

数字逻辑

第一章 数字系统

北京理工大学

计算机学院

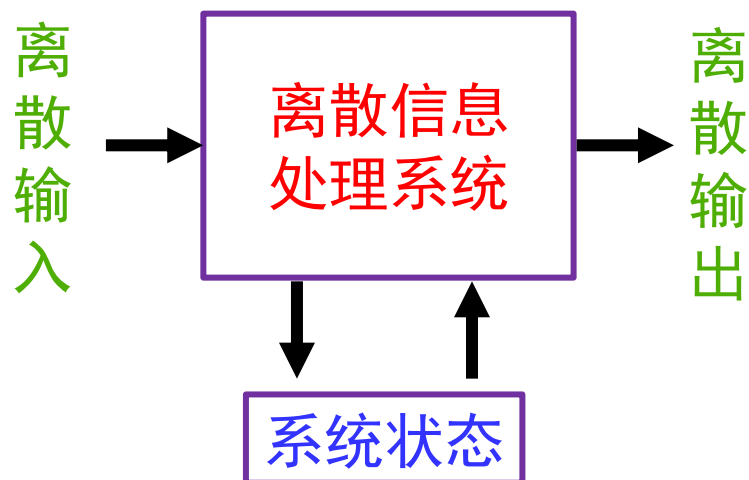
黄永刚

一. 数字系统

- 1. 定义
- 2. 分类
- 3. 典型系统

1. 定义

- 什么是数字系统?
- 是一种离散信息处理系统
 - 一组离散形式的信息作为输入
 - 离散的内部信息作为系统状态
 - 产生离散形式的系统输出



2. 分类

□ 根据是否有系统状态

□ 组合逻辑系统

➤ 无状态记忆功能

➤ 输出值只与当时系统输入有关

$$\text{output} = f(\text{input})$$

□ 时序逻辑系统

➤ 具备状态记忆功能

➤ 输出值与当时系统输入和状态都有关

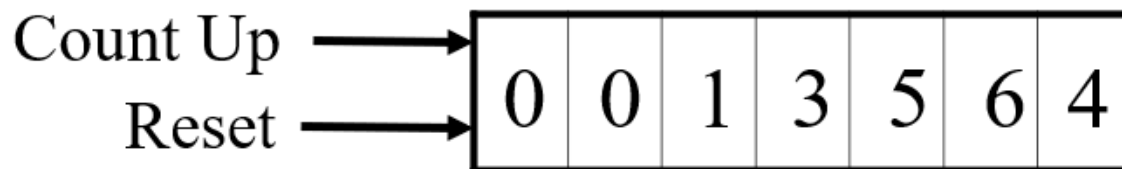
$$\text{output} = f(\text{state}, \text{input})$$

