第一讲 数据编码和布尔函数
Code & Boolean Function

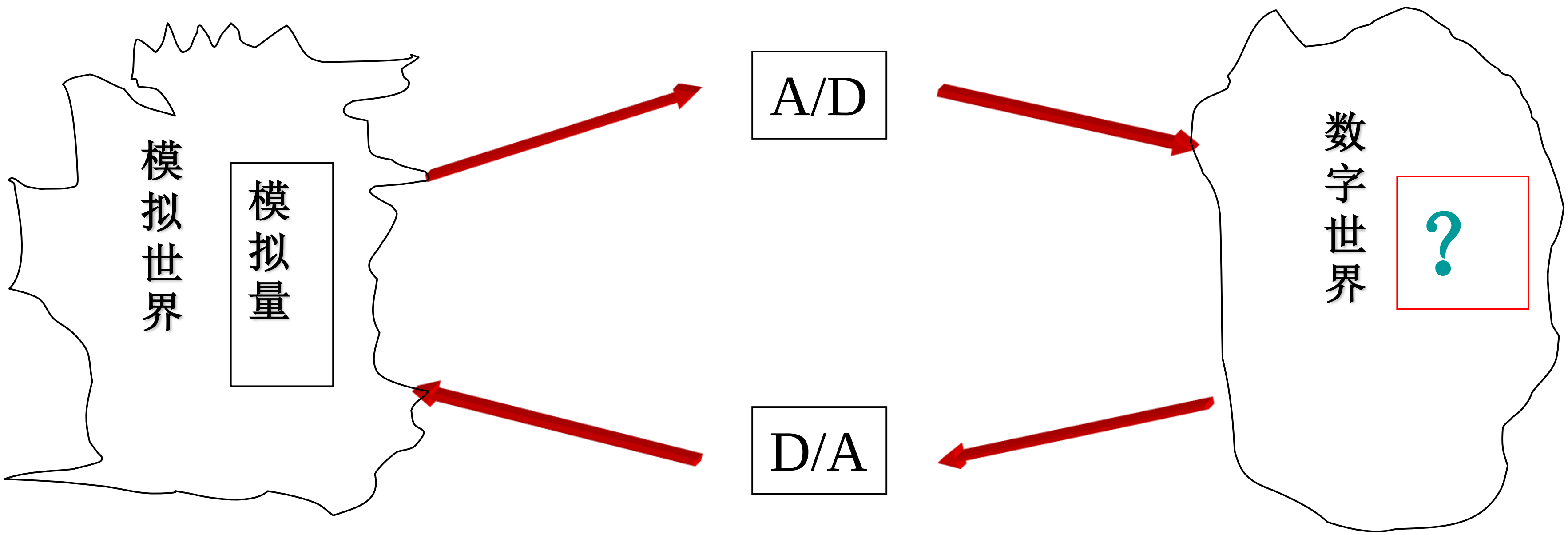
佟冬
tongdong@pku.edu.cn

微处理器研究开发中心（MPRC）
计算机科学技术系

北京大学

课程复习

- 数字系统
  - 对数字处理和存储的系统



抽象：复杂系统的设计方法

- 抽象：隐藏细节
- 限制：遵守约束
- 三“化”原则
  - 层次化Hierarchy
  - 模块化Modularity
  - 规整化Regularity

