Digital Systems Design and Laboratory [1. Number Systems and Conversion]

Chung-Wei Lin

cwlin@csie.ntu.edu.tw

CSIE Department

National Taiwan University

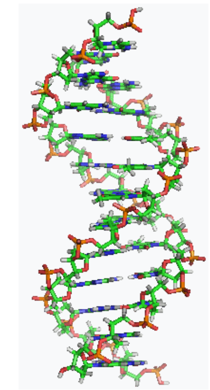
We would like to thank Prof. Hui-Ru Jiang (NTU) for contributing the lecture material

Outline

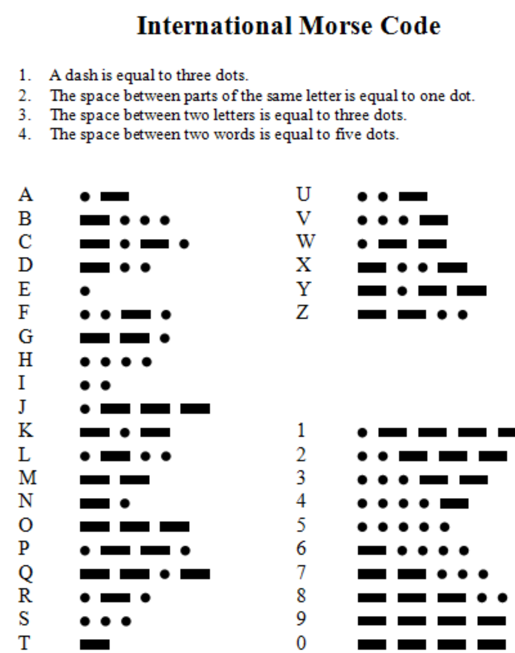
- **Digital Systems and Switching Circuits**
- Number Systems and Conversion
- Binary Arithmetic
- Representation of Negative Numbers
- Binary Codes

Historical Digital Systems

- ❑ Abacus
- ❑ Braille
- ❑ DNA
- ❑ Flag semaphore
- ❑ International maritime signal flags
- ❑ Morse code

