

2022年 秋季学期

### 几种常用的十进制代码

十进制数	8421码	余3码	2421码	5211码	余3循环码
0	0000	0011	0000	0000	0010
1	0001	0100	0001	0001	0110
2	0010	0101	0010	0100	0111
3	0011	0110	0011	0101	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1010	1101	1100	1111
8	1000	1011	1110	1101	1110
9	1001	1100	1111	1111	1010

wang\_hong@tsinghua.edu.cn 清华大学

1

2022年 秋季学期

### 格雷码

每一位的状态变化都按一定的顺序循环。

编码顺序依次变化，按表中顺序变化时，相邻代码只有一位改变状态。

编码顺序	二进制	格雷码	编码顺序	二进制	格雷码
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

wang\_hong@tsinghua.edu.cn 清华大学

2

2022年 秋季学期

### When choices aren't equally probable 变长编码

When the choices have different probabilities ( $p_i$ ), you get more information when learning of a unlikely choice than when learning of a likely choice

$\log_2(N/M)$  bits

Information from choice  $i = \log_2(1/p_i)$  bits  
 Average information from a choice =  $\sum (p_i) \log_2(1/p_i)$

Example

choice <sub>i</sub>	$p_i$	$\log_2(1/p_i)$
"A"	1/3	1.58 bits
"B"	1/2	1 bit
"C"	1/12	3.58 bits
"D"	1/12	3.58 bits

Average information  
 $= (.333)(1.58) + (.5)(1)$   
 $+ (2)(.083)(3.58)$   
 $= 1.626$  bits

Can we find an encoding where transmitting 1000 choices is close to 1626 bits on the average? Using two bits for each choice = 2000 bits

wang\_hong@tsinghua.edu.cn 清华大学

3

2022年 秋季学期

### Variable-length encodings

(David Huffman, MIT 1950)

Use shorter bit sequences for high probability choices, longer sequences for less probable choices

choice <sub>i</sub>	$p_i$	encoding
"A"	1/3	11
"B"	1/2	0
"C"	1/12	100
"D"	1/12	101

B C A B A D  
010011011101

Average information  
 $= (.333)(2) + (.5)(1) + (2)(.083)(3)$   
 $= 1.666$  bits

Transmitting 1000 choices takes an average of 1666 bits... better but not optimal

Huffman Decoding Tree

To get a more efficient encoding (closer to information content) we need to encode **sequences of choices**, not just each choice individually. This is the approach taken by most file compression algorithms...

wang\_hong@tsinghua.edu.cn 清华大学

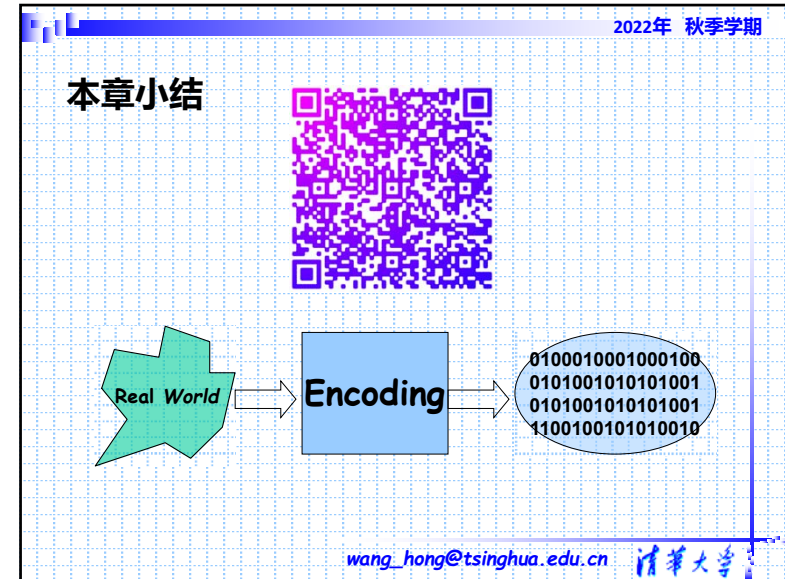
4

2022年 秋季学期

A 8.19	B 1.47	C 3.83	D 3.91
E 12.25	F 2.26	G 1.71	H 4.57
I 7.10	J 0.14	K 0.41	L 3.77
M 3.34	N 7.06	O 7.26	P 2.89
Q 0.09	R 6.85	S 6.36	T 9.41
U 2.58	V 1.09	W 1.59	X 0.21
Y 1.58	Z 0.08		

wang\_hong@tsinghua.edu.cn 清华大学

5



6

2022年 秋季学期

基本逻辑运算  
基本公式, 表示方法  
化简

## 第二章 逻辑代数基础

wang\_hong@tsinghua.edu.cn 清华大学

7

2022年 秋季学期

### George Boole, 1815-1864


- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen's College in Ireland
- Wrote *An Investigation of the Laws of Thought* (1854)
- Introduced binary variables
- Introduced the three fundamental logic operations: AND, OR, and NOT

