电子技术
Introduction to Electronics

By Bao Qilian

鲍其莲

电子技术

Course textbook

- *Digital Fundamentals*, Thomas L. Floyd, 10th Edition, Prentice Hall
- 模拟电路讲义
- 数字电子技术基础，阎石，第四版，高等教育出版社

Digital_ie@163.com
digital@

电子技术

Goals:
Gaining basic knowledge and skills in analysis and design of electronic circuits and digital circuits.

Grading Policy:
1.Coursework / Quiz /Attendance 30%
2. Final exam 70%

电子技术

The Digital System Hierarchy

Systems *(e.g., computers)*
↑
Functional Units
(e.g., Registers, Memories, Arithmetic units, etc.)
↑
Gates and Flip-Flops
↑
Electronic Circuits

电子技术

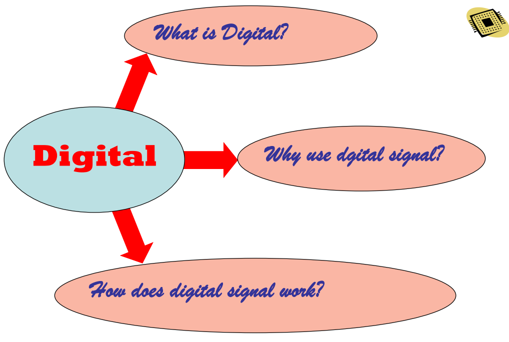
The Digital System Hierarchy

Systems *(e.g., computers)*
↑
Functional Units
(e.g., Registers, Memories, Arithmetic units, etc.)
↑
Gates and Flip-Flops
↑
Electronic Circuits

电子技术

Chapter 1 Digital Concepts

- *Objectives*
  - Digital and analog quantities
  - Binary digits, logic levels, and digital waveforms
  - Basic logic operations
  - Fixed function integrated circuits
- *Reading assignment*
  - Chapter 1(p2-p22)



What is Digital?

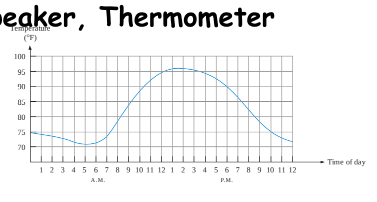
Analog ⇨ Digital

1.1 Digital and analog quantities

电子技术

- Analog signal（模拟信号）
  - Continues in time domain
  - Continues values

e.g. Speaker, Thermometer

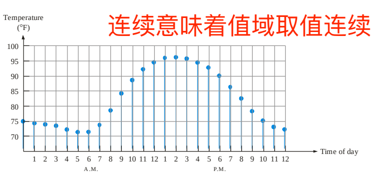


Sampled signal

电子技术

- Sampled signal（采样信号）
  - Discrete （离散） in time domain
  - Continues values

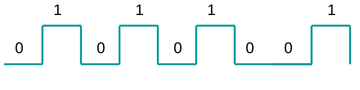
连续意味着值域取值连续



Digital Signal

电子技术

- Digital Signal （数字信号）
  - Discrete in time domain
  - Discrete values



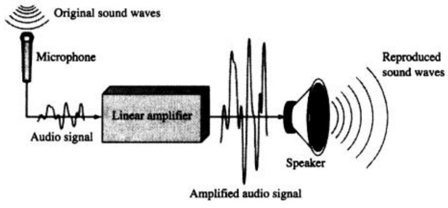
Characteristics of Digital Signal

电子技术

- Signals can assume only a discrete number of values, usually two.
- Noise immunity
- Unlimited precision: use multiple signals
- Simple, cheap and stable circuits
- Easy to detect one of two values - thresholds

Analog Example

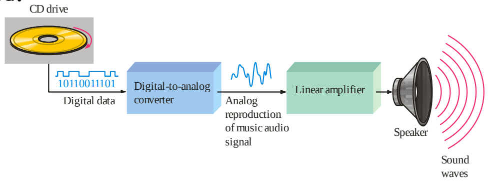
A public address system, used to amplify sound so that it can be heard by large audience, is one example of an application of analog electronics.



电子技术

A Mixed System

The compact disk (CD) player is an example of a system in which both digital and analog circuits are used.



电子技术

A Digital Example

A computer system is one example of an application of digital electronics

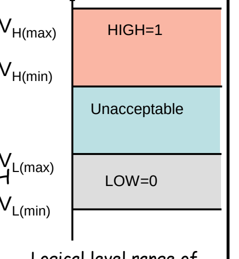
电子技术

1.2 Binary digits, logical levels and digital waveforms

- Binary
  - HIGH = 1, True $\rightarrow V_H$
  - LOW = 0, False $\rightarrow V_L$

Logical Level

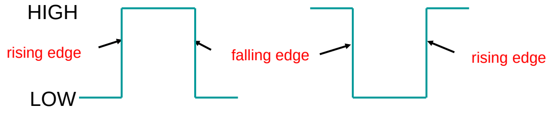
$V_{H(max)}$: maximum voltage value for HIGH
$V_{H(min)}$: minimum voltage value for HIGH
$V_{L(max)}$: maximum voltage value for LOW
$V_{L(min)}$: minimum voltage value for LOW