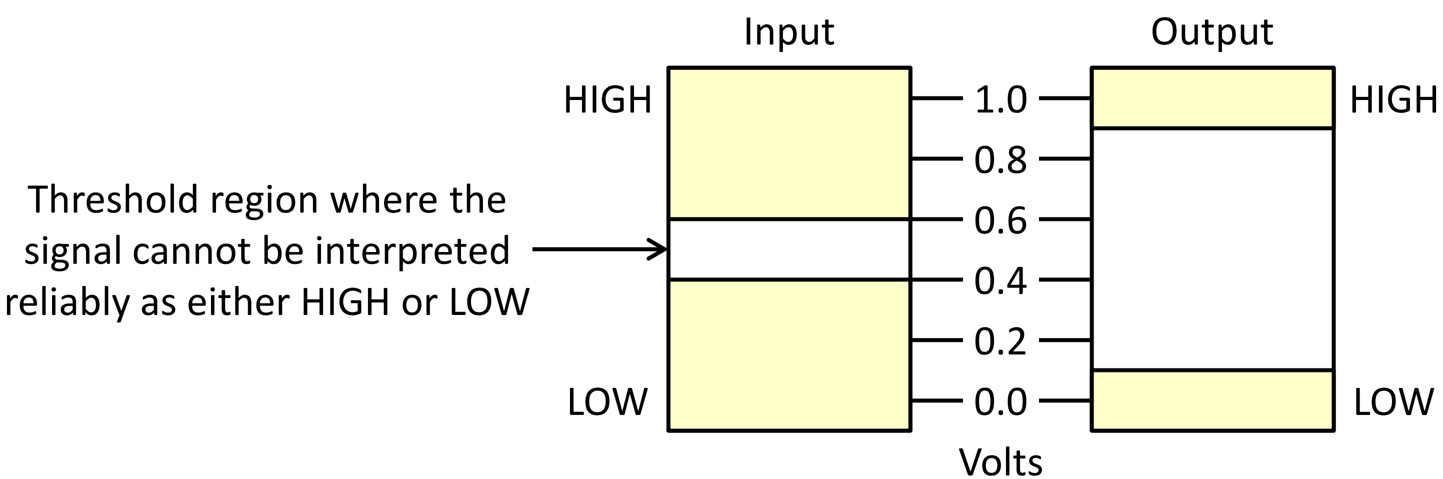


Source: Wikipedia



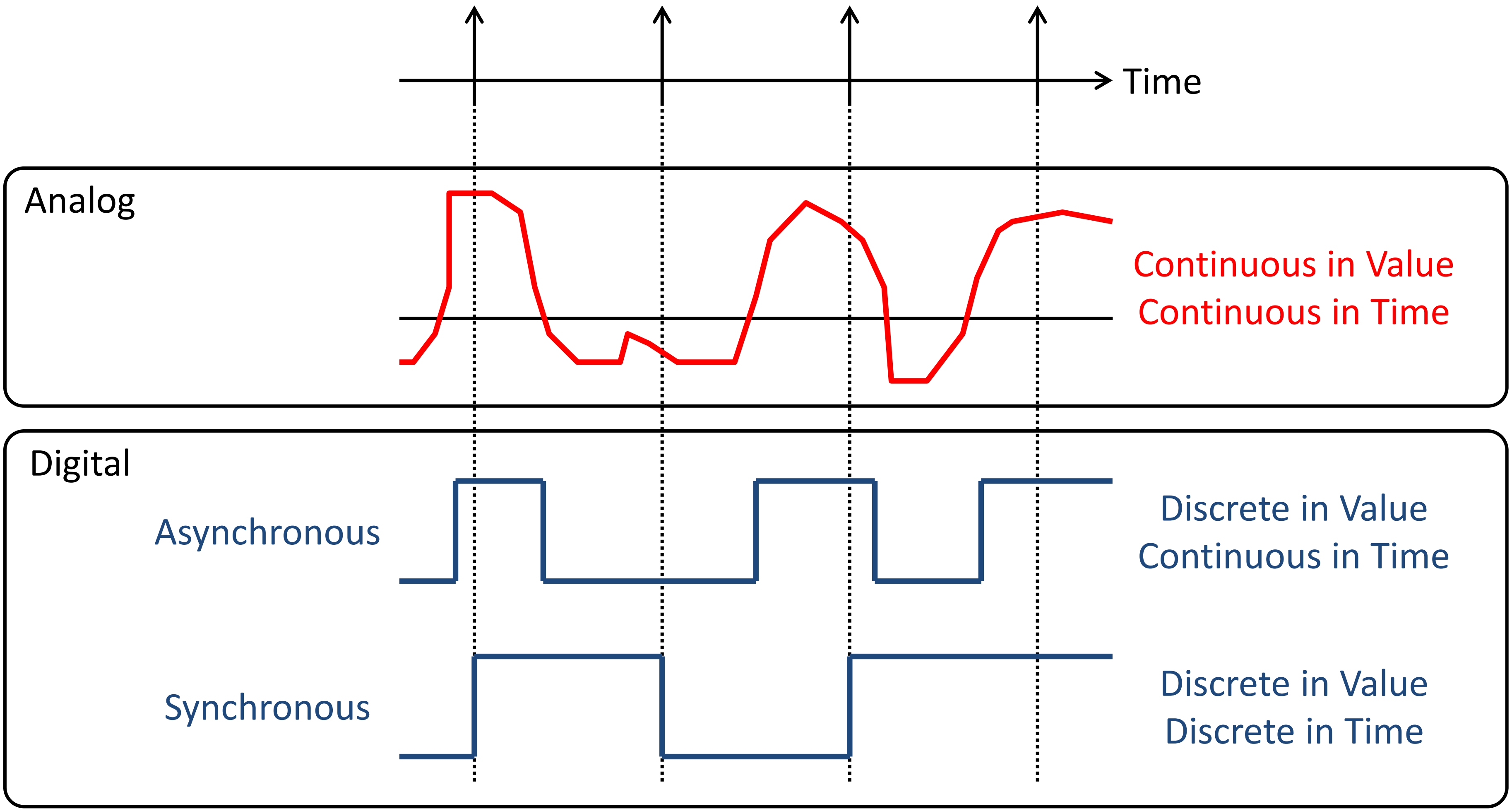
Digital vs. Analog

- The physical quantities or signals in
  - A digital system assumes only discrete values
    - Example: 0V and +1V
    - Greater accuracy and reliability (why?)