➤ 在离散时间点更新状态 > > 同步时序系统

➤ 在任何时间点更新状态 > > 异步时序系统

3 典型系统

□ 数字计数器



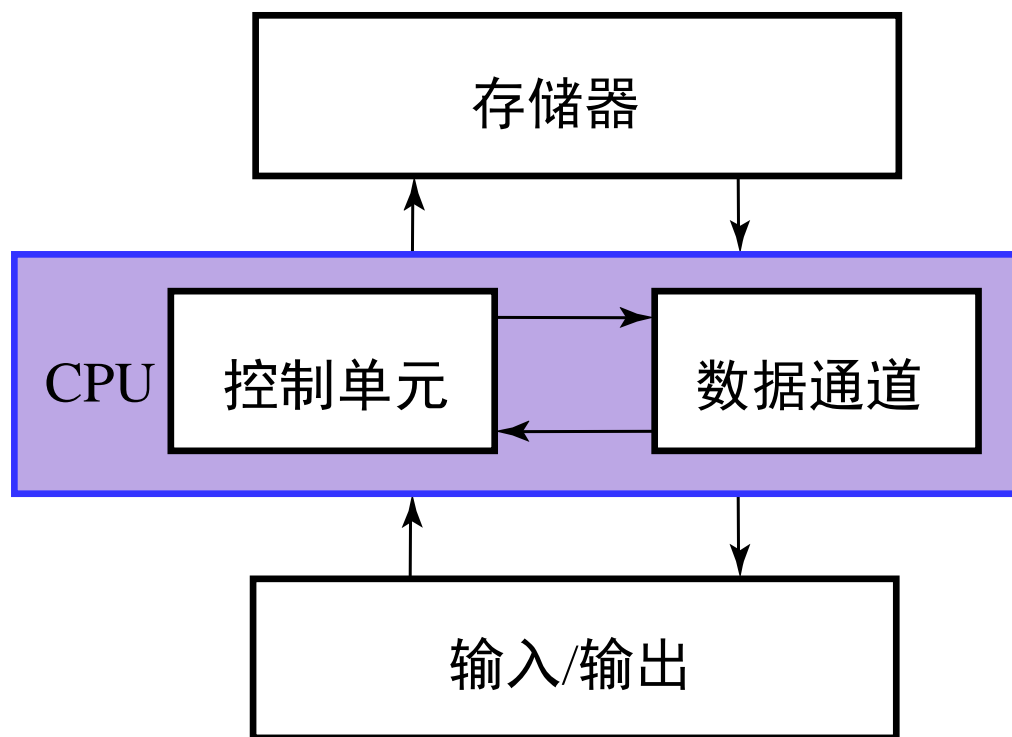
□ 输入：Count Up 、 Reset

□ 输出：显示

□ 状态：存储数字值

3. 典型系统