应用问题

算法

编程语言、程序

编译器、运行时环境

操作系统、虚拟机

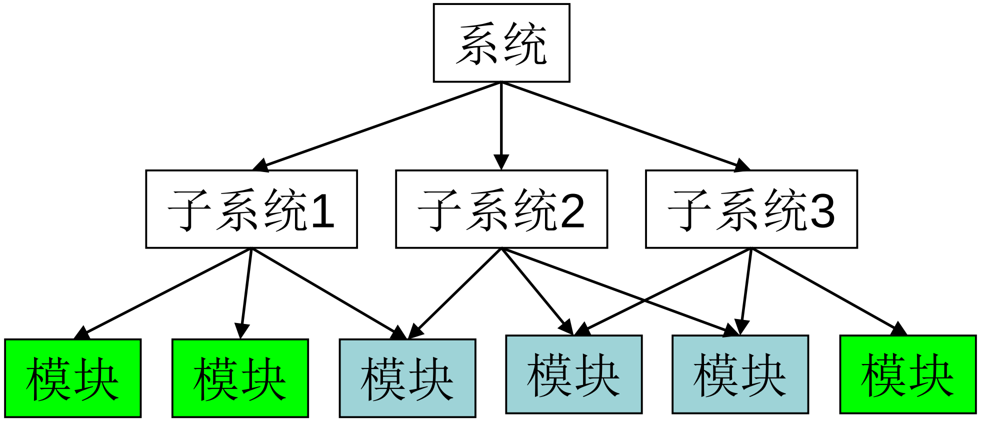
指令系统体系结构

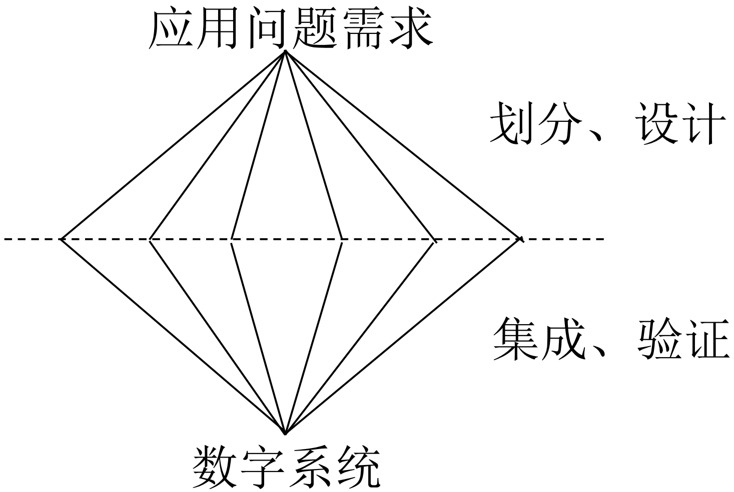
微体系结构、组成

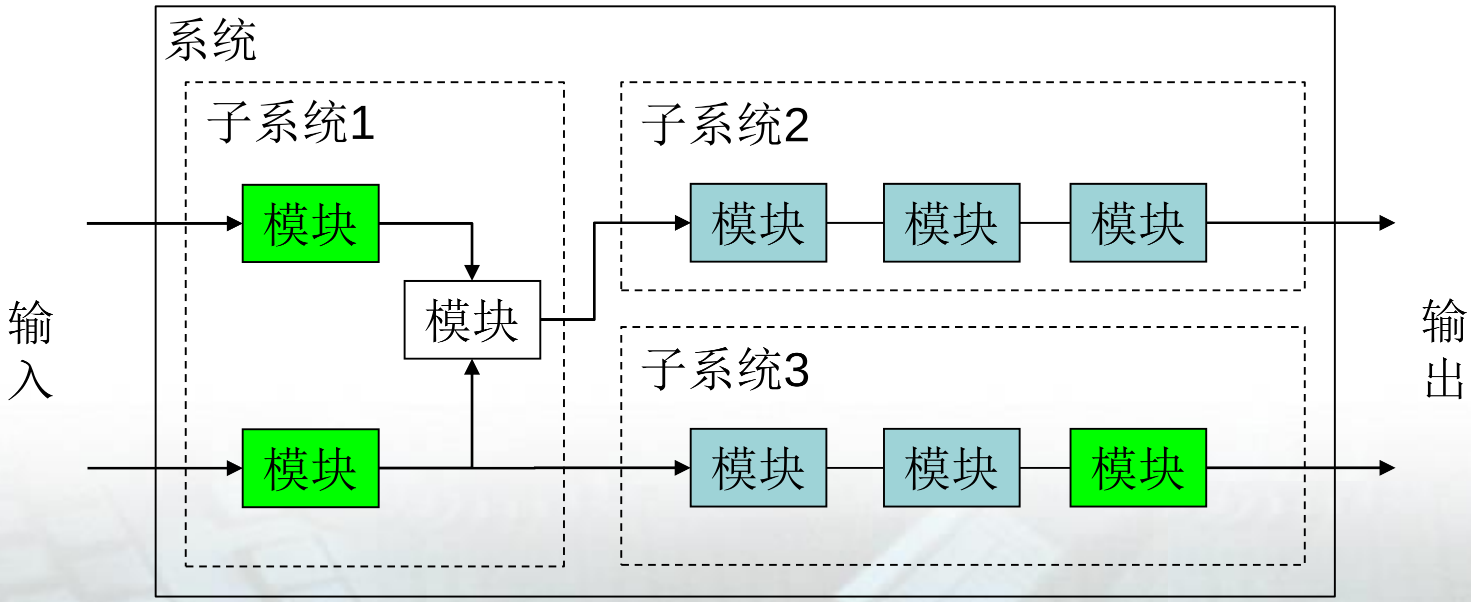
数字设计课程的首要内容 → 逻辑门电路、数字设计

模拟电路、电子器件、物理材料

三“化”原则：层次化、模块化和规整化







设计框图

如何做一个能计算的设备?