  - An analog system varies continuously over a specified range
    - Example: any value between 0V to +1V

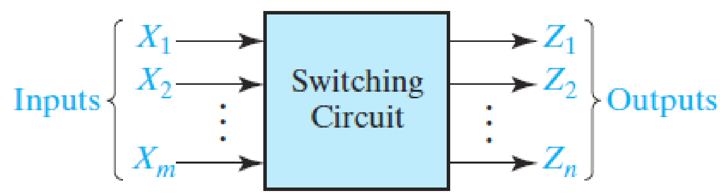
Signal Examples over Time



Digital Systems and Switching Circuits

- Subsystems of a digital system take the form of a switching circuit which has discrete inputs and outputs
  - Switching devices are generally **two-state** devices
    - i.e., output can assume only **two** different discrete values
  - It is natural to use **binary** numbers internally in digital systems

FIGURE 1-1
Switching Circuit



- Two types of switching circuits
  - Combinational circuits: outputs depend only on present inputs
    - Memoryless
  - Sequential circuits: outputs depend on both present and past inputs
    - In general, sequential circuits = combinational circuits + memory

Outline

- ❑ Digital Systems and Switching Circuits
- ❑ **Number Systems and Conversion**
- ❑ Binary Arithmetic
- ❑ Representation of Negative Numbers
- ❑ Binary Codes

Number Systems (1/2)

- Positional notation: each digit is multiplied by an appropriate power of base depending on its position in the number
  - The point separates the positive and negative powers of base
    - Example: decimal (base 10) numbers
      - $953.78_{10} = 9\text{x}10^{2} + 5\text{x}10^{1} + 3\text{x}10^{0} + 7\text{x}10^{-1} + 8\text{x}10^{-2}$
  - A positive number N with base R (positive integer, R>1):

    $N = (a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3})_R$

    $= a_4R^4+a_3R^3+a_2R^2+a_1R^1+a_0R^0+a_{-1}R^{-1}+a_{-2}R^{-2}+a_{-3}R^{-3}$

    - Base is also called radix
    - Base is indicated as subscript
  - Why do people use the decimal number system?

Number Systems (2/2)

Examples

- Decimal (base 10) numbers
  - $953.78_{10} = 9\text{x}10^{2} + 5\text{x}10^{1} + 3\text{x}10^{0} + 7\text{x}10^{-1} + 8\text{x}10^{-2}$
- Binary (base 2) numbers
  - $1011.11_{2} = 1\text{x}2^{3} + 0\text{x}2^{2} + 1\text{x}2^{1} + 1\text{x}2^{0} + 1\text{x}2^{-1} + 1\text{x}2^{-2}$
    $= 11.75_{10}$
- Octal (base 8) numbers
  - $147.3_{8} = 1\text{x}8^{2} + 4\text{x}8^{1} + 7\text{x}8^{0} + 3\text{x}8^{-1}$
    $= 103.375_{10}$
- Hexadecimal (base 16) numbers
  - Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - $\text{A2F}_{16} = 10\text{x}16^{2} + 2\text{x}16^{1} + 15\text{x}16^{0}$
    $= 2607_{10}$

