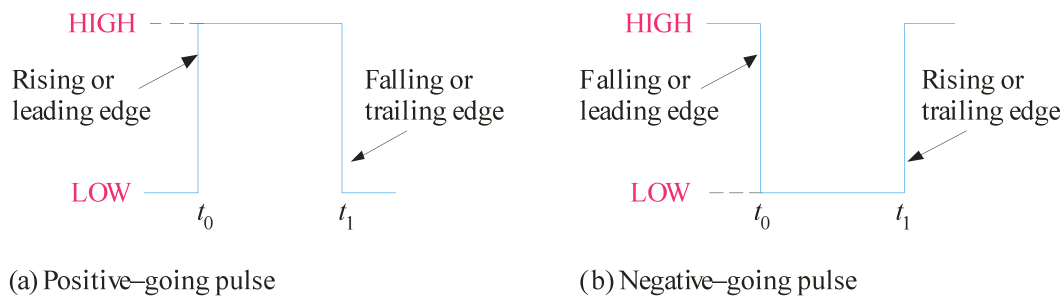
In binary, a single number is called a *bit* (for *b*inary dig*it*). A bit can have the value of either a 0 or a 1, depending on if the voltage is HIGH or LOW.



Summary

Digital Waveforms

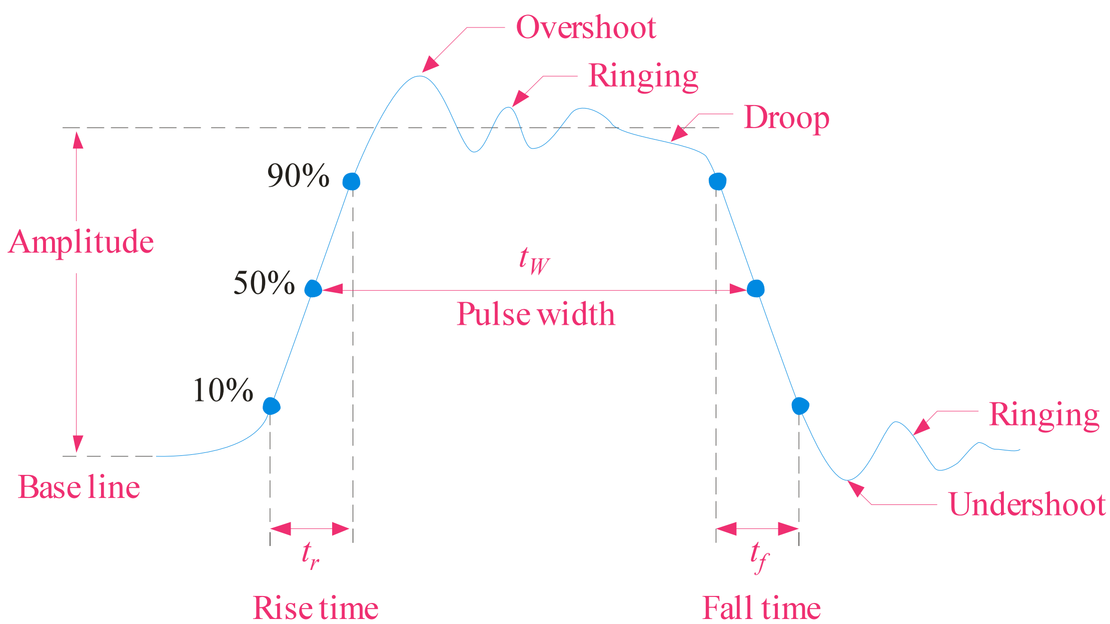
Digital waveforms change between the LOW and HIGH levels. A positive going pulse is one that goes from a normally LOW logic level to a HIGH level and then back again. Digital waveforms are made up of a series of pulses.



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.

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

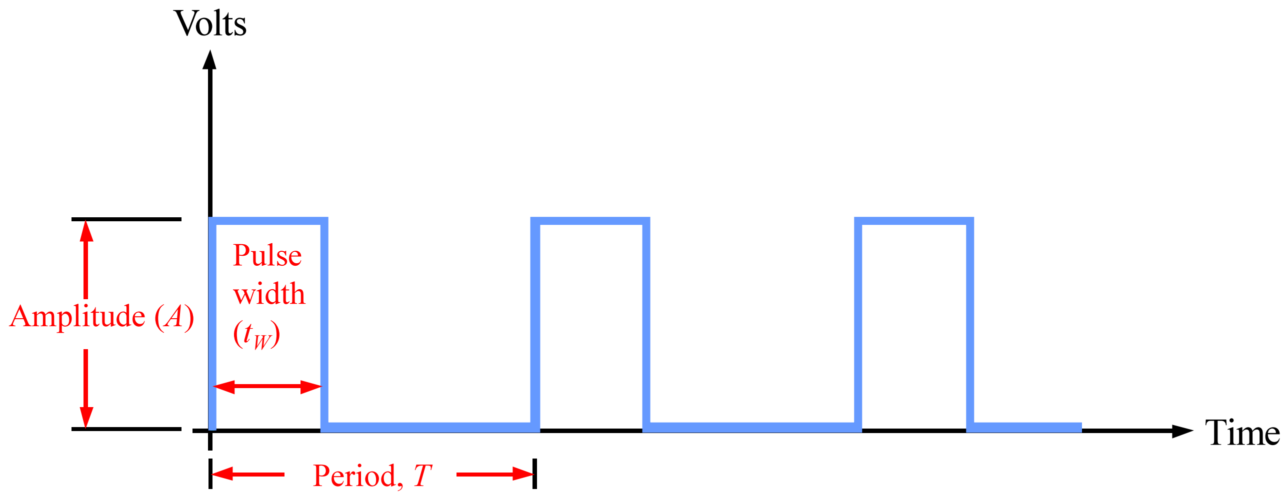
Solution

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

Summary

Pulse Definitions

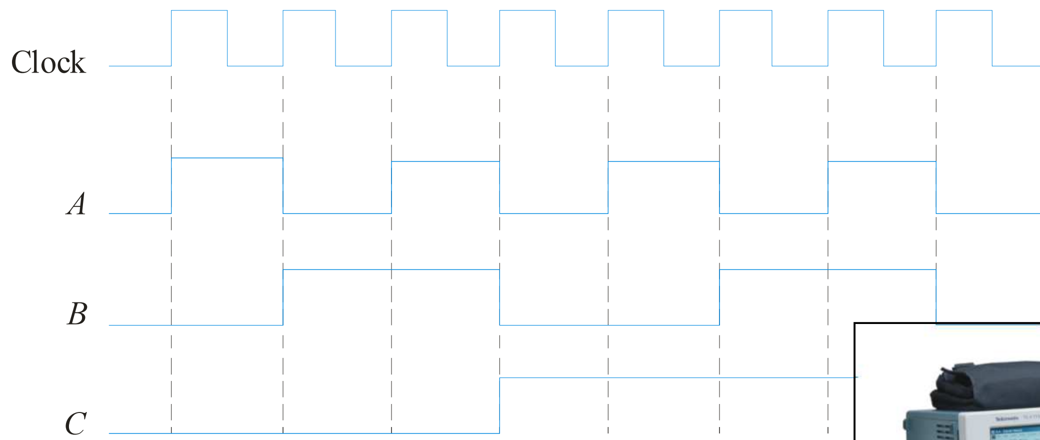
In addition to frequency and period, repetitive pulse waveforms are described by the amplitude (A), pulse width (t_W) and duty cycle. Duty cycle is the ratio of t_W to T.



