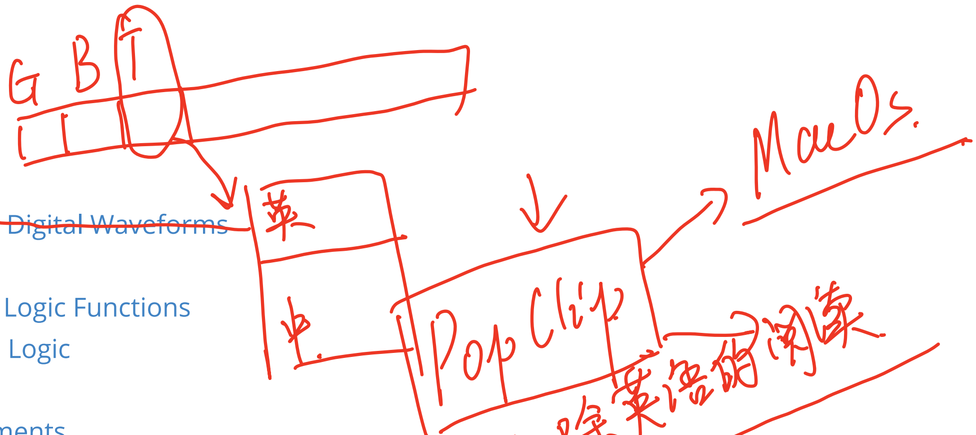


扫除英语的阅读

① PopClip

原英文教材 方便.

② 中文

问题1: 删减了.

问题2: 图 重新处理.

问题3: 作业. → 英文PDF中.

问题4: 移位寄存器. 康华光. 阎石.

③ 讲课:

★ 记笔记: 灵活 理解. / 悟性.

举一反三.

突出重点, 举例子, 引申. → 目的.

让你理解、

认真听, 听懂.

最容易的一门课.

线下课堂中，分阶段小测试

平时成绩	期末
2	8
5	5

最后商定比.

☆ chap1, chap2:

作业：课代表2：每做完2章交一次。纸/PDF.

考勤：点名 {线上点名：随机抽同学回答问题.
线下要严.

缺3次平时成绩0分

请假：假条.

考核：{试卷散、答题.
作业.

考核前：问题.

后：不会再接任何咨询

精力放在平时。抱.

in detail. X

学习方法：① 读英文原版.
② 上课听懂.
③ 作业独立完成

抄 X

模电.

二极管

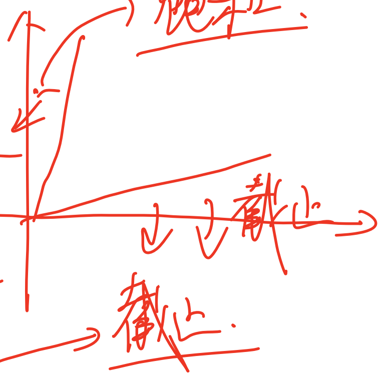
三极管 →

MOSFET / JFET } 三个放大电路 ← 小信号

振荡器（正负反馈.

功放. 功率 V/I

动放.（虚短、虚断）



饱和.

截止

数电：

状态1 → 导通

逻辑上正反两面.

状态2 → 截止.

三极管.

Logic. '0' / '1' T / F

二进制 逻辑.

十进制：0~9

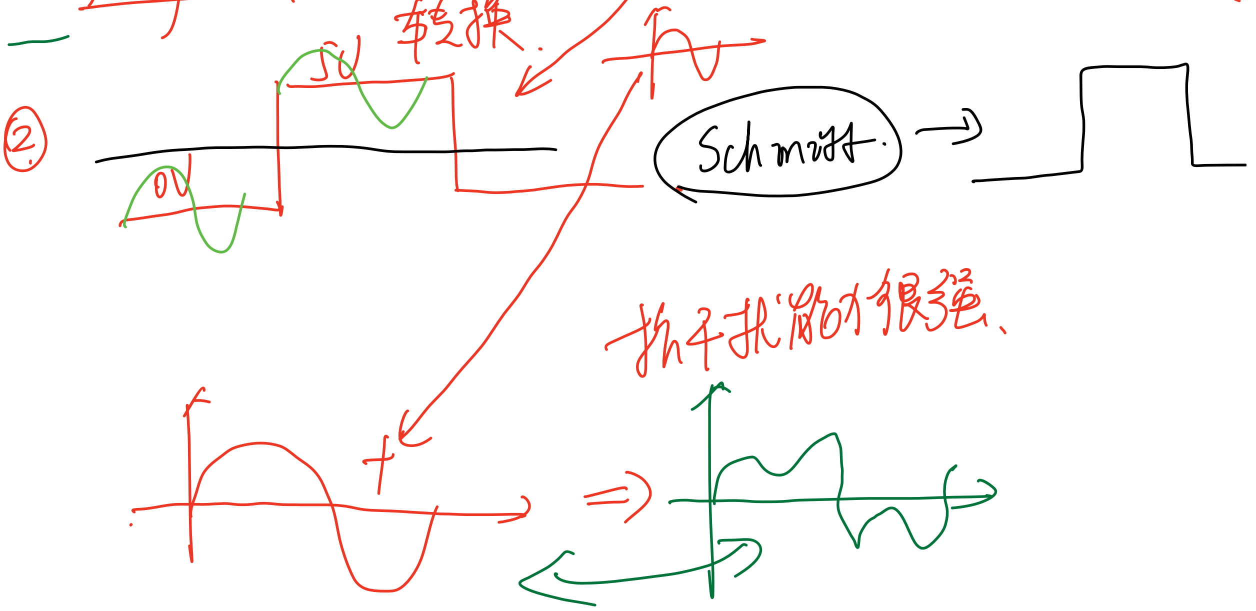
二 [illegible]

8进制、16进制

① 数制 2, 10, 8, 16 , 浮点数. , 计算器(存储)

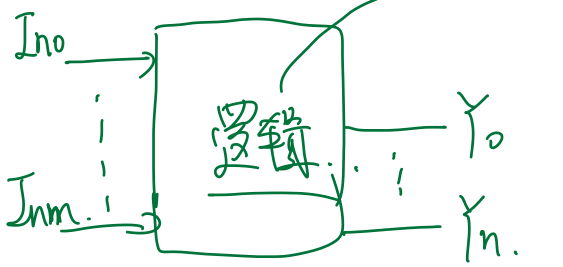
转换.

②



CPU. frequency.

③



逻辑表达.

Chap4: Boolean Algreba

④ AND OR. NOT, XOR. NAND

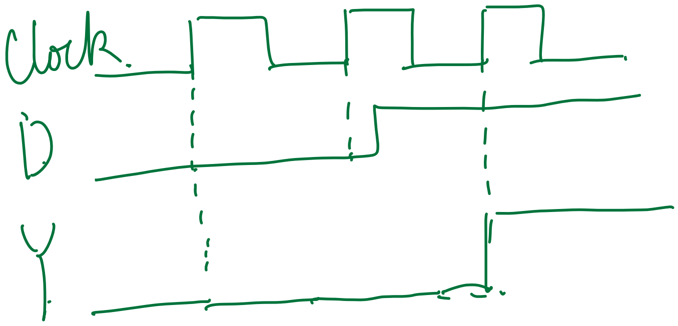
NOR. . . . . .

三人表决器 . 组合逻辑.

组合逻辑电路:
译码器
编码器.

⑤ 时序逻辑: 触发器. trigger.

时序:



三大方程 {
输入.
状态.
输出.

⑥ 时序电路: 移位寄存器.

计数器 ☆.

⑦ A/D D/A. 了解

学习仿真软件: LTspice. EDA 免费, 小、跨平台.

→ 设计电路

Chap1 Introductory Concepts

Digital and Analog Quantities

After completing this section, you should be able to

- Define analog
- Define digital
- Explain the difference between digital and analog quantities
- State the advantages of digital over analog
- Give examples of how digital and analog quantities are used in electronics

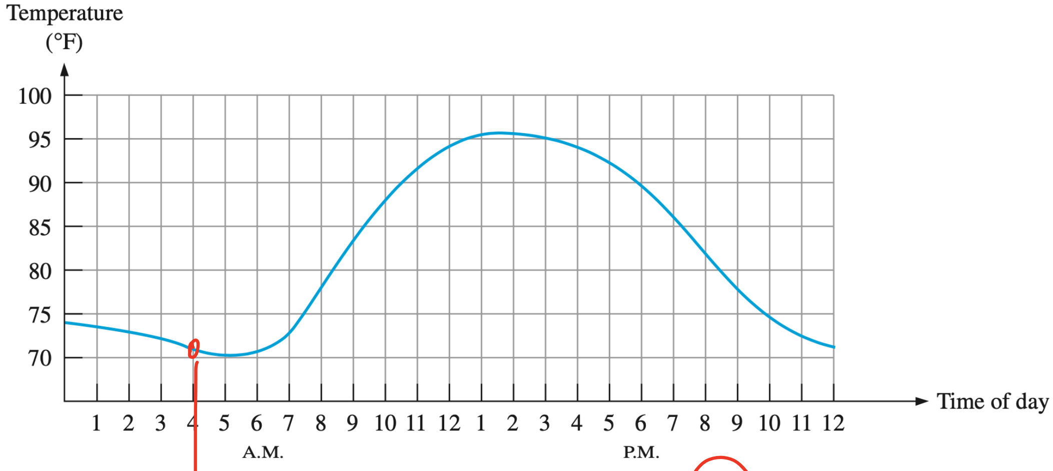


FIGURE 1–1 Graph of an analog quantity (temperature versus time).

1 0

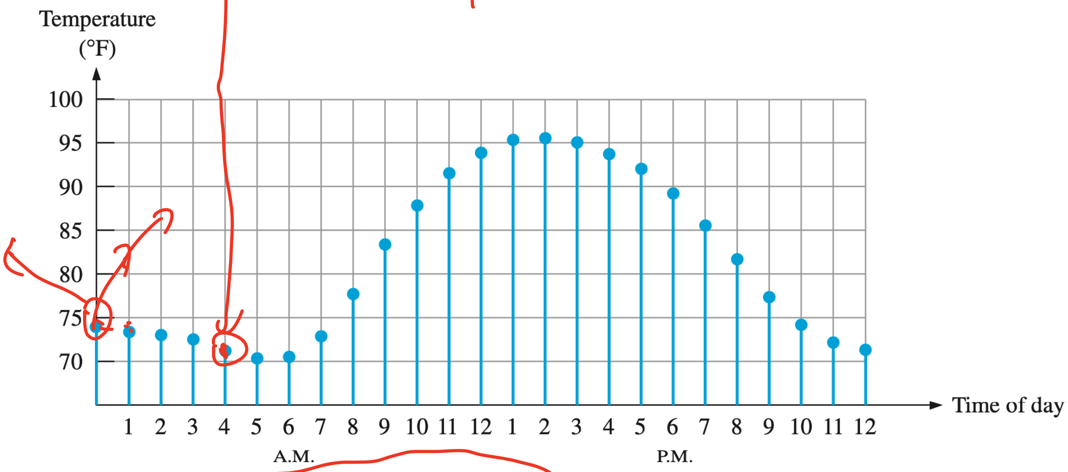


FIGURE 1–2 Sampled-value representation (quantization) of the analog quantity in Figure 1–1. Each value represented by a dot can be digitized by representing it as a digital code that consists of a series of 1s and 0s.

The Digital Advantage

Digital representation has certain advantages over analog representation in electronics applications. For one thing, digital data can be processed and transmitted more efficiently and reli- ably than analog data. Also, digital data has a great advantage when storage is necessary. For example, music when converted to digital form can be stored more compactly and reproduced with greater accuracy and clarity than is possible when it is in analog form. Noise (unwanted voltage fluctuations) does not affect digital data nearly as much as it does analog signals.

在电子应用中，数字表示比模拟表示具有某些优势。一方面，数字数据可以比模拟数据更有效、更可靠地处理和传输。此外，当需要存储时，数字数据具有很大的优势。例如，转换为数字形式的音乐比模拟形式的音乐可以更紧凑地存储，并以更高的准确性和清晰度进行再现。噪声（不需要的电压波动）对数字数据的影响几乎不如对模拟信号的影响。

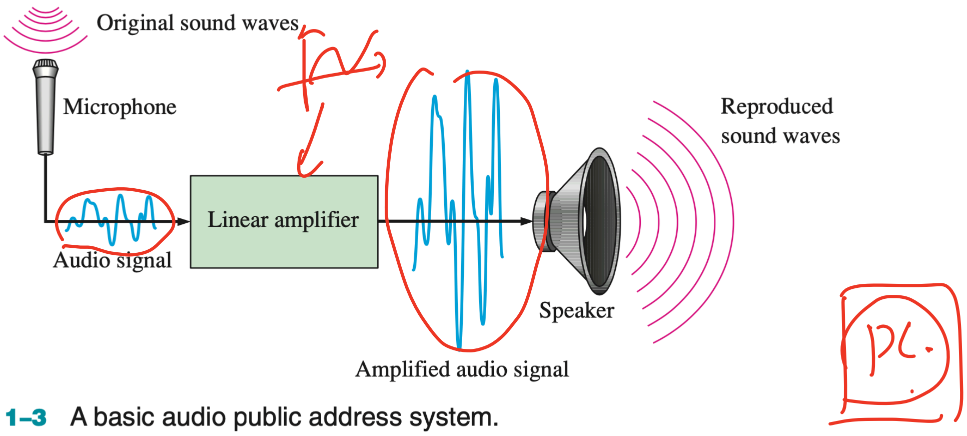


FIGURE 1–3 A basic audio public address system.

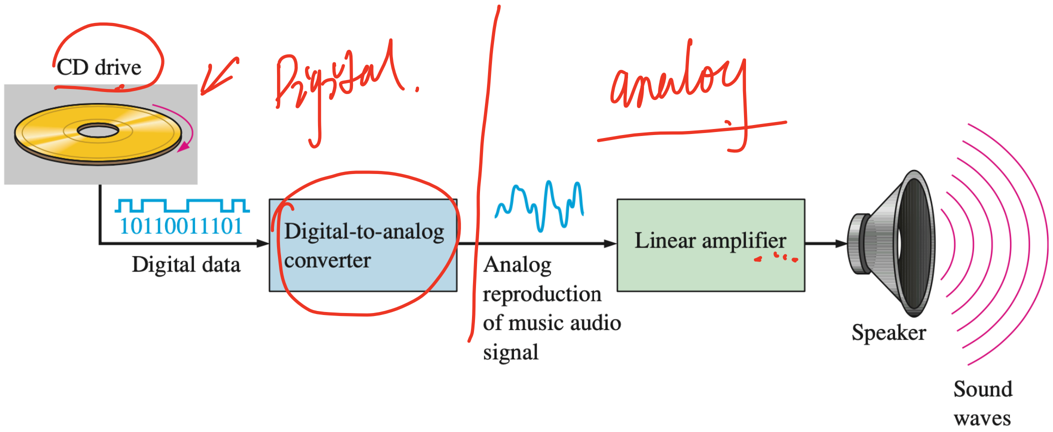


FIGURE 1–4 Basic block diagram of a CD player. Only one channel is shown.

Binary Digits, Logic Levels, and Digital Waveforms

After completing this section, you should be able to

- Define *binary*
- Define *bit*
- Name the bits in a binary system
- Explain how voltage levels are used to represent bits
- Explain how voltage levels are interpreted by a digital circuit
- Describe the general characteristics of a pulse
- Determine the amplitude, rise time, fall time, and width of a pulse
- Identify and describe the characteristics of a digital waveform
- Determine the amplitude, period, frequency, and duty cycle of a digital waveform
- Explain what a timing diagram is and state its purpose
- Explain serial and parallel data transfer and state the advantage and disadvantage of each

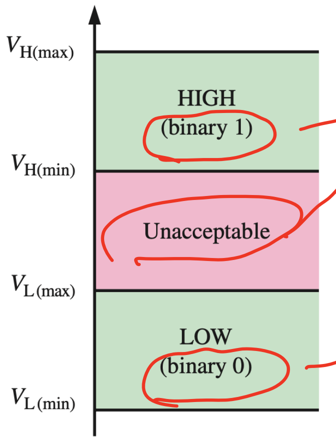


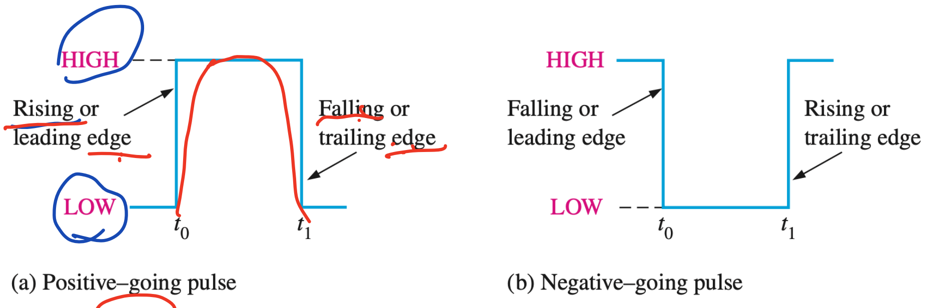
FIGURE 1–6 Logic level ranges of voltage for a digital circuit.