Conversion of Decimal Integer

- Convert a decimal **integer** to base R using **division**
  - $N = (a_n a_{n-1} \ldots a_2 a_1 a_0)_R = a_n R^n + a_{n-1} R^{n-1} + \ldots + a_3 R^3 + a_2 R^2 + a_1 R^1 + a_0$
  - $N/R = a_n R^{n-1} + a_{n-1} R^{n-2} + \ldots + a_3 R^2 + a_2 R^1 + a_1 = Q_1$ rem. $= a_0$
  - $Q_1/R = a_n R^{n-2} + a_{n-1} R^{n-3} + \ldots + a_3 R^1 + a_2 = Q_2$ rem. $= a_1$
  - $Q_2/R = a_n R^{n-3} + a_{n-1} R^{n-4} + \ldots + a_3 = Q_3$ rem. $= a_2$
  - Continue until …
  - $Q_i/R = 0$ rem. $= a_n$
  - Example: convert **53_{10}** to binary

2	53		
2	26	…… remainder = 1 = a_0	(**LSB**)
2	13	…… remainder = 0 = a_1	
2	6	…… remainder = 1 = a_2	$\mathbf{53_{10} = 110101_2}$
2	3	…… remainder = 0 = a_3	
2	1	…… remainder = 1 = a_4	
	0	…… remainder = 1 = a_5	(**MSB**)

Conversion of Decimal Fraction (1/2)

- Convert a decimal **fraction** to base R using **multiplication**
  - $F = (.a_{-1}a_{-2}a_{-3}\ldots a_{-m})_R = a_{-1}R^{-1}+a_{-2}R^{-2}+a_{-3}R^{-3}+\ldots+a_{-m}R^{-m}$
  - $FR = a_{-1} + a_{-2}R^{-1} + a_{-3}R^{-2} + \ldots + a_{-m}R^{-m+1} = a_{-1} + F_1$
  - $F_1R = a_{-2} + a_{-3}R^{-1} + \ldots + a_{-m}R^{-m+2} = a_{-2} + F_2$
  - $F_2R = a_{-3} + \ldots + a_{-m}R^{-m+3} = a_{-3} + F_3$
  - Continue until $F_i = 0$ or … (next slide)
  - Example: convert **$.375_{10}$** to binary

```
         .375
   x        2
   ----------
     (0).750   a_-1 = 0  (MSB)
   x        2
   ----------
     (1).500   a_-2 = 1        .375_10 = .011_2
   x        2
   ----------
     (1).000   a_-3 = 1  (LSB)
```

$a_{-1} = 0$ (**MSB**), $a_{-2} = 1$, $a_{-3} = 1$ (**LSB**)

$\mathbf{.375_{10} = .011_2}$

Conversion of Decimal Fraction (2/2)

- Sometimes, the result is a repeating fraction
  - Example: convert $.7_{10}$ to binary

```
     .7
  x   2
 -------
 (1).4
  x   2
 -------
 (0).8
  x   2
 -------
 (1).6
  x   2
 -------
 (1).2
  x   2
 -------
 (0).4      Repeating
```

$$.7_{10} = .1\ \underline{0110}\ \underline{0110}\ \ldots_{2}$$

Conversion between Two Bases (1/2)

- Convert between two bases R_1 and R_2 other than decimal
  - Base R_1 → base 10 → base R_2
  - Example: convert **231.3_4** to base 7
    - $231.3_4 = 2x4^2 + 3x4^1 + 1x4^0 + 3x4^{-1} = 45.75_{10}$

```
7 / 45
7 /  6 ...... remainder = 3
     0 ...... remainder = 6
```

$45_{10} = 63_7$

```
     .75
  x    7
 -------
  (5).25
  x    7
 -------
  (1).75   Repeating
```

$.75_{10} = .\underline{51}_7$

- $45.75_{10} = \mathbf{63.\underline{51}_7}$

Conversion between Two Bases (2/2)

- Convert between binary and octal/hexadecimal by inspection
  - Start at the binary point
  - Divide bits into groups of three/four
    - **Add 0's if necessary**
  - Replace each group by an octal/hexadecimal digit
- Binary to octal
  - $1001101.010111_2 = \underline{001}\ \underline{001}\ \underline{101}\ .\ \underline{010}\ \underline{111}_2 = 115.27_8$
- Binary to hexadecimal
  - $1001101.010111_2 = \underline{0100}\ \underline{1101}\ .\ \underline{0101}\ \underline{1100}_2 = 4D.5C_{16}$