Summary

Timing Diagrams

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

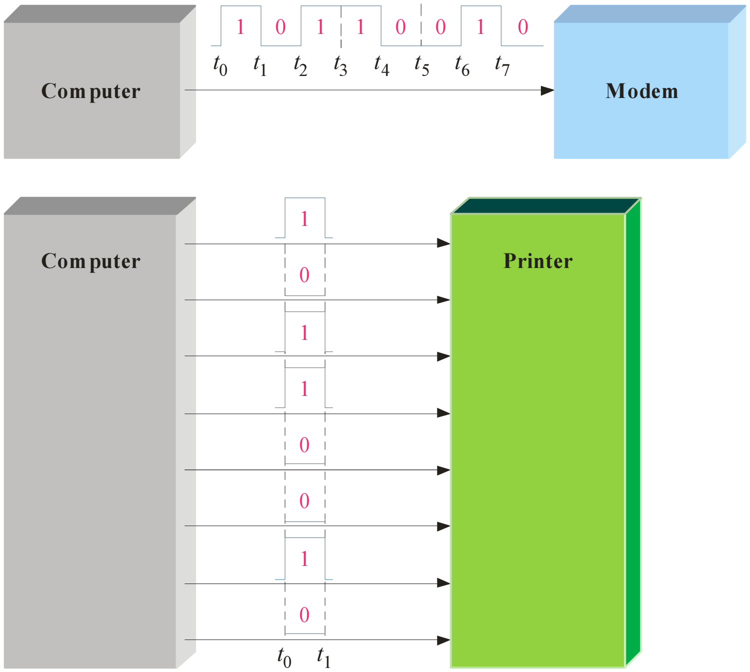


A diagram like this can be observed directly on a logic analyzer.

Summary

Serial and Parallel Data

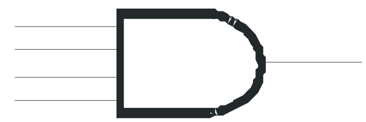
Data can be transmitted by either serial transfer or parallel transfer.



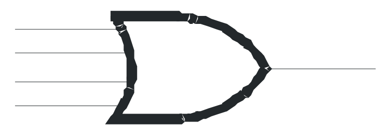
Summary

Basic Logic Functions

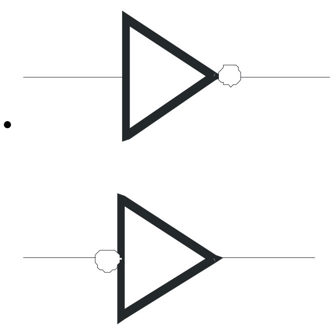
AND True only if *all* input conditions are true.



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



NOT Indicates the *opposite* condition.

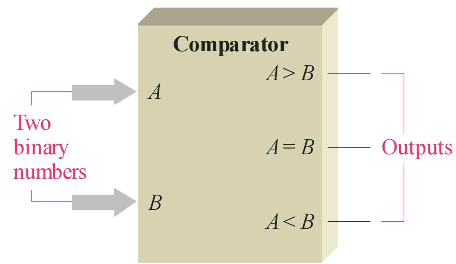


Summary

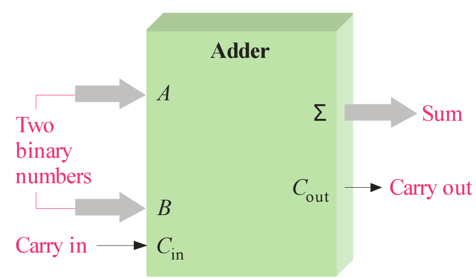
Basic System Functions

And, **or**, and **not** elements can be combined to form various logic functions. A few examples are:

