数字逻辑电路

目　录

第一章　数字逻辑基础

第一章　数制与编码

Number Systems and Codes

1.2.1 进位计数制

进位计数制简称“数制”。

(*Positional number system*) 。

如：1. 在日常计算中通常采用的是十进制计数制，计数规则“逢十进一”，

例：0,1,2,3,4,5,6,7,8,9,10,11, 12,…,99,100, …,;

2. 在计算机中多用的是二进制计数制，因为物理器件的输入、输出信号是用逻辑电平的两个状态0、1表示，

例：0,1,10,11,100,101,110,…；它是“逢二进一”；

3. 表示重量可以采用十进制或十六进制，例：“半斤八两”；

4. 表示时间的“分秒”，其计数规则采用的是六十进制，

例：1分= 60秒，1小时= 60分, …;

又例：“时”用的是二十四进制，“月”用的是十二进制，等;

5. 计件单位“打”或长度单位“英寸”用的是十二进制；等等。

例：十进制数　　1 2 4 6 3 8 5 3 4 5 .6 7 8 0 9 1

1. 特点：(1) 10个、有序的数字符号：0,1,2,3,4,5,6,7,8,9

(2) 小数点符号：“.”

(3)“逢十进一”的计数规则

其中：“十”为**进位基数(*Base / Radix*)**，简称**基数(*R*)**。

2. 表示法：并列表示法 ***Positional Notation***

多项式表示法 ***Polynomial Notation***

① 并列表示法

	万位	千位	百位	十位	个位		十分位	百分位	千分位	万分位	十万分位	百万分位
	10^4	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
例：十进制数	1	2	3	4	5	.	6	7	8	0	9	1

↑
小数点

如上所示，处在不同**位置**的数字具有不同的“**权(*Weight*)**”，所以并列表示法也称**位置表示法**。

数字逻辑电路

② 多项式表示法

将并列式按“权”展开为按权展开式，称为多项式表示法。如下例：

$$12345.67809$$

$$= 1\times10^{4} + 2\times10^{3} + 3\times10^{2} + 4\times10^{1} + 5\times10^{0} + 6\times10^{-1}$$

$$+ 7\times10^{-2} + 8\times10^{-3} + 0\times10^{-4} + 9\times10^{-5}$$

由此推出，任意一个十进制数 N 可以表示成：

① 并列表示法：

$$(N)_{10}=(K_{n-1}K_{n-2}\ldots K_1K_0.K_{-1}K_{-2}\ldots K_{-m})_{10}$$

$$(0\le K_i\le 9)$$

② 多项式表示法

$$(N)_{10}=(K_{n-1}\times 10^{n-1}+K_{n-2}\times 10^{n-2}+\ldots+K_1\times 10^1$$

$$+K_0\times 10^0+K_{-1}\times 10^{-1}+K_{-2}\times 10^{-2}+\ldots$$

$$+K_{-m}\times 10^{-m})_{10}$$

$$=\sum_{i=-m}^{n-1}K_i\times 10^i \qquad (0\le K_i\le 9)$$

数字逻辑电路

对于一个任意进制 R 的数 N，有 ：

- **特点：**1. **R个有序**的数字符号：0、1、 …、R-1 ;

  2. **小数点**符号：“.”

  3. “**逢R进一**”的计数规则

  其中：“**R**”为**进位基数**(Base / Radix)或**基数**。

例： R=2，二进制，数字符号有0、1，逢二进一；

R=16，十六进制，数字符号有0,1,2,3,4,5,6,7,8,9,A,B,C,D,
E, F(必须用单字符表示)，逢十六进一；

… …

- 表示法

① 并列表示法

$(N)_R = (A_{n-1} \ldots A_i \ldots A_1 A_0 . A_{-1} A_{-2} \ldots A_{-m})_R \quad (0 \le A_i \le R-1)$

(其中： n 整数位数， m 小数位数 ， $0 \le A_i \le R-1$)

当R=10时，则括号及括号外的基数R可以省略。

② 多项式表示法

$$(N)_R = (A_{n-1} \times 10^{n-1} + A_{n-2} \times 10^{n-2} + \ldots + A_1 \times 10^{1}$$

$$+ A_0 \times 10^{0} + A_{-1} \times 10^{-1} + \ldots + A_{-m} \times 10^{-m})_R$$

$$= \left(\sum_{i=-m}^{n-1} A_i \times 10^{i}\right)_R$$

例：$(1010)_2 = (1 \times 10^{11} + 0 \times 10^{10} + 1 \times 10^{1} + 0 \times 10^{0})_2$

$(1212)_3 = (1 \times 10^{10} + 2 \times 10^{2} + 1 \times 10^{1} + 2 \times 10^{0})_3$

不同进位计数制的数值具有等值关系。参见下表：

R=10	R=2	R=3	R=4	R=8	R=16
0	0	0	0	0	0
1	1	1	1	1	1
2	10	2	2	2	2
3	11	10	3	3	3
4	100	11	10	4	4
5	101	12	11	5	5
6	110	20	12	6	6
7	111	21	13	7	7
8	1000	22	20	10	8
9	1001	100	21	11	9
10	1010	101	22	12	A
11	1011	102	23	13	B
12	1100	110	30	14	C
13	1101	111	31	15	D
14	1110	112	32	16	E
15	1111	120	33	17	F
16	10000	121	100	20	10
17	10001	122	101	21	11
…	…	…	…	…	…

例：$(1010)_2 = (1\times10^{11}+0\times10^{10}+1\times10^{1}+0\times10^{0})_2 = 8+0+2+0 = 10$

$(1212)_3 = (1\times10^{10}+2\times10^{2}+1\times10^{1}+2\times10^{0})_3 = 27+18+3+2 = 50$

基数R大的好处

不考虑进位，一个数字符号，描述一种物理状态。

将0~3.3V的输入模拟电平信号转换为数字信号，假设将0~3.3V分成8段，则任意一个0~3.3V的电平替换为0~7之间的符号，相当于一个8进制数。（详见《AD转换与DA转换》章节）

假设将0~3.3V分成256段，则任意一个0~3.3V的电平数字化为0~255之间的符号，相当于一个256进制数。

基数R更大的数呢？

基数R小的好处

熟悉十进制数源自从小的习惯。

例如数青蛙，人有10个手指，所以10进制方便教小朋友：

1只青蛙 1张嘴， 2只眼睛 4条腿；

2只青蛙 2张嘴， 4只眼睛 8条腿；

3只青蛙 3张嘴， 6只眼睛 12条腿；

……

转换思路：

0只青蛙 0张嘴， 0只眼睛 0条腿；

1只青蛙 1张嘴， 10只眼睛 100条腿；

10只青蛙10张嘴，100只眼睛1000条腿；

11只青蛙11张嘴，110只眼睛1100条腿；

。。。

计算机使用的二进制中，称一个符号为一个Bit，只有0和1。

二进制数的运算

二进制数为计算机内部运算的基础。

(1) 运算规则： ＋ 、 －、 ×、 ÷

（ × 、 ÷ 运算可以由+、 －运算来实现）

加法规则：$0 + 0 = 0$　$0 + 1 = 1 + 0 = 1$　$1 + 1 = 10$

减法规则：$0 - 0 = 0$　$1 - 0 = 1$　$1 - 1 = 0$　$10 - 1 = 1$(借位)

乘法规则：$0 \times 0 = 0$　$0 \times 1 = 1 \times 0 = 0$　$1 \times 1 = 1$

除法规则　$0 \div 1 = 0$　$1 \div 1 = 1$　(0不能作除数)

举例： ① $1010 + 0110 = 10000$　② $1010 - 0110 = 0100$

③ $1010 \times 11 = 1010 + 10100 = 11110$ 乘法用加法实现

④ $1010 \div 10 = 101$ 除法用减法实现

```
 1010
–10       …够减，商 1
─────
  01
 –10      …不够减，商 0
─────
   10
  –10     …够减，商 1
─────
    0
```

(2) 常用的二进制常数。(R=2)。

i	R^i	i	R^i	i	R^i
-4	0.0625	2	4	8	256
-3	0.125	3	8	9	512
-2	0.25	4	16	10	1024
-1	0.5	5	32	11	2048
0	1	6	64	12	4096
1	2	7	128	13	8192

(3)二进制数的单位：

1位二进制数 = 1b(比特)　　1B（字节） = 8b

1K （千）= 2^{10}　1M（兆） = 2^{20}　1G（京） = 2^{30}

- 数制
  - ✓特点
  - ✓表示方法
  - –不同数制之间的转换
    - 十进制 ⟷ 其它进制
    - 二进制 ⟷ 其它进制
    - 任意进制 ⟷ 任意进制
    - 如果不能精确转换，怎么办？

1.2.2 进位计数制的相互转换

General Positional-number-system Conversions

数值转换： $(N)_\alpha \rightarrow (N')_\beta$

转换是按等值进行转换

1.2.2.1 多项式替代法 *Series Substitution*

例1：将$(1CE8)_{16}$转换为十进制。

$$
\begin{aligned}
(1CE8)_{16} &= (1\times10^{3}+C\times10^{2}+E\times10^{1}+8\times10^{0})_{16}\\
&= (1\times16^{3}+12\times16^{2}+14\times16^{1}+8\times16^{0})_{10}\\
&= (4096+3072+224+8)_{10}\\
&= (7400)_{10}\\
&= 7400
\end{aligned}
$$