二进制

0 F L
1 T H

high level

Low level.



(a) Positive–going pulse

(b) Negative–going pulse

FIGURE 1–7 Ideal pulses.

电压波形

tr → rise
↓
time

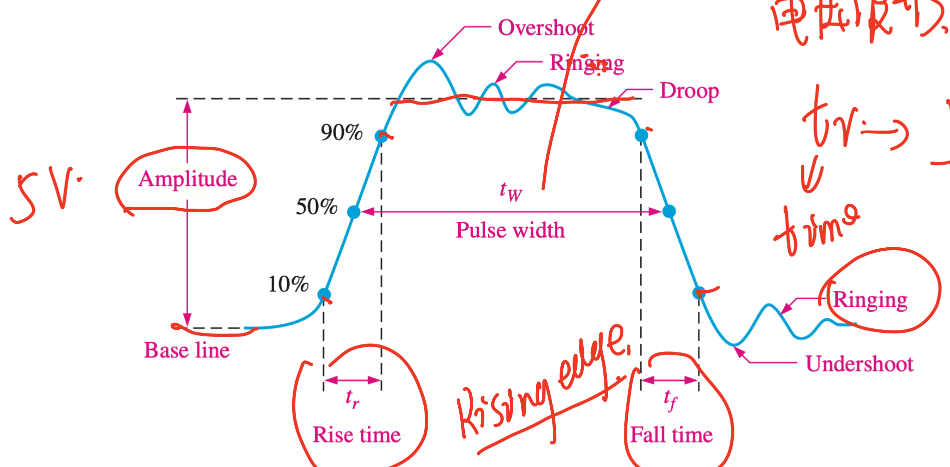
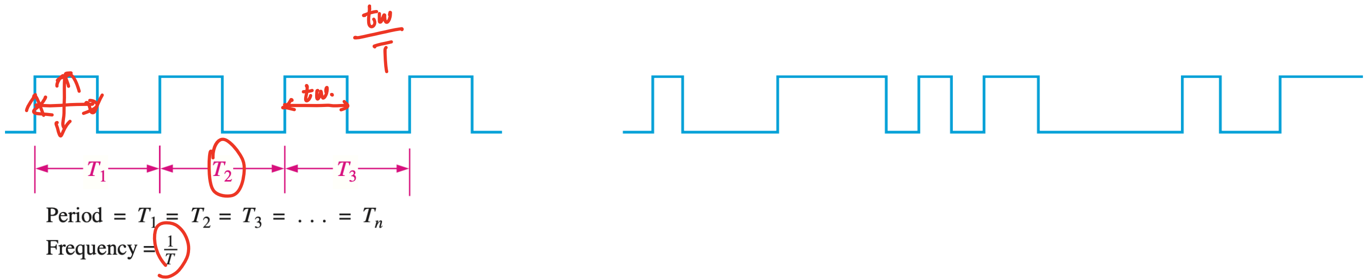


FIGURE 1–8 Nonideal pulse characteristics.



Period = $T_1 = T_2 = T_3 = \ldots = T_n$

Frequency = $\frac{1}{T}$

(a) Periodic (square wave)

(b) Nonperiodic

FIGURE 1–9 Examples of digital waveforms.

The frequency (f) of a pulse (digital) waveform is the reciprocal of the period. The relationship between frequency and period is expressed as follows:

KHz: 10^3 Hz
MHz: 10^6 Hz

ms = 10^{-3} s
μs = 10^{-6} s
ns = 10^{-9} s
ps = 10^{-12} s

$$f = \frac{1}{T} \rightarrow s$$ Equation 1–1

Hz.

$$T = \frac{1}{f}$$ Equation 1–2

An important characteristic of a periodic digital waveform is its **duty cycle**, which is the ratio of the pulse width (t_W) to the period (T). It can be expressed as a percentage.

$$\textbf{Duty cycle} = \left(\frac{t_W}{T}\right)100\%$$ Equation 1–3

EXAMPLE 1–1

A portion of a periodic digital waveform is shown in Figure 1–10. The measurements are in milliseconds. Determine the following:

(a) period **(b)** frequency **(c)** duty cycle $= \frac{1}{10} \times 100\% = 10\%$

T = 10ms.

$\frac{1}{T} = \frac{1}{10\text{ms}} = \frac{1}{10 \times 10^{-3}} = \frac{1000}{10} = 100$ Hz.

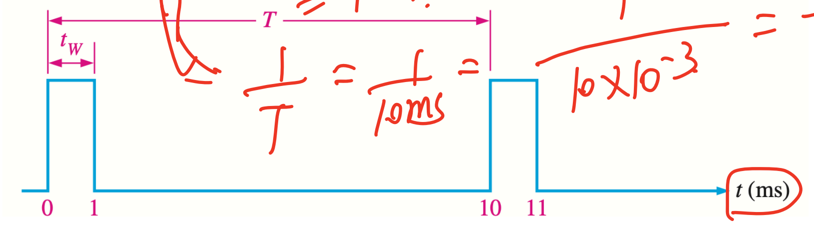
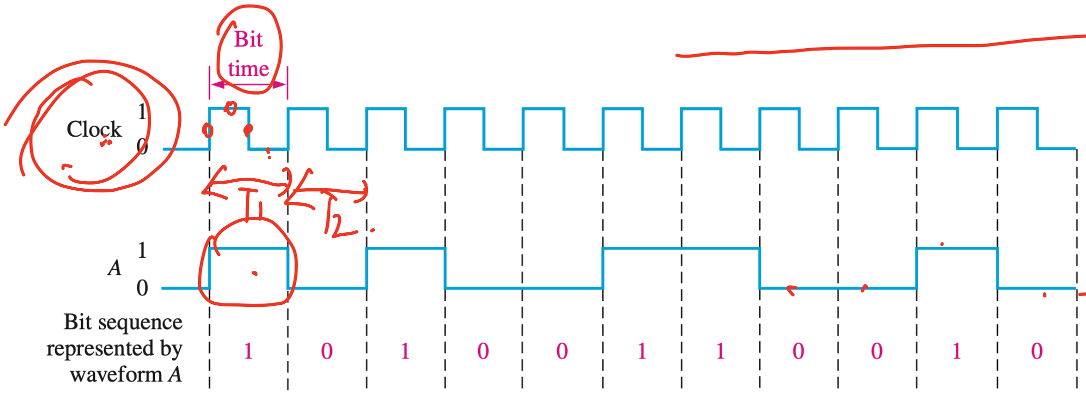


FIGURE 1–10

USB 网线



Frequency

通讯协议

FIGURE 1–11 Example of a clock waveform synchronized with a waveform representation of a sequence of bits.

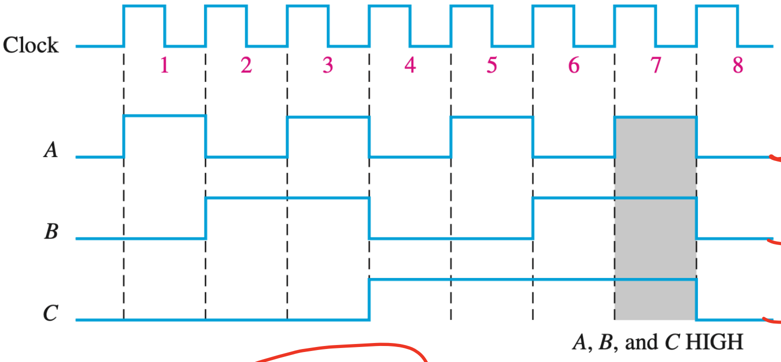
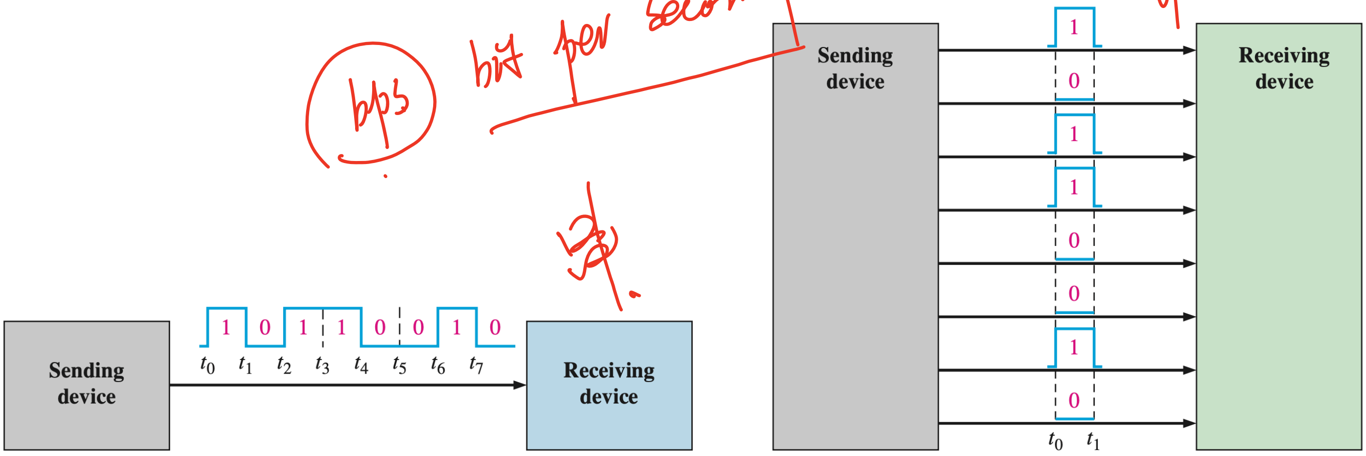


FIGURE 1–12 Example of a timing diagram.

bps bit per second

串

并



(a) Serial transfer of 8 bits of binary data. Interval t_0 to t_1 is first.

(b) Parallel transfer of 8 bits of binary data. The beginning time is t_0.

FIGURE 1–13 Illustration of serial and parallel transfer of binary data. Only the data lines are shown.

EXAMPLE 1–2

(a) Determine the total time required to serially transfer the eight bits contained in waveform *A* of Figure 1–14, and indicate the sequence of bits. The left-most bit is the first to be transferred. The 1 MHz clock is used as reference.

(b) What is the total time to transfer the same eight bits in parallel?

f = 1MHz.

$T = \frac{1}{f} = \frac{1}{1\times10^{6}}$

$= 1\times10^{-6}$ s

= 1 μs

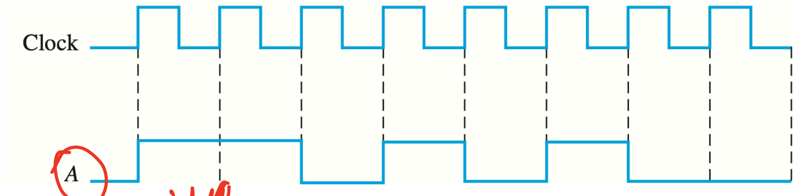


FIGURE 1–14

Basic Logic Functions

After completing this section, you should be able to

- List three basic logic functions
- Define the NOT function
- Define the AND function
- Define the OR function

NOT

The **NOT** function changes one logic level to the opposite logic level, as indicated in Figure 1–17. When the input is HIGH (1), the output is LOW (0). When the input is LOW, the output is HIGH. In either case, the output is *not* the same as the input. The NOT function is implemented by a logic circuit known as an **inverter**.

Input	Output
1	0
0.	1.

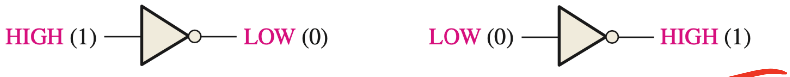


FIGURE 1–17 The NOT function.

AND

basic

The **AND** function produces a HIGH output only when all the inputs are HIGH, as indicated in Figure 1–18 for the case of two inputs. When one input is HIGH *and* the other input is HIGH, the output is HIGH. When any or all inputs are LOW, the output is LOW. The AND function is implemented by a logic circuit known as an *AND gate*.

Inputs		Output
0	0	0
0	1	0
1	0	0
1	1	1

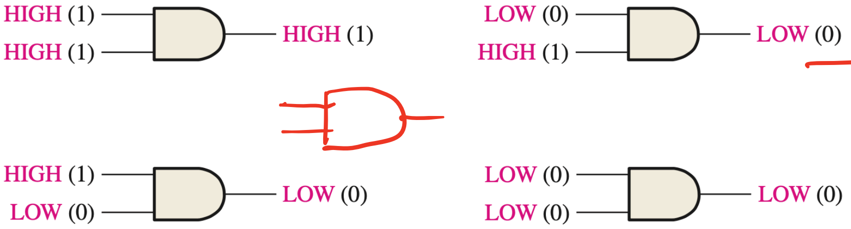
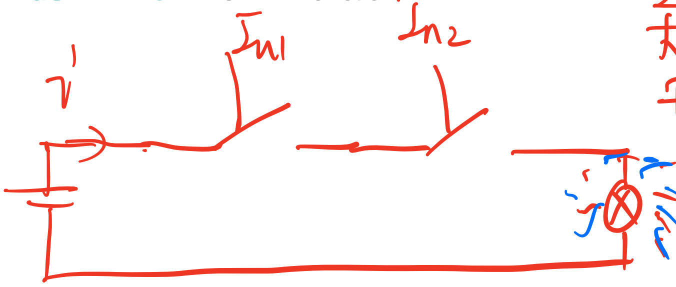


FIGURE 1–18 The AND function.



关: '1'
开: '0'

亮: '1'

'1'

求: 0

OR

The **OR** function produces a HIGH output when one or more inputs are HIGH, as indicated in Figure 1–19 for the case of two inputs. When one input is HIGH *or* the other input is HIGH *or* both inputs are HIGH, the output is HIGH. When both inputs are LOW, the output is LOW. The OR function is implemented by a logic circuit known as an *OR gate*.

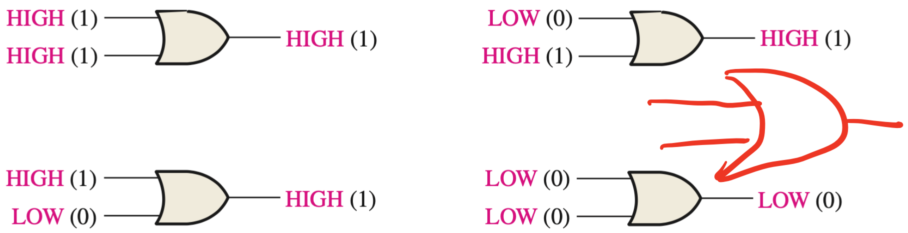


FIGURE 1–19 The OR function.

in1	in2	output
0	0	0
0	1	1
1	0	1
1	1	1

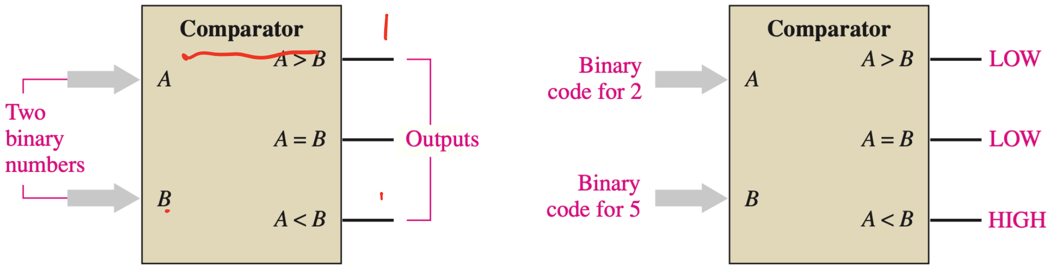
SECTION 1–3 CHECKUP

1. When does the NOT function produce a HIGH output?
2. When does the AND function produce a HIGH output?
3. When does the OR function produce a HIGH output?
4. What is an inverter?
5. What is a logic gate?