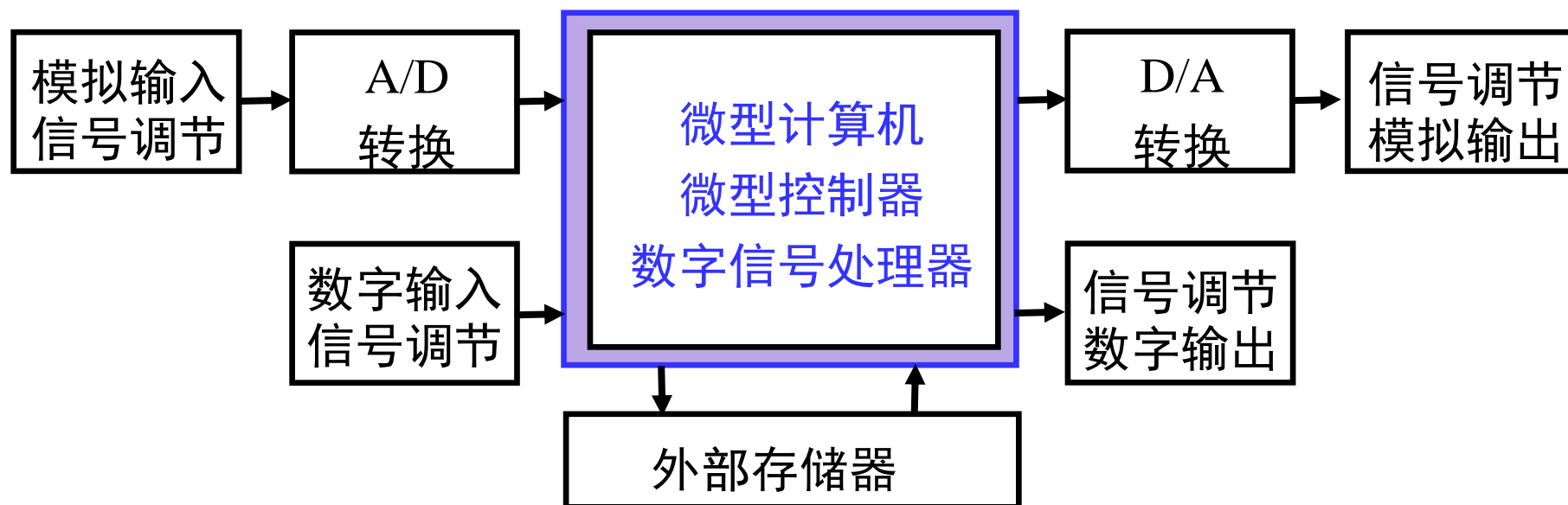
- 数字计算机
- 输入：键盘、鼠标、麦克风
- 输出：屏幕、扬声器



3. 典型系统

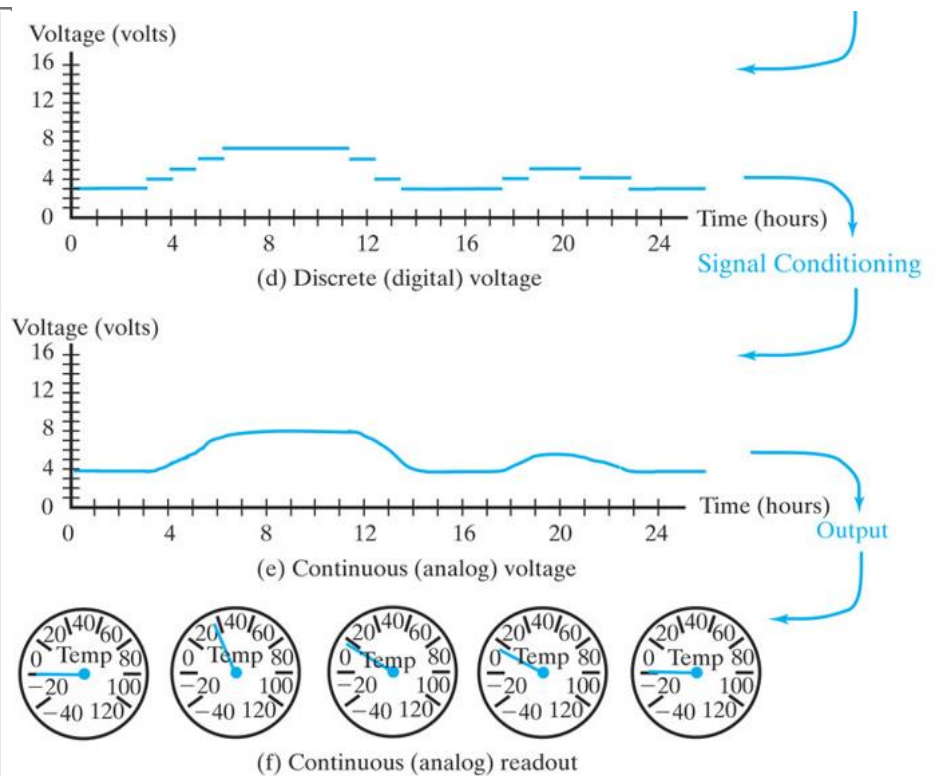
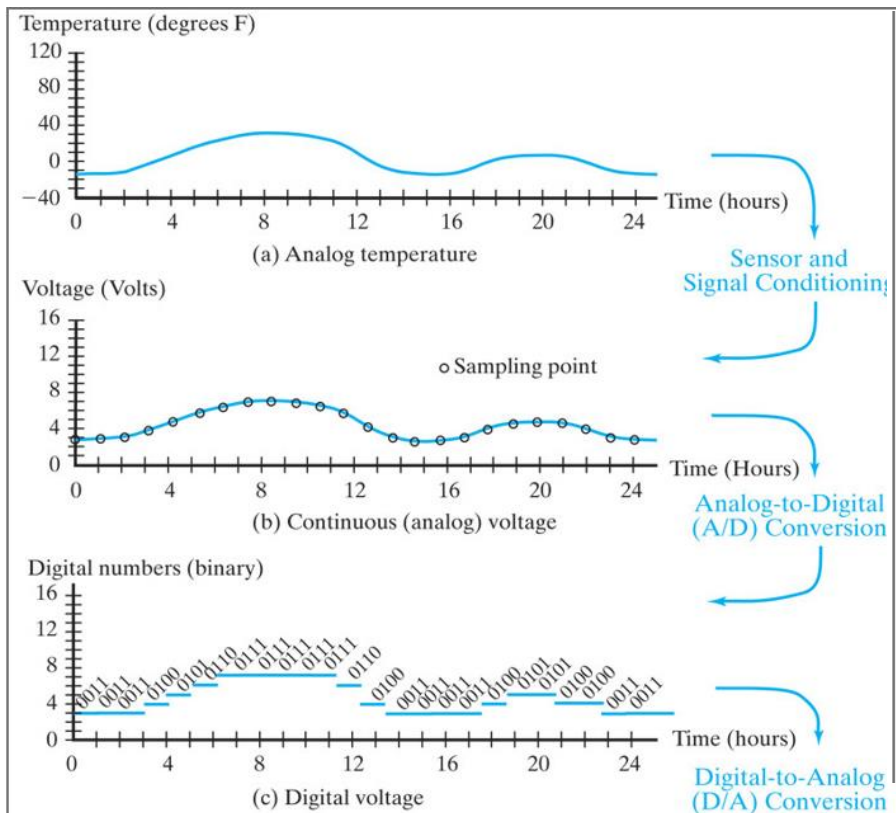
□ **嵌入式系统：** 计算机作为其他系统的内部部件