The comparison function



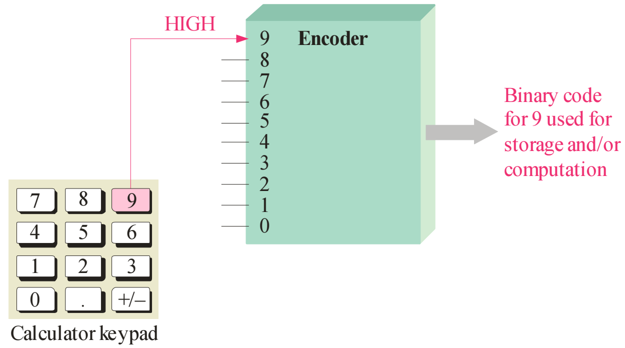
Basic arithmetic functions



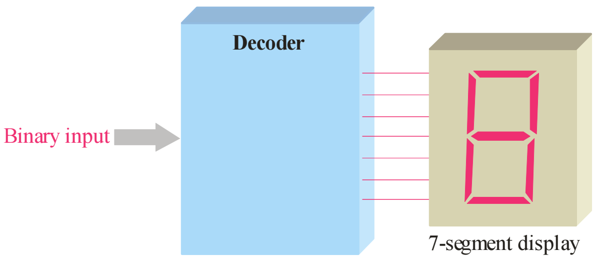
Summary

Basic System Functions

The encoding function



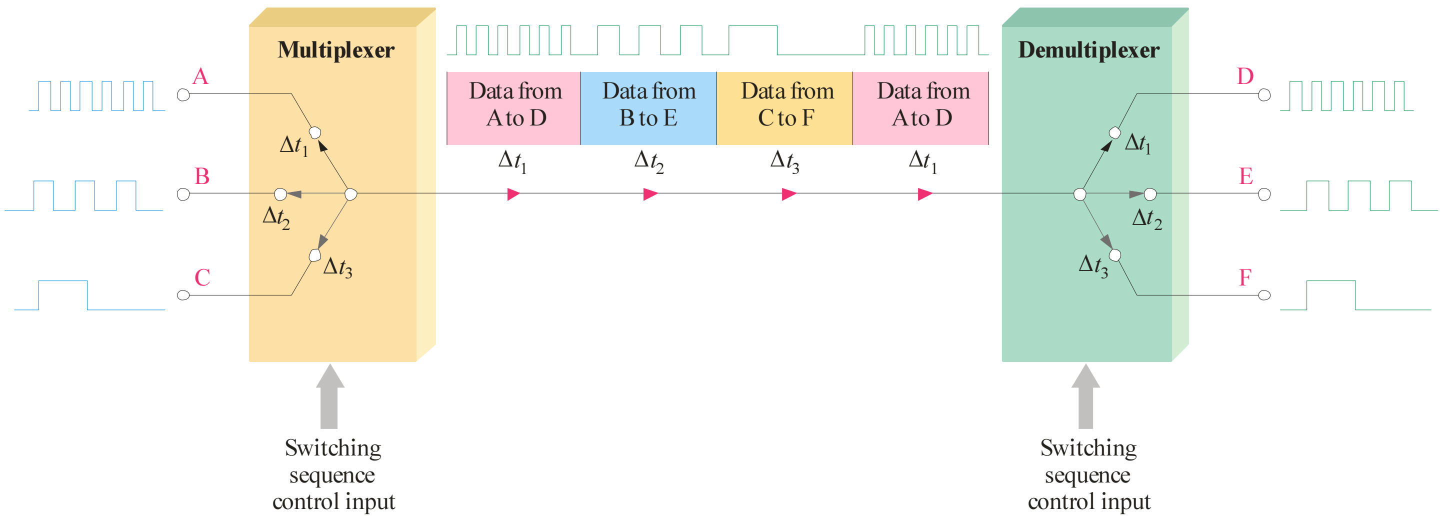
The decoding function



Summary

Basic System Functions

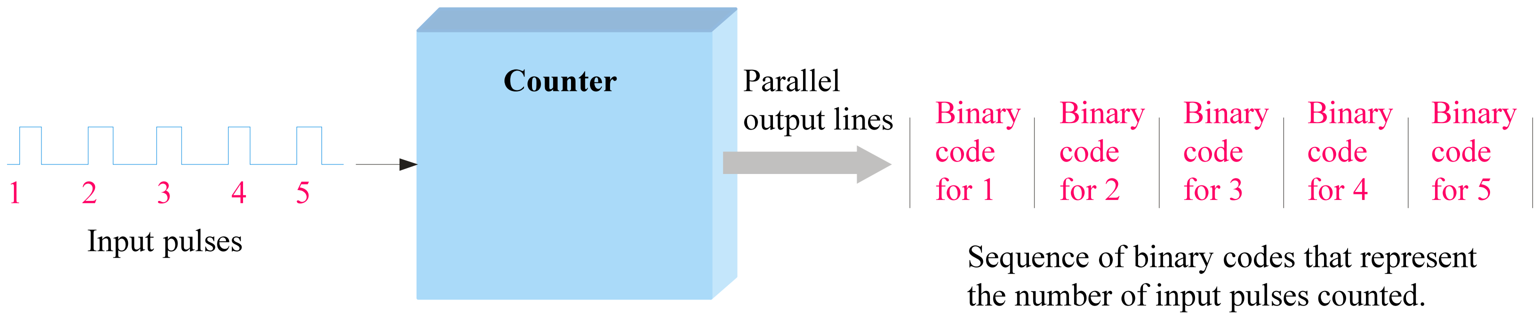
The data selection function



Summary

Basic System Functions

The counting function

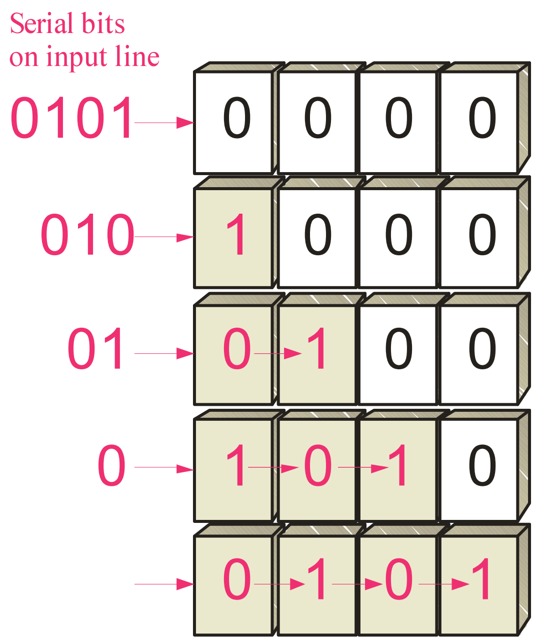


…and other functions such as code conversion and storage.

Summary

Basic System Functions

One type of storage function is the shift register, that moves and stores data each time it is clocked.



Initially the register contains only invalid data or all zeros as shown here.

First bit (1) is shifted serially into the register.

Second bit (0) is shifted serially into register and first bit is shifted right.

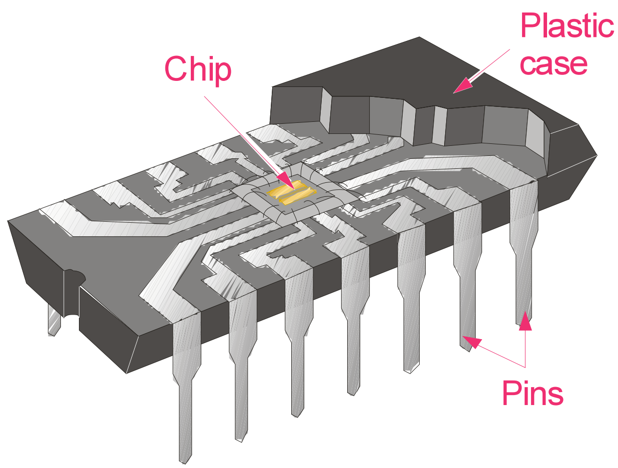
Third bit (1) is shifted into register and the first and second bits are shifted right.

Fourth bit (0) is shifted into register and the first, second, and third bits are shifted right. The register now stores all four bits and is full.

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

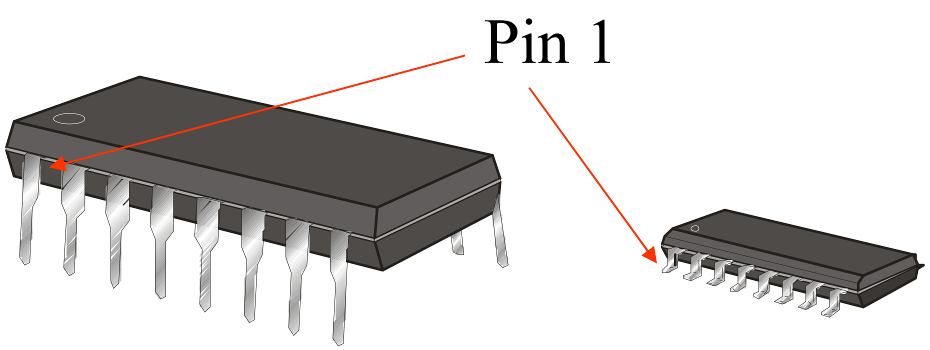
An example of laboratory prototyping is shown. The circuit is wired using DIP chips and tested.