Outline

- ❑ Digital Systems and Switching Circuits
- ❑ Number Systems and Conversion
- ❑ **Binary Arithmetic**
- ❑ Representation of Negative Numbers
- ❑ Binary Codes

Addition

- **Addition table**
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 (and carry 1 to the next column)
- **Example: add 13_{10} and 11_{10} in binary**

```
                1111  <------ Carries
13₁₀  =          1101
11₁₀  =  +       1011
         ------------
                11000   =  24₁₀
```

$$\begin{array}{rcrl} & & \color{red}{1111} & \leftarrow \text{Carries} \\ 13_{10} & = & 1101 & \\ 11_{10} & = & +\quad 1011 & \\ \hline & & 11000 & = 24_{10} \end{array}$$

Subtraction

- Subtraction table
  - 0 – 0 = 0
  - 1 – 0 = 1
  - 1 – 1 = 0
  - 0 – 1 = 1 (and borrow 1 from the next column)
    - Borrow 1 from the next column = subtract 1 at the next column and add 2 at the current column
- Example: subtract 19_{10} and 29_{10} in binary

```
                    1 <—— Borrow
29₁₀  =         11101
19₁₀  =  –      10011
         ______________
                01010   =  10₁₀
```

Multiplication

- Multiplication table
  - 0 x 0 = 0
  - 0 x 1 = 0
  - 1 x 0 = 0
  - 1 x 1 = 1
- Example: multiply 13_{10} and 11_{10} in binary

$$
\begin{array}{lcr}
13_{10} & = & 1101 \\
11_{10} & = & \times\ \ 1011 \\
\hline
 & & 1101 \\
 & & 1101 \\
 & & 0000 \\
 & & 1101 \\
\hline
 & & 10001111 \quad = 143_{10}
\end{array}
$$

Division

- Similar to (but easier than) decimal division
- Example: divide 145_{10} and 11_{10} in binary

$$
\begin{array}{rrl}
 & 1101 & = 13_{10} \\ \cline{2-2}
11_{10} = 1011 & \big)\,10010001 & = 145_{10} \\
 & 1011 & \\ \cline{2-2}
 & 1110 & \\
 & 1011 & \\ \cline{2-2}
 & 1101 & \\
 & 1011 & \\ \cline{2-2}
 & 10 & = 2_{10}
\end{array}
$$

Outline

- Digital Systems and Switching Circuits
- Number Systems and Conversion
- Binary Arithmetic
- **Representation of Negative Numbers**
- Binary Codes

Negative Numbers

- n = word length = number of bits
- Sign and Magnitude (SM)
  - 1-bit sign + (n-1)-bit magnitude
    - Example: 3_{10} = 0011 and -3 = 1011
  - Common for people but awkward for computers
- 1's complement
  - Complement N bits, i.e., $\overline{N} = (2^n - 1) - N$
    - Example: 3 = 0011 and $\overline{3}$ = 1100
- 2's complement
  - Complement N bits and then add 1, i.e., $N^* = 2^n - N = \overline{N} + 1$
  - Or complement all bits from MSB to the left of the rightmost 1
    - Example: 3 = 0011 and 3* = 1101

Signed Binary Integers

TABLE 1-1
Signed Binary Integers (word length: $n = 4$)

$+N$	Positive Integers (all systems)	$-N$	Negative Integers: Sign and Magnitude	Negative Integers: 2's Complement N^*	Negative Integers: 1's Complement $\overline{N}$
+0	0000	−0	1000	—	1111
+1	0001	−1	1001	1111	1110
+2	0010	−2	1010	1110	1101
+3	0011	−3	1011	1101	1100
+4	0100	−4	1100	1100	1011
+5	0101	−5	1101	1011	1010
+6	0110	−6	1110	1010	1001
+7	0111	−7	1111	1001	1000
		−8	—	1000	—

- For word length n = 4, there are 2^4 different permutations
  - SM and $\overline{N}$: [-7, …, -0, +0, …,+7], i.e., $[-2^{n-1}+1, 2^{n-1}-1]$
  - N*: [-8, …, +0, …, +7], i.e., $[-2^{n-1}, 2^{n-1}-1]$
- Always view the first bit as the sign bit
- Exercise: what is 1110_2?