- 微型计算机
- 微型控制器
- 数字信号处理器



3. 典型系统

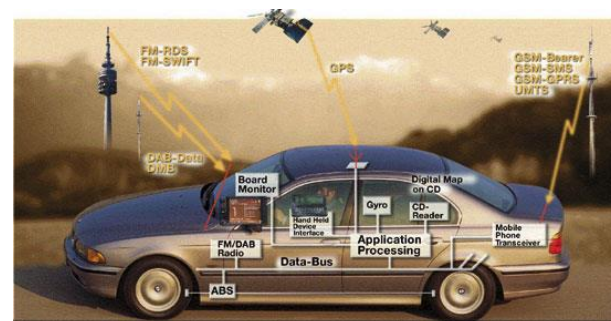
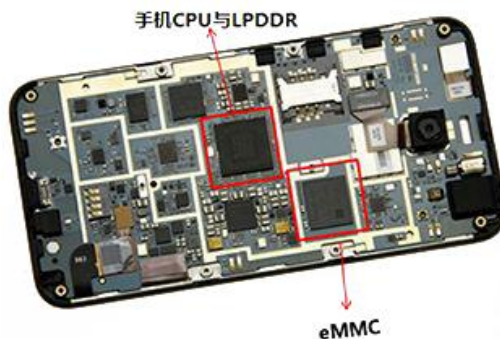
□ 嵌入式系统例子：测温



3. 典型系统

□ 其他例子

- 手机
- 相机
- 汽车
- 复印机
- 洗碗机
- GPS



二. 信息表示

- 1. 定义
- 2. 二进制表示

1. 定义

□ 信息

- 对物质世界与人类社会中存在现象的表示
- 可以消除不确定性
- 信息熵
- 熵增定律

□ 信号

- 信息表示的物理载体
- 模拟信号：连续的物理量
- 数字信号：离散的物理量

2. 二进制表示

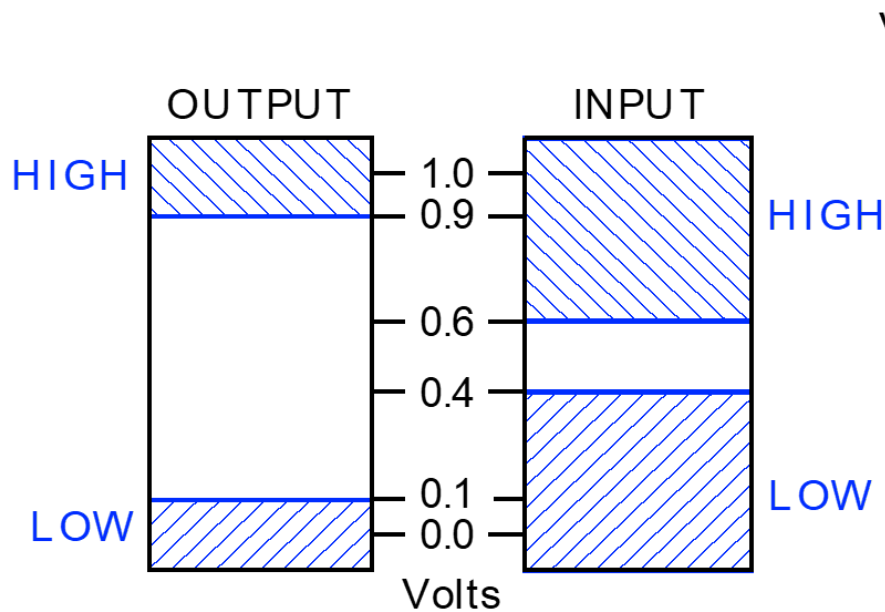
- 数字系统广泛采用两个离散值，称**二进制**
 - 数字**0和1**
 - 符号**真(T)和假(F)**
 - 符号**高(H)和低(L)**
 - 符号**开(On)和关(Off)**