Logical level range of voltage for a digital circuit

电子技术

Digital waveforms-Pulse(I)（脉冲）

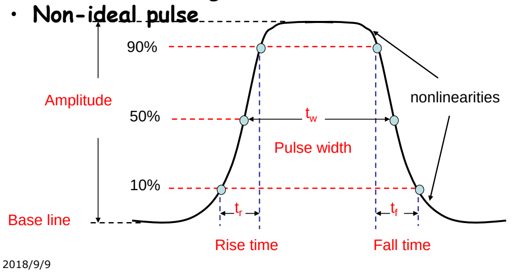


Positive-going pulse Negative-going pulse

电子技术

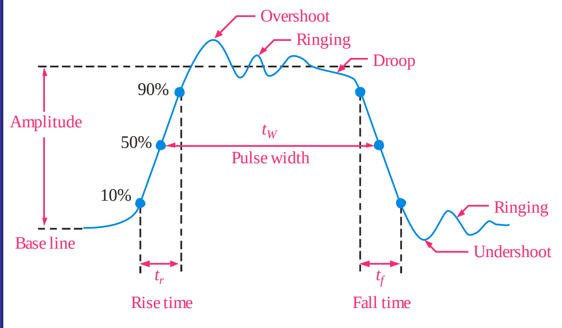
Pulse(II)

- Ideal pulse: the rising and falling edges are assumed to change in zero time.
- Non-ideal pulse



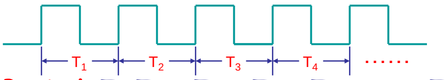
电子技术

Nonideal pulse characteristic



Digital waveform characteristics

- Periodic pulse（周期脉冲）



Period=$T=T_1= T_2= T_3= T_4=\cdots=T_n$

Frequency=$1/T$

- Non-periodic pulse

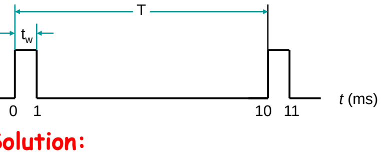
Duty cycle（占空比） -- the ratio of the pulse width (t_w) to the period (T)), be expressed as a percentage.

$$Duty\ cycle=\frac{t_w}{T}\times 100\%$$

Example:

A portion of a periodic digital waveform is shown in the figure. The measurements are in milliseconds. Determine the following:

(1)period (2)frequency (3) duty cycle



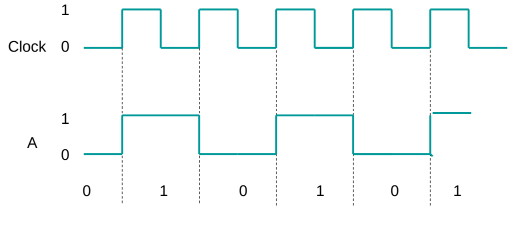
Solution:

(1) T=10 ms

(2) f=1/T=1/10ms=1/0.01s=100Hz

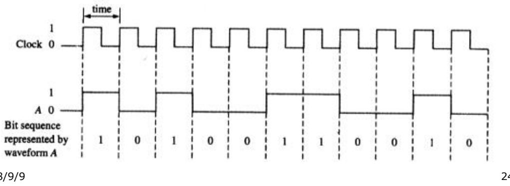
(3) Duty cycle=t_w/T=1ms/10ms×100%=10%

Example: Digital waveform carries binary information



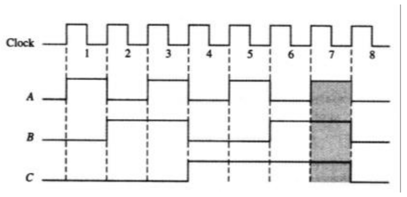
The Clock （时钟）

- In digital systems, all digital waveforms are synchronized (同步) with a basic timing waveform, called the clock. The clock is a periodic waveform. The clock waveform itself does not carry information.



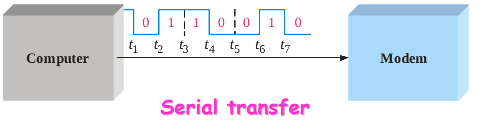
Timing diagrams（时序图）

- A timing diagram is a graph of digital waveforms showing the actual time relationship of two or more waveforms and how each changes in relation to the others



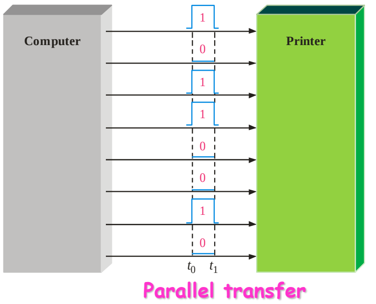
Data transfer

- When bits are transferred in serial form from one point to another, they are sent one bit at a time along a single conductor. To transfer *n* bits in series, it takes *n* time intervals.