Combinational and Sequential Logic Functions

After completing this section, you should be able to

- List several types of logic functions
- Describe comparison and list the four arithmetic functions
- Describe code conversion, encoding, and decoding
- Describe multiplexing and demultiplexing
- Describe the counting function
- Describe the storage function
- Explain the operation of the tablet-bottling system



(a) Basic magnitude comparator

(b) Example: *A* is less than *B* (2 < 5) as indicated by the HIGH output (*A* < *B*)

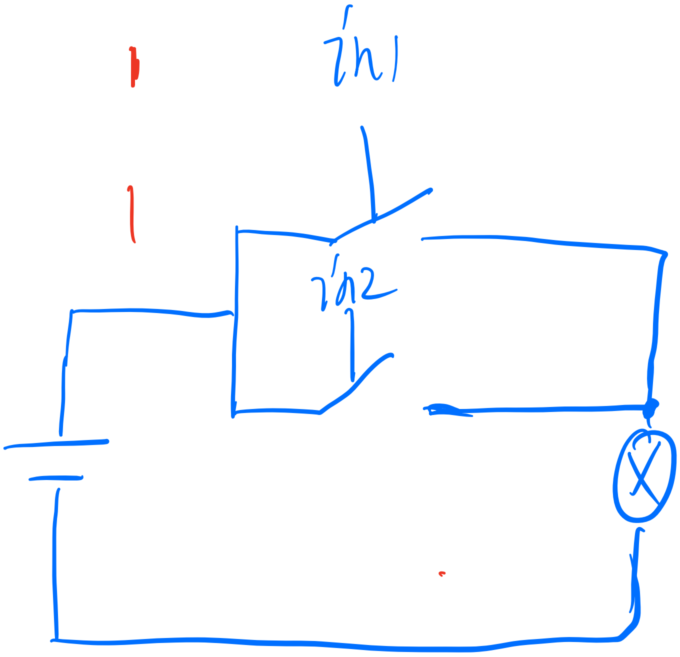
FIGURE 1–20 The comparison function.

in1	in2	output
0	0	0
0	1	1
1	0	1
1	1	1

逻辑抽象

开: '0'　亮: '1'
关: '1'　灭: '0'.

卧室

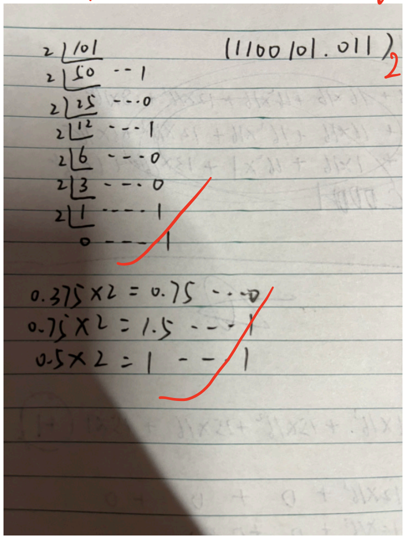


$(101.375)_{10} = (1100101.011)_2$

$= (145.3)_8$

$= (65.6)_{16}$

0 1100101.0110



2|101 (1100101.011)$_2$

2|50 ---1

2|25 ---0

2|12 ---1

2|6 ---0

2|3 ---0

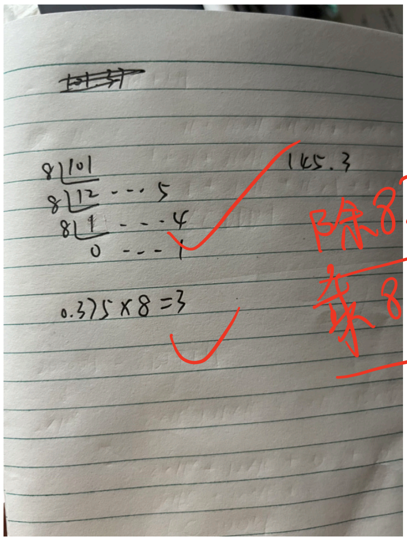
2|1 ---1

0 ---1

0.375×2 = 0.75 ---0

0.75×2 = 1.5 ---1

0.5×2 = 1 ---1



~~101.375~~

8|101 145.3

8|12 ---5

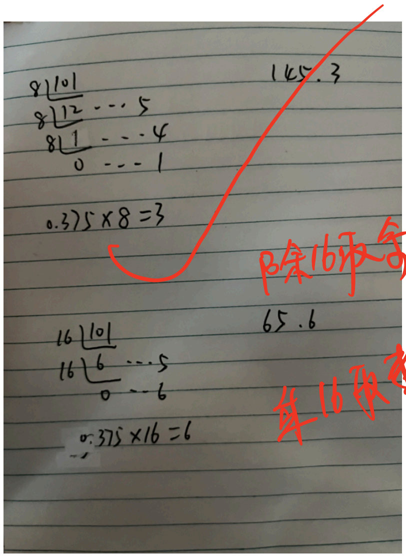
8|1 ---4

0 ---1

0.375×8 = 3

除8取余

乘8取整.



8|101 145.3

8|12 ---5

8|1 ---4

0 ---1

0.375×8 = 3

除16取余

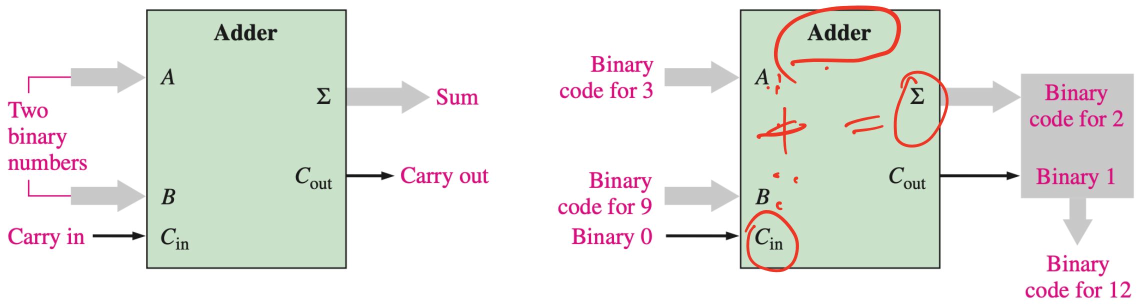
16|101 65.6

16|6 ---5

0 ---6

0.375×16 = 6

乘16取整.



(a) Basic adder

(b) Example: *A* plus *B* (3 + 9 = 12)

FIGURE 1–21 The addition function.

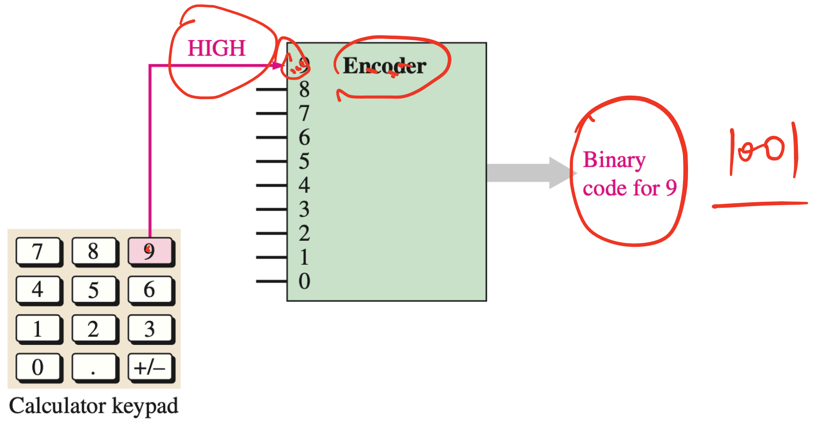


FIGURE 1–22 An encoder used to encode a calculator keystroke into a binary code for storage or for calculation.

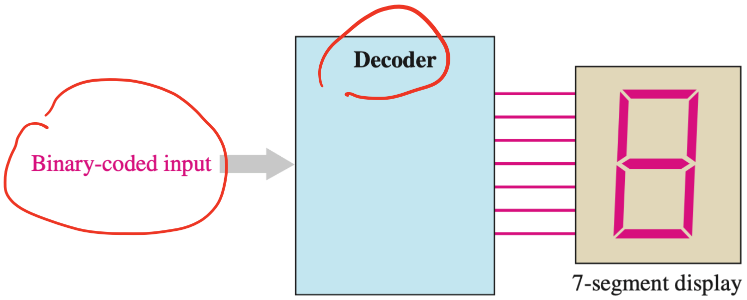


FIGURE 1–23 A decoder used to convert a special binary code into a 7-segment decimal readout.

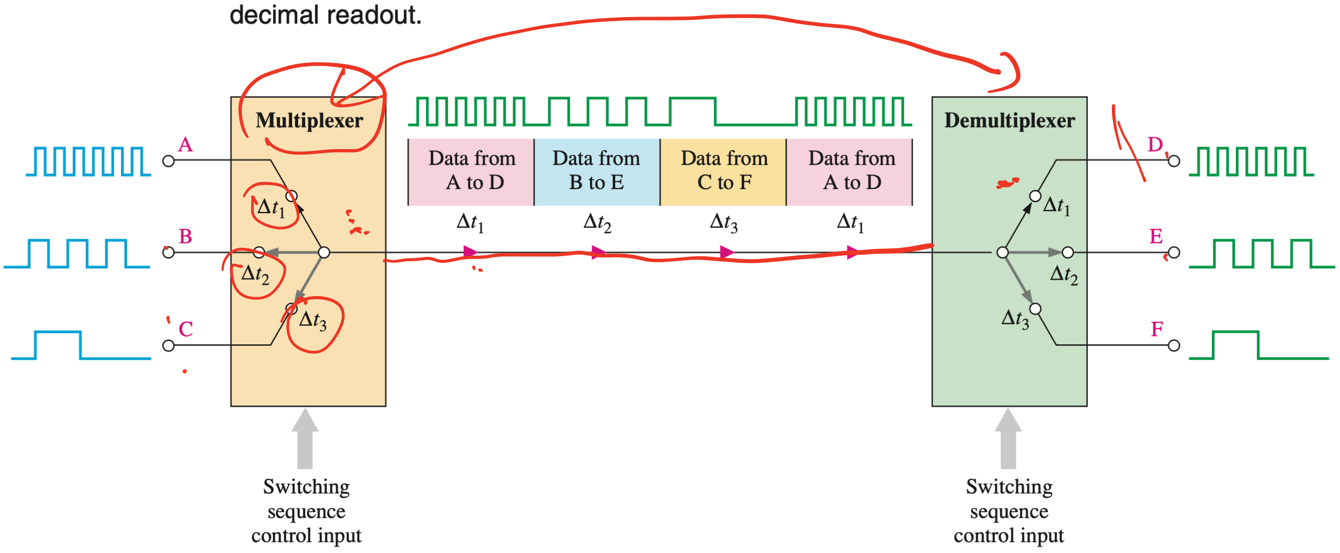
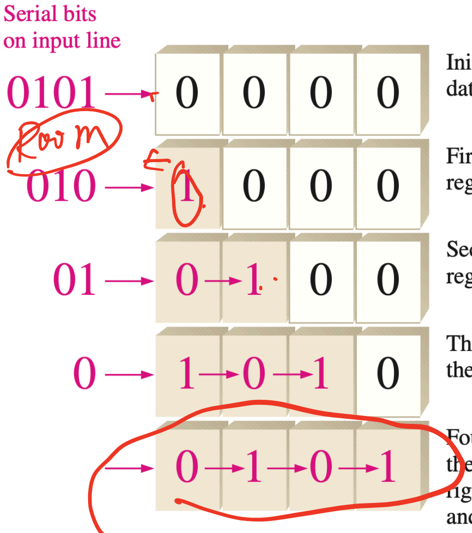


FIGURE 1–24 Illustration of a basic multiplexing/demultiplexing application.



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

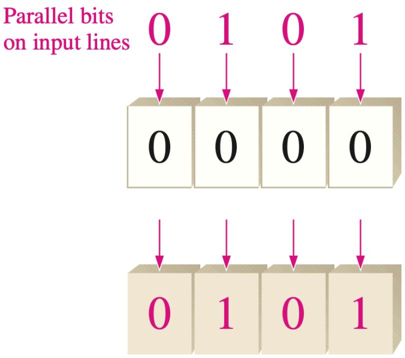
First bit (1) is shifted serially into the register.

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.

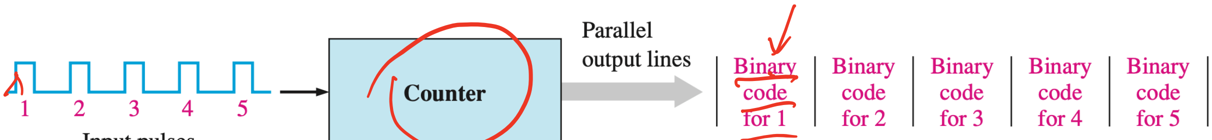
FIGURE 1–25 Example of the operation of a 4-bit serial shift register. Each block represents one storage "cell" or flip-flop.



Initially, the register is empty, containing only nondata zeros.

All bits are shifted in and stored simultaneously.

FIGURE 1–26 Example of the operation of a 4-bit parallel shift register.



Sequence of binary codes that represent the number of input pulses counted.

FIGURE 1–27 Illustration of basic counter operation.

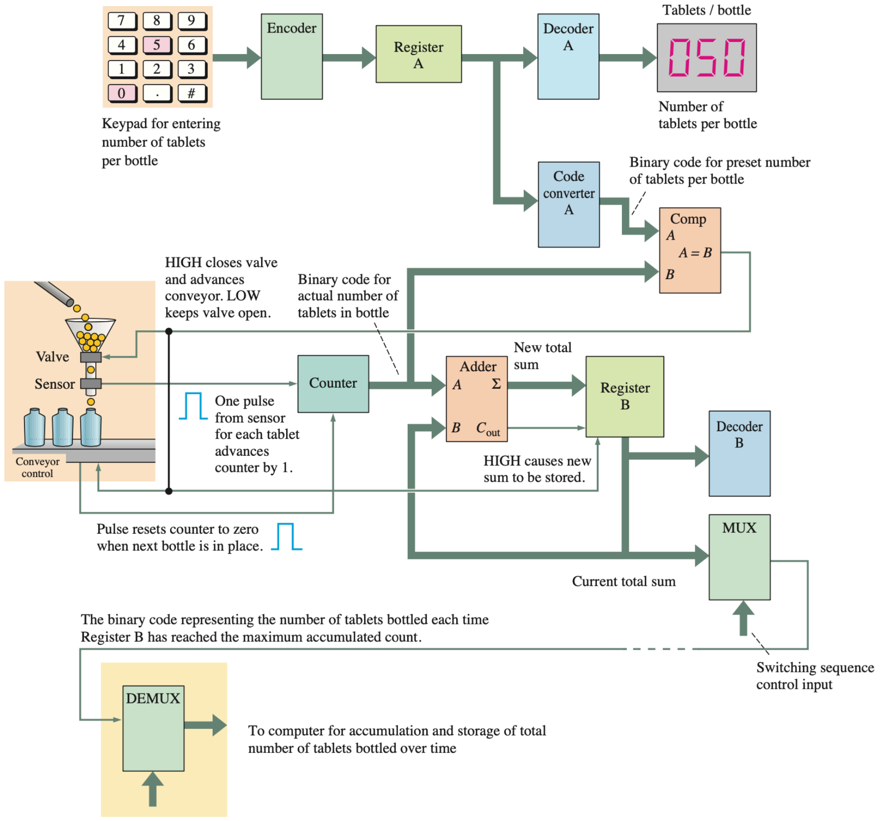

7
8
9
4
5
6
1
2
3
0
.
#
Keypad for entering number of tablets per bottle
Encoder
Register A
Decoder A
Tablets / bottle
050
Number of tablets per bottle
Code converter A
Binary code for preset number of tablets per bottle
Comp
A
A = B
B
HIGH closes valve and advances conveyor. LOW keeps valve open.
Binary code for actual number of tablets in bottle
Valve
Sensor
Counter
One pulse from sensor for each tablet advances counter by 1.
Adder
A
Σ
B
C_{out}
New total sum
Register B
HIGH causes new sum to be stored.
Decoder B
Conveyor control
Pulse resets counter to zero when next bottle is in place.
MUX
Current total sum
The binary code representing the number of tablets bottled each time Register B has reached the maximum accumulated count.
Switching sequence control input
DEMUX
To computer for accumulation and storage of total number of tablets bottled over time

FIGURE 1–28 Block diagram of a tablet-bottling system.

Introduction to Programmable Logic

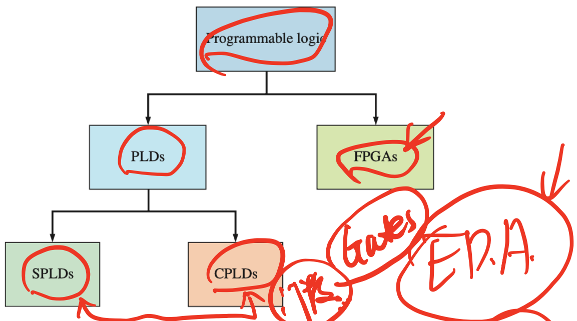
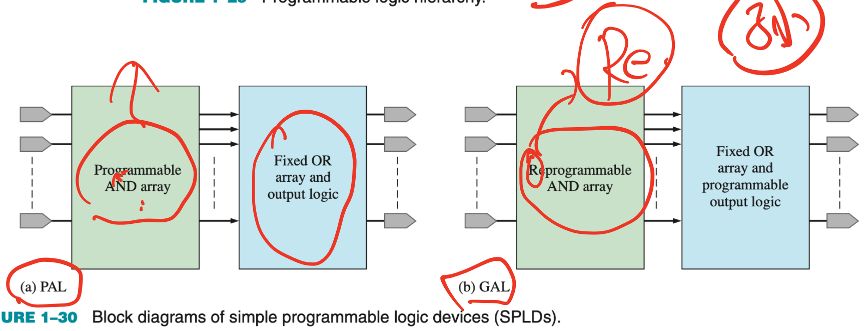


FIGURE 1–29 Programmable logic hierarchy.