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

Summary

Integrated Circuits

DIP chips and surface mount chips



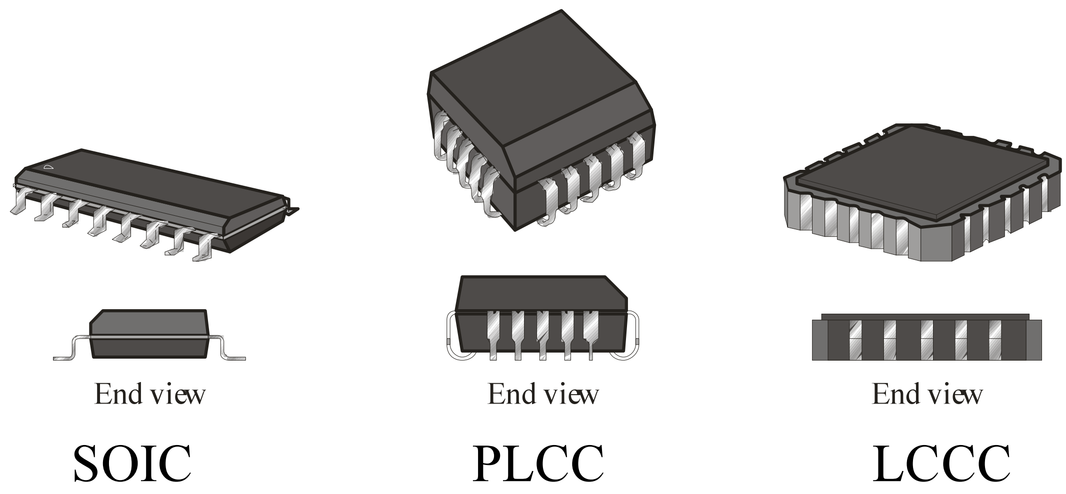
Dual in-line package

Small outline IC (SOIC)

Summary

Integrated Circuits

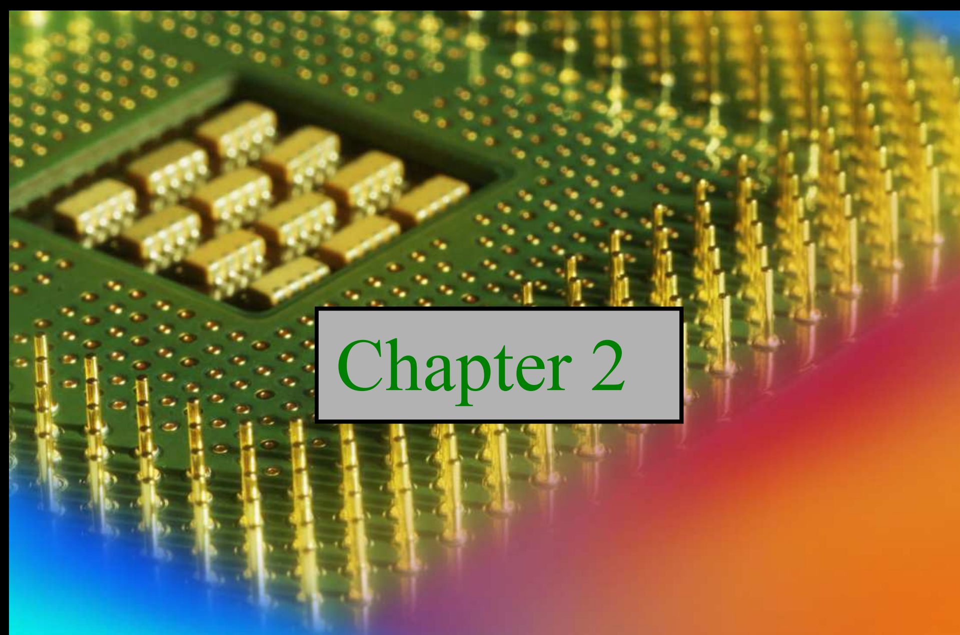
Other surface mount packages:



Digital Fundamentals

Tenth Edition

Floyd



Chapter 2

Summary

Decimal Numbers

The position of each digit in a weighted number system is assigned a weight based on the **base** or **radix** of the system. 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 $10^0 = 1$:

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

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

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

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:

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

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

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

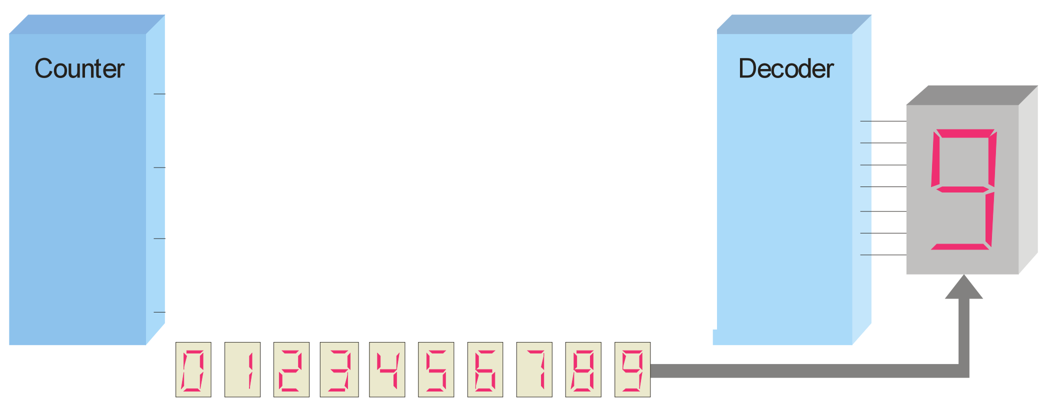
Summary

Binary Numbers

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

Notice the pattern of zeros and ones in each column.

Digital counters frequently have this same pattern of digits:



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.

$2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 . \; 2^{-1} \quad 2^{-2}$

$32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \; . \; \tfrac{1}{2} \quad \tfrac{1}{4}$

$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 . \; 0 \quad 1$

$32 \qquad +4 \quad +1 \quad +\tfrac{1}{4} = 37\tfrac{1}{4}$

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 x 2 = 0.376	carry = 0	MSB
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	

Answer = .00110 (for five significant digits)

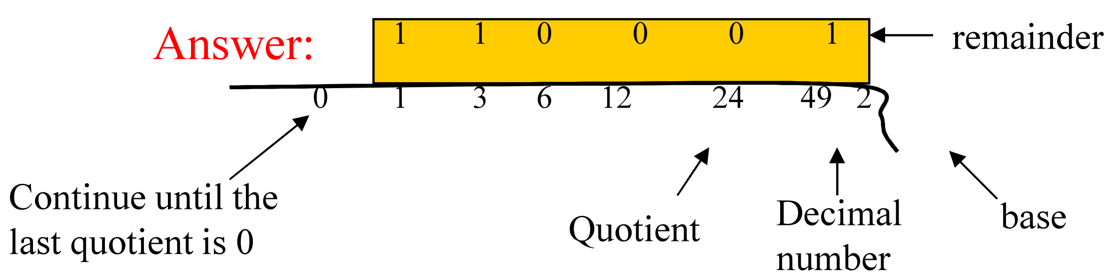
Summary

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

0 1 1 1	
00111	7
10101	21
1 1 1 0 0	= 28

Summary

Binary Subtraction

The rules for binary subtraction are

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

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

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

Solution

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

Summary

Binary Multiplication/ Division

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

Perform the following binary multiplications:

(a) 11×11 **(b)** 101×111

Solution

(a)

$$\begin{array}{r} 11 \\ \times\, 11 \\ \hline 11 \\ +11 \\ \hline \mathbf{1001} \end{array} \qquad \begin{array}{r} 3 \\ \times\, 3 \\ \hline 9 \end{array}$$

Partial products: 11, $+11$

(b)

$$\begin{array}{r} 111 \\ \times\, 101 \\ \hline 111 \\ 000 \\ +111 \\ \hline \mathbf{100011} \end{array} \qquad \begin{array}{r} 7 \\ \times\, 5 \\ \hline 35 \end{array}$$

Partial products: 111, 000, $+111$

Related Problem Multiply 1101×1010.