Serial transfer

- When bits are transferred in parallel form, all the bits in a group are sent out on separate lines at the same time. There is one line for each bit. To transfer *n* bits in parallel, it takes one time interval.



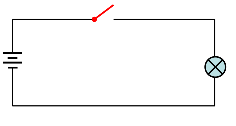
Parallel transfer

1.3 Basic logic operation

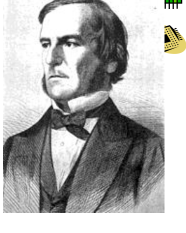
- Problems can be described by logic functions.
- True/false—how to represent

Logic operation

- True-light on/High
- False-light off/Low

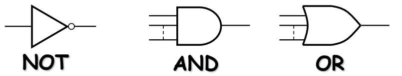


- In 1850s, the Irish logician and mathematician George Boole developed a mathematical system for formulating logic statements with symbols so that problems can be written and solved in a manner similar to ordinary algebra.



Logic gates

- In Boolean algebra, there are three basic logic operations: NOT, AND, and OR.
- circuit that performs a specified basic logic operation is called a logic gate.
- The standard distinctive shape symbols for the three basic logic gates are shown below.

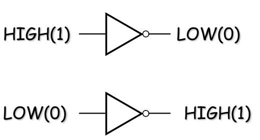


NOT AND OR

- The true/false statements mentioned earlier are represented by a HIGH (true) and a LOW (false).
- AND and OR gates can have any number of inputs.
- Each of the three basic logic operations produces a unique response (output) to a given set of conditions (inputs).

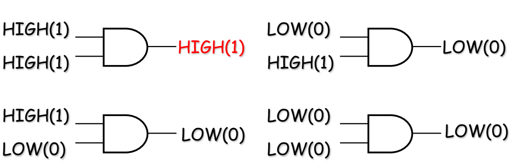
The NOT Operation

- The NOT operation changes one logic level to the opposite logic level.
- The NOT operation is implemented by a logic circuit known as an inverter (NOT gate).



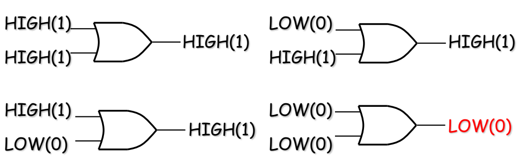
The AND Operation

- The AND operation produces a HIGH output only if all the inputs are HIGH. When any or all inputs are LOW, the output is LOW.
- The AND operation is implemented by a logic circuit known as an AND gate.



The OR Operation

- The OR operation produces a LOW output only if all the inputs are LOW. When any or all inputs are HIGH, the output is HIGH.
- The OR operation is implemented by a logic circuit known as an OR gate.

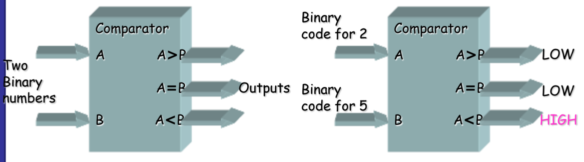


1.4 Fixed function integrated circuits

- The three basic logic elements AND, OR, and NOT can be combined to form more complex logic circuits.
- Some of the common logic functions are comparison, arithmetic, code conversion, encoding, decoding, data selection (multiplexing, demultiplexing), storage, and counting.

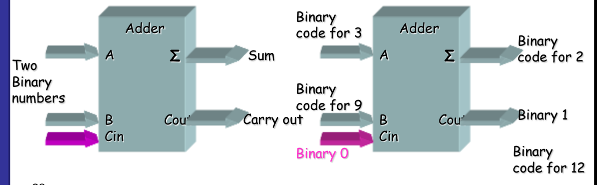
a. The Comparison Function

- Magnitude comparison is performed by a logic circuit called a comparator. A comparator compares two quantities and indicates if they are equal or not equal and, if not equal, which is greater.



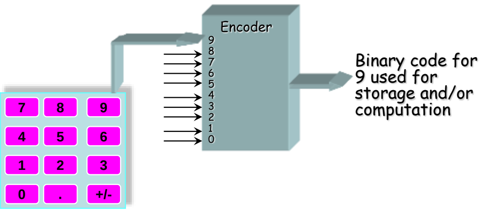
b. The Arithmetic Function

- Addition/subtraction/multiplication/division is performed by a logic circuit called an adder/subtracter/multiplier/divider.
- An adder adds two binary numbers and generates a sum and a carry.



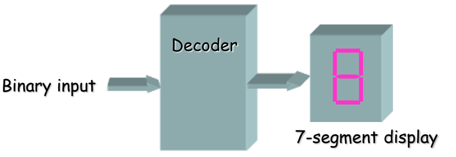
c. The Encoding Function

- The encoding function is performed by a logic circuit called an encoder. The encoder converts information, such as a decimal number or alphabetic character, into some coded form.



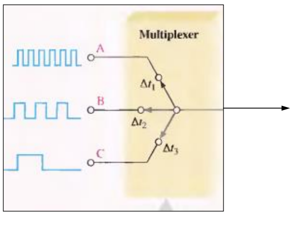
d. The Decoding Function

- The decoding function is performed by a logic circuit called an decoder. The decoder converts coded information, such as a binary number, into noncoded form.