Re
Programmable AND array
Fixed OR array and output logic
Reprogrammable AND array
Fixed OR array and programmable output logic

(a) PAL

(b) GAL

FIGURE 1–30 Block diagrams of simple programmable logic devices (SPLDs).

GAL

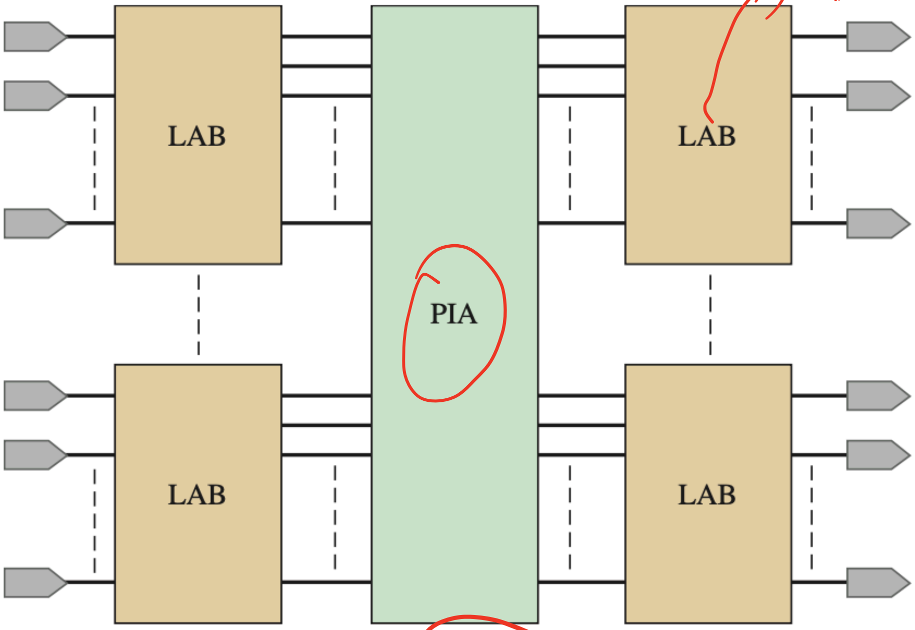


FIGURE 1–32 General block diagram of a CPLD.

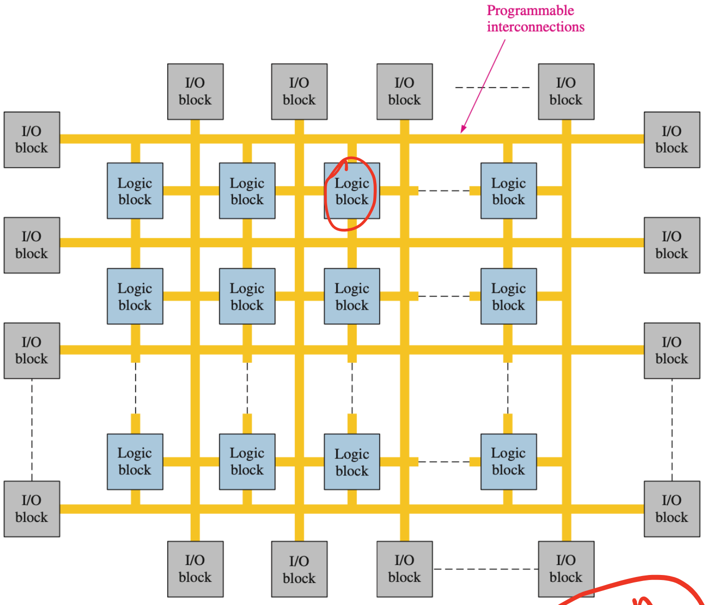


FIGURE 1–34 Basic structure of an FPGA.

FPGA

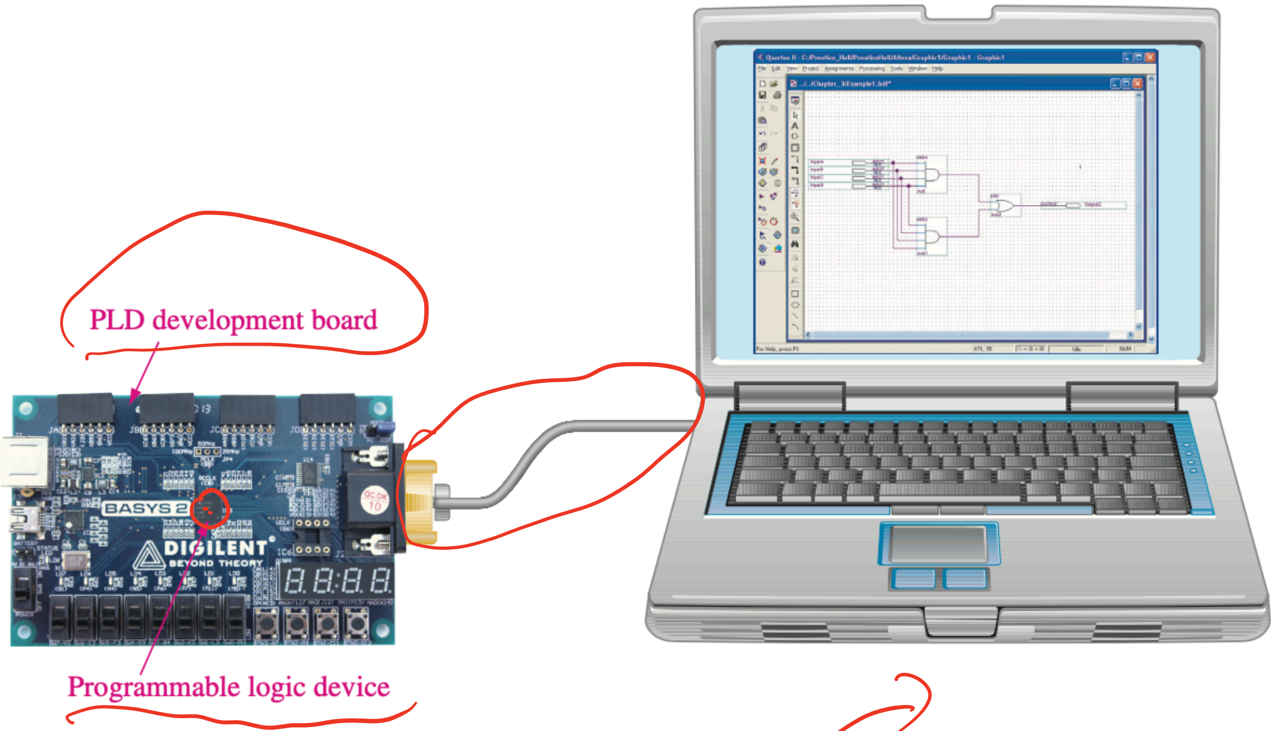


FIGURE 1–36 Basic setup for programming a PLD or FPGA. Graphic entry of a logic circuit is shown for illustration. Text entry such as VHDL can also be used. (Photo courtesy of Digilent, Inc.)

Verilog 类C

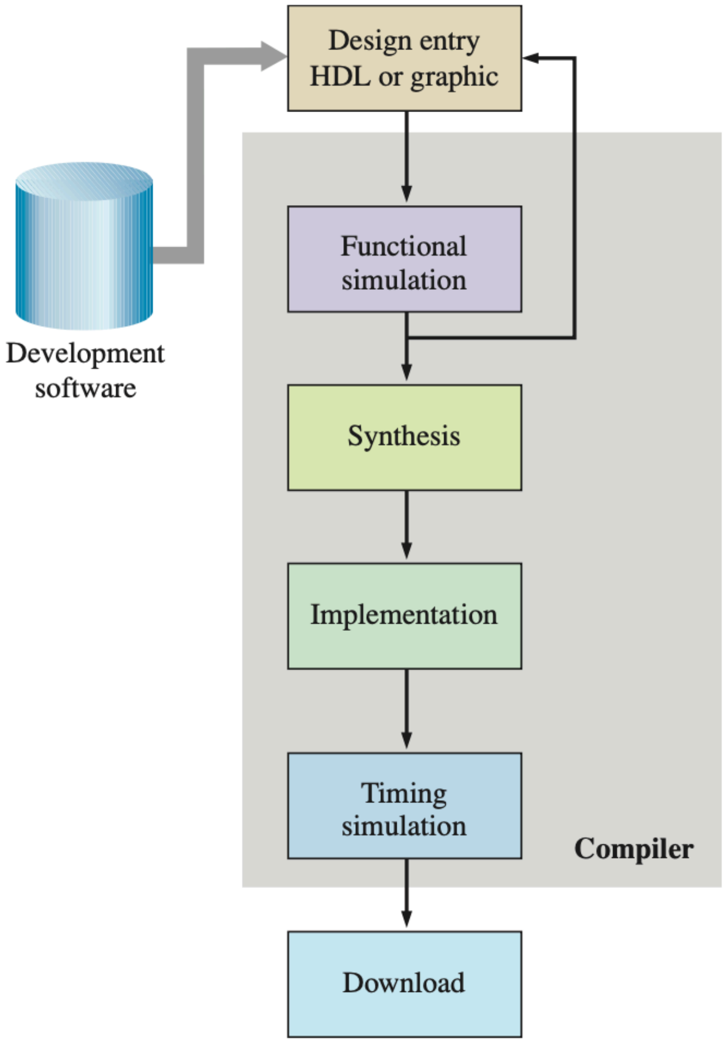


FIGURE 1–37 Basic programmable logic design flow block diagram.

Fixed-Function Logic Devices

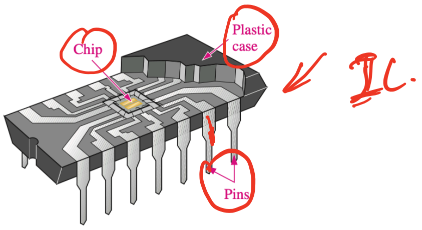
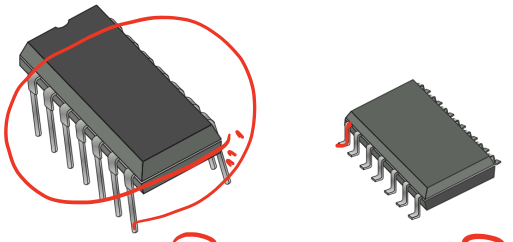


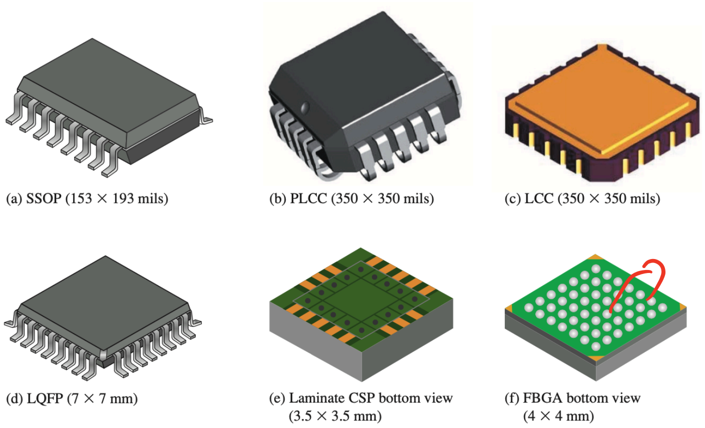
FIGURE 1–38 Cutaway view of one type of fixed-function IC package (dual in-line package) showing the chip mounted inside, with connections to input and output pins.



(a) Dual in-line package (DIP)

(b) Small-outline IC (SOIC)

FIGURE 1–39 Examples of through-hole and surface-mounted devices. The DIP is larger than the SOIC with the same number of leads. This particular DIP is approximately 0.785 in. long, and the SOIC is approximately 0.385 in. long.



(a) SSOP (153 × 193 mils)

(b) PLCC (350 × 350 mils)

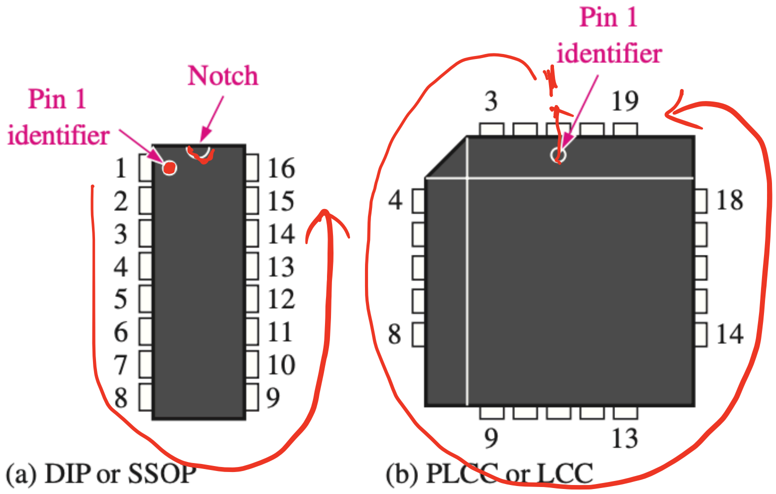
(c) LCC (350 × 350 mils)

(d) LQFP (7 × 7 mm)

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

(f) FBGA bottom view (4 × 4 mm)

FIGURE 1–40 Examples of SMT package configurations. Parts (e) and (f) show bottom views.



(a) DIP or SSOP

(b) PLCC or LCC

FIGURE 1–41 Pin numbering for two examples of standard types of IC packages. Top views are shown.

Test and Measurement Instruments

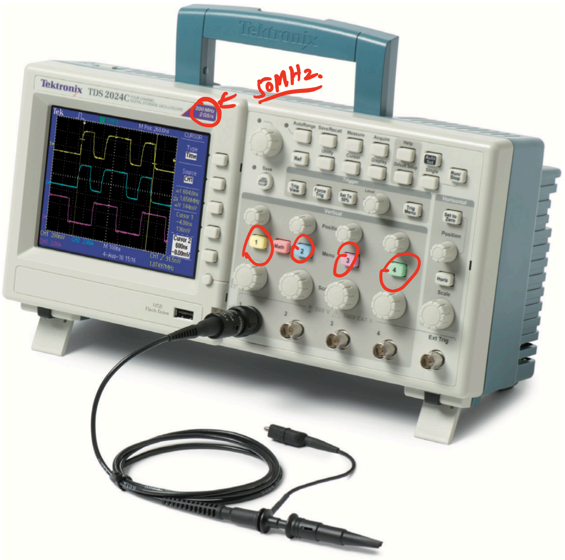


FIGURE 1–42 Typical digital oscilloscope with voltage probe. Used with permission from Tektronix, Inc.

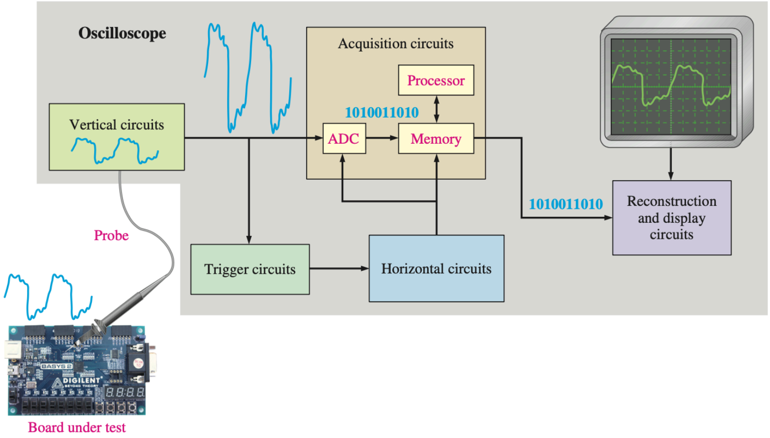


FIGURE 1–43 Block diagram of a digital oscilloscope. (Photo courtesy of Digilent, Inc.)

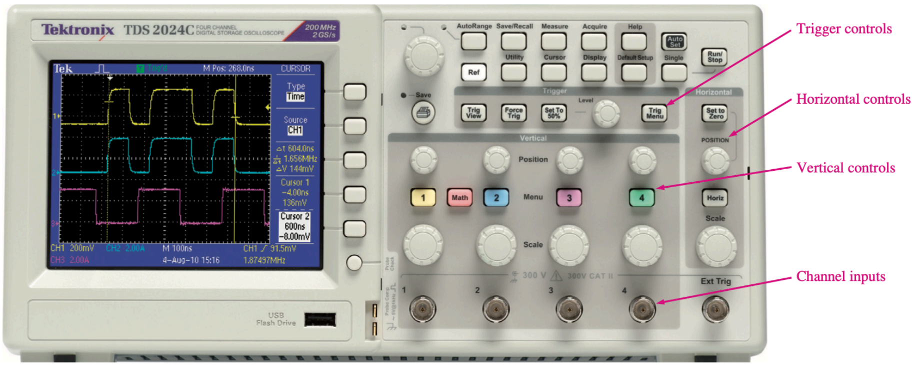


FIGURE 1–44 A typical digital oscilloscope front panel. Numbers below screen indicate the values for each division on the vertical (voltage) and horizontal (time) scales and can be varied using the vertical and horizontal controls on the scope. Used with permission from Tektronix, Inc.