十进制的R可以省略

由上例可以了解多项式替代法的转换步骤，归纳如下，

$$(N)_{\alpha} = (A_{n-1}A_{n-2}\ldots A_1 A_0 . A_{-1} A_{-2} \ldots A_{-m})_{\alpha}$$

$$= (A_{n-1}\times 10^{n-1} + A_{n-2}\times 10^{n-2} + \ldots + A_1\times 10^{1} + A_0 \times 10^{0} + A_{-1}\times 10^{-1} + A_{-2}\times 10^{-2} + \ldots + A_{-m}\times 10^{-m})_{\alpha}$$

$$= (A_{n-1}\times \alpha^{n-1} + A_{n-2}\times \alpha^{n-2} + \ldots + A_1\times \alpha^{1} + A_0\times \alpha^{0} + A_{-1}\times \alpha^{-1} + A_{-2}\times \alpha^{-2} + \ldots + A_{-m}\times \alpha^{-m})_{\beta} = (N')_{\beta}$$

注意：多项式替代法是在β进制下完成 $(N)_{\alpha}$ 到 $(N')_{\beta}$ 的转换的，因此，要求熟悉β进制的算术运算规则。

例2：将$(121.2)_3$转换为二进制。

$$(121.2)_3 = (1\times10^2 + 2\times10^1 + 1\times10^0 + 2\times10^{-1})_3$$

$$= (1\times11^{10} + 10\times11^1 + 1\times11^0 + 10\times11^{-1})_2$$

$$= (1001 + 110 + 1 + 0.101010\ldots)_2$$

$$= (10000.101010\ldots)_2$$

例3：将$(1234)_{10}$转换为十六进制。

$$(1234)_{10} = (1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0)_{10}$$

$$= (1\times A^3 + 2\times A^2 + 3\times A^1 + 4\times A^0)_{16}$$

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

当α为任意进制而β为十进制时，用此方法进行转换。而当β不为十进制时，则不用此方法进行转换。

1.2.2.2 基数乘除法 *Radix Multiply Divide Methed*

要点：(1) 在α进制下完成$(N)_\alpha \to (N')_\beta$的转换

(2) 整数部分转换用基数除法

(3) 小数部分转换用基数乘法

1. 整数转换(基数除法) *Radix Divide Methed*

设：$(N)_\alpha = (N')_\beta = (b_{n-1} b_{n-2} \ldots b_1 b_0)_\beta \quad (0 \le b_i \le \beta-1)$

$$= (b_{n-1} \times 10^{n-1} + b_{n-2} \times 10^{n-2} + \ldots + b_1 \times 10^1 + b_0 \times 10^0)_\beta$$

将b_i、10转换成α进制下的数，则

$$(N)_\alpha = (b_{n-1} \times \beta^{n-1} + b_{n-2} \times \beta^{n-2} + \ldots + b_1 \times \beta^1 + b_0 \times \beta^0)_\alpha$$

$(0 \le b_i \le \beta-1)$

数字逻辑电路

以下讨论在α进制下进行：

令：$N= b_{n-1}\times\beta^{n-1}+b_{n-2}\times\beta^{n-2}+\ldots+b_1\times\beta^1+b_0$

等式两边同除以β，则

$$N/\beta= \underbrace{b_{n-1}\times\beta^{n-2}+b_{n-2}\times\beta^{n-3}+\ldots+b_1\times\beta^0}_{\text{第一个商}}+\boxed{b_0/\beta}$$

则得到 第一个商

第一个余数 b_0

又∵ $0\le b_0\le \beta-1$

∴ 得到第一个余数b_0

数字逻辑电路

又令：

$$N_1 = b_{n-1} \times \beta^{n-2} + b_{n-2} \times \beta^{n-2} + \ldots + b_2 \times \beta^1 + b_1$$

等式两边同除以β，则

$$N_1/\beta = \underbrace{b_{n-1} \times \beta^{n-3} + b_{n-2} \times \beta^{n-4} + \ldots + b_2 \times \beta^0}_{\text{第二个商}} + b_1/\beta$$

同理得到

第二个余数 b_1

又∵ $0 \le b_1 \le \beta - 1$

∴ 得到第二个余数b_1

依次类推，令：

$$N_{i-1} = b_{n-1} \times \beta^{n-i-1} + b_{n-2} \times \beta^{n-i-2} + \ldots + b_i \times \beta^1 + b_{i-1}$$

等式两边同除以β，则

$$N_{i-1}/\beta = b_{n-1} \times \beta^{n-i-2} + b_{n-2} \times \beta^{n-i-3} + \ldots + b_i \times \beta^0 + b_{i-1}/\beta$$

∴ 得到 第 i 个商

∴ 和第i个余数b_i

第i -1个余数 b_{i-1}

直至，令：　$N_{n-2}= b_{n-1}\times\beta^{n-2} + b_{n-2}$

等式两边再同除以β，则

$$N_{n-2}/\beta= \underbrace{b_{n-1}\times \beta^{0}}_{\text{第 n-1个商}} + b_{n-2}/\beta$$

∴　得到　第 n-1个商

第n-1个
余数 b_{n-2}

最后 令：$N_{n-1}= b_{n-1}$

则　$N_{n-1} / \beta= b_{n-1} / \beta$

此时商为零，得到第n个 余数 b_{n-1}，则转换结束。

这种方法也称为“除基取余”。

数字逻辑电路

例1 将十进制的179 转换成二进制数。

$179 \div 2 = 89$ … …余1（b_0）

$89 \div 2 = 44$ … …余1（b_1）

$44 \div 2 = 22$ … … 余0（b_2）

$22 \div 2 = 11$ … … 余0（b_3）

$11 \div 2 = 5$ … …余1（b_4）

$5 \div 2 = 2$ … …余1（b_5）

$2 \div 2 = 1$ … …余0（b_6）

$1 \div 2 = 0$ … … 余1（b_7）

即 $(179)_{10} = (10110011)_2$

例2 将十进制的3417 转换成十六进制数。

$$
\begin{array}{l|l}
16 & 3417 \;\ldots\;\ldots\; 余9\ (b_0) \\ \hline
\quad 16 & 213 \;\ldots\;\ldots\; 余5\ (b_1) \\ \hline
\qquad 16 & 13 \;\ldots\;\ldots\; 余13\ (即D)\ \ (b_2) \\ \hline
\end{array}
$$

即 $(3417)_{10} = (D59)_{16}$

2. 小数转换(基数乘法) *Radix Multiply Methed*

设：

$$(N)_\alpha = (N')_\beta$$

$$= (0.C_{-1}C_{-2}\ldots C_{-m})_\beta$$

$$= (C_{-1}\times 10^{-1} + C_{-2}\times 10^{-2} + \ldots + C_{-m}\times 10^{-m})_\beta$$

$$(0 \le C_i \le \beta - 1)$$

将 b_i、10转换成α进制下的数，则

$$(N)_\alpha = (C_{-1}\times\beta^{-1}+C_{-2}\times\beta^{-2}+\ldots+C_{-m}\times\beta^{-m})_\alpha$$

$$(0 \le C_i \le \beta - 1)$$

以下讨论在α进制下进行：

令：$N = C_{-1}\times\beta^{-1} + C_{-2}\times\beta^{-2} + \ldots + C_{-m}\times\beta^{-m}$

等式两边同乘以β，则

$$N\times\beta = \boxed{C_{-1}} + \underbrace{C_{-2}\times\beta^{-1} + \ldots + C_{-m}\times\beta^{-m+1}}$$

∴ 得到

整数部分的第一位 C_{-1}

小数部分

$\because\ 0 \le C_i \le \beta - 1$

又令：$N_1 = C_{-2} \times \beta^{-1} + C_{-3} \times \beta^{-2} + \dots + C_{-m} \times \beta^{-m+1}$

等式两边同乘以β，则

$$N_1 \times \beta = C_{-2} + \underbrace{C_{-3} \times \beta^{-1} + \dots + C_{-m} \times \beta^{-m+2}}_{\text{小数部分}}$$

整数部分的第二位 C_{-2}

∴ 得到

依次类推，又令：

$$N_{i-1}=C_{-i}\times\beta^{-1}+C_{-i-1}\times\beta^{-2}+\ldots+C_{-m}\times\beta^{-m+i-1}$$

等式两边同乘以β，则

$$N_{i-1}\times\beta=\boxed{C_{-i}+}\,\underbrace{C_{-i-1}\times\beta^{-1}+\ldots+C_{-m}\times\beta^{-m+i}}_{\text{小数部分}}$$

整数部分的第i位 C_{-i}

∴ 得到

直至，令：$N_{m-2}= C_{-m+1} \times \beta^{-1} + C_{-m} \times \beta^{-2}$

等式两边再同乘以β，则

$N_{m-2} \times \beta = C_{-m+1} + C_{-m} \times \beta^{-1}$

∴ 得到

整数部分的第m-1位 C_{-m+1}

小数部分

最后，令：$N_{m-1}= C_{-m} \times \beta^{-1}$

则 $N_{m-1} \times \beta= C_{-m}$

得到第m位 C_{-m}，则转换结束。

这种方法也称为“乘基取整”。

口诀：

除基取余，乘基取整，近点先得。

例3 将 $(0.375)_{10}$ 转换成二进制数。

$$\begin{array}{r l}
0\;.\;375 & \\
\times \qquad 2 & \\
\hline
[0]\;.\;750 & \cdots\cdots\ C_{-1}=0 \\
\times \qquad 2 & \\
\hline
[1]\;.\;500 & \cdots\cdots\ C_{-2}=1 \\
\times \qquad 2 & \\
\hline
[1]\;.\;000 & \cdots\cdots\ C_{-3}=1
\end{array}$$

即 $(0.375)_{10} = (0.011)_2$

例4 将$(0.4321)_{10}$转换成十六进制数。

$N_0 = 0.4321$

$N_0 \times \beta = 0.4321 \times 16 = 6.9136$ $\quad N_1 = 0.9136$ $\quad C_{-1} = 6$

$N_1 \times \beta = 0.9136 \times 16 = 14.6176$ $\quad N_2 = 0.6176$ $\quad C_{-2} = 14(E)$