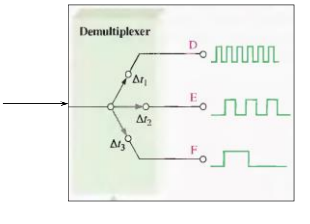
e. The Multiplexing Function

- The multiplexing function is performed by a logic circuit called a multiplexer, or mux for short. The mux switches digital data from several input lines onto a single output line in a specified time sequence.



f. The DeMultiplexing Function

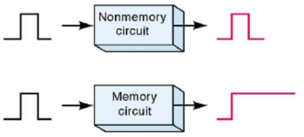
- The demultiplexing function is performed by a logic circuit called a demultiplexer, or demux for short. The demux switches digital data from one input line to several output lines in a specified time sequence.



g. The Storage Function

Storage is a function that is required in most digital systems, and its purpose is to retain binary data for a period of time.

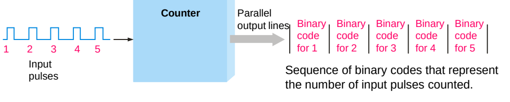
Common types of storage devices are flip-flops, registers, semiconductor memories, magnetic disks, magnetic tape, and optical disks.



- The flip-flop is a bistable (two stable states) logic circuit that can store only one bit at a time, either a 1 or a 0. The output of a flip-flop indicates which bit it is storing.
- A register is formed by combining several flip-flops so that groups of bits can be stored.
- In addition to storing bits, registers can be used to shift the bits from one position to another within the register or out of the register to another circuit; therefore, these devices are known as shift registers.

h. The Counting Function

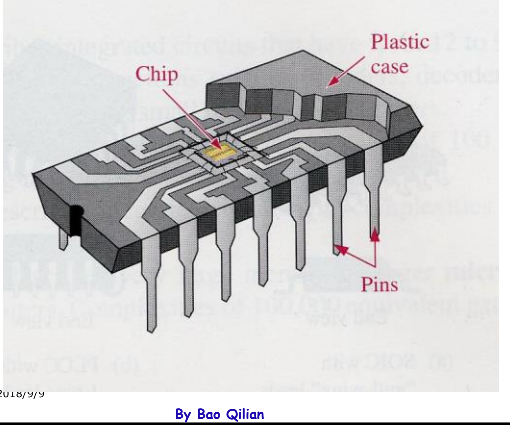
- The counting function is performed by a logic circuit called a counter. The basic purpose of counters is to count events or to generate a particular code sequence.



Digital Integrated Circuits (IC)

A single chip of integrated circuits（集成电路） is an electronic circuit that is constructed entirely on a single small chip of silicon. All the components that make up the circuit , including transistors, diodes, resistors, and capacitors, are an integral part of that single chip.

The following figure shows a cutaway view of one type of IC package, with the circuit chip shown within the package. Points on the chip are connected the the package pins to allow input and output connections to the outside world.



IC Packages（封装）

- **Through-hole package: e.g. dual-in-line package (DIP), the most common type.**
- **Surface-mounted technology (SMT):**

  e.g. SOIC (small-outline IC),

  PLCC (plastic leaded chip carrier)

  LCCC (leadless ceramic chip carrier),

  FP (flat pack)

  SSOP (shrink small-outline package)

  TSSOP (thin shrink small-outline package)

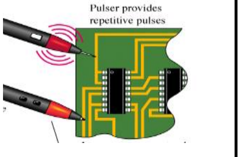
  TVSOP (thin very small-outline package)

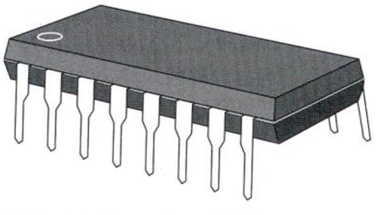
DIP

- The through-hole packages have pins (leads) that are inserted through holes in the PC board and can be soldered to conductors on the opposite side. The most common type of through-hole package is the dual-in-line package (DIP).

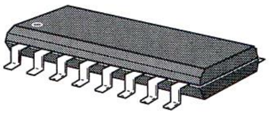
SMT

- **The pins of the surface-mount technology (SMT) packages are soldered directly to the conductors on one side of the board, leaving the other side free for additional circuits.**

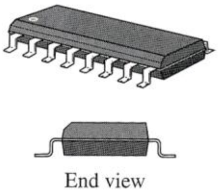




(a) Dual-in-line package (DIP)

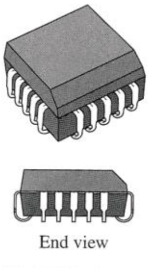


(b) Small-outline IC (SOIC)