EXAMPLE 1–3

Based on the readouts, determine the amplitude and the period of the pulse waveform on the screen of a digital oscilloscope as shown in Figure 1–48. Also, calculate the frequency.

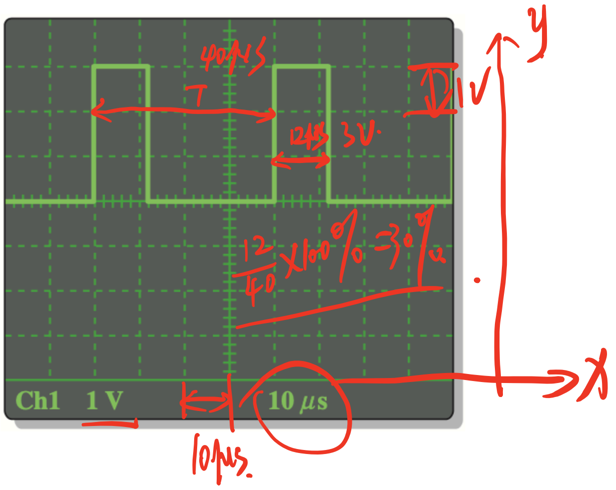
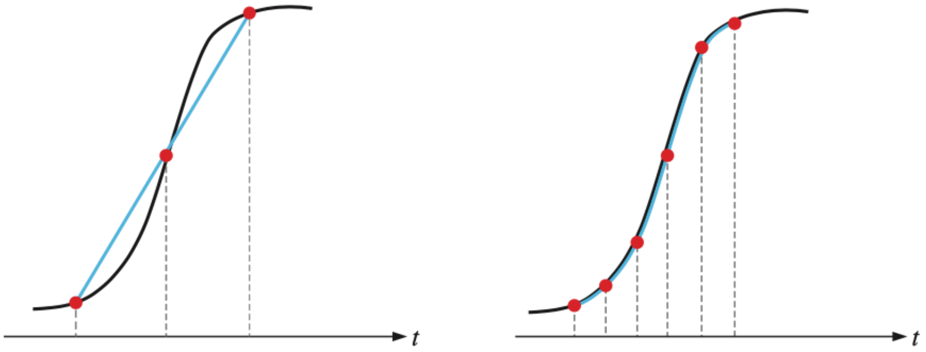


FIGURE 1–48

Sampling Rate

The **sampling rate** is the rate at which the analog-to-digital converter (ADC) in the oscilloscope is clocked to digitize the incoming signal. The sampling rate and bandwidth are not directly related, but the sampling rate should be at least five times the bandwidth. Figure 1–49 illustrates the difference between a low sampling rate and a much higher sampling rate. Part (a) shows how a sampling rate that is too low distorts the shape of the rising edge. In part (b), the higher sampling rate results in a much more accurate representation of the rising edge. When the sampling rate is sufficiently high, the signal can be precisely reproduced.



(a) Low sampling rate

(b) Higher sampling rate

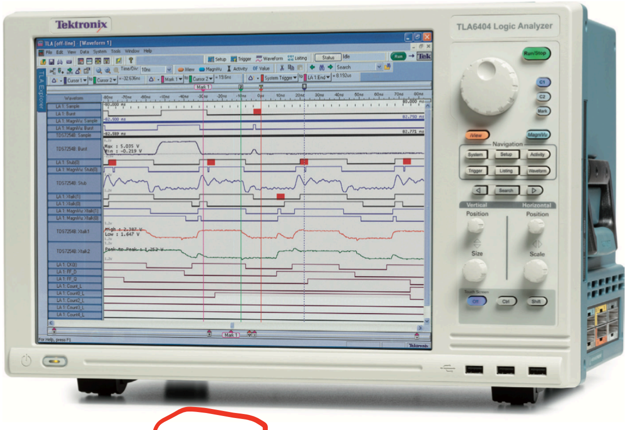
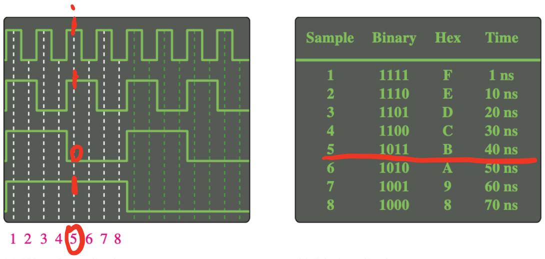


FIGURE 1–50 Typical logic analyzer. Used with permission from Tektronix, Inc.



Sample	Binary	Hex	Time
1	1111	F	1 ns
2	1110	E	10 ns
3	1101	D	20 ns
4	1100	C	30 ns
5	1011	B	40 ns
6	1010	A	50 ns
7	1001	9	60 ns
8	1000	8	70 ns

(a) Waveform display

(b) Listing display

FIGURE 1–52 Two logic analyzer display modes.

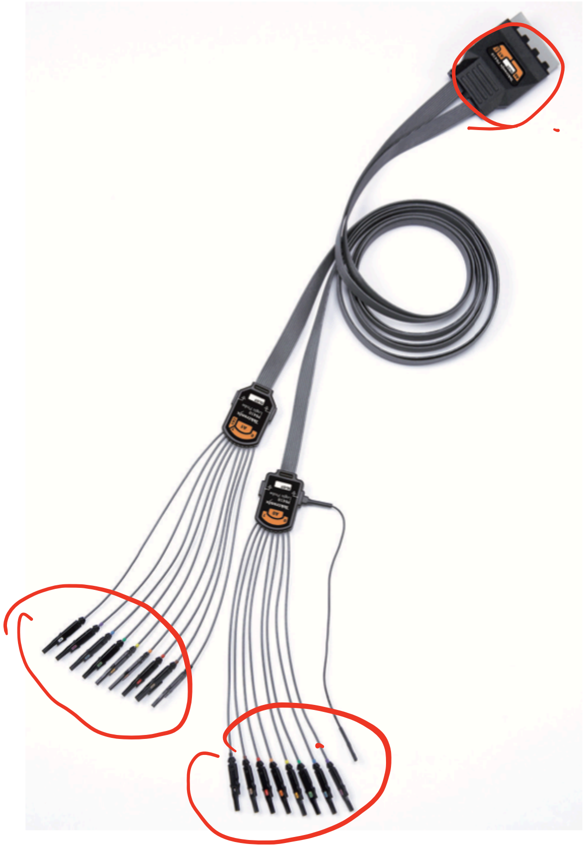
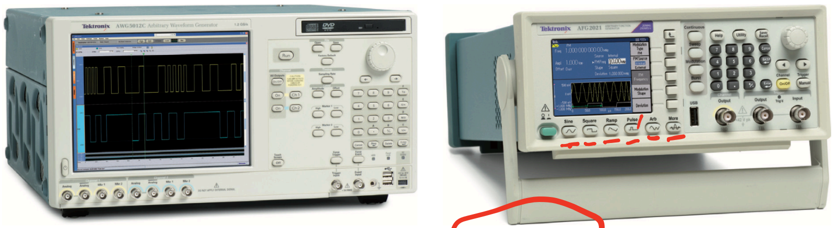


FIGURE 1–53 A typical multichannel logic analyzer probe. Used with permission from Tektronix, Inc.



(a) Arbitrary waveform generator

(b) Function generator

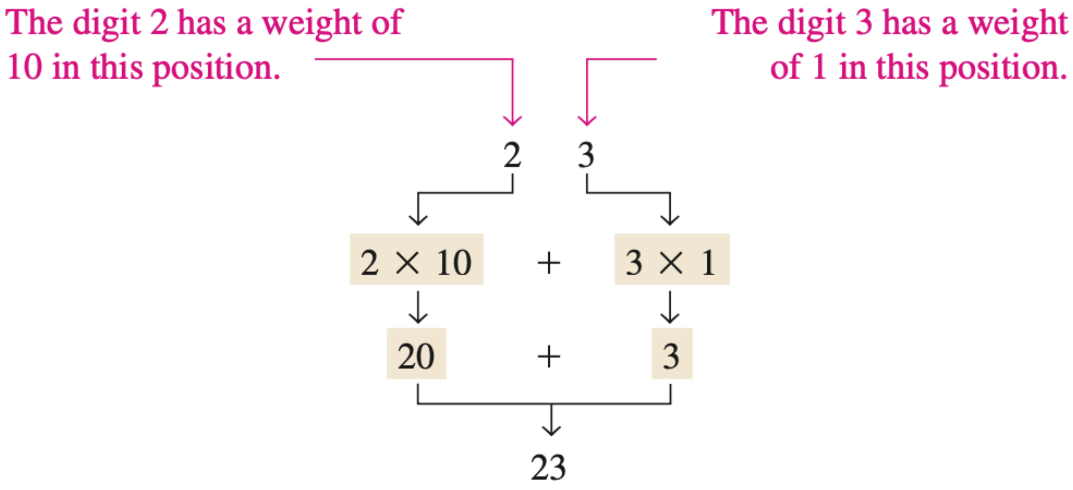
FIGURE 1–54 Typical signal generators. Used with permission from Tektronix, Inc.

FIGURE 1–55 Typical DMMs. Used with permission from (a) B+K Precision®; (b) Fluke

FIGURE 1–56 Typical bench-type dc power supply. Used with permission from Tektronix, Inc.

Chap2 Number Systems, Operations, and Codes

2–1 Decimal Numbers


The digit 2 has a weight of 10 in this position.
The digit 3 has a weight of 1 in this position.
2
3
2 × 10
+
3 × 1
20
+
3
23

digits: The decimal number system has ten digits.

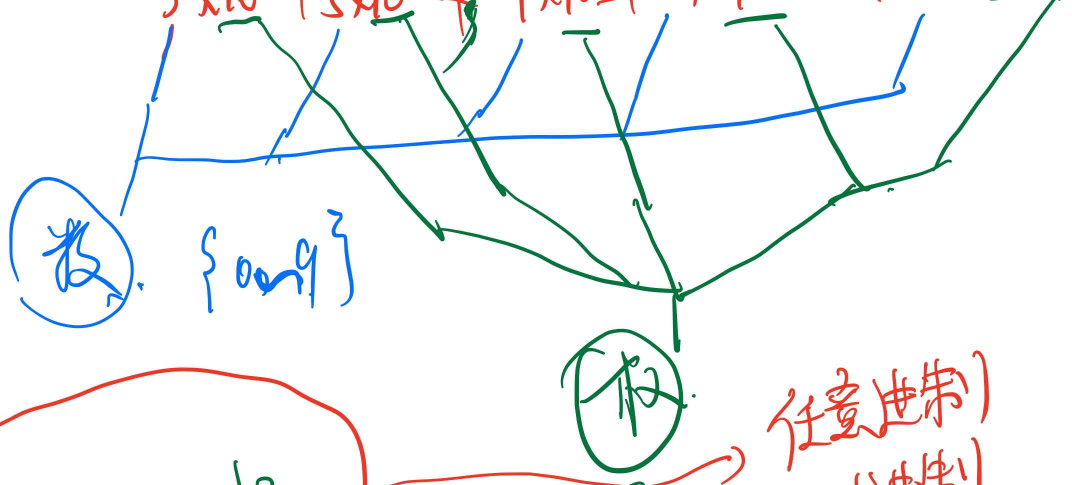
weight:The decimal number system has a base of 10.

2–2 Binary Numbers

十进制：

35.125

$3\times10^1+5\times10^0+1\times10^{-1}+2\times10^{-2}+5\times10^{-3}$



任数 = Σ 数 × 权

↓

进制

→ 进制

位置

→ 任意进制

→ 十进制

10 转化方法

二进制：　数：0, 1

$(101.11)_2$

$=1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

$=4+0+1+0.5+0.25$

$=(5.5)$.

十进制 → 二进制：

$(13.75)_{10} \longrightarrow (1101.11)_2$

(除2取余)

$$\begin{array}{r|l l}
2 & 13 & 1 \\
2 & 6 & 0 \\
2 & 3 & 1 \\
2 & 1 & 1 \\
 & 0 &
\end{array}$$

L ↑ H

(乘2取整)

$$\begin{array}{r l}
 & 0.75 \\
\times & 2 \\ \hline
 & 0.5 & 1 \\
\times & 2 \\ \hline
 & 0 & 1
\end{array}$$

$$(\overset{3}{1}\overset{2}{1}\overset{1}{0}\overset{0}{1})_2 \Rightarrow 1\times 2^3 + 1\times 2^2 + 0 + 1\times 2^0$$

$$= 8 + 4 + 0 + 1$$

$$= 13$$

$$(0.11)_2 \Rightarrow 1\times 2^{-1} + 1\times 2^{-2} = 0.5 + 0.25$$

$$= 0.75.$$

7 1

八进制：$(0 \sim 7)$

310.5

$= 3\times8^2 + 1\times8^1 + 0\times8^0 + 5\times8^{-1}$

$= 3\times64 + 8 + 0 + \frac{5}{8}$

16进制：$(0 \sim 9,\ A \sim F)$

$AE.34$

$= 10\times16^1 + 14\times16^0 + 3\times16^{-1} + 4\times16^{-2}$

$= 16\times10 + 14 + \frac{3}{16} + \frac{4}{16^2}$

任意进制 → 十进制 转化方法。

二进制 ⟷ 八进制：[0~7]
↓
111

十六

$(001101.110)_2 \longrightarrow (15.6)_8$

$(1101.1100)_2 \longrightarrow (D.C)_{16}$

8进制	⟷	二进制
0		000
1		001
2		010
3		011
4		100
5		101
6		110
7		111

16进制：二进制

0 ⟷ 0000

⋮

F ⟷ 1111

TABLE 2–2

Binary weights.

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

Binary-to-Decimal Conversion

The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0.

EXAMPLE 2–3

Convert the binary whole number 1101101 to decimal.

Solution

Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

$$\begin{array}{rccccccc} \text{Weight:} & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ \text{Binary number:} & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}$$

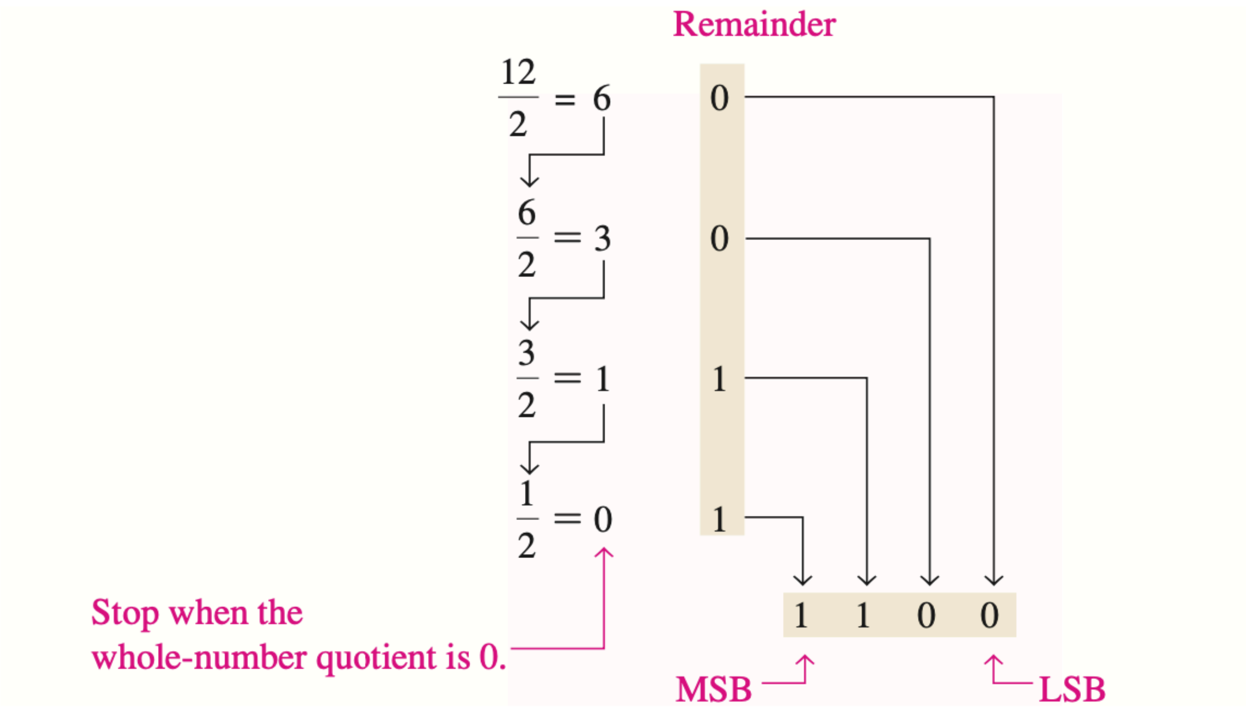
$$\begin{aligned} 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

Related Problem

Convert the binary number 10010001 to decimal.