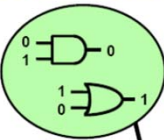

Scanned at the American Institute of Physics

wang\_hong@tsinghua.edu.cn 清华大学

8

2022年 秋季学期



(The Vacuum Tube)

Lee de Forest, 1906

**Digital Electronics**

- Despite existence of **relays** and introduction of **vacuum tube** in 1906, digital electronics did not emerge for thirty years!
- Claude Shannon notices similarities between Boolean algebra and electronic telephone switches
- Shannon's 1937 MIT Master's Thesis introduces the world to **binary digital electronics**

wang\_hong@tsinghua.edu.cn 清华大学

9

2022年 秋季学期

## 逻辑运算

当二进制代码表示不同逻辑状态时，可以按一定的规则进行推理运算。

wang\_hong@tsinghua.edu.cn 清华大学

10

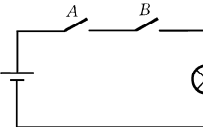
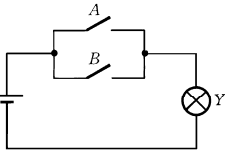
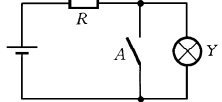
2022年 秋季学期

## 一、三种基本运算

与 (AND)

或 (OR)

非 (NOT)

(a)

(b)

(c)

以A=1表示开关A合上，A=0表示开关A断开；  
以Y=1表示灯亮，Y=0表示灯不亮；  
三种电路的因果关系不同：

wang\_hong@tsinghua.edu.cn 清华大学

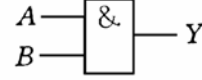
11

2022年 秋季学期


## 与

- 条件同时具备，结果发生
- $Y = A \text{ AND } B = A \& B = A \cdot B = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



与



wang\_hong@tsinghua.edu.cn 清华大学

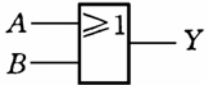
12

2022年 秋季学期


### 或

- 条件之一具备，结果发生
- $Y = A \text{ OR } B = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



或



wang\_hong@tsinghua.edu.cn 清华大学

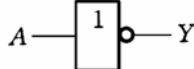
13

2022年 秋季学期


### 非

- 条件不具备，结果发生
- $Y = A' = \text{NOT } A$

A	Y
0	1
1	0



非




wang\_hong@tsinghua.edu.cn 清华大学

14

2022年 秋季学期


### 几种常用的复合逻辑运算

- 与非
- 或非
- 与或非



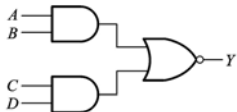
与非

 $Y = (A \cdot B)'$



或非

 $Y = (A + B)'$



与或非

 $Y = (A \cdot B + C \cdot D)'$

wang\_hong@tsinghua.edu.cn 清华大学

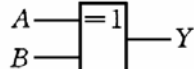
15

2022年 秋季学期


### 几种常用的复合逻辑运算

- 异或
- $Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



异或



wang\_hong@tsinghua.edu.cn 清华大学

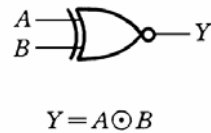
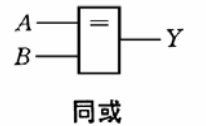
16

## 几种常用的复合逻辑运算

## • 同或

$$Y = A \odot B = (A \oplus B)'$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



## 二、逻辑代数的基本公式和常用公式

## 基本公式

## 常用公式

基本逻辑运算：与，或，非  
基本公式：  
基本定理：  
表示方法：  
化简：

## 基本公式

## 证明方法：推演 真值表

- 根据与、或、非的定义，得表2.3.1的布尔恒等式

序号	公 式	序号	公 式
		10	$1' = 0; 0' = 1$
1	$0 A = 0$	11	$1 + A = 1$
2	$1 A = A$	12	$0 + A = A$
3	$A A = A$	13	$A + A = A$
4	$A A' = 0$	14	$A + A' = 1$
5	$A B = B A$	15	$A + B = B + A$
6	$A (B C) = (A B) C$	16	$A + (B + C) = (A + B) + C$
7	$A (B + C) = A B + A C$	17	$A + B C = (A + B)(A + C)$
8	$(A B)' = A' + B'$	18	$(A + B)' = A' B'$
9	$(A')' = A$		