$N_2 \times \beta = 0.6176 \times 16 = 9.8816$ $\quad N_3 = 0.8816$ $\quad C_{-3} = 9$

$N_3 \times \beta = 0.8816 \times 16 = 14.1056$ $\quad N_4 = 0.1056$ $\quad C_{-4} = 14(E)$

即 $(0.4321)_{10} \approx (0.6E9E)_{16}$

注意：

在转换中，若小数部分最终为零，则表明此数是准确转换；但若小数部分为循环小数或无限不循环小数，则表明此数不是准确转换，即转换有误差。

N的表示

- $(N) = ((((b_{n-1} \times \beta + b_{n-2}) \times \beta + \ldots + b_1) \times \beta + b_0).$
  $(\beta^{-1} \times (b_{-1} + \ldots + \beta^{-1} \times (b_{-m+1} + b_{-m} \times \beta^{-1}))))$

  $(0 \le b_i \le \beta - 1)$

- 无论整数小数，基数乘除法最先转换出的是最靠近小数点的系数。

- （数值转换是考查点）

1.2.2.3 任意两种进制之间的转换

General positional-number-system Conversions

$$(N)_\alpha \rightarrow (N')_\beta$$

1. 若熟悉α进制（十进制）的运算规则，则采用基数乘除法完成转换；
2. 若熟悉β进制十进制的运算规则，则采用多项式替代法完成转换；
3. 若不熟悉α 、β进制的运算规则，则可利用十进制作为转换桥梁，如下图所示：

$$(N)_\alpha \xrightarrow{\text{多项式替代法}} (N')_{10} \xrightarrow{\text{基数乘除法}} (N'')_\beta$$

例3 将$(1023.231)_4$转换成五进制数。

第一步： $(1023.231)_4 \xrightarrow[\text{替代法}]{\text{多项式}} (\quad ? \quad)_{10}$

$(1023.231)_4$

$= (1\times10^3+0\times10^2+2\times10^1+3\times10^0+2\times10^{-1}+3\times10^{-2}+1\times10^{-3})_4$

$= (1\times4^3+0\times4^2+2\times4^1+3\times4^0+2\times4^{-1}+3\times4^{-2}+1\times4^{-3})_{10}$

$= 64+0+8+3+0.5+0.1875+0.015625$

$= 75.703125$

数字逻辑电路

第二步： $(75.703125)_{10} \xrightarrow[\text{除法}]{\text{基数乘}} (\quad ? \quad)_5$

整数部分与小数部分分别转换。

整数部分

$$
\begin{array}{r|l}
5 & 75 \quad \cdots\cdots\ b_0 = 0 \\
\hline
5 & 15 \quad \cdots\cdots\ b_1 = 0 \\
\hline
5 & 3 \quad \cdots\cdots\ b_2 = 3 \\
\hline
 & 0
\end{array}
$$

小数部分

$$
\begin{array}{rl}
0\ .\ 703125 & \\
\times \quad\quad\quad 5 & \\
\hline
[3]\ .\ 515626 & \cdots\cdots\ C_{-1} = 3 \\
\times \quad\quad\quad 5 & \\
\hline
[2]\ .\ 578125 & \cdots\cdots\ C_{-2} = 2 \\
\times \quad\quad\quad 5 & \\
\hline
[2]\ .\ 890625 & \cdots\cdots\ C_{-3} = 2 \\
\times \quad\quad\quad 5 & \\
\hline
[4]\ .\ 453125 & \cdots\cdots\ C_{-4} = 4
\end{array}
$$

即 $(75.703125)_{10} \approx (300.3224)_5$

$\therefore \quad (1023.231)_4 \approx (300.3224)_5$

1.2.2.4 直接转换法

Conversion Between

Base α and Base β when $\beta = \alpha^k$

二进制数是计算机内部真正使用的数，但由于它的表示既易出错也不易交流，故常常用八进制或十六进制的形式表示。

二进制*Binary*，简称B，如 $(10)_2 = (10)_B$ ；

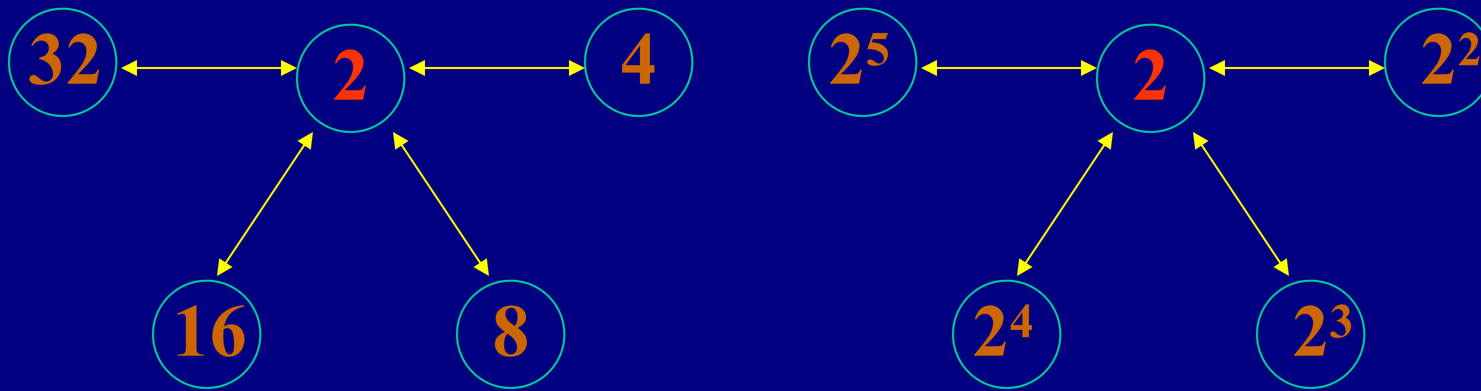
八进制*Octal*，简称O ，如 $(10)_8 = (10)_O$ ；

十六进制*Hexadecimal*，简称H ，如 $(10)_{16} = (10)_H$ 。

转换： $(N)_\alpha \rightarrow (N')_\beta$

当基数α 、β是2的幂次方时，可以进行直接转换。

直接转换图示：



基数为 2^k 的进位制是将一个k位二进制字符串用一位数字字符表示，如下表中$(5)_8=(101)_2$、 $(5)_{16}=(0101)_2$

因此，可用划分相应字符串的方法实现基数为 2^k 的进位制与二进制之间的直接转换。

基数为2与基数为2^k的关系：

R=2	R=4	R=8	R=16
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	10	4	4
101	11	5	5
110	12	6	6
111	13	7	7
1000	20	10	8
1001	21	11	9
1010	22	12	A
1011	23	13	B

R=2	R=4	R=8	R=16
1100	30	14	C
1101	31	15	D
1110	32	16	E
1111	33	17	F
10000	100	20	10
10001	101	21	11
10010	102	22	12
10011	103	23	13
10100	110	24	14
10101	111	25	15
10110	112	26	16
…	…	…	…

$(N)_2 = (N')_{2^k}$ 直接转换的步骤：

(1) 将二进制数用并列表示法表示；

(2) 以小数点为中心，分别向左、向右分组，每k位一组；

(3) 注意小数部分的右补零；

(4) 每组用一位基数为2^k的进位制数表示，则转换完成。

$$(N)_2 = (K_{n-1}K_{n-2}\ldots K_1K_0 \,.\, K_{-1}K_{-2}\ldots K_{-m})_2$$

向左分组　　向右分组

例1　将$(11111010.0111)_2$转换为八进制数。

二进制数　011 111 010 . 011 100

八进制数　3　7　2 .　3　4

即　$(11111010.0111)_2 = (372.34)_8$

$(N)_{2^k} = (N')_2$ 直接转换的步骤：

(1) 将基数为 2^k 进制的数用并列表示法表示；

(2) 以小数点为中心，分别向左、向右分组；

即：基数为2^k的进位制数的每一位用一个k位二进制 字符串表示，则转换完成。

$$(N)_{2^k} = (K_{n-1} K_{n-2} \ldots K_1 K_0 . K_{-1} K_{-2} \ldots K_{-m})_2$$

向左分组　　向右分组

例2 将$(213.01)_4$转换为二进制数。

四进制数	2	1	3	.	0	1
二进制数	10	01	11	.	00	01

即 $(213.01)_4 = (100111.0001)_2$

$(N)_{2^k} = (N')_{2^j}$ 直接转换的步骤：

$$2^k \longrightarrow 2 \longrightarrow 2^j$$

即：转换以二进制为“桥梁”完成。

例3 将$(AF.16C)_{16}$转换为八进制数。

十六进制数	A	F	.	1	6	C		
二进制数	1010	1111	.	0001	0110	1100		
二进制数	010	101	111	.	000	101	101	100
八进制数	2	5	7	.	0	5	5	4

二进制数：1 0 1 0 1 1 1 1 . 0 0 0 1 0 1 1 0 1 1 0 0

即 $(AF.16C)_{16} = (257.0554)_8$

数字逻辑电路

1.2.2.5 数制转换时小数位数的确定

问题的提出：在进行$(N)_{\alpha} \rightarrow (N')_{\beta}$的数制转换时，会出现小数部分不能精确转换的情况，那么，转换后的小数部分应是怎样的呢？

① 小数位数受机器字长的限制而确定；
② 由题目给定小数的位数；
③ 保证转换成β进制后维持与α进制相同的精度。

举例说明：如果一个十进制数的精度是0.001，转换到四进制数时这个数的小数部分应该取几位？

用0.326与0.327作比较：

0.326	0.327
$\times\quad 4$	$\times\quad 4$
$[1].304 \ldots b_{-1}$	$[1].308 \ldots b_{-1}$
$\times\quad 4$	$\times\quad 4$
$[1].216 \ldots b_{-2}$	$[1].232 \ldots b_{-2}$
$\times\quad 4$	$\times\quad 4$
$[0].864 \ldots b_{-3}$	$[0].928 \ldots b_{-3}$
$\times\quad 4$	$\times\quad 4$
$[3].456 \ldots b_{-4}$	$[3].712 \ldots b_{-4}$
$\times\quad 4$	$\times\quad 4$
$[\mathbf{1}].824 \ldots \mathbf{b_{-5}}$	$[\mathbf{2}].848 \ldots \mathbf{b_{-5}}$