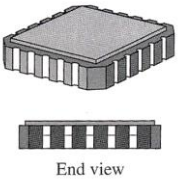
End view

(a) SOIC with "gull-wing" leads



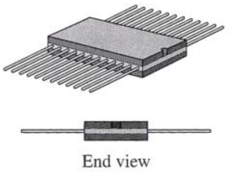
End view

(b) PLCC with J-type leads



End view

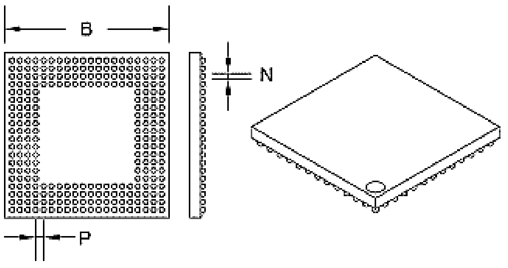
(c) LCCC with no leads (contacts are part of case)



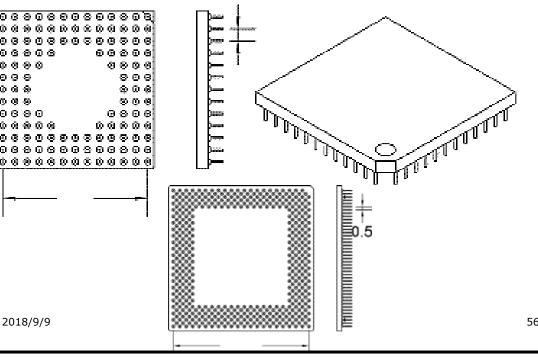
End view

(d) Flat pack (FP) with straight leads

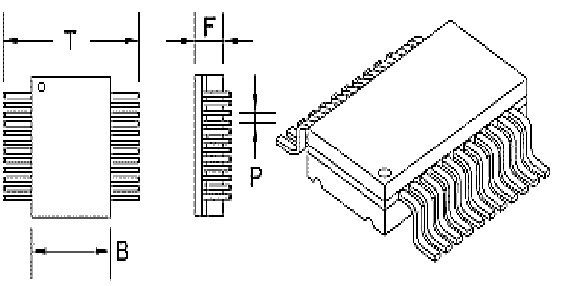
BGA (Ball Grid Array)



PGA (Pin Grid Array) Package



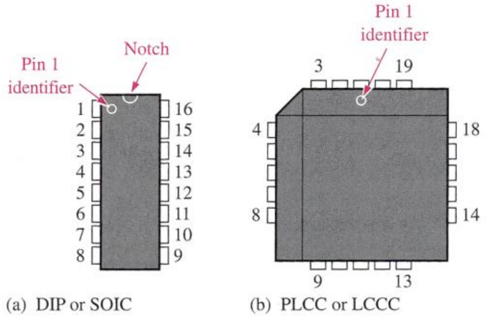
SOIC



Pin Numbering

All IC packages have a standard format for numbering pins (leads). Pin1 is indicated an identifier that can be either a small dot, a notch, or a beveled edge.

The pin number increase as you go coputer clockwise.



(a) DIP or SOIC (b) PLCC or LCCC

Summary

- Digital and analog quantities
- Binary digits, logic levels, and digital waveforms
- Basic logic operations
- Fixed function integrated circuits

Chapter 2 Number systems, operations, and codes

- Objectives
  - Number systems (Binary, Decimal, Hexadecimal)
  - Conversions
  - Arithmetic operations
  - BCD and digital codes
- Reading assignments
  - Chapter2 p38 – p84

2.1 Number Systems（数制）

- Number systems
  - Decimal numbers
  - Binary numbers
  - Hexadecimal numbers
  - Octal numbers

Number systems	digits	examples
Decimal numbers	0,1,2,3,4,5,6,7,8,9	1234.5678
Binary numbers	0,1	11011.111
Hexadecimal numbers	0,1,2,3,4,5,6,7,8,9, a,b,c,d,e,f	A50F.12D
Octal numbers	0,1,2,3,4,5,6,7	3472.123

2.2 Conversion of different number systems

Sum-of-Weight expression:

$(N)_r = C_{n-1}r^{n-1} + C_{n-2}r^{n-2} + \ldots + C_1 r^1 + C_0 r^0 + C_{-1}r^{-1} + \ldots + C_{-m}r^{-m}$

Where:

r — base of number system
c — character from the character set of the base
N — number to be represented in base r
n — the number of digits in the integer portion of N
m — the number of digits in the fractional portion of N

e.g.

$(1234.5678)_{10} = 1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1} + 6\times10^{-2} + 7\times10^{-3} + 8\times10^{-4}$

$(11011.011)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$

Base r to Decimal

Apply Sum-of-Weight equation:

Binary-to-Decimal:

e.g. $(11011.011)_2$

$= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$

$= 16+8+0+2+1+0.5+0.25+0.125 = (27.875)_{10}$

Octal-to-Decimal:

e.g. $(672.12)_8$

$= 6\times8^2 + 7\times8^1 + 2\times8^0 + 1\times8^{-1} + 2\times8^{-2}$

$= 384+56+2+0.125+0.0625 = (442.1875)_{10}$

Hexadecimal-to-Decimal

e.g.