Perform the following binary divisions:

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

Solution

(a)

$$\begin{array}{r} \mathbf{10} \\ 11\overline{)110} \\ 11 \\ \hline 000 \end{array} \qquad \begin{array}{r} 2 \\ 3\overline{)6} \\ 6 \\ \hline 0 \end{array}$$

(b)

$$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ 10 \\ \hline 10 \\ 10 \\ \hline 00 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ 6 \\ \hline 0 \end{array}$$

Related Problem Divide 1100 by 100.

Summary

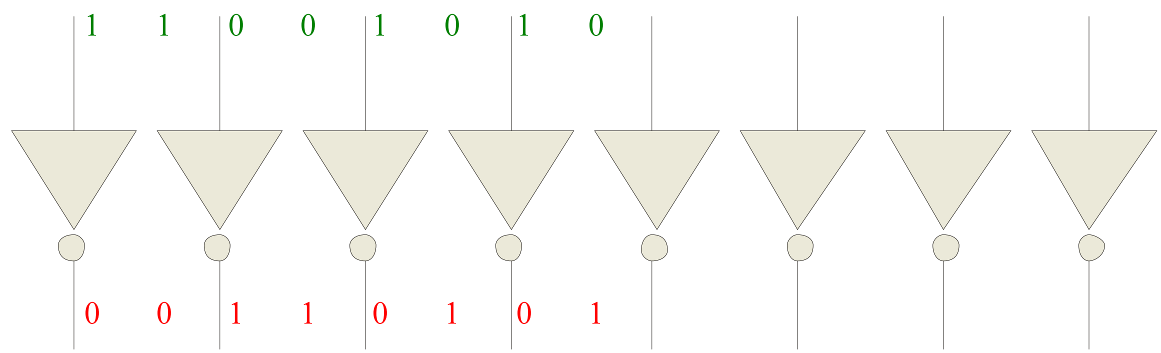
1's Complement

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.

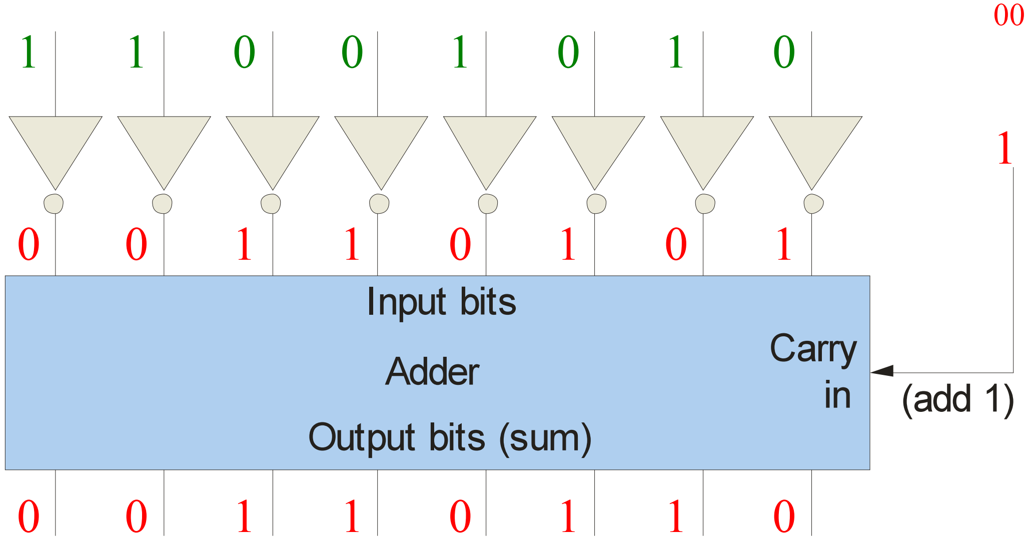
Recall that the 1's complement of 11001010 is

00110101 (1's complement)

+1

00110110 (2's complement)

To form the 2's complement, add 1:



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 (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

Summary

Floating Point Numbers

Floating point notation is capable of representing very large or small numbers by using a form of scientific notation. A 32-bit single precision number is illustrated.

S	E (8 bits)	F (23 bits)

Sign bit   Biased exponent (+127)   Magnitude with MSB dropped

Example Express the speed of light, c, in single precision floating point notation. ($c = 0.2998 \times 10^9$)

Solution In binary, $c = 0001\ 0001\ 1101\ 1110\ 1001\ 0101\ 1100\ 0000_2$.

In scientific notation, $c = 1.001\ 1101\ 1110\ 1001\ 0101\ 1100\ 0000 \times 2^{28}$.

S = 0 because the number is positive. $E = 28 + 127 = 155_{10} = 1001\ 1011_2$.

F is the next 23 bits after the first 1 is dropped.

In floating point notation, c =

0	10011011	001 1101 1110 1001 0101 1100

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:

$$\begin{array}{rl} 00011110 & = +30 \\ 00001111 & = +15 \\ \hline 00101101 & = +45 \end{array} \qquad \begin{array}{rl} 00001110 & = +14 \\ 11101111 & = -17 \\ \hline 11111101 & = -3 \end{array} \qquad \begin{array}{rl} 11111111 & = -1 \\ 11111000 & = -8 \\ \hline \cancel{1}\,11110111 & = -9 \end{array}$$

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:

$$
\begin{array}{r}
01000000 = +128 \\
01000001 = +129 \\
\hline
10000001 = -\cancel{126}
\end{array}
\qquad
\begin{array}{r}
10000001 = -127 \\
10000001 = -127 \\
\hline
\cancel{1}00000010 = \cancel{+2}
\end{array}
$$

Discard carry

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

Summary

Arithmetic Operations with Signed 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:

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

2's complement subtrahend and add:

$$\begin{array}{r} 00011110 = +30 \\ 11110001 = -15 \\ \hline \cancel{1}00001111 = +15 \end{array} \qquad \begin{array}{r} 00001110 = +14 \\ 00010001 = +17 \\ \hline 00011111 = +31 \end{array} \qquad \begin{array}{r} 11111111 = -1 \\ 00001000 = +8 \\ \hline \cancel{1}00000111 = +7 \end{array}$$

Discard carry

Discard carry

Summary

Hexadecimal Numbers

Hexadecimal uses sixteen characters to represent numbers: the numbers 0 through 9 and the alphabetic characters A through F.

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

Large binary number can easily be converted to hexadecimal by grouping bits 4 at a time and writing the equivalent hexadecimal character.