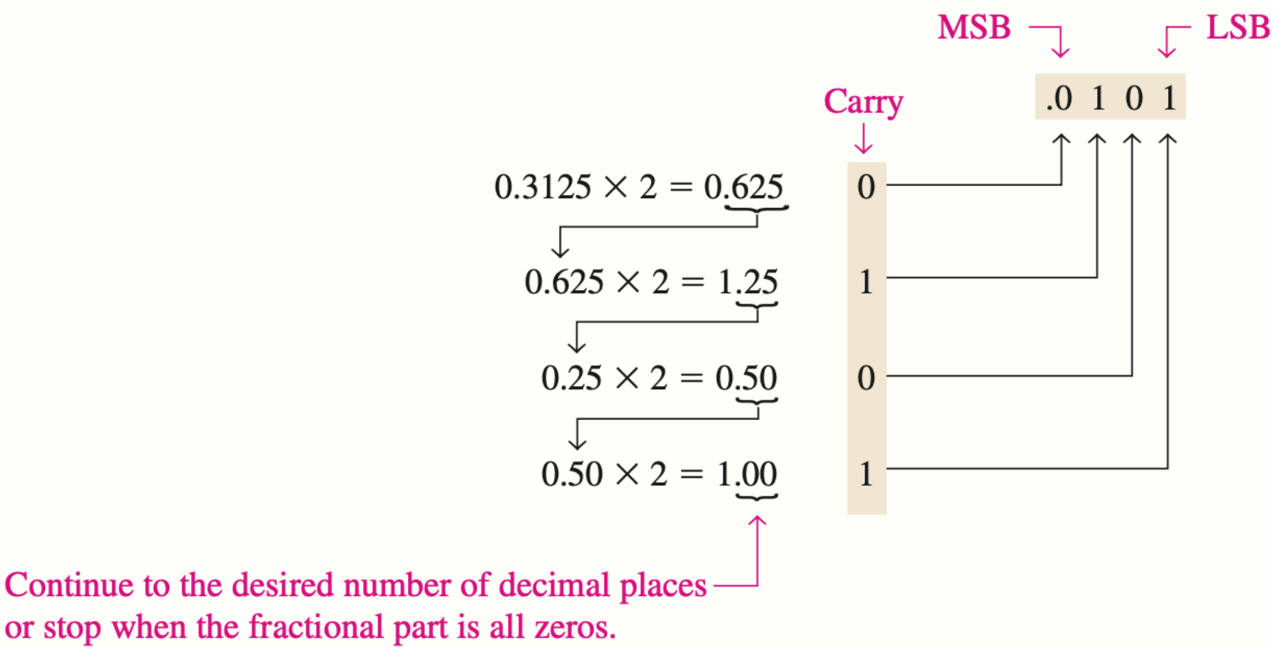
2–3 Decimal-to-Binary Conversion

Repeated Division-by-2 Method



For example:

Repeated Multiplication by 2


MSB
LSB
.0 1 0 1
Carry
$0.3125 \times 2 = 0.\underbrace{625}$
$0.625 \times 2 = 1.\underbrace{25}$
$0.25 \times 2 = 0.\underbrace{50}$
$0.50 \times 2 = 1.\underbrace{00}$
0
1
0
1
Continue to the desired number of decimal places or stop when the fractional part is all zeros.

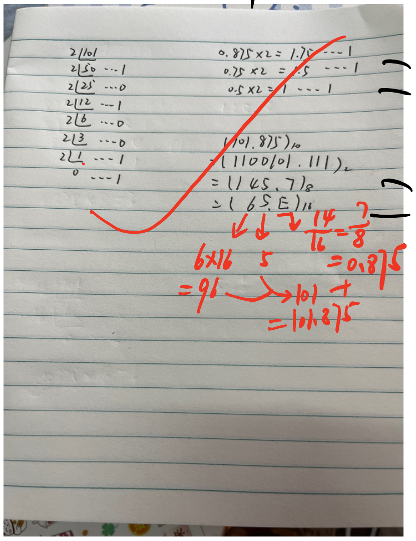
For example:

Test:

$(101.875)_{10} = (\qquad\qquad)_2$

$= (\qquad\qquad)_8$

$= (\qquad\qquad)_{16}$



2|101
2|50 ···1
2|25 ···0
2|12 ···1
2|6 ···0
2|3 ···0
2|1 ···1
0 ···1

$0.875\times2 = 1.75$ ···1
$0.75\times2 = 1.5$ ···1
$0.5\times2 = 1$ ···1

$(101.875)_{10}$
$= (1100101.111)_2$
$= (145.7)_8$
$= (65.E)_{16}$

$6\times16 = 96$
5
$\frac{14}{16} = \frac{7}{8} = 0.875$
$\rightarrow 101 + = 101.875$

2–4 Binary Arithmetic

Add binary numbers

$$\begin{array}{rrl} & 1011 & \rightarrow 11 \\ + & 0011 & \rightarrow 3 \\ \hline & 1110 & \rightarrow 14 \end{array}$$

Subtract binary numbers

$$\begin{array}{rrl} & 1010 & \rightarrow 10 \\ - & 0111 & \rightarrow -7 \\ \hline & 0011 & \rightarrow 3 \end{array}$$

Multiply binary numbers

$$\begin{array}{rrl} & 1010 & \rightarrow 10 \\ \times & 0011 & \times\ 3 \\ \hline & 1010 & \\ & 1010 & \\ + & 000000 & \\ \hline & 11110 & 30 \end{array}$$

Divide binary numbers

$16 = 2^4$ $16+8+4+2$

2–5 Complements of Binary Numbers

Convert a binary number to its 1's complement

11.0101010...

~~00~~11) 10.110

− 11

100

− 11

00100

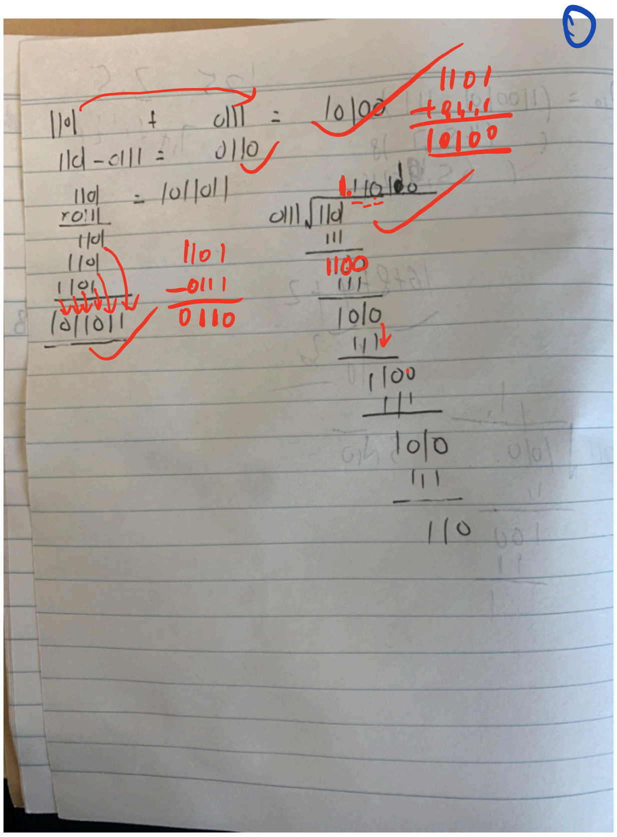
011

001

3) 10

9

1



1101 + 0111 = 10100

1101 − 0111 = 0110

1101 x 0111 = 1011011

1101
1101
1101

1011011

1101
+0111
10100

1101
−0111
0110

1.110100
0111) 1101
111
1100
111
1010
111
1100
111
1010
111
110

Convert a binary number to its 2's complement using either of two methods

EXAMPLE 2–12

Find the 2's complement of 10110010.

Solution

10110010	Binary number
01001101	1's complement
+ 1	Add 1
01001110	2's complement

Related Problem

Determine the 2's complement of 11001011.

2–6 Signed Numbers

The Sign Bit

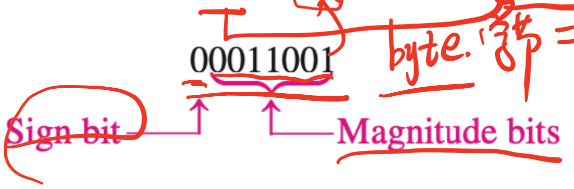
100M/8 = 12.5MB

The left-most bit in a signed binary number is the **sign bit**, which tells you whether the number is positive or negative.

位. 100 Mbps

A 0 sign bit indicates a positive number, and a 1 sign bit indicates a negative number.

byte. 字节 = 8b



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

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

For example:

0 ~ 255

8位数: $0 \sim 2^8 - 1$

补码 0 000 0000 ⟷ 1111 1111

$-(2^{n-1} - 1)$

$2^8 - 1$

原码 1111 1111 ⟷ 0 111 1111

Range of Signed Integer Numbers

-128 ±0

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

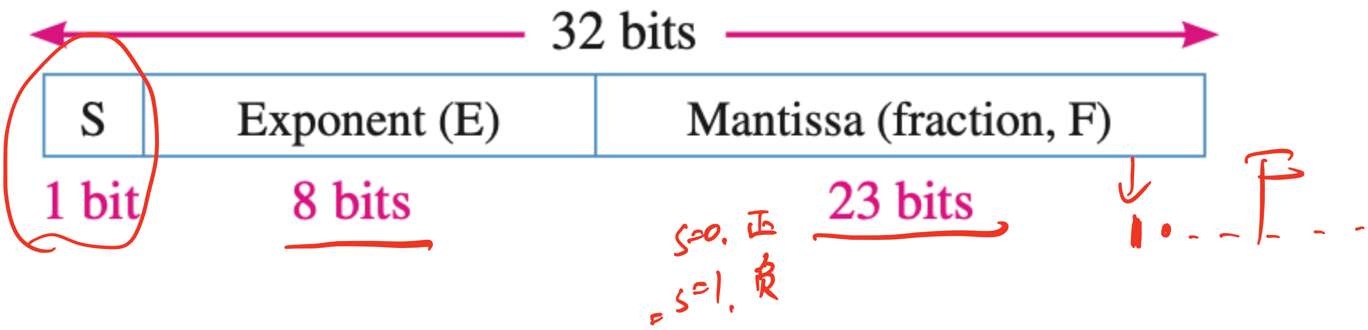
-127 127

Floating-Point Numbers

A **floating-point number** (also known as a *real number*) consists of two parts plus a sign. The **mantissa** is the part of a floating-point number that represents the magnitude of the number and is between 0 and 1. The **exponent** is the part of a floating-point number that represents the number of places that the decimal point (or binary point) is to be moved.

浮点数（也称为实数）由两部分加上一个符号组成。尾数是浮点数的一部分，代表数字的大小，介于 0 和 1 之间。指数是浮点数的一部分，代表小数点（或二进制小数点）的位数。）将被移动。

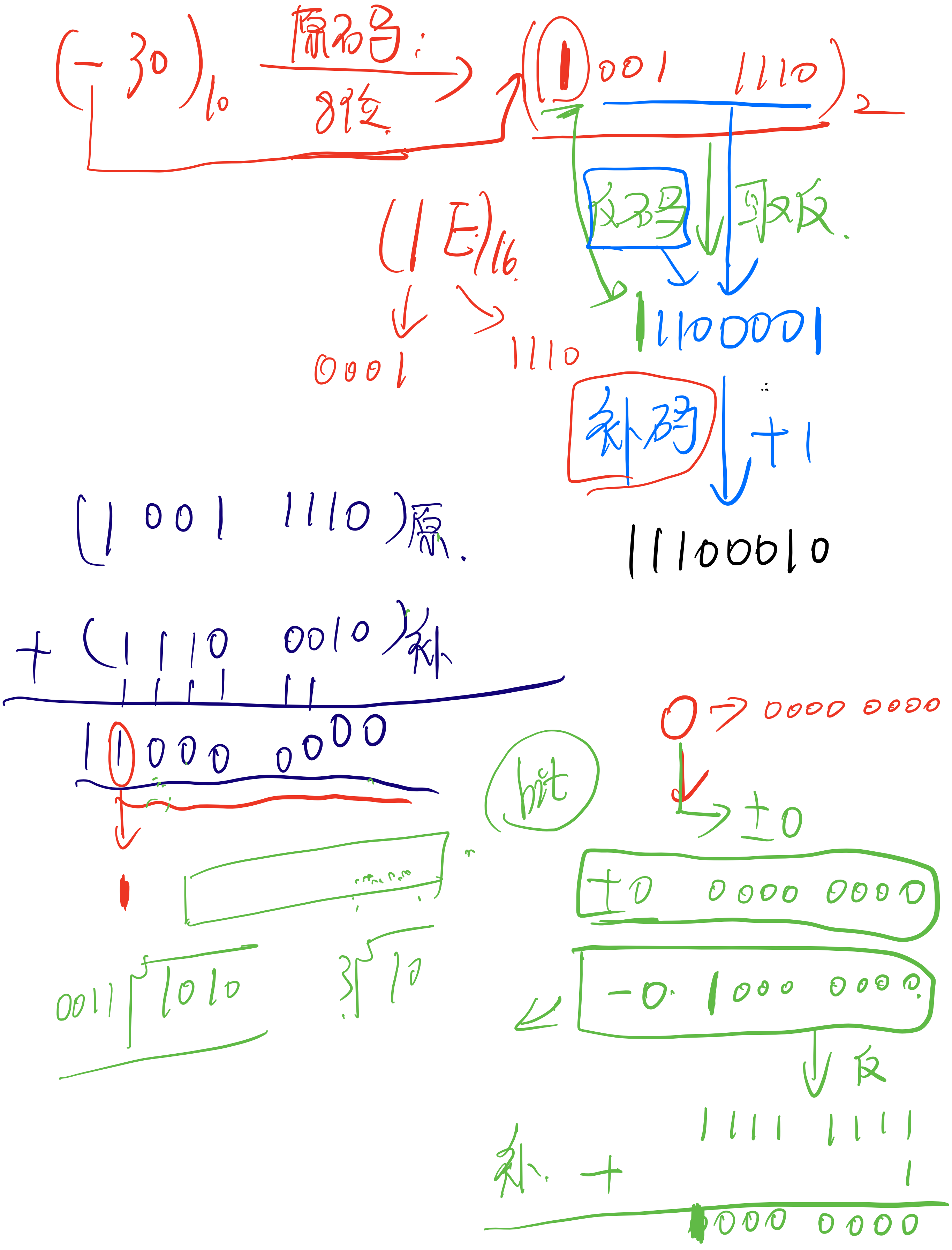
Single-Precision Floating-Point Binary Numbers

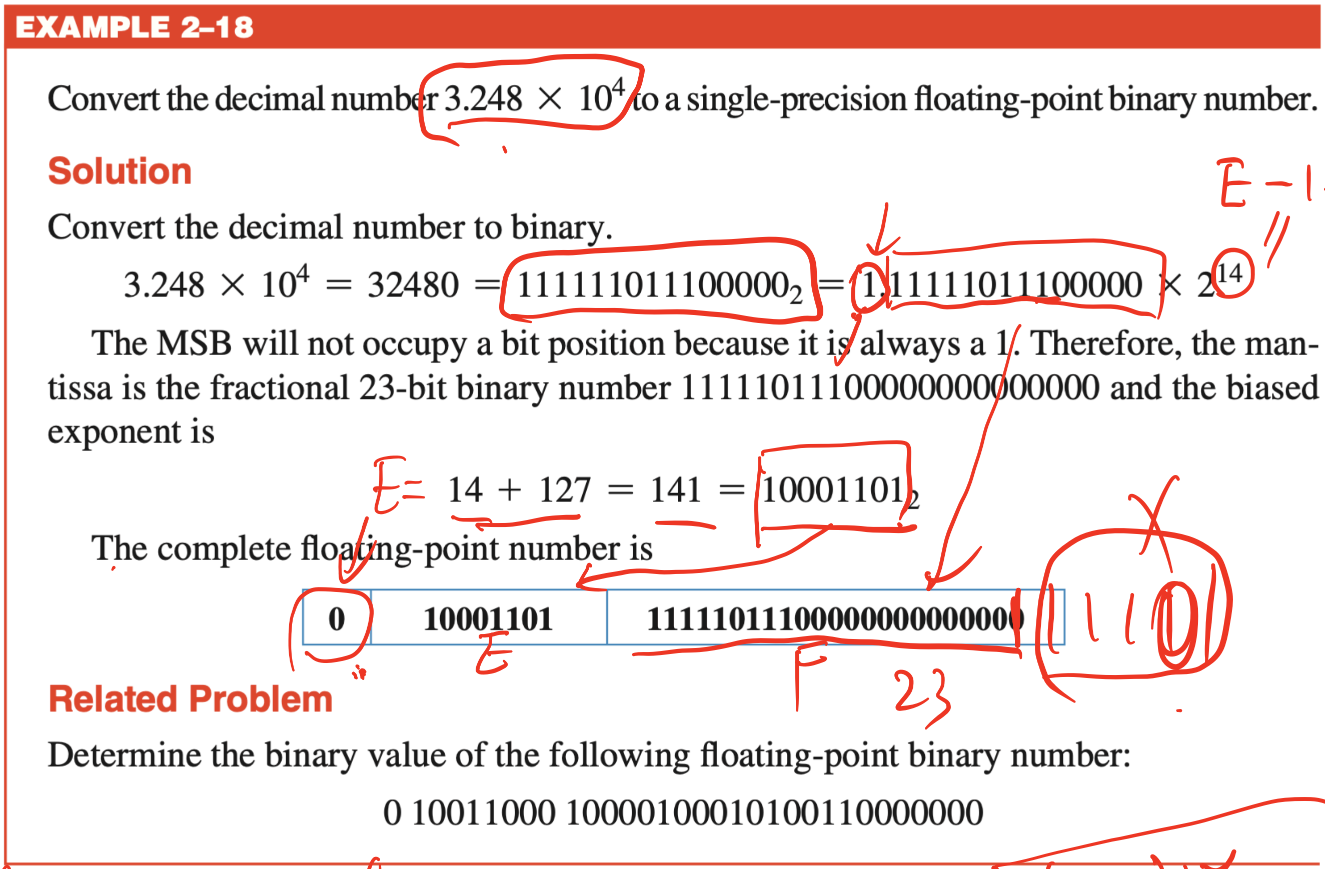


$$\text{Number} = (-1)^{\mathrm{S}}\underline{(1+\mathrm{F})}\left(2^{\mathrm{E}-127}\right)$$

There are two exceptions to the format for floating-point numbers: The number 0.0 is repre- sented by all 0s, and infinity is represented by all 1s in the exponent and all 0s in the mantissa.