$(1aef.ea)_{16}$

$= 1\times16^3 + 10\times16^2 + 14\times16^1 + 15\times16^0 + 14\times16^1 + 10\times16^{-2}$

$= 4096+2560+224+15+0.875+0.0390625$

$= (6895.9140625)_{10}$

Decimal to Base r

- Conversion of integer numbers–
  Repeated Division-by-r (Successive Division)(除r取余法)
- Conversion of fractional numbers—
  Repeated Multiplication-by-r
  (Successive Multiplication)（乘r取整法）

Repeated Division-by-r

e.g. Convert $(119)_{10}$ to Binary

Remainder

2 | 119	1	LSB-Least Significant Bit
2 | 59	1	
2 | 29	1	
2 | 14	0	
2 | 7	1	
2 | 3	1	
2 | 1	1	MSB-Most Significant Bit
0		

1110111

Stop when the quotient is 0.

repeated division-by-r

Steps:

1. Divide the quotient by two (or *r*) and record the remainder.
2. Repeat step (1) until the quotient is equal to zero (0).
3. The first remainder produced is the LSB in the binary number and the last remainder (R) the MSB. Accordingly, the binary number is then written (from left to right) with the MSB occurring first.

Repeated Multiplication-by-r

e.g. Convert $(0.3125)_{10}$ to Binary

Carry

$0.3125 \times 2 = 0.625$	0	MSB
$0.625 \times 2 = 1.25$	1	
$0.25 \times 2 = 0.50$	0	
$0.5 \times 2 = 1.00$	1	LSB

.0101

Stop when the fractional part is all 0.

Binary to Octal Conversion

e.g. $(111011001.010011)_2 = (731.23)_8$

111,011,001.010,011

7 3 1 . 2 3

Partition the binary number into groups of three, starting at the binary point and going left and right.

Binary to Hexadecimal Conversion

e.g. $(11011001.01001101)_2$

1101,1001.0100,1111

D 9 . 4 F

Partition the binary number into groups of four, starting at the binary point and going left and right.

Note:

Add 0s when not enough bits for grouping.

e.g. $(11010110.0101)_2$

= $(011,010,110.010,100)_2$

$=(326.24)_8$

$(1011110.101)_2$

= $(0101,1110.1010)_2$

$=(5E.A)_{16}$

Octal or Hexadecimal to Binary Conversion

$(343.126)_8=(011,100,011.001,010,110)_2$

$(FE8A.6B)_{16}=(1111,1110,1000,1010.0110,1011)_2$

2.3 Binary Arithmetic

Basics of binary addition, subtraction, multiplication, and division.

Binary Addition

Four basic rules for adding binary bits :

- 0+0=0 Sum of 0 with a carry（进位） of 0
- 0+1=1 Sum of 1 with a carry of 0
- 1+0=1 Sum of 1 with a carry of 0
- 1+1=0 Sum of 0 with a carry of 1

Note: The first three rules result in a single bit and in the fourth rule the addition of two 1s yields a binary two (10), with a carry of 1 over to the left.

Addition with a carry（带进位加法）

e.g. 11+1

	Carry	Carry	
	1	1	
	0	1	1
	+0	0	1
	1	0	0
	Sum=1	Sum=0	Sum=0
	Carry=0	Carry=1	Carry=1

Addition with a carry

When carry bit is 1, the four rules are:

- 1+0+0=1 Sum of 1 with a carry of 0
- 1+0+1=0 Sum of 0 with a carry of 1
- 1+1+0=0 Sum of 0 with a carry of 1
- 1+1+1=1 Sum of 1 with a carry of 1

Carry bits

电子技术

Example: Add the following binary numbers:

(a)11+11 (b) 110+100

Solution:

$$\begin{array}{rr} \text{(a)} & 11 \\ & +11 \\ \hline & 110 \end{array} \qquad \begin{array}{rr} \text{(b)} & 110 \\ & +100 \\ \hline & 1010 \end{array}$$

电子技术

Binary Subtraction

Four basic rules for subtracting bits:

- 0 – 0 = 0
- 1 – 1 = 0
- 1 – 0 = 1
- 10 – 1 = 1 0-1 with a borrow of 1

Note: The fourth rule shows a borrow from the next column to the left. A borrow is required when subtracting a 1 from a 0.

电子技术

Example: Subtract 011 from 101.

Solution:

A 1 is borrowed, a 0 is left in this column.

Borrow from left column, make a 10 in this column

$$\begin{array}{rrr} 0 & 10 & \\ 1 & 0 & 1 \\ -0 & 1 & 1 \\ \hline 0 & 1 & 0 \end{array}$$

电子技术

Binary Multiplication

Four basic rules for multiplying bits:

- 0 × 0 = 0
- 0 × 1 = 0
- 1 × 0 = 0
- 1 × 1 = 1

**Multiplication is performed with binary numbers in the same manner as with decimal numbers. It involves forming partial products, shifting each successive partial product left one place, and then adding all the partial products.