即：转换后精度应该是**0.00001**。

数字逻辑电路

设：α进制的小数有k位，转换成β进制后具有相同精度的小数是 j 位，则

$$(0.1)_{\alpha}^{k} = (0.1)_{\beta}^{j}$$

即：

$$\left(\frac{1}{10}\right)_{\alpha}^{k} = \left(\frac{1}{10}\right)_{\beta}^{j}$$

在十进制中可表示为：

$$\left(\frac{1}{\alpha}\right)^{k} = \left(\frac{1}{\beta}\right)^{j}$$

即

$$\alpha^{k} = \beta^{j}$$

两边取对数

$$\log \alpha^{k} = \log \beta^{j}$$

即

$$k \log \alpha = j \log \beta \qquad j = k\frac{\log \alpha}{\log \beta}$$

则 j 应满足不等式：

$$k\frac{\log \alpha}{\log \beta} \le j < k\frac{\log \alpha}{\log \beta} + 1$$

例：将$(0.4321)_{10}$转换成十六进制时，小数位数应取几位？

j 应满足下列不等式：

$$4\frac{\log 10}{\log 16} \le j < 4\frac{\log 10}{\log 16} + 1$$

即

$$3.320 \le j < 4.320$$

所以，小数位数应取4位。

- 总结数制转换
  - 任意进制→ 十进制　　多项式替代法
  - 十进制→ 任意进制　　基数乘除法
  - 任意进制→ 任意进制

    多项式替代法+ 基数乘除法
  - 2的整数次幂进制之间　　分组直接转换

  - 注意哪些转换会有转换误差
  - 当出现转换不尽时，小数位数该如何确定

习题

数字逻辑电路

1.2.3 带符号数的代码表示

Signed Number Representation

带符号数	真值	符号位	数值位
x	+5	+	5
y	−7	−	7

所谓“带符号数的代码表示”是指

带符号数的数值位和符号位都用统一的代码形式表示，即仅取 0 和 1 两种数字符号表示。

有三种代码表示：原码、反码和补码。

- 有关带符号数
  - 代码表示系统(符号位+ 数值位)

    原码、反码、补码

    (生成规则， 表示范围)

  - 加减运算规则(符号位+ 数值位)

1.2.3.1 原码 *Ture form*

(符号-数值表示法 *Signed-magnitude Representation*)

1. 原码的形成规则：

真值 x	符号位 S	数值位 (二进制数)
原码 $[x]_{原}$	＋→0 － →1	不变 不变

例1. 用8位二进制代码表示的原码

$x = +5 \quad [x]_{原} = 00000101$

$y = -7 \quad [y]_{原} = 10000111$

例2. 由原码写出所对应的十进制数值：

$[x]_{原} = 01010101 \quad x = +85$； $[x]_{原} = 11010101 \quad x = -85$；

$[x]_{原} = 01111111 \quad x = +127$； $[x]_{原} = 11111111 \quad x = -127$；

$[x]_{原} = 00000000 \quad x = +0$； $[x]_{原} = 10000000 \quad x = -0$；

2. 将真值 x 变换成原码 $[x]_{原}$ 的变换公式：

在一个 n 位原码系统中（包括一位符号位），则

$$[x]_{原} = \begin{cases} x & 0 \le x < 2^{n-1} \\ 2^{n-1} - x & -2^{n-1} < x \le 0 \end{cases}$$

3. 原码的加法运算： $[z]_{原} = [x]_{原} + [y]_{原}$

① 当 $[x]_{原}$、$[y]_{原}$ 的符号 S 相同时，则 $[z]_{原}$ 的符号同 $[x]_{原}$ 的符号，$[z]_{原}$ 的数值为两加数的和；

{ if.else 效率太低
有无懒人方法？

② 当 $[x]_{原}$、$[y]_{原}$ 的符号 S 相异时，先判断两数的绝对值的大小，$[z]_{原}$ 的符号同大数的原码符号，$[z]_{原}$ 的数值为用大数的绝对值减去小数的绝对值的差。

$$\begin{array}{lr} +5 & 00101 \\ -5 & +\ 10101 \\ \hline & 11010 \end{array}$$

加出一个负数

1.2.3.2 反码 *Ngative Number*

(对“1”补 *One's-complement Representation*)

1. 反码的形成规则：

真值 x	符号位 S	数值位 (二进制数)
反码 $[x]_{反}$	$+ \rightarrow 0$ $- \rightarrow 1$	不变 按位变反

例1. 用8位二进制代码表示的原码、反码

$x = +5 \quad [x]_{原} = 00000101 \quad [x]_{反} = 00000101$

$y = -7 \quad [y]_{原} = 10000111 \quad [y]_{反} = 11111000$

例2. 由反码写出所对应的十进制数值：

至少在加减上 if/else 省了.

$[x]_{反} = 01010101 \quad x = +85;$ $\qquad [x]_{反} = 11010101 \quad x = -42;$

42 = 00101010 $\qquad$ 32+8+2

$[x]_{反} = 01111111 \quad x = +127;$ $\qquad [x]_{反} = 11111111 \quad x = -0;$

$[x]_{反} = 00000000 \quad x = +0;$ $\qquad [x]_{反} = 10000000 \quad x = -127;$

2. 将真值 x 变换成反码 $[x]_{反}$ 的变换公式：

在一个 n 位反码系统中，则

$$[x]_{反} = \begin{cases} x & 0 \leq x < 2^{n-1} \\ (2^{n}-1) + x & -2^{n-1} < x \leq 0 \end{cases}$$

反码运算时可以将符号位与数值位一起参与运算，但有点Bug

1.2.3.3 补码 *tow's-complement Representation*

1. 补码的形成规则：

8位 负多一个

-128 ~ +127

真值 x	符号位 S	数值位 (二进制数)
补码 $[x]_{补}$	＋→0 － →1	不变 按位变反+1

例1. 用8位二进制代码表示的原码、补码

x = ＋5　$[x]_{原}$ = 00000101　$[x]_{补}$ = 00000101

y = －7　$[y]_{原}$ = 10000111　$[y]_{补}$ = 11111001

11111000+1

例2. 由补码写出所对应的十进制数值：

00101011

$[x]_{补}$ = 01010101　x =＋85；　$[x]_{补}$= 11010101　x =－43；

$[x]_{补}$ = 01111111　x =＋127；　$[x]_{补}$= 11111111　x =－1；

$[x]_{补}$ = 00000000　x =＋0；　$[x]_{补}$= 10000000　x =－128；

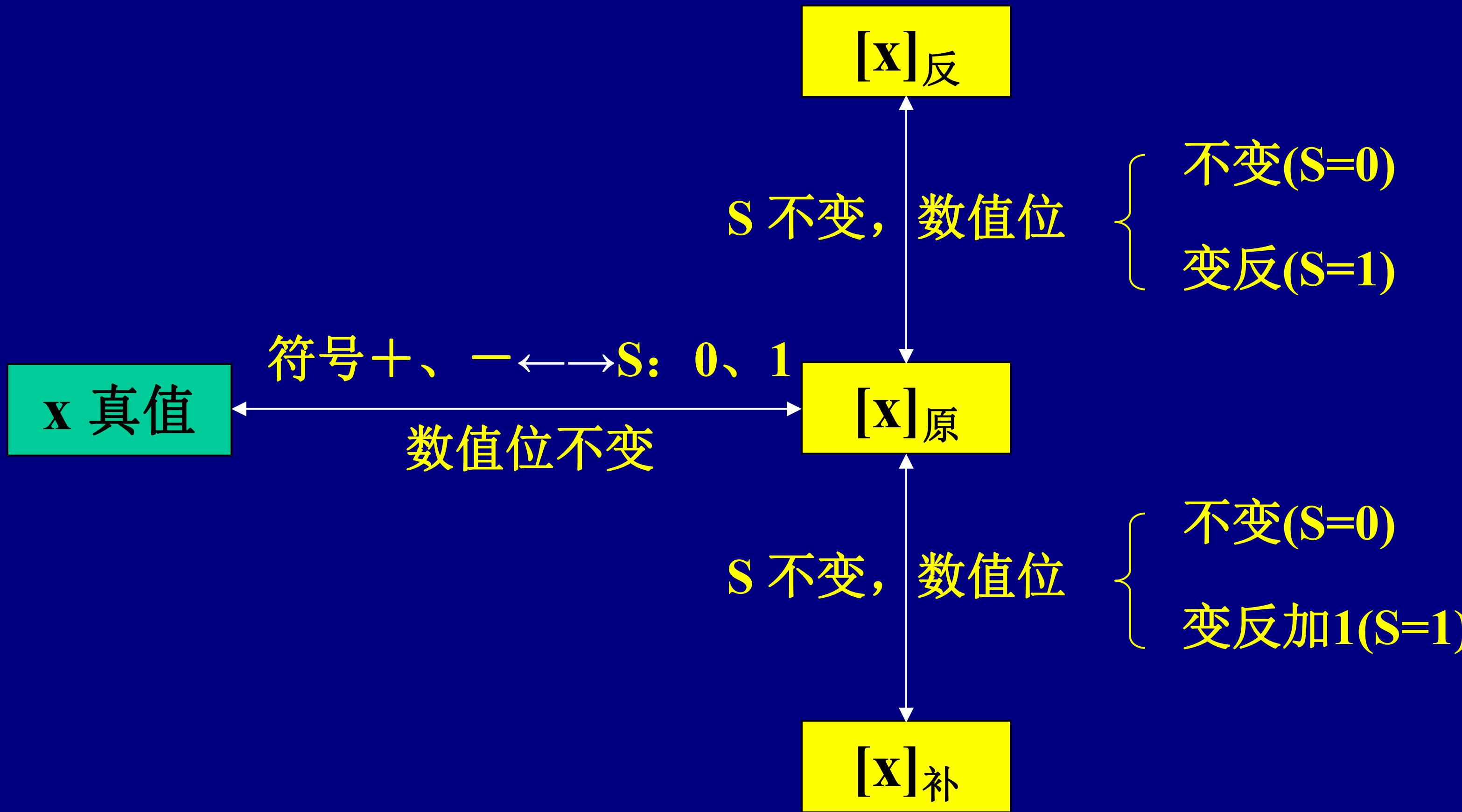
2. 将真值 x 变换成补码 $[x]_{补}$ 的变换公式：

在一个n位补码系统中，则

$$[x]_{补} = \begin{cases} x & 0 \le x < 2^{n-1} \\ 2^n + x & -2^{n-1} \le x < 0 \end{cases}$$