公式 (17)  $A + B C = (A + B)(A + C)$  (真值表法) :

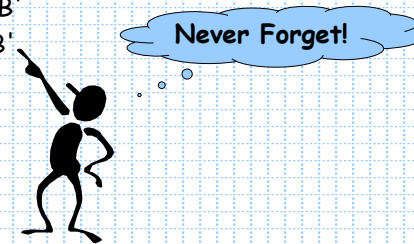
A	B	C	BC	A+BC	A+B	A+C	(A+B)(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

公式 (17)  $A + BC = (A+B)(A+C)$  的证明:

$$\begin{aligned}
 \text{右} &= (A+B)(A+C) \\
 &= A + AB + AC + BC \\
 &= A(1+B+C) + BC \\
 &= A + BC = \text{左}
 \end{aligned}$$

德·摩根定理 (公式8, 18)

$$\begin{aligned}
 (A B)' &= A' + B' \\
 (A+B)' &= A' B'
 \end{aligned}$$



若干常用公式

序 号	公 式
21	$A + A B = A$
22	$A + A' B = A + B$
23	$A B + A B' = A$
24	$A (A + B) = A$
25	$A B + A' C + B C = A B + A' C$ $A B + A' C + B C D = A B + A' C$
26	$A (AB)' = A B' ; A' (AB)' = A'$

三、逻辑代数的基本定理

- 代入定理

----在任何一个包含A的逻辑等式中, 若以另外一个逻辑式代入式中A的位置, 则等式依然成立。

## 代入定理

- 应用举例：

公式17

$$\begin{aligned}
 A+BC &= (A+B)(A+C) \\
 A+B(CD) &= (A+B)(A+CD) \\
 &= (A+B)(A+C)(A+D)
 \end{aligned}$$

## 代入定理

- 应用举例：  
式 (8)

$$(A \cdot B)' = A' + B'$$

以  $B \cdot C$  代入  $B$ 

$$\begin{aligned}
 (A \cdot B \cdot C)' &= A' + (BC)' \\
 &= A' + B' + C'
 \end{aligned}$$

## 三、逻辑代数的基本定理

- 反演定理

-----对任一逻辑式

$$Y \Rightarrow Y'$$

变换顺序 先括号，  
然后与，最后或

$$\bullet \Rightarrow +, + \Rightarrow \bullet, 0 \Rightarrow 1, 1 \Rightarrow 0,$$

原变量  $\Rightarrow$  反变量反变量  $\Rightarrow$  原变量基本公式8,18  
摩根定理不属于单个变量的  
上的反号保留不变

## 反演定理

- 应用举例：

$$Y = A(B+C) + CD$$

$$Y' = (A' + B'C')(C' + D')$$

$$= A'C' + B'C' + A'D' + B'C'D'$$

### 三、逻辑代数的基本定理

- 对偶定理

-----若两个逻辑式相等，则对偶式也成立

$$Y \Rightarrow Y^D$$

$$0 \rightarrow 1; 1 \rightarrow 0$$

$$+ \rightarrow \cdot; \cdot \rightarrow +$$

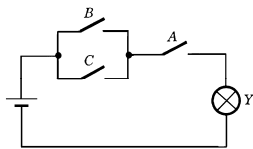
### 四、逻辑函数及其表示方法

逻辑函数  $Y=F(A,B,C,\dots)$

----若以逻辑变量为输入，运算结果为输出，则输入变量值确定以后，输出的取值也随之而定。输入/输出之间是一种函数关系。

注：在二值逻辑中，输入/输出都只有两种取值0/1。

#### 举例：一家三口的投票机制



$$Y = A \cdot (B + C)$$



A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

#### 逻辑函数的表示方法

- 真值表
- 逻辑式
- 逻辑图
- 波形图
- 卡诺图
- EDA中 硬件描述语言

各种表示方法之间可以相互转换



2022年 秋季学期

- 真值表——厘清思路

输入变量				输出	
A	B	C	...	$Y_1$	$Y_2$ ...
遍历所有可能的输入变量的取值组合				输出对应的取值	

输出之间有无相关性?