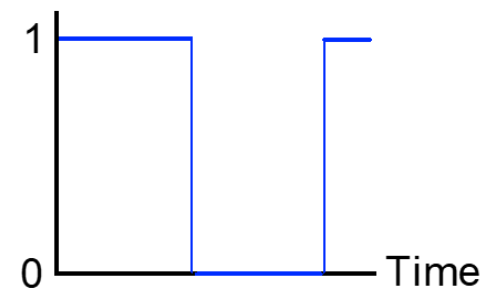
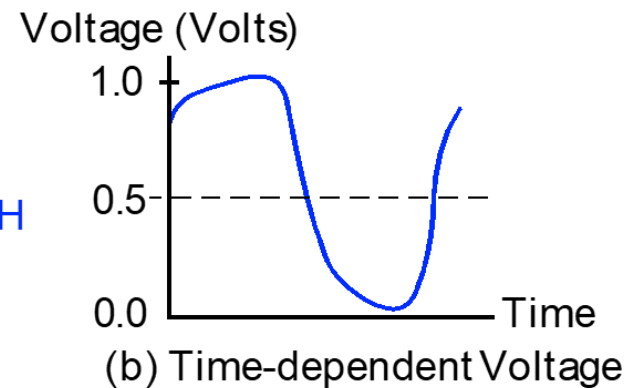
2. 二进制表示

□ 可以用不同物理量来实现二进制

➤ 电压、磁场方向、电荷量



(a) Example voltage ranges



(c) Binary model of time-dependent voltage

三. 数制

- 1. 定义
- 2. 运算
- 3. 转换

1. 定义

- 数的表示规则称为**数制**
- **基底(r)**: 一个数制所包含的数字符号的个数

二进制	0, 1
八进制	0, 1, 2, 3, 4, 5, 6, 7
十进制	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
十六进制	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- **权(r^i)**: 数字符号的**位置决定的值**
- **值**是各位**数字值与其权之积的总和**

$$(101.01)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$

1. 定义

- 二进制数: 1001B $(1001)_2$
- 八进制数: 1001O(Q) $(1001)_8$
- 十进制数: 1001D $(1001)_{10}$
- 十六进制数: 1001H $(1001)_{16}$
- 二进制单位
 - $2^{10} = 1024 = \text{Kilo} = \mathbf{K}$
 - $2^{20} = 1048576 = \text{Mega} = \mathbf{M}$
 - $2^{30} = \text{Giga} = \mathbf{G}$
 - $2^{40} = \text{Tera} = \mathbf{T}$

2. 运算

□ 二进制加减乘：同十进制

Carries:	00000	101100
Augend:	01100	10110
Addend:	+10001	+10111
Sum:	<u>11101</u>	<u>101101</u>

Borrows:	00000	00110
Minuend:	10110	10110
Subtrahend:	-10010	-10011
Difference:	<u>00100</u>	<u>00011</u>

Multiplicand:	1011
Multiplier:	× 101
	<u>1011</u>
	0000
	1011
Product:	<u>110111</u>

	00110
10011	11110
-11110	-10011
<u> </u>	<u>-01011</u>

2. 运算

□ 八十六进制加减乘

- 将同一列对应位转换为十进制
- 计算完后再转换为原进制

Hexadecimal

$$\begin{array}{r} 59F \\ E46 \\ \hline 13E5 \end{array}$$

Equivalent Decimal Calculation

$$\begin{array}{r} 1 \leftarrow \\ 5 \\ 14 \\ \hline 19 = 16 + 3 \end{array} \quad \text{Carry}$$

$$\begin{array}{r} 1 \leftarrow \\ 9 \quad 15 \\ 4 \quad 6 \\ \hline 14 = E \quad 21 = 16 + 5 \end{array} \quad \text{Carry}$$