Example Express $1001\ 0110\ 0000\ 1110_2$ in hexadecimal:

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

Summary

Hexadecimal Numbers

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

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

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:

4096 256 16 1

1 A 2 F_{16}

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

Summary

Hexadecimal Conversions

Convert the following binary numbers to hexadecimal:

(a) 1100101001010111 **(b)** 111111000101101001

Solution

(a) $\underbrace{1100}_{\text{C}}\underbrace{1010}_{\text{A}}\underbrace{0101}_{\text{5}}\underbrace{0111}_{\text{7}} = \mathbf{CA57_{16}}$

(b) $\underbrace{0011}_{\text{3}}\underbrace{1111}_{\text{F}}\underbrace{0001}_{\text{1}}\underbrace{0110}_{\text{6}}\underbrace{1001}_{\text{9}} = \mathbf{3F169_{16}}$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

Determine the binary numbers for the following hexadecimal numbers:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solution

(a) $\underbrace{1}_{1}\underbrace{0000}_{0}\underbrace{1010}_{A}\underbrace{0100}_{4}$ → **1000010100100**

(b) $\underbrace{1100}_{C}\underbrace{1111}_{F}\underbrace{1000}_{8}\underbrace{1110}_{E}$ → **1100111110001110**

(c) $\underbrace{1001}_{9}\underbrace{0111}_{7}\underbrace{0100}_{4}\underbrace{0010}_{2}$ → **1001011101000010**

In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

Summary

Hexadecimal Conversions

Convert the following hexadecimal numbers to decimal:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solution Remember, convert the hexadecimal number to binary first, then to decimal.

(a)
$$\underbrace{0001}_{1}\underbrace{1100}_{C} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28_{10}}$$

(b)
$$\underbrace{1010}_{A}\underbrace{1000}_{8}\underbrace{0101}_{5} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693_{10}}$$

Convert the following hexadecimal numbers to decimal:

(a) $E5_{16}$ **(b)** $B2F8_{16}$

Solution Recall from Table 2–3 that letters A through F represent decimal numbers 10 through 15, respectively.

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229_{10}}$

(b)
$$\begin{aligned} B2F8_{16} &= (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1) \\ &= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1) \\ &= 45{,}056 + 512 + 240 + 8 = \mathbf{45{,}816_{10}} \end{aligned}$$

Summary

Hexadecimal Conversions

Convert the decimal number 650 to hexadecimal by repeated division by 16.

Solution

Hexadecimal remainder

$$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 = \text{A}$$

$$\frac{40}{16} = 2.5 \longrightarrow 0.5 \times 16 = 8 = 8$$

$$\frac{2}{16} = 0.125 \longrightarrow 0.125 \times 16 = 2 = 2$$

Stop when whole number quotient is zero.

2 8 A Hexadecimal number

MSD LSD

Summary

Hexadecimal Addition

Add the following hexadecimal numbers:

(a) $23_{16} + 16_{16}$ **(b)** $58_{16} + 22_{16}$ **(c)** $2B_{16} + 84_{16}$ **(d)** $DF_{16} + AC_{16}$

Solution

(a)

$$\begin{array}{r} 23_{16} \\ +16_{16} \\ \hline \mathbf{39}_{16} \end{array}$$

right column: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$

left column: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b)

$$\begin{array}{r} 58_{16} \\ +\ 22_{16} \\ \hline \mathbf{7A}_{16} \end{array}$$

right column: $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$

left column: $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$

(c)

$$\begin{array}{r} 2B_{16} \\ +\ 84_{16} \\ \hline \mathbf{AF}_{16} \end{array}$$

right column: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$

left column: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

(d)

$$\begin{array}{r} DF_{16} \\ +\ AC_{16} \\ \hline \mathbf{18B}_{16} \end{array}$$

right column: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$

$27_{10} - 16_{10} = 11_{10} = B_{16}$ with a 1 carry

left column: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$

$24_{10} - 16_{10} = 8_{10} = 8_{16}$ with a 1 carry

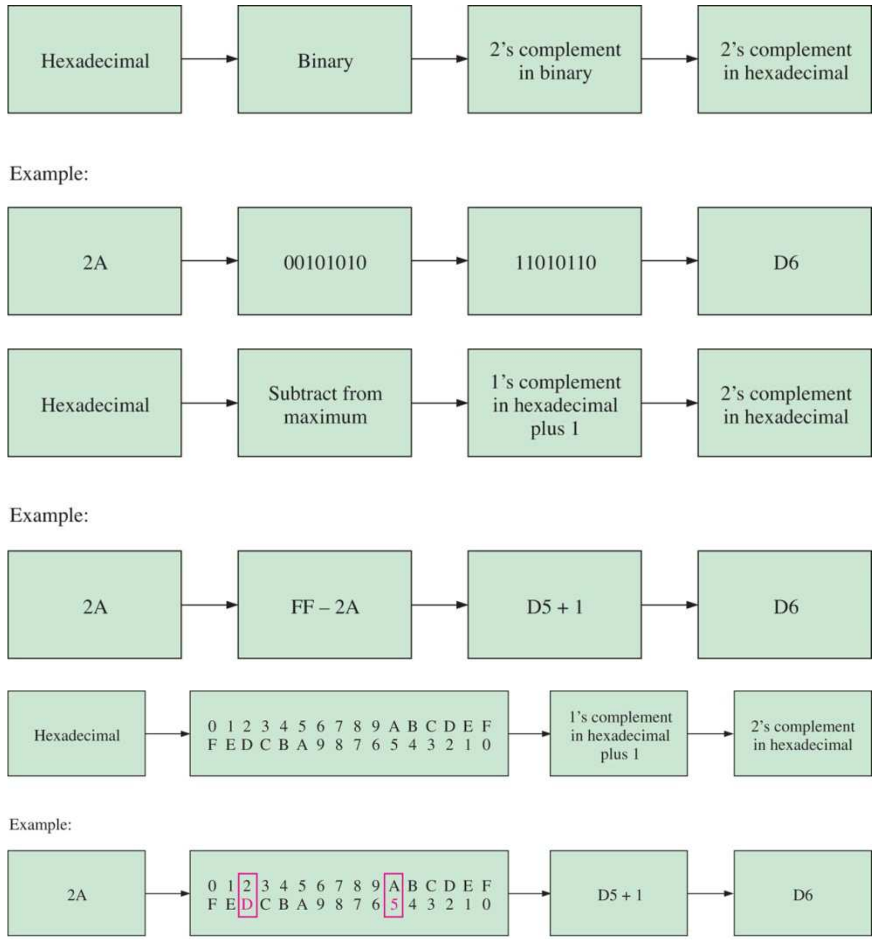
Summary

Hexadecimal Subtraction

Method I

Method II

Method III



Summary

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.

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

Example Express $1\ 001\ 011\ 000\ 001\ 110_2$ in octal:

Solution Group the binary number by 3-bits starting from the right. Thus, 113016_8

Summary

Octal Numbers

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

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

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

Example Express 3702_8 in decimal.