运算时将符号位与数值位一起参与运算

原码、反码和补码之间的关系如下图



1.2.4 带符号数的加、减运算

Signed Number Addition and Subtraction

- 原码　加减法有不同的规则，关键是要判大小；
- 反码　$[x+y]_{反}=[x]_{反}+[y]_{反}$；

　　$[x-y]_{反}=[x]_{反}+[-y]_{反}$； 只处理加法

- 补码　$[x+y]_{补}=[x]_{补}+[y]_{补}$；

　　$[x-y]_{补}=[x]_{补}+[-y]_{补}$；

反码与补码的运算规则：

- 反码 $[x+y]_{反}=[x]_{反}+[y]_{反}$；

  $[x-y]_{反}=[x]_{反}+[-y]_{反}$；
- 补码 $[x+y]_{补}=[x]_{补}+[y]_{补}$；

  $[x-y]_{补}=[x]_{补}+[-y]_{补}$；

① 减法运算：按加法运算完成；

② 符号位S的处理：S 被看成一位数码，并与数值位一样，按同样的加法规则进行处理，所得结果的符号位即是正确结果的符号位。

③ 进位的处理：反码运算时符号位产生的进位要加到和数的最低位上去，补码运算时符号位产生的进位要丢掉。

原码、反码与补码的运算举例：

例1 求 $z = x - y$， 其中 $x = +1010$， $y = +0011$

10 − 3.

(1) 原码运算： $[x]_{原} = 01010$ $[y]_{原} = 00011$

∵ x 绝对值 > y 绝对值

∴ $[z]_{原} = [01010 - 00011]_{原} = 00111$ $z = +0111$

(2) 补码运算： $[x]_{补} = 01010$

$[-y]_{补} = [-0011]_{补} = 11101$

∵
$$\begin{array}{r} 01010 \\ +11101 \\ \hline \boxed{1}\,00111 \end{array}$$

丢掉← 1

∴ $[z]_{补} = 00111$

$z = +0111$

数字逻辑电路

原码、反码与补码的运算举例：

例1　求 $z = x - y$，　其中 $x = +1010$，$y = +0011$

(3)反码运算：　$[x]_{反} = 01010$

$$[-y]_{反} = [-\ 0011\ ]_{反} = 11100$$

$$\because \quad \begin{array}{r} 01010 \\ +11100 \\ \hline \boxed{1}\,00110 \\ +\qquad 1 \\ \hline 00111 \end{array}$$

丢不掉

补码解决了这个多出一个1，更加健壮"

$\therefore$　$[z]_{反} = 00111$

$z = +0111$

二进制加减

考试不做要求

原码、反码与补码的运算举例：

例2　求 $z = x - y$，其中 $x = +0011$，$y = +1010$

(1) 原码运算：　$[x]_{原} = 00011$　　　$[y]_{原} = 01010$

∵ x 绝对值 < y 绝对值

∴ $[z]_{原} = [-(01010 - 00011)]_{原} = -00111$　$z = -0111$

(2) 补码运算：　$[x]_{补} = 00011$

$[-y]_{补} = [-1010]_{补} = 10110$

∵
$$\begin{array}{r} 00011 \\ +10110 \\ \hline 11001 \end{array}$$

∴ $[z]_{补} = 11001$

$z = -0111$

原码、反码与补码的运算举例：

例2　求 $z = x - y$，　其中 $x = +0011$，$y = +1010$

(3)反码运算：　$[x]_{反} = 00011$

$[-y]_{反} = [-1010]_{反} = 10101$

$$
\because \quad
\begin{array}{r}
00011 \\
+\,10101 \\
\hline
11000
\end{array}
$$

要求：
给真值，
写出原反补

$\therefore \quad [z]_{反} = 11000$

$z = -0111$

11011
00110
00001

求补数: $[x]_{补} \xrightarrow{求补数} [-x]_{补}$

取反加1

无视符号

反码　$[x+y]_{反}=[x]_{反}+[y]_{反}$;

　　　$[x-y]_{反}=[x]_{反}+[-y]_{反}$;

补码　$[x+y]_{补}=[x]_{补}+[y]_{补}$;

　　　$[x-y]_{补}=[x]_{补}+[-y]_{补}$;

$$\begin{array}{r} 985 \\ -\ 211 \\ \hline 774 \end{array} \qquad \begin{array}{r} 985 \\ +\ 789 \\ \hline \cancel{1}774 \end{array}$$

有权码: 变减为加, 进位舍弃

怎么求反，怎么求补？

- 总结带符号数
  - 代码表示系统(符号位+ 数值位)
    - 原码、反码、补码 ⟷ 真值
  - 加减运算规则
    - 只做加法
    - 符号位、数值位变反、变补时一起处理
    - 符号位、数值位做二进制加法时，一起处理
    - 补码的优点在哪里

1.2.5 十进制数的常用代码

Binary code for decimal numbers

1. 十进制数的代码表示

- 既具有二进制数的形式，又具有十进制数的特点，即用四位二进制数的代码表示一位十进制数；
- 可按位直接相互转换；
- 可按位直接运算。

2. 常用的代码

“8421”码（BCD码 *Binary coded decimal*）

“2421”码

余3码 *(Excess-3)*

⋮

数字逻辑电路

表1—3 三种十进制数的代码表示法

十进制整数	8421码	2421码	余3码 最简单
0	0000	0000	0011
1	0001 顺着写	0001 前5个	0100 中间
2	0010	0010 后5个	0101 10个
3	0011	0011 有权码	0110
4	0100	0100 $(5)_{10}$	0111
5	0101	1011 $(11)_{10}$.	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9 这些人不应出现	1001	1111	1100
无效码区 Unused code wrds	1010、1011、1100、1101、1110、1111	0101、0110、0111、1000、1001、1010	0000、0001、0010、1101、1110、1111

3. 代码的特点

设：代码表示为 $A_3\,A_2\,A_1\,A_0$

会考填空.

代码	对应的十进制数值	代码直接按位转换
8421码	有权码 (*Weighted code*) $8A_3+4A_2+2A_1+1A_0$	8421. $(13)_{10}=(00010011)_{BCD}$ 0 $(1011101010000)_{BCD}=(1750)_{10}$
2421码	有权码、对9自补码 7 2 1101+0010=9=>是7 $2A_3+4A_2+2A_1+1A_0$	$(13)_{10}=(00010011)_{2421}$ 0010→9 -6 $(1110110110000)_{2421}=(1750)_{10}$ 后5.减6 后5.减6
余3码	无权码、对9自补码 $8A_3+4A_2+2A_1+1A_0$ -0011	$(13)_{10}=(01000110)_{余3}$ $(1001010100000011)_{余3}=(1750)_{10}$ 4余3, 10余3 · 8余3 3余3

三种十进制数代码的分布图

四位二进制代码	8421码	2421码	余3码
0000	0000 0	0000 0	0000 非码区
0001	0001 1	0001 1	0001
0010	0010 2	0010 2	0010
0011	0011 3	0011 3	0011 0
0100	0100 4	0100 4	0100 1
0101	0101 5	0101 非码区	0101 2
0110	0110 6	0110	0110 3
0111	0111 7	0111	0111 4
1000	1000 8	1000	1000 5
1001	1001 9	1001	1001 6
1010	1010 非码区	1010	1010 7
1011 11	1011	1011 5	1011 8
1100	1100	1100 6	1100 9
1101 13	1101	1101 7	1101 非码区
1110	1110	1110 8	1110
1111	1111	1111 9	1111

正好互反

互反

4. 代码运算　　$C = A + B$

若按二进制数直接运算，则运算结果要修正。

例：一位代码的加法运算，如下：

8421修正器（不用记）

8421码 8	2421码	余3码
$C_8C_4C_2C_1 = A_8A_4A_2A_1 + B_8B_4B_2B_1$	$C_2C_4C_2C_1 = A_2A_4A_2A_1 + B_2B_4B_2B_1$	$C_4C_3C_2C_1 = A_4A_3A_2A_1 + B_4B_3B_2B_1$
1.当0000≤C≤1001，不需修正 例：0011＋0100 2. 当1010≤C≤1111，C＋0110 例：0111＋0100 3.当C≥10000，C＋0110 例：1001＋1001	修正的算法较复杂（略）	1.当C ≤ 1111，C－0011 例：0011＋0100 2.当C≥10000，C＋0011 例：1100＋0100

- 作业

1.2.6 可靠性编码 Error Detection Codes and Correction Codes

目的：解决代码在形成或传输过程中可能会发生的错误，提高系统的安全性。

方法：1. 使代码自身具有一种特征或能力；
2. 增加信息位之间的运算，如异或运算⊕；
3. 增加校验位 。

作用：1. 不易出错；
2. 若出错时易发现错误；（低级）
3. 出错时易查错且易纠错。（高级）

- 关于可靠性编码
  - 格雷码
  - 奇偶校验码
  - 海明校验码

作用，产生

1.2.6.1 格雷码（Gray)