2. 运算

□ 八十六进制加减乘

- 将同一列对应位转换为十进制
- 计算完后再转换为原进制

Octal	Octal	Decimal	Octal
7 6 2	5×2	$= 10 = 8 + 2$	$= 12$
4 5	$5 \times 6 + 1$	$= 31 = 24 + 7$	$= 37$
4 6 7 2	$5 \times 7 + 3$	$= 38 = 32 + 6$	$= 46$
3 7 1 0	4×2	$= 8 = 8 + 0$	$= 10$
4 3 7 7 2	$4 \times 6 + 1$	$= 25 = 24 + 1$	$= 31$
	$4 \times 7 + 3$	$= 31 = 24 + 7$	$= 37$

3. 转换

□ 非十进制→十进制

➤ 按权展开

$$(110101.11)_2 = 32 + 16 + 4 + 1 + 0.5 + 0.25 = (53.75)_{10}$$

□ 非十进制相互转化

➤ 1个八进制位=3个二进制位

➤ 1个十六进制位=4个二进制位

➤ 以小数点为界


$$(010\ 110\ 001\ 101\ 011.111\ 100\ 000\ 110)_2 = (26153.7406)_8$$

$$(3A6.C)_{16} = 0011\ 1010\ 0110.1100 = (1110100110.11)_2$$


3. 转换

□ 十进制→非十进制

- 整数部分：除_r取余
- 小数部分：乘_r取余

$41/2 = 20 + 1/2$	Remainder = 1		Least significant digit
$20/2 = 10$	= 0		
$10/2 = 5$	= 0		
$5/2 = 2 + 1/2$	= 1		
$2/2 = 1$	= 0		
$1/2 = 0 + 1/2$	= 1		Most significant digit

$$(41)_{10} = (101001)_2$$

$0.6875 \times 2 = 1.3750$	Integer = 1		Most significant digit
$0.3750 \times 2 = 0.7500$	= 0		
$0.7500 \times 2 = 1.5000$	= 1		
$0.5000 \times 2 = 1.0000$	= 1		Least significant digit

$$(0.6875)_{10} = (0.1011)_2$$

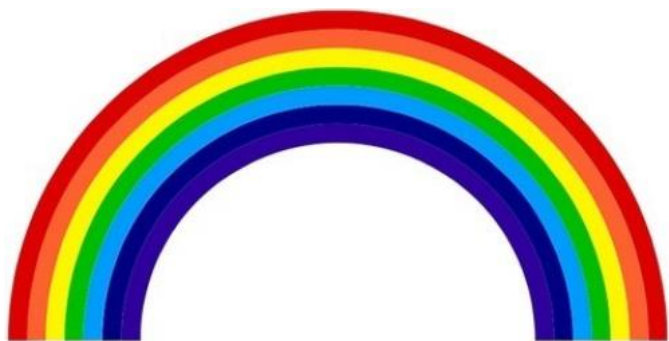
四. 编码

- 1. 二进制编码
- 2. BCD码
- 3. 格雷码
- 4. 字符编码
- 5. 校验位

1. 二进制编码

- 通过0、1排列的组合方式表示数据
- n 位二进制编码可以表示 2^n 个数据
- 数据类型
 - 数值型：能进行算术运算，如整数、小数
 - 非数值型：一般不需算术运算，如字符、控制符
- 表达 M 个数据需要二进制位数
$$\lceil \log_2(M) \rceil$$

1. 二进制编码



颜色	二进制
红	000
橙	001
黄	010
绿	011
蓝	101
靛	110
紫	111

2. BCD码

- 采用二进制编码十进制数
- 用4位二进制表示0~9, 6个冗余

Decimal	8,4,2,1	Excess3	8,4, -2, -1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