真值表的长度与宽度?

wang\_hong@tsinghua.edu.cn 清华大学

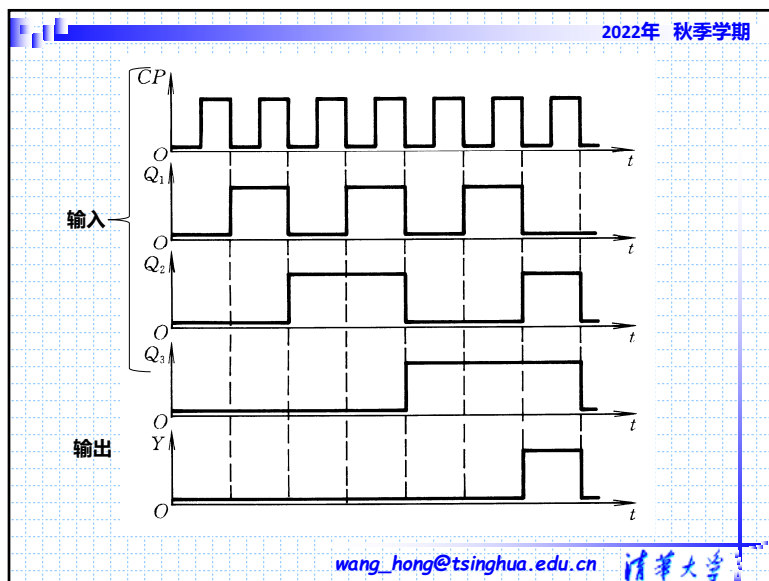
33

2022年 秋季学期

- 逻辑式——简洁  
将输入/输出之间的逻辑关系用 **与/或/非** 的运算式表示就得到逻辑式。
- 逻辑图——电路连接  
用逻辑图形符号表示逻辑运算关系，与逻辑电路的实现相对应。
- 波形图——实验室测试  
将输入变量所有取值可能与对应输出按时间顺序排列起来画成时间波形。

wang\_hong@tsinghua.edu.cn 清华大学

34



35

2022年 秋季学期

- 卡诺图——直观
- EDA中的描述方式
  - HDL (Hardware Description Language)
    - VHDL (Very High Speed Integrated Circuit ...)
    - Verilog HDL
  - EDIF
  - DTIF
  - ...

wang\_hong@tsinghua.edu.cn 清华大学

36

2022年 秋季学期

- **Hardware description language (HDL):**
  - specifies logic function only
  - Computer-aided design (CAD) tool produces or synthesizes the optimized gates
- Most commercial designs built using HDLs
- Two leading HDLs:
  - Verilog
    - developed in 1984 by Gateway Design Automation
    - IEEE standard (1364) in 1995
    - Extended in 2005 (IEEE STD 1800-2017)
  - VHDL
    - Developed in 1981 by the Department of Defense
    - IEEE standard (1076) in 1987
    - Updated in 2008 (IEEE STD 1076-2017)

wang\_hong@tsinghua.edu.cn 清华大学

37

2022年 秋季学期

## HDL to Gates

- **Simulation**
  - Inputs applied to circuit
  - Outputs checked for correctness
  - Millions of dollars saved by debugging in simulation instead of hardware
- **Synthesis**
  - Transforms HDL code into a *netlist* describing the hardware (i.e., a list of gates and the wires connecting them)

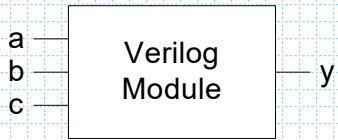
**IMPORTANT:**  
When using an HDL, think of the hardware the HDL should produce

wang\_hong@tsinghua.edu.cn 清华大学

38

2022年 秋季学期

## Verilog Modules



```

graph LR
    a --> VM[Verilog Module]
    b --> VM
    c --> VM
    VM --> y
  
```

- Two types of Modules:
  - **Behavioral:** describe what a module does
  - **Structural:** describe how it is built from simpler modules

wang\_hong@tsinghua.edu.cn 清华大学

39

2022年 秋季学期

## Verilog

```

module example(input logic a, b, c,
               output logic y);
  assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;
endmodule
  
```

- module/endmodule: required to begin/end module
- example: name of the module
- Operators:
  - ~: NOT
  - &: AND
  - |: OR

wang\_hong@tsinghua.edu.cn 清华大学

40

2022年 秋季学期

## Simulation

### Verilog:

```

module example(input  logic a, b, c,
               output logic y);
    assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;
endmodule

```

Now: 800 ns

Signal	Value
a	0
b	0
c	0
y	0

wang\_hong@tsinghua.edu.cn 清华大学

41

2022年 秋季学期

## Verilog:

```

module example(input  logic a, b, c,
               output logic y);
    assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;
endmodule

```

### Synthesis:

wang\_hong@tsinghua.edu.cn 清华大学

42