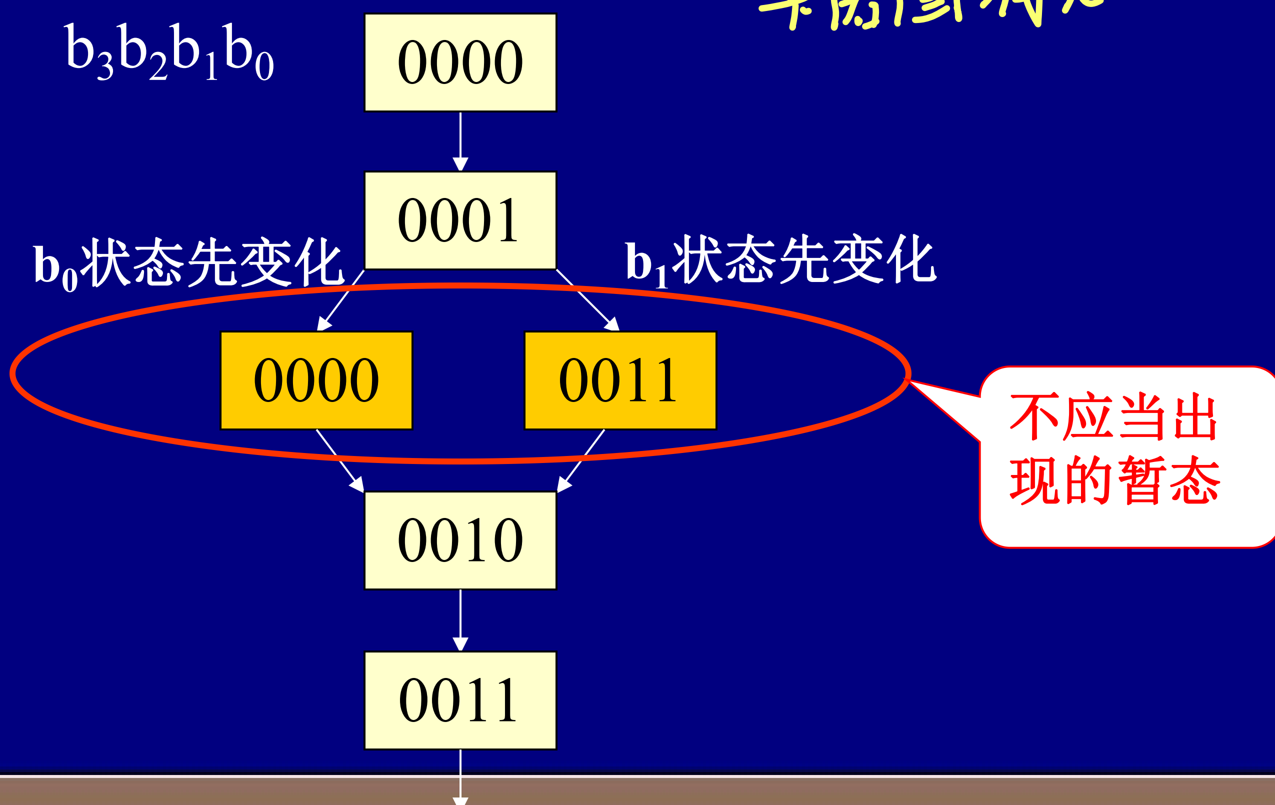
二→格雷标准.考

特点： 任意两个相邻数的代码只有一位二进制数不同。

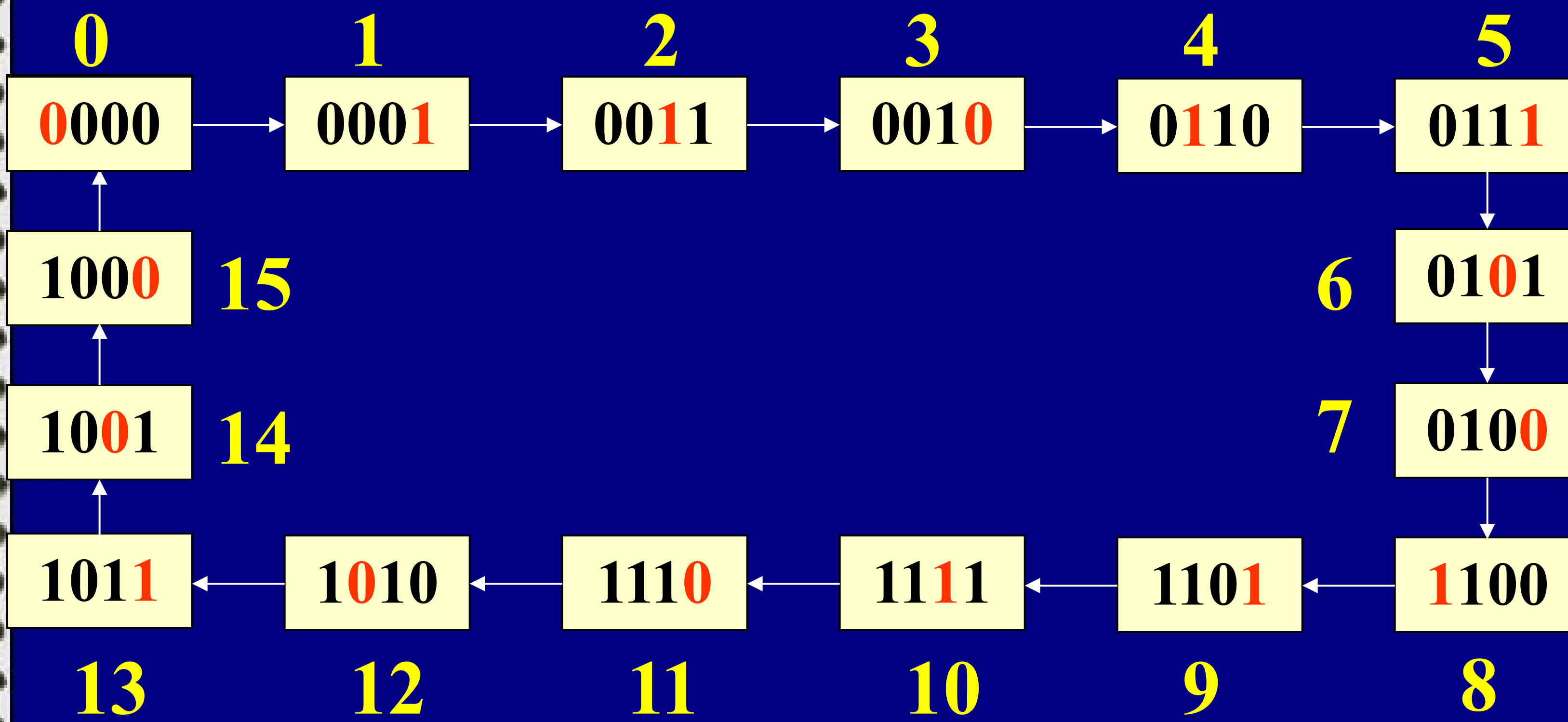
目的： 解决代码生成时发生的错误。

例：四位二进制加1计数器，工作时有如下情况出现：

卡诺图有见



但是，当选用典型Gray码设计加1计数器时，就不会出现上述情况。如下： 比较精密



Gray码还包括步进码、十进制Gray 码等。参见书p15.表1－4 。

表1－4　几种Gray码、步进码和二进制码对照表

会：2进制 VS 典型的Gray. 异或得来

十进制数	二进制数	典型Gray	十进制 Gray码(1)	十进制 Gray码(2)	步进码
0	0000	0000	0000	0000	00000
1	0001	0001	0001	0001	00001
2	0010	0011	0011	0011	00011
3	0011	0010	0010	0010	00111
4	0100	0110	0110	0110	01111
5	0101	$\boxed{0111}$	1110	0111	11111
6	0110	0101	1010	0101	11110
7	0111	0100	1011	0100	11100
8	1000	1100	1001	1100	11000
9	1001	1101	1000	1000	10000
10	1010	1111			
11	1011	1110			
12	1100	1010			
13	1101	1011			
14	1110	1001			
15	1111	1000			

少=10/32

五位(缺点)

从图形去写

卡诺!

格雷有5W种
典型只有一种

给一个二典格雷，
要写出二进制

只要相邻就是Gray码
但方便实现

数字逻辑电路

异或、“模2和”的运算规则如下

不同就对　添彩

2026个1异或 ⇒ 0
2025个1 第0位 1 ⇒ 1

$0\oplus0=0$　　$1\oplus1=0$　　$0\oplus1=1$　　$1\oplus0=1$

$0\oplus A=A$　　$1\oplus A=\overline{A}$　　$A\oplus A=0$　　$\overline{A}\oplus A=1$

(一样)　(一定不一样)

$A=\begin{cases}1\Rightarrow0\\0\Rightarrow1\end{cases}$

$1\oplus1\oplus1=0\oplus1=1$

用真值表验证：

$(1\oplus1)\oplus(1\oplus1)=0\oplus0=0.$

A	$0\oplus A$	$1\oplus A$	$A\oplus A$	$A\oplus\overline{A}$
0	0	1	0	1
1	1	0	0	1

数字逻辑电路

典型Gray码 这么算(独一无二) 由二进制码生成典型Gray码

只有典格雷可以 — 格码

通过异或运算⊕完成：$G_i = B_{i+1} \oplus B_i$ ✓ 不用推导证明 记住.