BCD码

2. BCD码

□ BCD码

- 8421码
- 最简单，最直觉
- 加权码，权值8,4,2,1
- 1010到1111 无意义

□ 进制转换 VS 编码

- 进制转换: $13_{10} = 1101_2$
- 编码: $13 \Leftrightarrow 0001|0011$

3. 格雷码

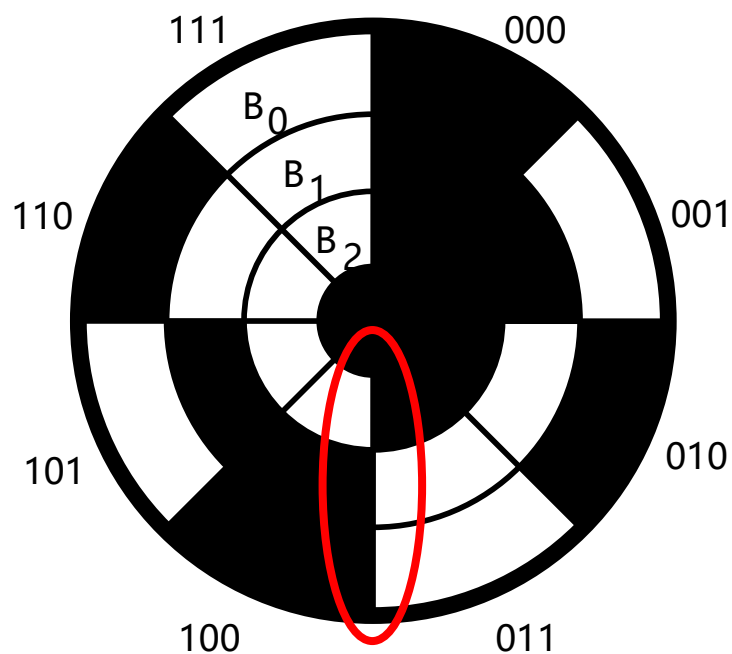
- 计数过程中，相邻编码之间**只有一位不同**
- 格雷码**不唯一**

十进制	8,4,2,1	Gray
0	0000	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

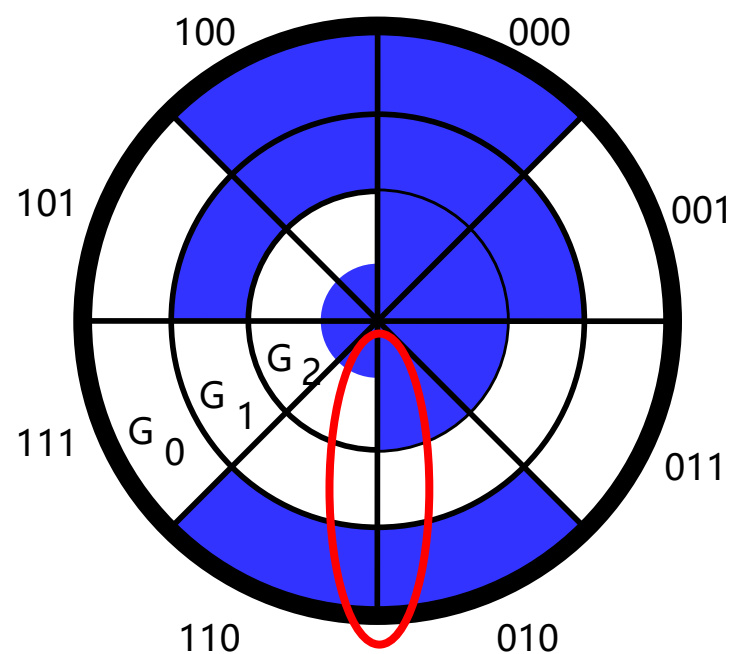
3. 格雷码

□ 特点

- 计数电路中，位翻转次数少，功耗低
- 光学轴角编码器转动中，避免错误编码



BCD码

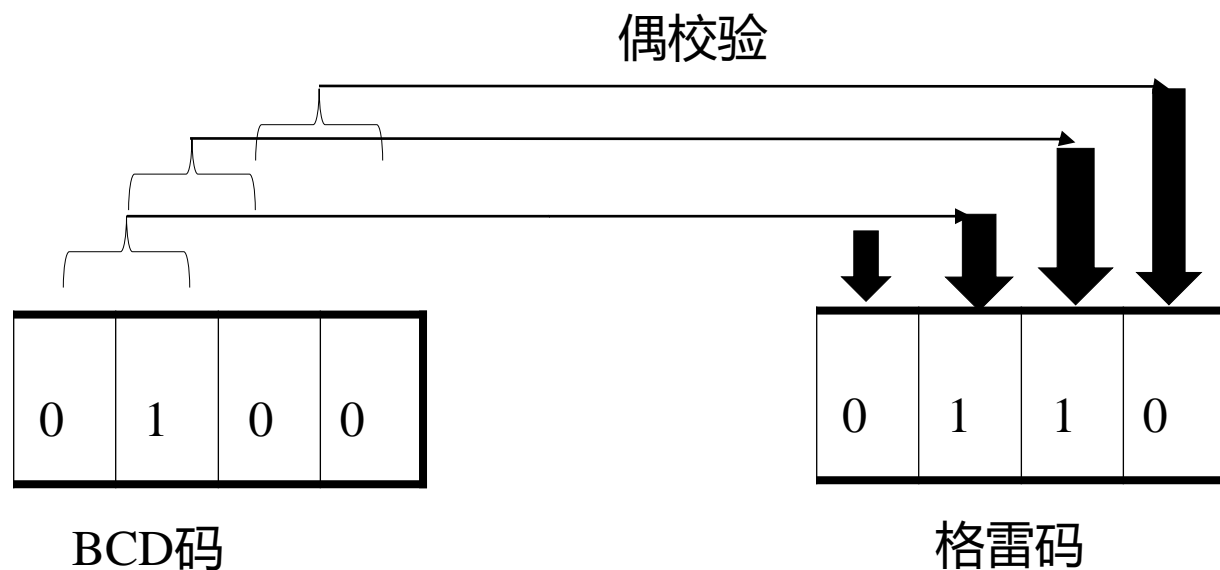


格雷码

3. 格雷码

□ 编制 n (偶数)位二进制计数序列

- 前半：左最高位为0，往右各位为原二进制编码的每一位与它左边相邻位的偶校验
- 后半：前半逆序排列，左最高位为1



3. 格雷码

十进制数	自然二进制数	格雷码	十进制数	自然二进制数	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

3. 格雷码

□ 自动编制算法

- 1位格雷码有两个码字
- $(n+1)$ 位格雷码 = n 位格雷码(顺序)加前缀0 + n 位格雷码(逆序)加前缀1

0	00	000
1	01	001
	11	011
	10	010
		110
		111
		101
		100

<http://blog.csdn.net/>

4. 字符编码

□ ASCII码

- 美国信息交换标准编码
- 7位二进制编码，低->高， $B_1 \dots B_7$
- 128个字符： 94可打印+34个控制