Addition of 2's Complement Numbers (1/2)

- Steps
  - Add just as if all numbers are positive
  - Ignore the carry, if any, from the sign bit
- Cases (assume A > 0, B > 0, and word length = n)
  - Case 1: A + B and $|A + B| < 2^{n-1}$ → Correct
  - Case 2: A + B and $|A + B| \geq 2^{n-1}$ → Wrong (overflow)
  - Case 3: A – B and A < B → Correct
  - Case 4: – A + B and A ≤ B → Correct (ignore the carry)
  - Case 5: – A – B and $|A + B| \leq 2^{n-1}$ → Correct (ignore the carry)
  - Case 6: – A – B and $|A + B| > 2^{n-1}$ → Wrong (overflow)

Case 1		Case 2		Case 3		Case 4		Case 5		Case 6	
+3	0011	+5	0101	+5	0101	−5	1011	−3	1101	−5	1011
+4	0100	+6	0110	−6	1010	+6	0110	−4	1100	−6	1010
+7	0111		1011	−1	1111	+1	(1) 0001	−7	(1) 1001		(1) 0101

Addition of 2's Complement Numbers (2/2)

- Why to ignore the carry, i.e., subtract 2^n ?
  - Add(-A, +B) where B > A
    - $A^* + B = (2^n - A) + B = 2^n + (B - A)$
  - Add(-A, -B) where $A + B \leq 2^{n-1}$
    - $A^* + B^* = (2^n - A) + (2^n - B) = 2^n + 2^n - (A + B) = 2^n + (A + B)^*$
- How to detect overflow?
  - Check the sign
    - (+) + (+) becomes (-)
    - (-) + (-) becomes (+)

Addition of 1's Complement Numbers

End-around carry

- Add just as if all numbers are positive
- Add the carry out back to the rightmost bit

Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
+3 0011 +4 0100 +7 0111	+5 0101 +6 0110 1011	+5 0101 -6 1001 -1 1110	-5 1010 +6 0110 +1 (1) 0000 1 0001	-3 1100 -4 1011 -7 (1) 0111 1 1000	-5 1010 -6 1001 (1) 0011 1 0100

How to detect overflow?

- Check the sign
  - (+) + (+) becomes (-)
  - (-) + (-) becomes (+)

Outline

- Digital Systems and Switching Circuits
- Number Systems and Conversion
- Binary Arithmetic
- Representation of Negative Numbers
- **Binary Codes**

Decimal Digits to Binary Codes

- Input/output interface generally uses decimal digits
  - How to code decimal digits using binary codes?
  - Choose 10 elements from 16 binary numbers of 4 bits
  - Binary-Coded-Decimal (BCD)
    - Example: 9**3**7.**2**5 → 1001 **0011** 0111 . **0010** 0101

For error checking

For analog quantity

(BCD+3)

TABLE 1-2
Binary Codes for Decimal Digits

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

Only 1 bit difference for two successive digits

Warning: Conversion or Coding?

- Do NOT mix up
  - Conversion of a decimal number to a binary number
  - Coding a decimal digit with a binary code
- Example
  - Conversion: $13_{10} = 1101_2$
  - Coding: `13 = 0001 0011`

Text to Binary Codes

- ASCII
  - American Standard Code for Information Interchange
  - Developed from telegraph code
  - English alphanumeric symbols
  - 7 bits
  - 94 printable characters are numbered 32_{10} to 126_{10}
- Unicode
  - https://en.wikipedia.org/wiki/Unicode
- UTF-8
  - https://en.wikipedia.org/wiki/UTF-8
- Big-5
  - Traditional Chinese characters
  - https://en.wikipedia.org/wiki/Big5