Solution Start by writing the column weights:

512 64 8 1

3 7 0 2_8

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

Summary

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	00010000
11	1011	00010001
12	1100	00010010
13	1101	00010011
14	1110	00010100
15	1111	00010101

Summary

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 Addition

Example I

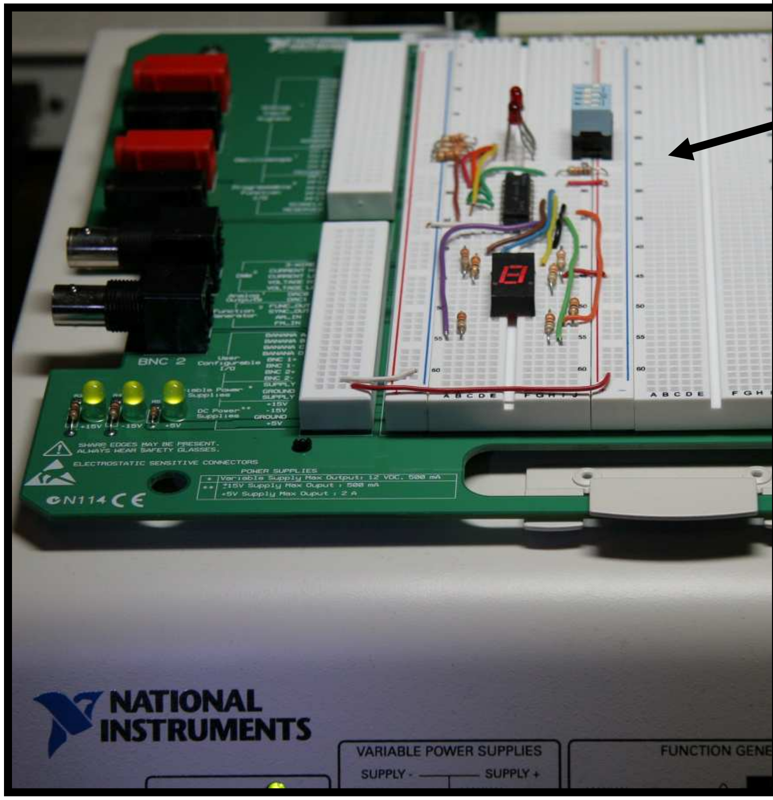
0001	0110		16
+ 0001	0101		+ 15
0010	1011	Right group is invalid (>9), left group is valid.	31
	+ 0110	Add 6 to invalid code. Add carry, 0001, to next group.	
0011	**0001**	Valid BCD number	
↓	↓		
3	1		

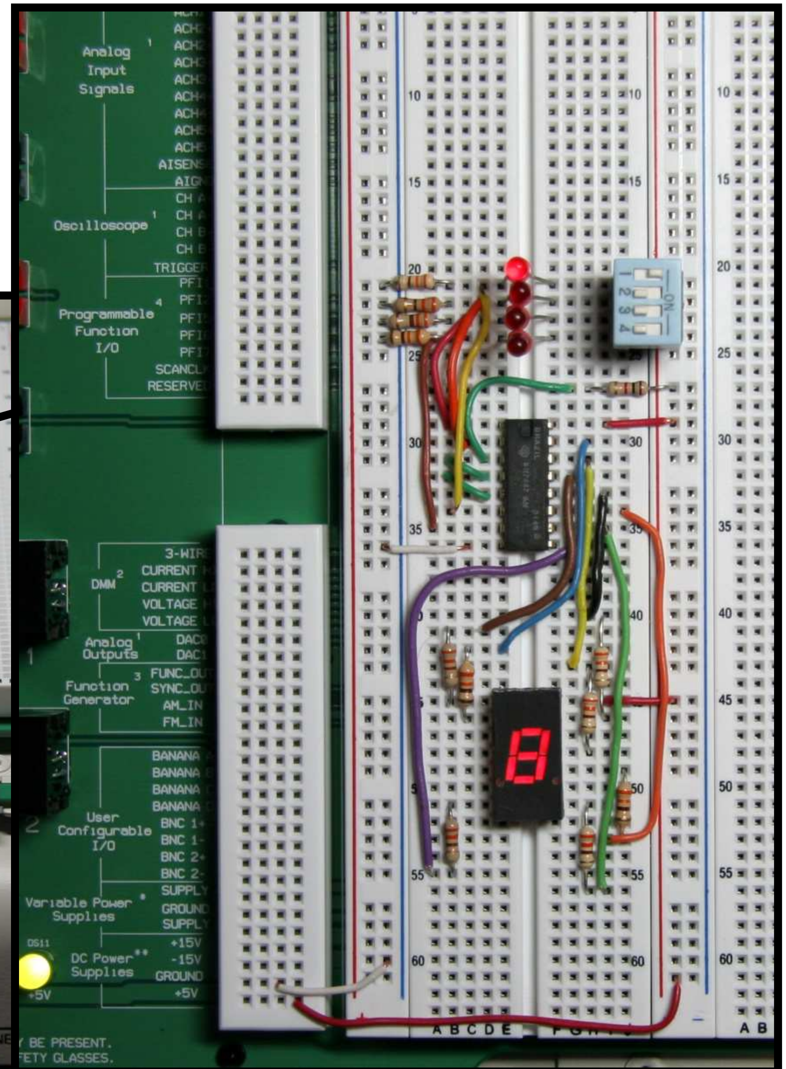
Example II

	0110	0111		67
	+ 0101	0011		+ 53
	1011	1010	Both groups are invalid (>9)	120
	+ 0110	+ 0110	Add 6 to both groups	
0001	**0010**	**0000**	Valid BCD number	
↓	↓	↓		
1	2	0		