- 特性
 - 0~9: $30_{16} \sim 39_{16}$
 - A~Z: $41_{16} \sim 5A_{16}$
 - a~z: $61_{16} \sim 7A_{16}$
 - 大小写转换: 翻转 B_6

4. 字符编码

$B_4B_3B_2B_1$	$B_7B_6B_5$							
	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

4. 字符编码

□ Unicode

- 万国码、统一码
- 可表示几乎所有语言中的字符与文字
- 对字符规定了唯一代码点, U+0030表示0
- 不同方案
 - UTF-8: 1~4字节, 与ASCII兼容
 - UTF-16: 2或4字节
 - UTF-32: 4字节

4. 字符编码

□ UTF-8编码

- 单字节，第1位为0，7位为代码点(ASCII)
- n字节
 - 第1个字节n个1，1个0，后面字节前两位为10
 - 代码点位从后往前填入，剩下补0

代码点范围 (十六进制)	UTF-8 编码 (二进制, 其中 x 位为代码点位)
U+0000 0000~U+0000 007F	0xxxxxxx
U+0000 0080~U+0000 07FF	110xxxxx 10xxxxxx
U+0000 0800~U+0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
U+0001 0000~U+0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

5. 校验位

- 检测数据传输中可能存在的错误
- 增加1位：表示编码中1个数是奇数或偶数
- 偶校验：偶数个1，校验位0
- 奇校验：奇数个1，校验位0

偶校验

0 1 0 0 0 0 0 1

1 1 0 1 0 1 0 0

奇校验

1 1 0 0 0 0 0 1

0 1 0 1 0 1 0 0

5. 校验位

□ 不足

- 无法确定错误位
- 无法检测偶数个位出错