TABLE 1-3 ASCII Code

	ASCII Code								ASCII Code								ASCII Code						
Character	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Character	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Character	A_6	A_5	A_4	A_3	A_2	A_1	A_0
space	0	1	0	0	0	0	0	@	1	0	0	0	0	0	0	'	1	1	0	0	0	0	0
!	0	1	0	0	0	0	1	A	1	0	0	0	0	0	1	a	1	1	0	0	0	0	1
"	0	1	0	0	0	1	0	B	1	0	0	0	0	1	0	b	1	1	0	0	0	1	0
#	0	1	0	0	0	1	1	C	1	0	0	0	0	1	1	c	1	1	0	0	0	1	1
$	0	1	0	0	1	0	0	D	1	0	0	0	1	0	0	d	1	1	0	0	1	0	0
%	0	1	0	0	1	0	1	E	1	0	0	0	1	0	1	e	1	1	0	0	1	0	1
&	0	1	0	0	1	1	0	F	1	0	0	0	1	1	0	f	1	1	0	0	1	1	0
'	0	1	0	0	1	1	1	G	1	0	0	0	1	1	1	g	1	1	0	0	1	1	1
(	0	1	0	1	0	0	0	H	1	0	0	1	0	0	0	h	1	1	0	1	0	0	0
)	0	1	0	1	0	0	1	I	1	0	0	1	0	0	1	i	1	1	0	1	0	0	1
*	0	1	0	1	0	1	0	J	1	0	0	1	0	1	0	j	1	1	0	1	0	1	0
+	0	1	0	1	0	1	1	K	1	0	0	1	0	1	1	k	1	1	0	1	0	1	1
,	0	1	0	1	1	0	0	L	1	0	0	1	1	0	0	l	1	1	0	1	1	0	0
–	0	1	0	1	1	0	1	M	1	0	0	1	1	0	1	m	1	1	0	1	1	0	1
.	0	1	0	1	1	1	0	N	1	0	0	1	1	1	0	n	1	1	0	1	1	1	0
/	0	1	0	1	1	1	1	O	1	0	0	1	1	1	1	o	1	1	0	1	1	1	1
0	0	1	1	0	0	0	0	P	1	0	1	0	0	0	0	p	1	1	1	0	0	0	0
1	0	1	1	0	0	0	1	Q	1	0	1	0	0	0	1	q	1	1	1	0	0	0	1
2	0	1	1	0	0	1	0	R	1	0	1	0	0	1	0	r	1	1	1	0	0	1	0
3	0	1	1	0	0	1	1	S	1	0	1	0	0	1	1	s	1	1	1	0	0	1	1
4	0	1	1	0	1	0	0	T	1	0	1	0	1	0	0	t	1	1	1	0	1	0	0
5	0	1	1	0	1	0	1	U	1	0	1	0	1	0	1	u	1	1	1	0	1	0	1
6	0	1	1	0	1	1	0	V	1	0	1	0	1	1	0	v	1	1	1	0	1	1	0
7	0	1	1	0	1	1	1	W	1	0	1	0	1	1	1	w	1	1	1	0	1	1	1
8	0	1	1	1	0	0	0	X	1	0	1	1	0	0	0	x	1	1	1	1	0	0	0
9	0	1	1	1	0	0	1	Y	1	0	1	1	0	0	1	y	1	1	1	1	0	0	1
:	0	1	1	1	0	1	0	Z	1	0	1	1	0	1	0	z	1	1	1	1	0	1	0
;	0	1	1	1	0	1	1	[	1	0	1	1	0	1	1	{	1	1	1	1	0	1	1
<	0	1	1	1	1	0	0	\	1	0	1	1	1	0	0	|	1	1	1	1	1	0	0
=	0	1	1	1	1	0	1	]	1	0	1	1	1	0	1	}	1	1	1	1	1	0	1
>	0	1	1	1	1	1	0	^	1	0	1	1	1	1	0	~	1	1	1	1	1	1	0
?	0	1	1	1	1	1	1	—	1	0	1	1	1	1	1	delete	1	1	1	1	1	1	1

Q&A