设：二进制码 B　　　$0 \quad B_{n-1} \quad B_{n-2} \quad \cdots \quad B_1 \quad B_0$

$\oplus \quad \oplus \quad \cdots \quad \oplus \quad \oplus$

典型Gray码G　　　$G_{n-1} \quad G_{n-2} \quad \cdots \quad G_1 \quad G_0$

0101

$0 \quad B_3 B_2 B_1 B_0$

$\oplus \quad \oplus \quad \oplus \quad \oplus$

0 1 1 1

例：二进制码 B　　　0　1　1　1　0

$\oplus \quad \oplus \quad \oplus \quad \oplus$

典型Gray码G　　　1　0　0　1

∴ 由二进制码生成典型Gray码 可以看

$$G_0 = B_1 \oplus B_0$$

$$G_1 = B_2 \oplus B_1$$

$$\vdots$$

$$G_i = B_{i+1} \oplus B_i$$

$$\vdots$$

$$G_{n-2} = B_{n-1} \oplus B_{n-2}$$

$$G_{n-1} = 0 \oplus B_{n-1} = B_{n-1}$$

反之，由典型Gray码也可以得到二进制码，如下：

设：典型Gray码 $G_{n-1}\ G_{n-2}\ \dots\ G_1\ G_0$

$\because \quad G_i = B_{i+1} \oplus B_i$

则等式两边同时 $\oplus\ B_{i+1}$：

$$G_i \oplus B_{i+1} = B_{i+1} \oplus B_i \oplus B_{i+1}$$

$\therefore \quad B_i = G_i \oplus B_{i+1}$

故 $\quad B_{n-1} = G_{n-1} \oplus B_n = G_{n-1} \oplus 0 = G_{n-1}$

$$B_{n-2} = G_{n-2} \oplus B_{n-1}$$

$$\vdots$$

$$B_0 = G_0 \oplus B_1$$

设：典型Gray码 G　　$G_{n-1}\,G_{n-2}\ \cdots\ G_1\ \ G_0$

二进制码 B　　$B_{n-1}\ \ B_{n-2}\ \cdots\ B_1\ \ B_0$

例：典型Gray码G　　1　1　1　0

二进制码 B　　1　0　1　1

低位算完往上传，效率比较低下

另一种解决方案

B_{n} → 直接取反方 所有异或

同步进行.

$$B_{n-1} = G_{n-1}$$

$$B_{n-2} = G_{n-2} \oplus B_{n-1} = G_{n-2} \oplus G_{n-1}$$

$$B_{n-3} = G_{n-3} \oplus B_{n-2} = G_{n-3} \oplus G_{n-2} \oplus G_{n-1}$$

$$\vdots$$

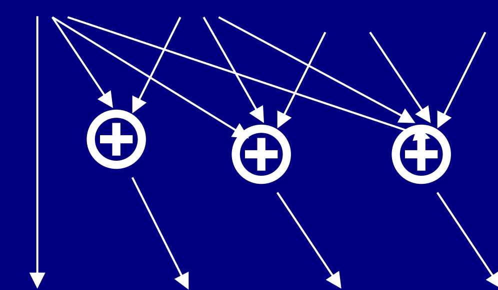
$$B_0 = G_0 \oplus B_1 = G_0 \oplus G_1 \oplus \ldots \oplus G_{n-2} \oplus G_{n-1}$$

74LS 民用产品

54LS 军用产品 成本高

特种性能

典型Gray码G　　1　1　1　0



二进制码 B　　1　0　1　1

十进制 典型 Gray 异或算的 / 设计其余Gray 不能算

要模10 步进码 / (2)种Gray.

典型Gray码的一个特点：

所有对应于十进制数2^m-1(m为正整数)的Gray：

???

仅在m位上有1，其他位都为0。

2个1 不可以回0.

可以回0 模2/4/8.

m	十进制数 2^m - 1	典型Gray码
1	1（0001）	0001
2	3（0011）	0010
3	7（0111）	0100
4	15（1111）	1000

考过 模6格雷码计数器设计

数字逻辑电路

表1－4　几种Gray码、步进码和二进制码对照表

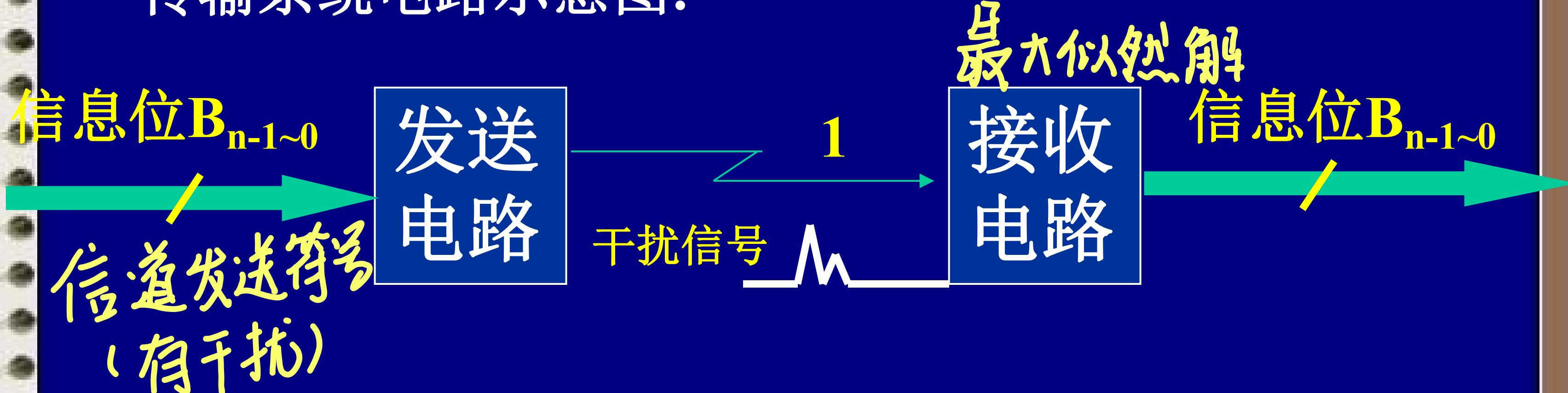
十进制数	二进制数	典型Gray	十进制Gray码(1)	十进制Gray码(2)	步进码
0	0000	0000	0000	0000	00000
1	0001	0001	0001	0001	00001
2	0010	0011	0011	0011	00011
3	0011	0010	0010	0010	00111
4	0100	0110	0110	0110	01111
5	0101	0111	1110	0111	11111
6	0110	0101	1010	0101	11110
7	0111	0100	1011	0100	11100
8	1000	1100	1001	1100	11000
9	1001	1101	1000	1000	10000
10	1010	1111			
11	1011	1110			
12	1100	1010			
13	1101	1011			
14	1110	1001			
15	1111	1000			

数字逻辑电路

校验码和纠错码 *Codes for detecting and Correcting Errors*

误码率曲线★

传输系统电路示意图:



问题：信息在传输过程中受外界干扰而出错，

且绝大多数为单错。一个错误（多错不是误码是系统该重造了）

精准发现

解决方法：① 增加校验位（P）；第n位 信息论（矩阵分析）

② 通过异或运算⊕； 概率论

1.2.6.2 奇偶校验码 *Parity Code*

校验码:
(单错情况下)

信息位 $B_{n-1\sim0}$	校验位 P

$= P_{偶}$

即 $\overline{\oplus}\ B_{n-1} \oplus B_1 \oplus B_0 \oplus P_{偶} = 0$

- 偶校验:

"使"

校验码P的取值使校验码中"1"的个数是偶数;

$$\boxed{P_{偶}} = B_{n-1} \oplus B_{n-2} \oplus \ldots \oplus B_1 \oplus B_0$$

$P_{偶}$由 $\sum^{n-1} \oplus B_k$ 决定

认知上 补上一位. $P_{偶}=0/1$.
使1个数最终有2k个

- 奇校验:

校验码P的取值使校验码中"1"的个数是奇数;

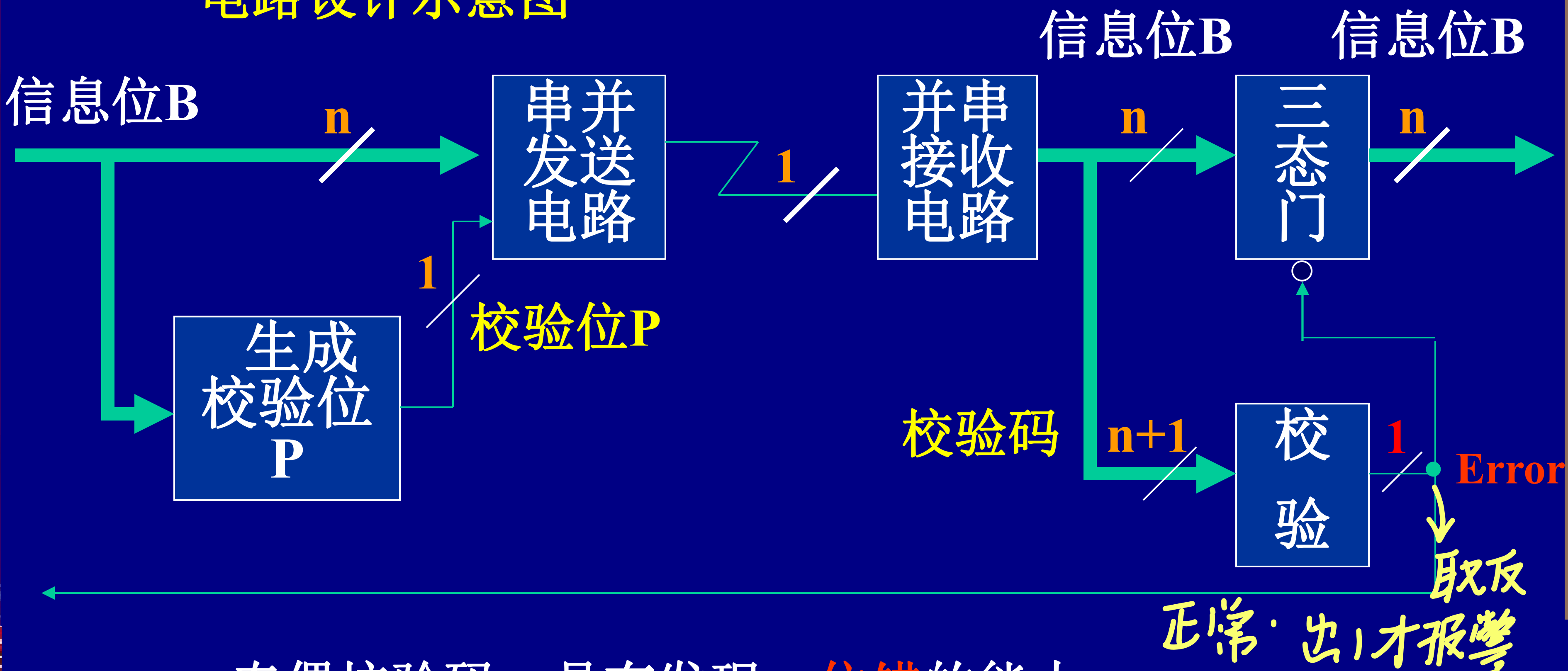
(用的人少)