浮点数的格式有两个例外：数字 0.0 由全 0 表示，无穷大由指数中的全 1 和尾数中的全 0 表示。

$(-30)_{10}$
原码：
8位
$(1001\ 1110)_2$
$(1E)_{16}$
0001
1110
反码
取反
1110 0001
补码
+1
1110 0010
$(1001\ 1110)$原
+ $(1110\ 0010)$补
1 1 000 0000
0011 1010
3 10
bit
0 → 0000 0000
±0
±0 0000 0000
−0 1000 0000
反
1111 1111
补：+ 1
1 0000 0000



EXAMPLE 2–18

Convert the decimal number 3.248×10^4 to a single-precision floating-point binary number.

Solution

Convert the decimal number to binary.

E − 127

$$3.248 \times 10^4 = 32480 = 111111011100000_2 = 1.11111011100000 \times 2^{14}$$

The MSB will not occupy a bit position because it is always a 1. Therefore, the mantissa is the fractional 23-bit binary number 11111011100000000000000 and the biased exponent is

$$E = 14 + 127 = 141 = 10001101_2$$

The complete floating-point number is

0	10001101	11111011100000000000000
	E	F 23

X 1101

Related Problem

Determine the binary value of the following floating-point binary number:

0 10011000 10000100010100110000000

Matlab: 存入一个很小数 再读出来， 舍入误差

2–7 Arithmetic Operations with Signed Numbers

Addition

The two numbers in an addition are the **addend** and the **augend**. The result is the **sum**. There are four cases that can occur when two signed binary numbers are added.

1. Both numbers positive
2. Positive number with magnitude larger than negative number
3. Negative number with magnitude larger than positive number
4. Both numbers negative

Let's take one case at a time using 8-bit signed numbers as examples. The equivalent decimal numbers are shown for reference.

Both numbers positive:

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

Addition of two positive numbers yields a positive number.

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

Positive number with magnitude larger than negative number:

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

Addition of a positive number and a smaller negative number yields a positive number.

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

Negative number with magnitude larger than positive number:

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

Addition of a positive number and a larger negative number or two negative numbers yields a negative number in 2's complement.

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

Both numbers negative:

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

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

$$\begin{array}{r} -\ \ 1 \\ \hline 11110001 \\ 负\ 10001110 \\ \hline \end{array}$$

Overflow Condition

When two numbers are added and the number of bits required to represent the sum exceeds the number of bits in the two numbers, an **overflow** results as indicated by an incorrect sign bit.

$$\begin{array}{lrr} 正 & 01111101 & 125 \\ +\ \ 正 & +\ 00111010 & +\ 58 \\ \hline 负\ 1 & 10110111 & \boxed{183} \rightarrow 2^7-1 \\ & & >= 127 \end{array}$$

overflow error.

Sign incorrect

Magnitude incorrect

2^{n-1}

$\sim (2^{n-1}-1)$

An overflow can occur only when both numbers are positive or both numbers are nega- tive. If the sign bit of the result is different than the sign bit of the numbers that are added, overflow is indicated.

Subtraction

P+P=P N+N=N

P+P=N N+N=P Overflow

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

EXAMPLE 2–20

Perform each of the following subtractions of the signed numbers:

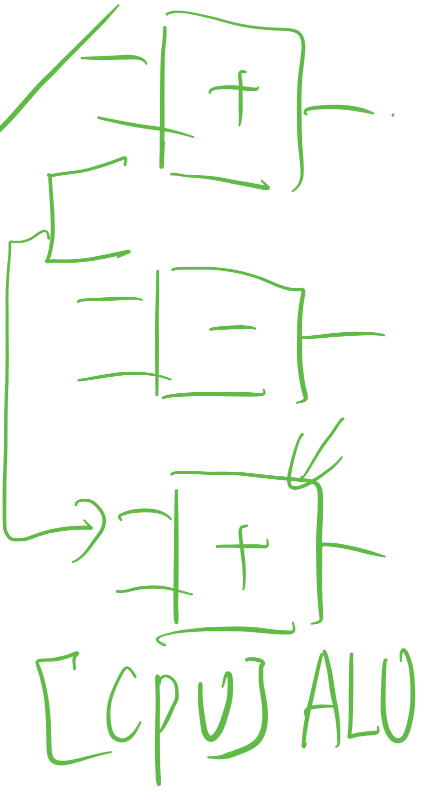
(a) 00001000 − 00000011 **(b)** 00001100 − 11110111

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

15
−6
9

P
00001111
N + 11111010
1 00001001 9
P

−6. 1000 0110
1111 1001
+ 1
1111 1010



15 99
−6 → 100 − 6 = 94

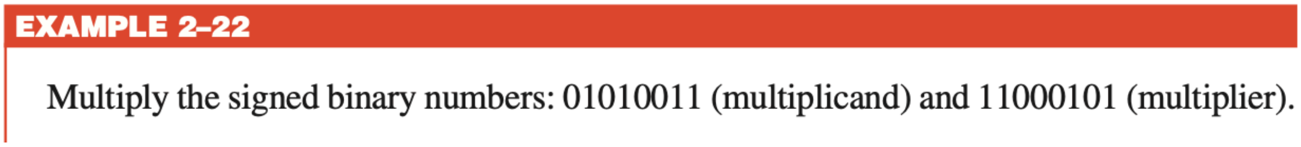
15
94
~~1~~09

Multiplication

The numbers in a multiplication are the **multiplicand**, the **multiplier**, and the **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.**



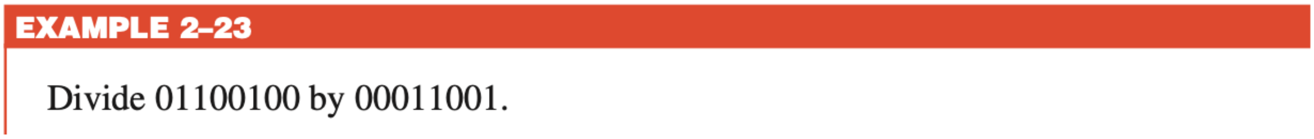
EXAMPLE 2–22

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

Division

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

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



EXAMPLE 2–23

Divide 01100100 by 00011001.

2–8 Hexadecimal Numbers

The **hexadecimal** number system has a base of sixteen; that is, it is composed of 16 **numeric** and alphabetic **characters**.

Test: $(-111-30)_{10}$

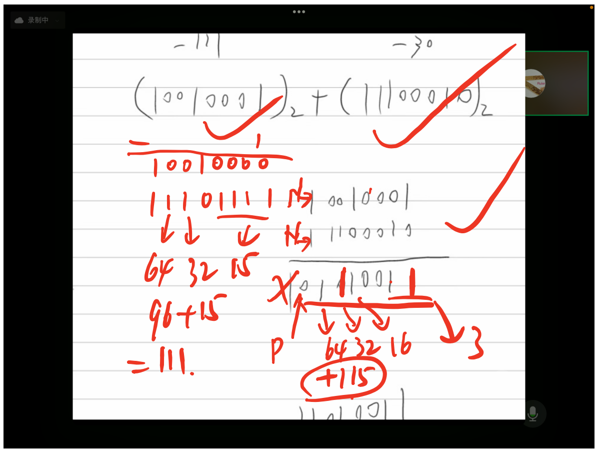
8 bits

补 补

$(\qquad)_2 + (\qquad)_2$

$= (\qquad)_2$

Overflow:



录制中

-111 -30

$(10010001)_2 + (11100010)_2$

$= 100100000$

$11101111\ 1$ $N_1 \rightarrow 10010001$

$N_2 \rightarrow 11100010$

$64\ 32\ 15$ $\cancel{1}01110011$

$96+15$ P $64\ 32\ 16$ 3

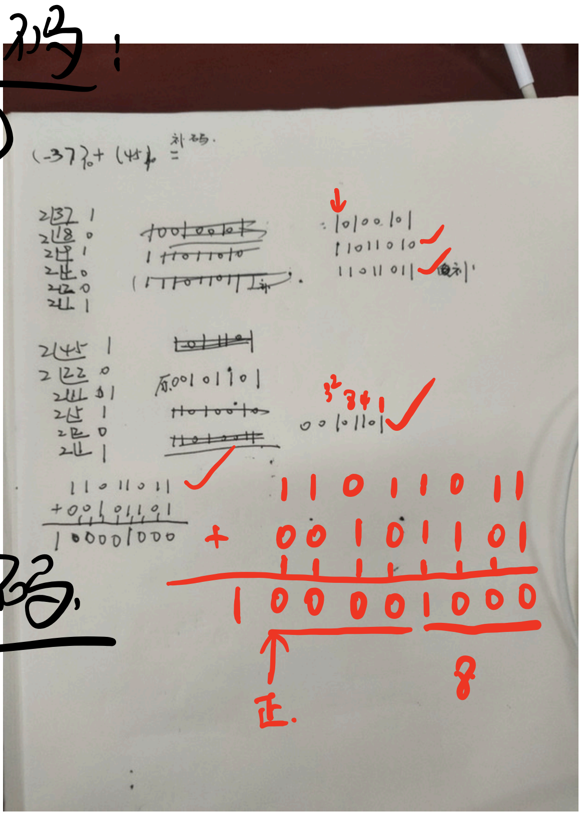
$=111.$ $+115$

11010011

(8位) 补码:

$(-37)_{10} + (45)_{10} \Rightarrow$

同符号的，会发生溢出。



$(-37)_{10} + (45)_{10} =$ 补码.

2|37 1
2|18 0
2|9 1
2|4 0
2|2 0
2|1 1

~~100100101~~
~~11011010~~
~~11011011~~ 补

10100101
11011010
11011011 补

2|45 1
2|22 0
2|11 1
2|5 1
2|2 0
2|1 1

~~1011101~~
原 00101101
~~11010010~~
~~11010011~~

00101101

```
  11011011
+ 00101101
 100001000
```

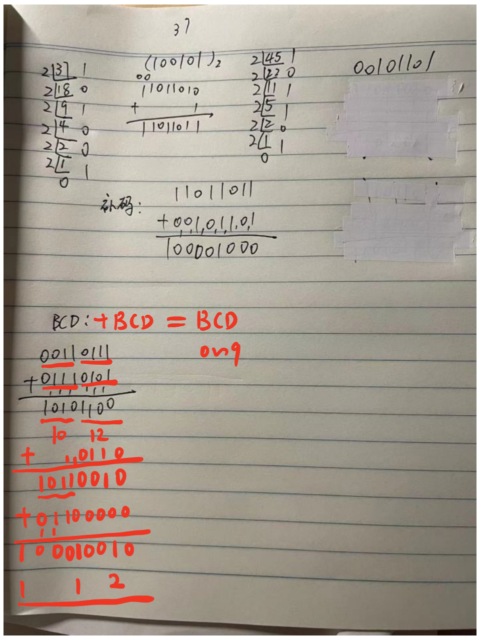
```
  11011011
+ 00101101
 100001000
```

正.

8

(8位) BCD码

$(37+75)_{10} \Rightarrow$



37

2|37 1
2|18 0
2|9 1
2|4 0
2|2 0
2|1 1
0

$(100101)_2$

```
  11011010
+        1
  11011011
```

2|45 1
2|22 0
2|11 1
2|5 1
2|2 0
2|1 1
0

00101101

补码:
```
  11011011
+ 00101101
 100001000
```

BCD: +BCD = BCD

or 9

```
  0011 0111
+ 0111 0101
  1010 1100
    10   12
+    0110
  10110010
+ 01100000
 100010010
 1   1   2
```

+6，从低位开始.

↓

高位

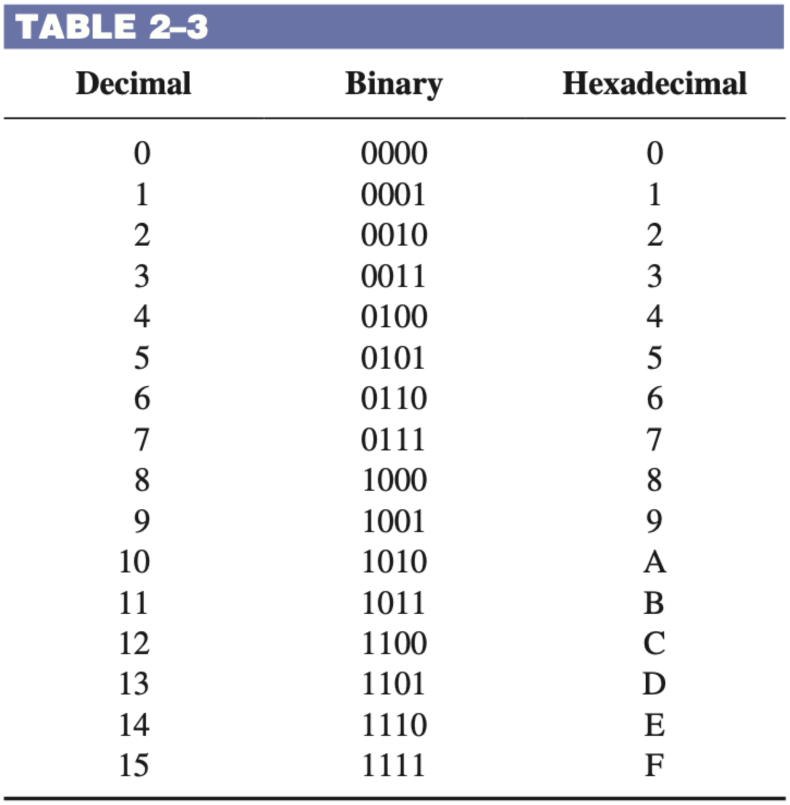
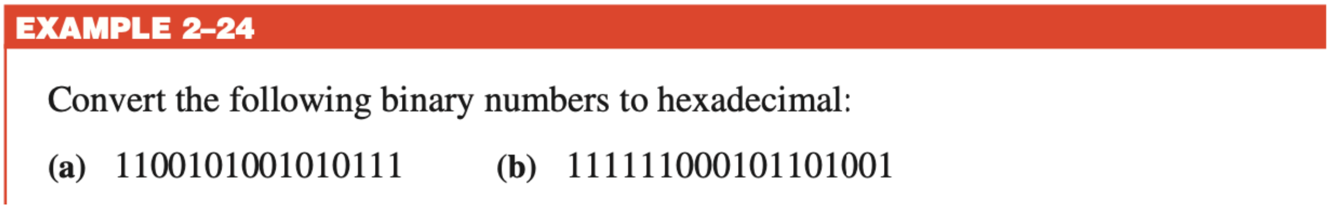