电子技术

Example: Perform the following binary multiplications:

(a) 11 × 11 (b) 101 × 11

Solution:

$$\begin{array}{rr} \text{(a)} & 11 \\ & \times 11 \\ \hline & 11 \\ & +11 \\ \hline & 1001 \end{array} \qquad \begin{array}{rr} \text{(b)} & 101 \\ & \times 11 \\ \hline & 101 \\ & +101 \\ \hline & 1111 \end{array}$$

电子技术

Binary Division

Example: Perform the following binary divisions:

(a) 110÷11 (b) 110 ÷10

Solution:

$$\text{(a)}\ \begin{array}{r} 10 \\ 11 \overline{)110} \\ 11 \\ \hline 000 \end{array} \qquad \text{(b)}\ \begin{array}{r} 11 \\ 10 \overline{)110} \\ 10 \\ \hline 10 \\ 10 \\ \hline 00 \end{array}$$

Signed Numbers

The sign bit （符号位）: the left-most bit in a signed binary number

0 : positive

1 : negative

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

Sign-Magnitude System

+25: 00011001

-25: 10011001

↑ Sign bit ↑ Magnitude bits

Complement（补码） of a negative binary number

- 1's complement of binary numbers

(First changing the sign bit from 1 to 0)

Changing all the 1s to 0s and 0s to 1s.

e.g. 1 0 0 1 0 1 0 1 → Binary number

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

1 1 1 0 1 0 1 0 → 1's complement

（反码：除符号位求反）

2's complement of binary numbers

(2's complement) = (1's complement)+ 1

Example: Find the 2's complement of 10000010 (-2).

Solution:

10000010	→ Binary number
11111101	→ 1's complement
+ 1	→ Add 1
11111110	→ 2's complemt

Complement of a positive binary number

Complement of a positive binary number is equal to the true binary number

Range of 1's complement:

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

Range of 2's complement:

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

Ex. 1000: -8 (-0)

2.4 Arithmetic operations with signed number

Addition: the two numbers in an addition are the addend and augend. The result is the sum.

Four cases can occur:

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

1)Both numbers positive:

addend>0, augend>0

00000111	7
+00000100	+ 4
00001011	11

2) Positive number with magnitude larger than negative number:

addend>0,augend<0, |addend|>|augend|

00001111		15
+11111010	←10000110	+ -6
Discard carry → 100001001		9

3) Negative number with magnitude larger than positive number:

addend>0,augend<0, |addend|<|augend|

```
10010110→   00010000        16
           +11101000       +-24
           ---------       ----
            11111000         -8
```

$\underbrace{11111000}_{\text{2's complement of 8}}$

8→00001000→11110111+1→11111000

1's complement 2's complement

4) Both negative numbers:

addend<0,augend<0

```
                11111011   ←10000101    -5
               +11110111   ←10001001  + -9
               ---------              ----
Discard carry →111110010               -14
```

$\underbrace{11110010}_{\text{2's complement}}$

Conclusion:

- **If the sum>0, the result is in true binary.**

  In this case, the 2's complement form is the same of the true binary.
- **If the sum<0, the result is in 2's complement form.**

Note:

Overflow Condition: when two numbers are added and the sum should be represent by more bits than bits in the two numbers, it results an incorrect sign bit. An overflow can occur only when two numbers are both positive or negative.

```
 01111101
+00111010
---------
 10110111
```

Incorrect sign bit Incorrect magnitude

Alternative Method for Addition

Using 2's complement for addition.

$$[x+y]_{2c}=[x]_{2c}+[y]_{2c}$$

Steps:

1. Change the magnitude parts of two number to 2's complement form. Note that the 2's complement of positive number is itself.
2. Add two numbers. Discard the carry bit.

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 a subtraction is called the difference.

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

$$[x-y]_{2c}=[x+(-y)]_{2c}=[x]_{2c}+[-y]_{2c}$$

Example: Perform each of the following subtractions of the singed numbers:

(a)00001000-00000011

(b)00001100-11110111

(c)11100111-00010011

(d)10001000-11100010

Solution: There are four cases in this example

2018/9/9 97

By Bao Qilian

(a) 8-3=8+(-3)=5

```
                 00001000  ← Minuend
                +11111101  ← 2's complement of subtrahend
Discard carry → 100000101  ← Difference
```

(b) 12-(-9)=12+9=21

```
 00001100  ← Minuend
+00001001  ← 2's complement of subtrahend
 00010101  ← Difference
```

2018/9/9 98

By Bao Qilian

(c) -25-(+19)=-25+(-19)=-44

```
                 11100111  ← Minuend
                +11101101  ← 2's complement of subtrahend
Discard carry → 111010100  ← Difference(2's complement)
```

(d) -120-(-30)=-120+30=-90

```
 10001000  ← Minuend
+11101001  ← 2's complement of subtrahend
 10100110  ← Difference
```

2018/9/9 99

By Bao Qilian

Multiplication

Partial Product Method Steps:

1. Determine if the signs of the multiplicand and multiplier are the same or different. If same, the product is positive. If different, the product is negative.
2. Change any negative number to true 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.)
3. Starting with the LSB of multiplier, generate the partial product. Shift each successive partial product one bit to the left.
4. Add each successive partial product to the sum of the previous partial products to get the final product.
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.