BCD

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





Summary

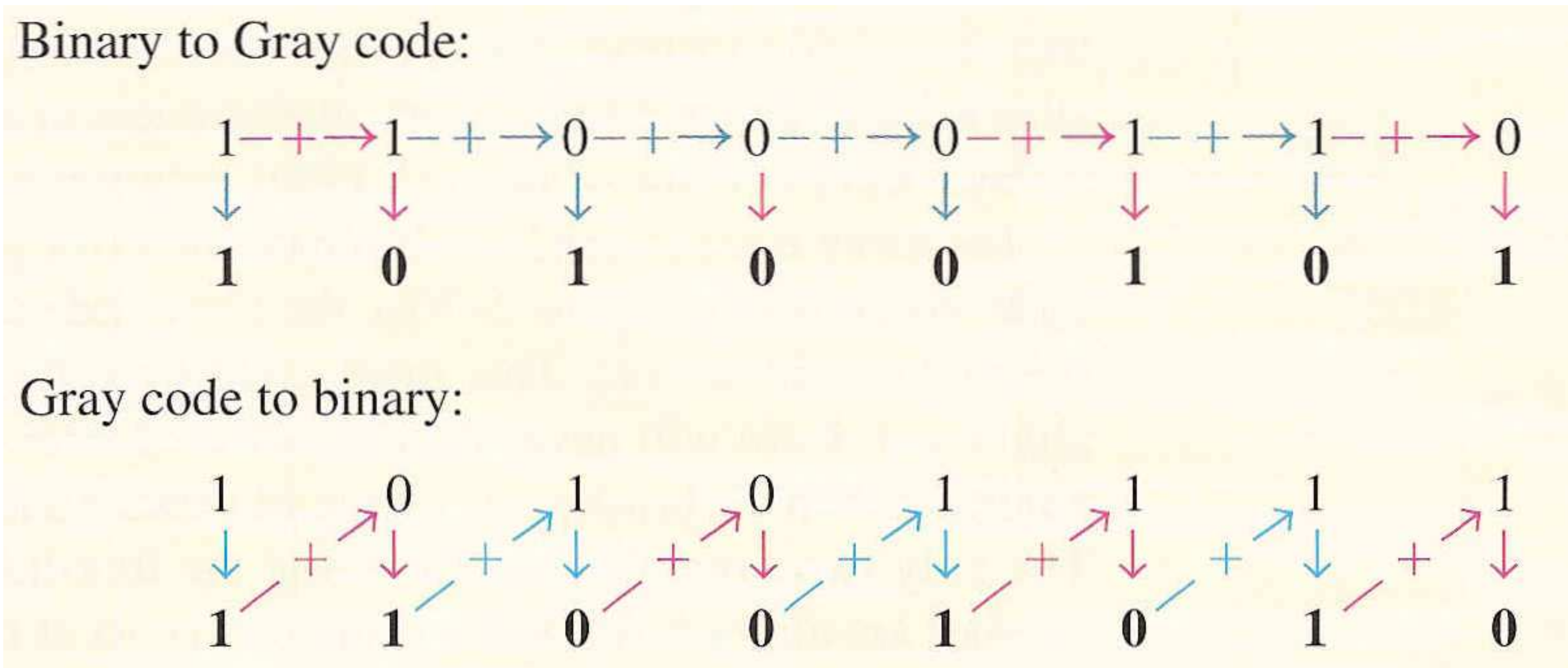
Gray code

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

Summary

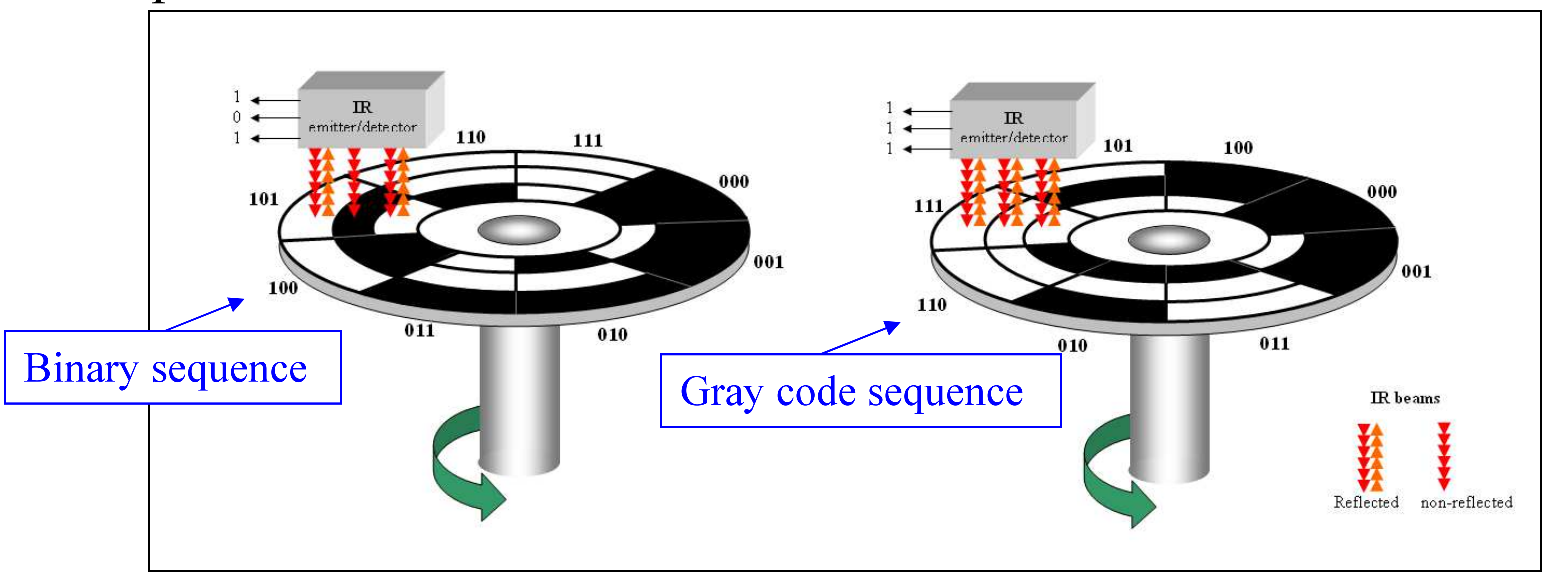
Gray code conversion



Summary

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.



Summary

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.

Summary

Parity Method

The parity method is a method of error detection for simple transmission errors involving one bit (or an odd number of bits). A parity bit is an "extra" bit attached to a group of bits to force the number of 1's to be either even (even parity) or odd (odd parity).

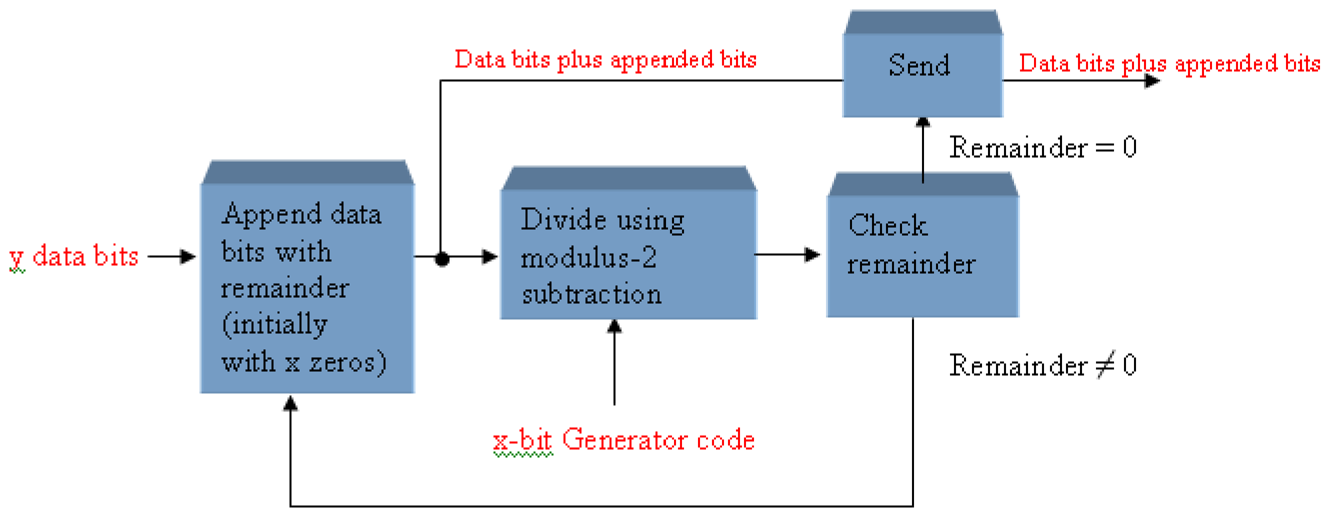
Example The ASCII character for "a" is 1100001 and for "A" is 1000001. What is the correct bit to append to make both of these have odd parity?

Solution The ASCII "a" has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII "A" has an even number of bits that are equal to 1; therefore the parity bit is 1.

Summary

Cyclic Redundancy Check

The cyclic redundancy check (CRC) is an error detection method that can detect multiple errors in larger blocks of data. At the sending end, a checksum is appended to a block of data. At the receiving end, the check sum is generated and compared to the sent checksum. If the check sums are the same, no error is detected.



Homework 6

- 2-2 (7-9 only f-h)
- 2-3 (11-14 only a-c)
- 2-4 (15-18 only a and b)
- 2-5 (21,22 only a and b)
- 2-6 (30)
- 2-7 (31 only a)
- 2-8 (37, 38 only a and b)