TABLE 2–3

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

Binary-to-Hexadecimal Conversion

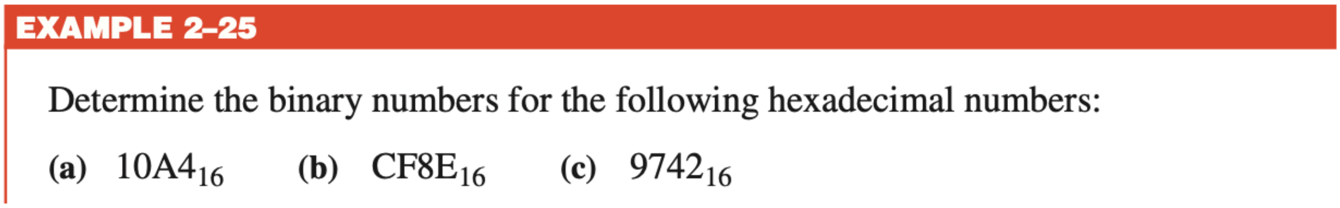


EXAMPLE 2–24

Convert the following binary numbers to hexadecimal:

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

Hexadecimal-to-Binary Conversion

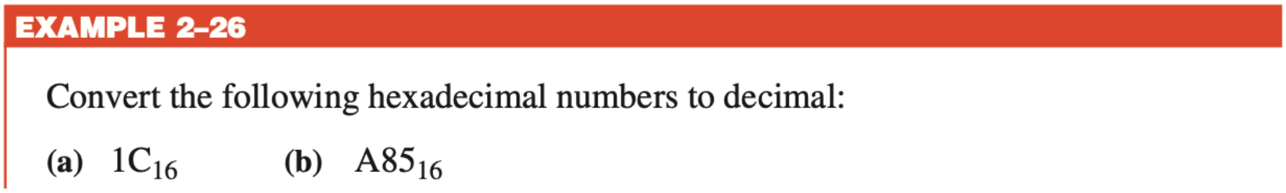


EXAMPLE 2–25

Determine the binary numbers for the following hexadecimal numbers:

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

Hexadecimal-to-Decimal Conversion

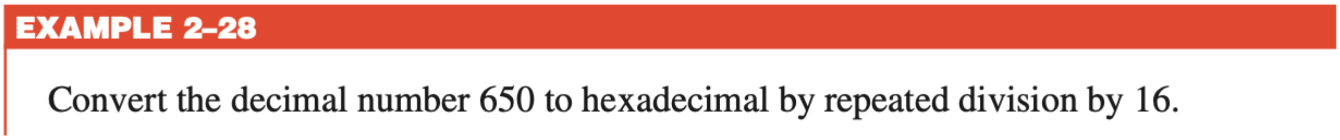


EXAMPLE 2–26

Convert the following hexadecimal numbers to decimal:

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

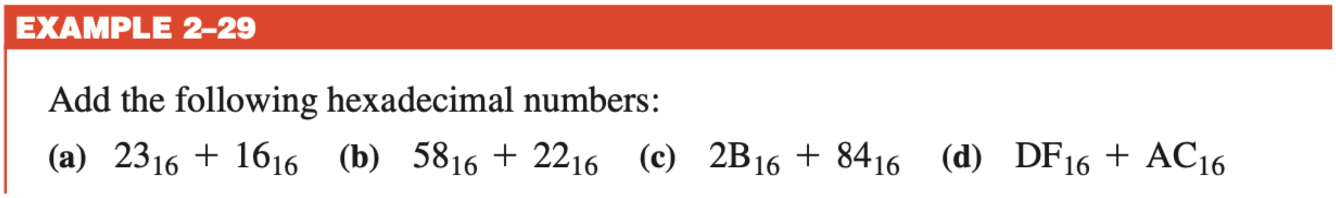
Decimal-to-Hexadecimal Conversion



EXAMPLE 2–28

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

Hexadecimal Addition



EXAMPLE 2–29

Add the following hexadecimal numbers:

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

2–9 Octal Numbers

After completing this section, you should be able to u Write the digits of the octal number system

- Convert from octal to decimal
- Convert from decimal to octal
- Convert from octal to binary
- Convert from binary to octal

2–10 Binary Coded Decimal (BCD)

Binary coded decimal (BCD) is a way to express each of the decimal digits with a binary code.

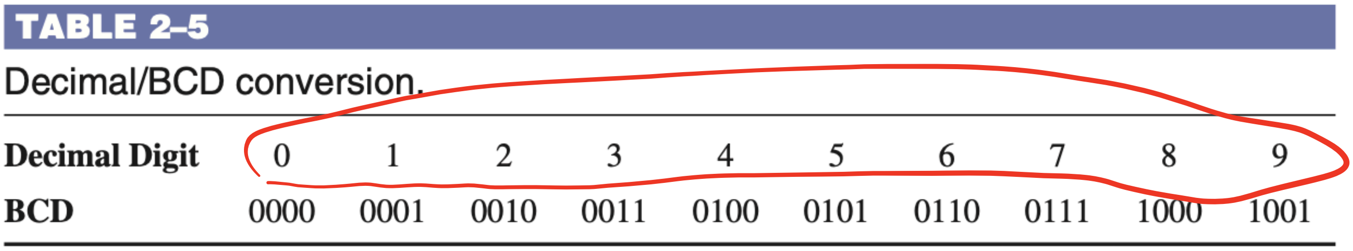
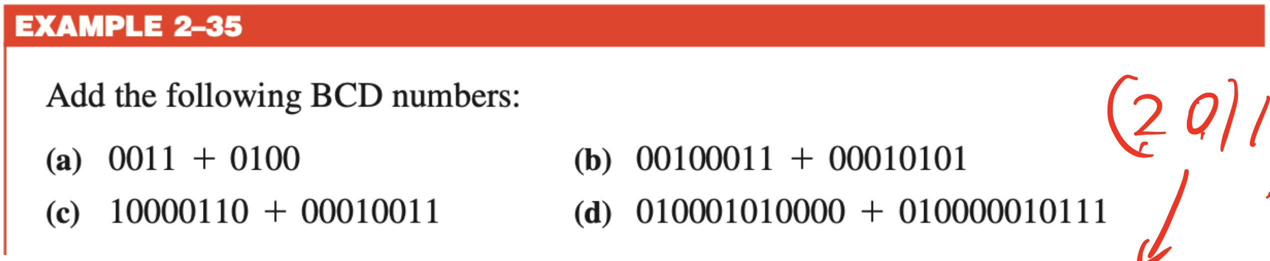


TABLE 2–5

Decimal/BCD conversion.

Decimal Digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001



EXAMPLE 2–35

Add the following BCD numbers:

(a) 0011 + 0100 **(b)** 00100011 + 00010101

(c) 10000110 + 00010011 **(d)** 010001010000 + 010000010111

$(20)_{16}$

$(32)_{10} \rightarrow (\,00100000\,)_2$

$\rightarrow (\,\underline{0011}\,\underline{0010}\,)_{\underline{BCD}}$

$(34)_{10} + (47)_{10}$

$\Rightarrow (0011\ 0100)_{BCD} + (0100\ 0111)_{BCD}$

$$\begin{array}{rll} & 0011 & 0100 \\ + & 0100 & 0111 \\ \hline & 0111 & 1011 \\ & (7 & B)_{BCD} \end{array} \qquad \begin{array}{r} 34 \\ +47 \\ \hline 81 \end{array}$$

X

if > 9, +6

$$\begin{array}{rll} + & 0000 & 0110 \\ & 1111 & 11 \\ \hline & (1000 & 0001)_{BCD} \\ & 8 & 1 \end{array}$$

2–11 Digital Codes

The Gray Code

格雷码.

The **Gray code** is unweighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions.

TABLE 2–6

Four-bit Gray code.

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

只有1位发生变化

无权值

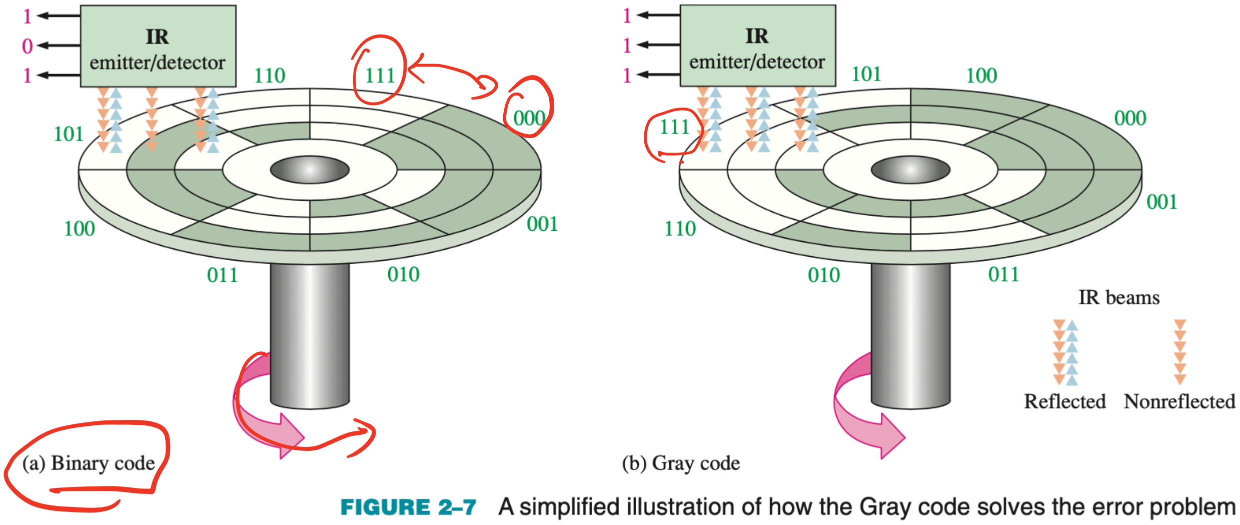


FIGURE 2–7 A simplified illustration of how the Gray code solves the error problem in shaft position encoders. Three bits are shown to illustrate the concept, although most shaft encoders use more than 10 bits to achieve a higher resolution.

ASCII

ASCII is the abbreviation for American Standard Code for Information Interchange. Pro- nounced “askee,” ASCII is a universally accepted alphanumeric code used in most comput- ers and other electronic equipment.

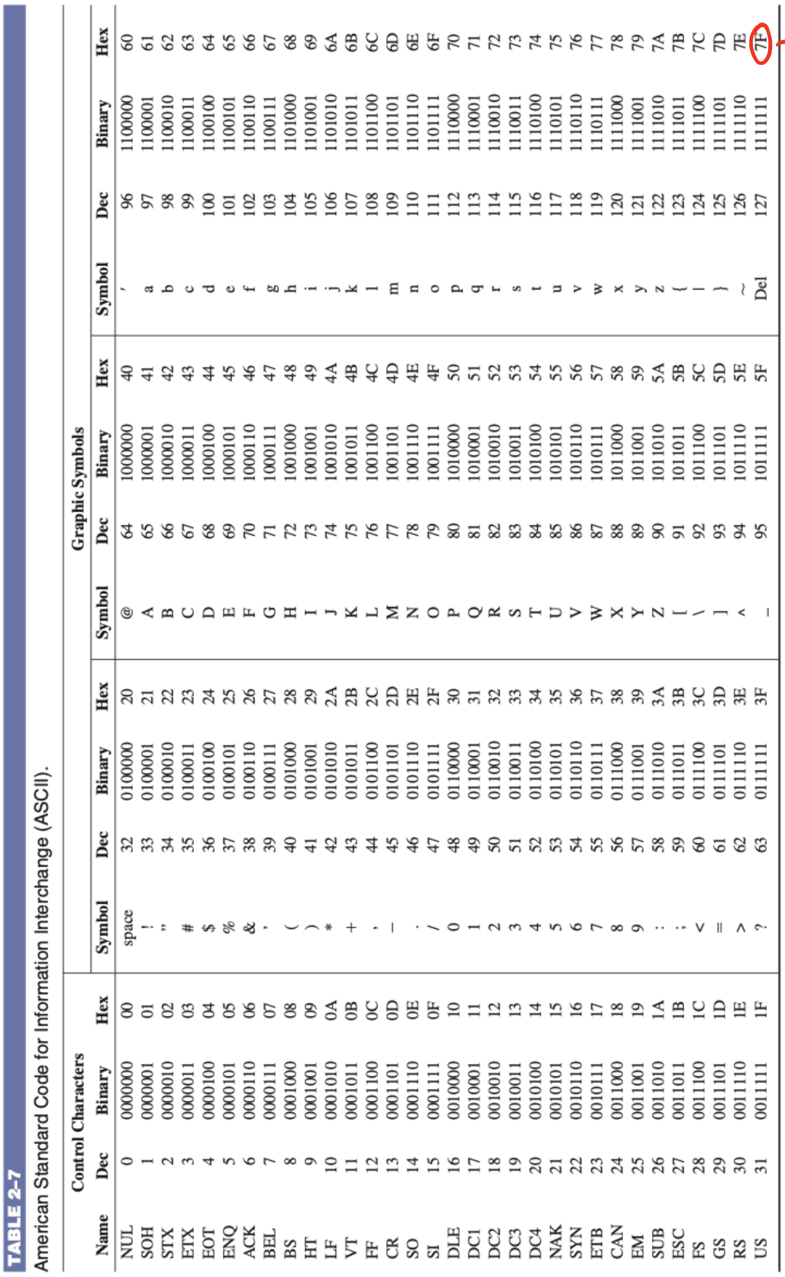


TABLE 2-7

American Standard Code for Information Interchange (ASCII).

Control Characters				Graphic Symbols											
Name	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	-	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

→7

2^8-1

=127个

Unicode

Unicode provides the ability to encode all of the characters used for the written languages of the world by assigning each character a unique numeric value and name utilizing the universal character set (UCS). It is applicable in computer applications dealing with multi- lingual text, mathematical symbols, or other technical characters.

Unicode 通过使用通用字符集 (UCS) 为每个字符分配一个唯一的数值和名称，提供了对用于世界书面语言的所有字符进行编码的能力。它适用于处理多语言文本、数学符号或其他技术字符的计算机应用程序。

2–12 Error Codes

1101 → 1111

error

Parity Method for Error Detection

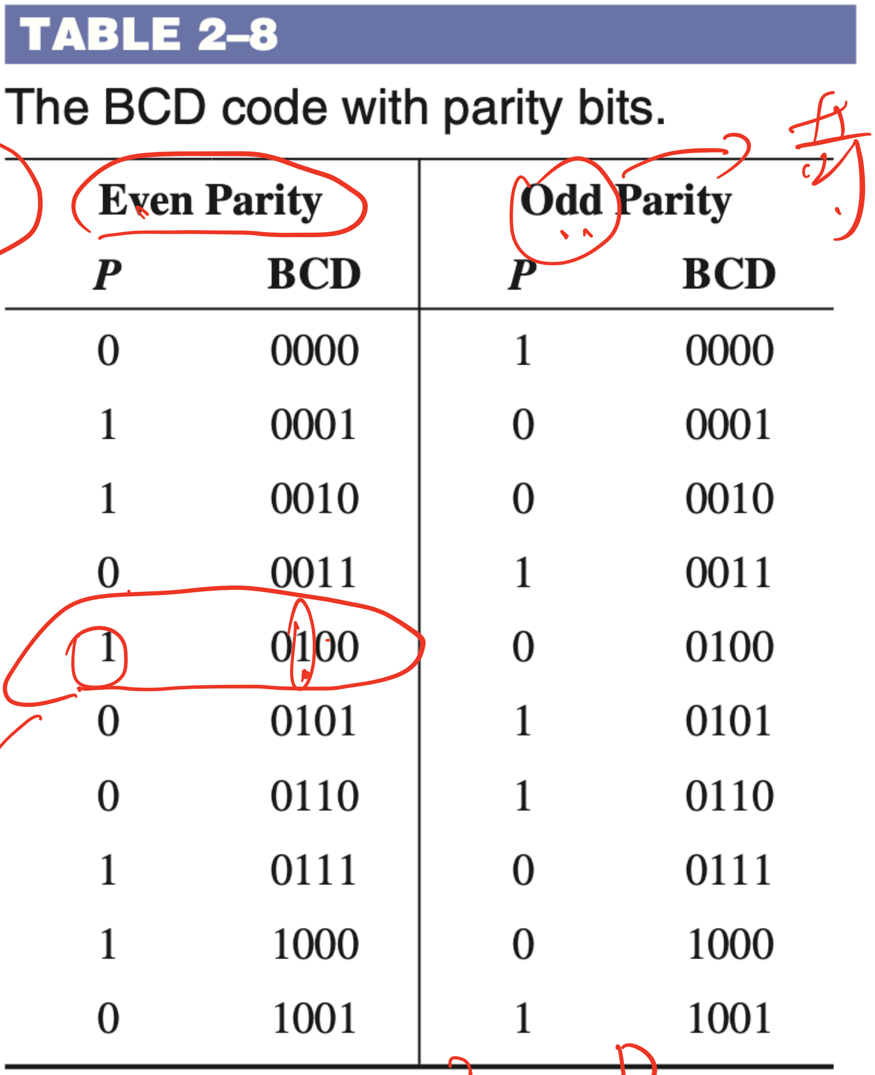


TABLE 2–8

The BCD code with parity bits.

偶

奇

Even Parity		Odd Parity	
P	BCD	*P*	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

P D

1 0100

发

P D

1 0110 X

收

D

1 0111

Chap1, 2. 作业 上网查看。