2018/9/9 100

By Bao Qilian

Example: Multiply the signed binary number: 01010011 and 11000101(2's complement form).

Step1. The sign bits of the two numbers are different. So the product is negative.

Step2. Take the 2's complement of the multiplier to put it **in true form**.
e.g.11000101→10111011

Step3. The multiplication proceeds as follows. Notice that only the magnitude bits are used in these steps.

2018/9/9 101

By Bao Qilian

```
        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 sum
```

2018/9/9 102

By Bao Qilian

Division

When two binary numbers are divided, both numbers must be in true form.

Step1. Determine if the sings of the dividend and divisor are the same or different. If same, the quotient is positive, otherwise, the quotient is negative. Initialized the quotient is zero.

Step2. Subtract the divisor from the dividend using 2's complement addition to get the first partial remainder and add 1 to the quotient. If the partial remainder is positive, go to step3, else the division is complete.

Step3. 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.

Example. Divide 01100100 by 00011001.

Solution

Step1. The signs of the both numbers are positive. The quotient is initially: 00000000.

Step2. Subtract the divisor from the dividend using 2's complement addition (remember that final carries are discarded). Add 1 to quotient. 00000000+00000001=00000001

```
  01100100  ← Dividend
 +11100111  ← 2's complement of subtrahend
  01001011  ← Positive 1st partial remainder
```

Step3. Subtract the divisor from the 1st partial remainder using 2's complement addition.

```
  01001011  ← Dividend
 +11100111  ← 2's complement of subtrahend
  00110010  ← Positive 2nd partial remainder
```


Example. Divide 01100100 by 00011001.

Solution

```
quotient=0000000 →  01100100  ← Dividend
                   +11100111  ← 2's complement of divisor
quotient=0000001 →  01001011  ← Positive 1st partial remainder
                   +11100111  ← 2's complement of divisor
quotient=0000010 →  00110010  ← Positive 2nd partial remainder
                   +11100111  ← 2's complement of divisor
quotient=0000011 →  00110010  ← Positive 3rd partial remainder
                   +11100111  ← 2's complement of divisor
quotient=0000100 →  00000000  ← Zero remainder
```


Conlusion

Binary Addition
Binary Subtraction
Binary Multiplication
Binary Division

→

Addition
2's complement
Bit Shift

2.5 Binary Coded Decimal (BCD)

- BCD weighted codes (only represented numbers):
  8421, Ex-3, 7421, 5311,84-2-1,5421
- Self-complement BCD codes
  Ex-3,631-1,2421
  Unit Distance Code---Gray code
- Alphanumeric codes---ASCII (American Standard Code for information interchange) code, they are used to represent numbers, letters, symbols, and instructions.

8421 code

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

Binary	Decimal	8421	2421	5211	EX-3	Gray Code
0000	0	0000	0000	0000	0011	0000
0001	1	0001	0001	0001	0100	0001
0010	2	0010	0010	0100	0101	0011
0011	3	0011	0011	0101	0110	0010
0100	4	0100	0100	0111	0111	0110
0101	5	0101	1011	1000	1000	0111
0110	6	0110	1100	1001	1001	0101
0111	7	0111	1101	1100	1010	0100
1000	8	1000	1110	1101	1011	1100
1001	9	1001	1111	1111	1100	1101
		8421	2421	5211		

ASCII

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

The 8421 Code

- To express any decimal number in BCD, simply replace each decimal digit with the appropriate 4-bit code.
- [Example 2-33] Convert each of the following decimal numbers to BCD.
  (a) 35 (b) 98 (c) 170 (d) 2469
- To determine a decimal number from a BCD number, start at the right-most bit and break the code into groups of four bits, then write the decimal digit represented by each 4-bit group.
- [Example 2-34] Convert each of the following BCD codes to decimal:
  (a) 10000110 (b) 001101010001 (c) 1001010001110000

BCD Addition

- BCD is a numerical code and can be used in arithmetic operations. Here is how to add two BCD numbers:
  - Add the two BCD numbers, using the rules for binary addition in section 2.4.
  - If a 4-bit sum is equal to or less than 9, it is a valid BCD number.
  - 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 invalid codes and returned the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.
- [Example] Add the BCD numbers: 00010110+00010101.

Parity Method for Error Detection

Many systems use a parity bit （奇偶校验位）as a means for bit error detection. Any group of bits contain either an even or an odd number of 1s.A parity bit is attached to a group of bits to make the total number of 1s in a group always even or always odd.

Parity Method for Error Detection

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

Number of 1 is odd

电子技术

Summary

- Number systems
- Conversion in different number systems
- Binary Arithmetic
- Arithmetic for signed mubers
- Binary coded decimal

电子技术

Homework 2

Chapter2 Problems:

4, 6, 12, 16, 22,