$$P_{奇} = B_{n-1} \oplus B_{n-2} \oplus \ldots \oplus B_1 \oplus B_0 \oplus 1$$

信息条 + $P_{奇}$ (0/1) { 偶个1 补1; 否则 补0

数字逻辑电路

电路设计示意图



奇偶校验码：具有发现一位错的能力。

例：偶校验结果

$$Error_{偶} = B_{n-1} \oplus B_{n-2} \oplus \ldots \oplus B_1 \oplus B_0 \oplus P_{偶}$$

若 $Error_{偶}=0$ 则 传输正确

$Error_{偶}=1$ 传输出错

1.2.6.3 海明校验码 *Hamming codes* $\boxed{\text{考试不考}}$

- 目的：不仅能检测出单错，还能校正单错 3角形思想
- 方法：增加校验位及相应的异或运算

以四位信息位$B_4\,B_3\,B_2\,B_1$为例，在传输前生成它的海明校验码：

$$\begin{array}{lcccccccc} & 2^3 & & & & 2^2 & & 2^1 & 2^0 \\ \text{(1) 位序：} & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ & P_4 & B_4 & B_3 & B_2 & P_3 & B_1 & P_2 & P_1 \end{array}$$

必须3个校位

$(n-1)$个

① 几个校码
② 在哪放P_n.
③ 怎计算P_n
④ 收端怎发现

(2) 校验位的生成公式：

$$P_3 = B_4 \oplus B_3 \oplus B_2$$

（偶校验）

$$P_2 = B_4 \oplus B_3 \oplus B_1$$

$$P_1 = B_4 \oplus B_2 \oplus B_1$$

纠错目标 001 → B_1错　7种可能 $\left\{\begin{array}{l}1\text{个正确} \\ 3\text{个单错} \\ 4\text{个校验}\end{array}\right\} 8$

表1-6 “8421”海明码 4位校验 { 1个正确 4个单错 11个校验 } 16

位序 / N	7	6	5	4	3	2	1
	B_4	B_3	B_2	P_3	B_1	P_2	P_1
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0

对传输后的海明码进行检错和校错：

(3) 校验和：$S_3 = B_4 \oplus B_3 \oplus B_2 \oplus P_3$

$S_2 = B_4 \oplus B_3 \oplus B_1 \oplus P_2$

$S_1 = B_4 \oplus B_2 \oplus B_1 \oplus P_1$

① 当$S_3\,S_2\,S_1 = 0$时，接收到的信息是正确的；

② 当$1 \le S_3\,S_2\,S_1 \le 7$时，则$S_3\,S_2\,S_1$所表示的二进制值就是出错的那一位的位序值。

例：若接收到的海明码为：

7	6	5	4	3	2	1
B_4	B_3	B_2	P_3	B_1	P_2	P_1
0	1	0	1	0	1	0

6位出错

则 $S_3\,S_2\,S_1 = 110$，表示第6位(B_3)出错，改0为1。

表1-7 出错表的确定

$S_3 =$	$B_4 \oplus$	$B_3 \oplus$	$B_2 \oplus$	P_3				
$S_2 =$	$B_4 \oplus$	B_3		$\oplus$	$B_1 \oplus$	P_2		
$S_1 =$	B_4	$\oplus$	B_2	$\oplus$	B_1	$\oplus$	P_1	
$S_3\ S_2\ S_1$	111	110	101	100	011	010	001	000
出错位序列	7	6	5	4	3	2	1	
出错位	B_4	B_3	B_2	P_3	B_1	P_2	P_1	

校验码自己都错，错S只有自己

B4必在S1. S2 S3出现. 都能看见B4.

1. 每个校验位P必分布在2^k位上，使其仅在一个校验和S中出现;

2. 信息位B分布在非2^k位上，使其在一个以上的校验和S中出现;

3. 若传送后海明码中的某一位出错，则将影响它所在的校验和S_i，故能得到它的位序值，即可实现其单错的定位和校错。

设：信息位n位，校验位k位

则 $(2^k - 1) - k \geq n$

或 $(2^k - 1) \geq n + k$

如下表所列：

4位校验

校验位数k	1	2	3	4	5	6	7	8
最大信息位数n	0	1	4	11	26	57	120	247
海明码位数 $(2^k - 1)$	1	3	7	15	32	63	127	255

74海明码　　增加奇偶校验扩展位

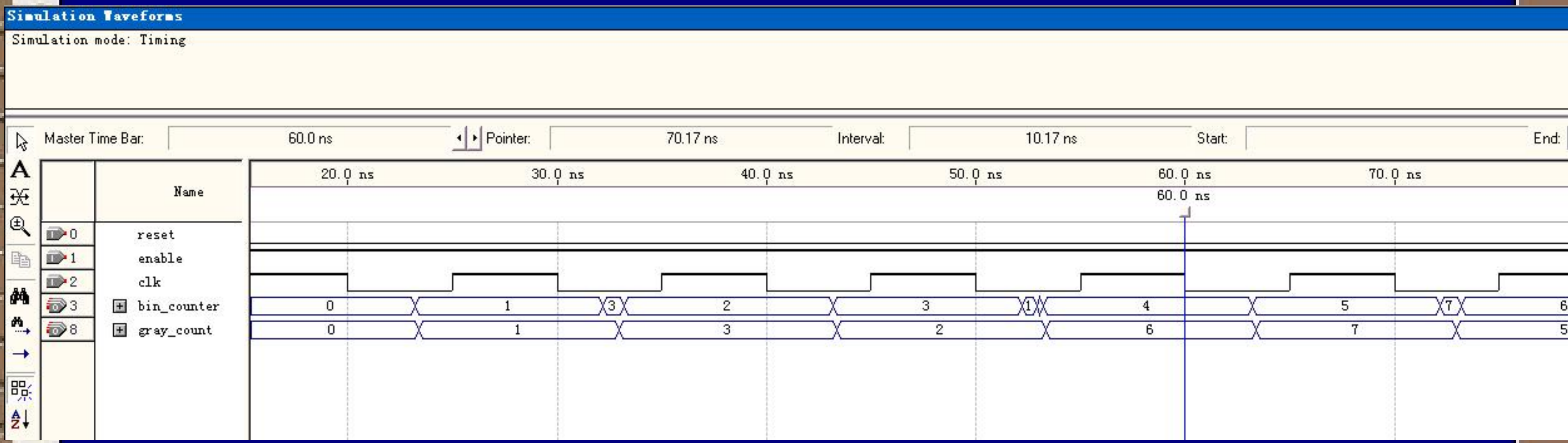
84增余海明码更为常用

可靠性编码

格雷码　奇偶校验码　海明码

格雷码

000, 001, 011, 010, 110, 111, 101, 100



000, 001, 010, 100, 011, 101, 110, 111

One hot编码：001, 010, 100

奇偶校验码

000 - 001 - 011 海明距离为2 站远一点

★研究码间距 d, 可查错 $d-1$ 不具备纠正

星

000 - 001 - 011 - 111 海明距离为3

海明 $d=5$. 检4 纠2

- 总结可靠性编码
  - 格雷码——作用，特点

    产生中间错

    典型格雷码—二进制码 (考)

    十进制数的格雷码

    步进码 ★设计模n格雷计数
  - 奇偶校验码——作用

    补一位异或算出

    偶校验
  - 海明校验码——作用 纠单错

    海明距离 信道角度

    海明码的位序 3→7

    校验位的生成 4→11 …

    利用校验和进行校验，确定是否出错及出错位置。

习题

第一章 数制和编码

1.1 进位计数制(多项式替代法)

1.2 各种进位计数制的相互转换(重点， 多项式替代法， 基数乘除法， 直接转换法)

1.3 带符号数的代码表示 正负数

1.4 带符号数的加减法(补码系统运算最简单)

1.5 十进制数的常用代码(8421码， 2421码， 余3码)

1.6 可靠性编码(格雷码， 典型格雷码， 奇偶校验码， 海明校验码)

数字逻辑电路

第一章 数制和编码

1.1 进位计数制(多项式替代法)

1.2 各种进位计数制的相互转换(重点， 多项式替代法， 基数乘除法， 直接转换法)

1.3 带符号数的代码表示

1.4 带符号数的加减法(补码系统运算最简单)

1.5 十进制数的常用代码(8421码， 2421码， 余3码)

1.6 可靠性编码(格雷码， 典型格雷码， 奇偶校验码， 海明校验码)