人的计算方法

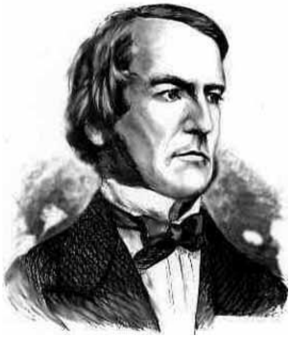
计算方法描述（输入、输出、功能）

↓ ?

计算设备

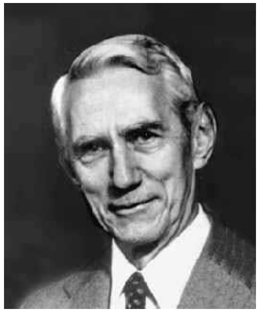
数字设计的基础

- 1850s, George Boole
  - 将逻辑表述映射到符号
  - 采用数学的方法处理逻辑推理



"An investigation into the Laws of Thought"

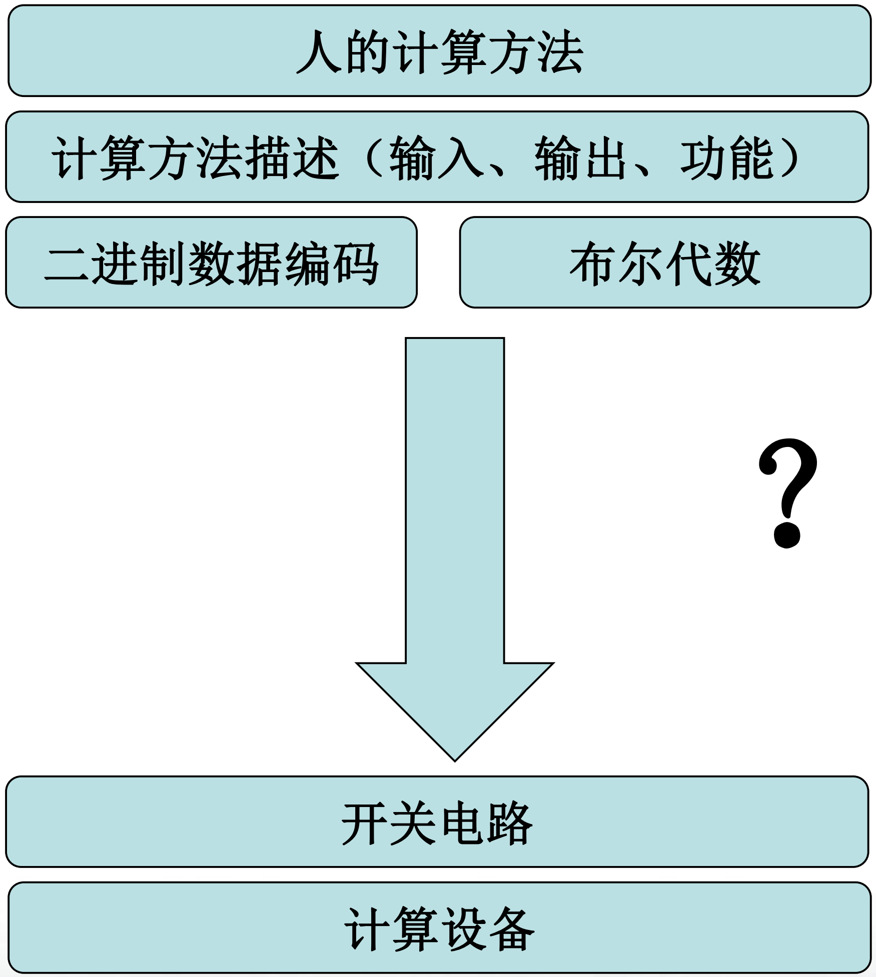
- *1938s,* Claude Elwood Shannon
  - 将布尔代数和硬件开关相联系
  - 第一次提出bit（比特）表示信息



"A Symbolic Analysis of Relay and Switching Circuits(1938)" Master thesis in MIT.

  - 1941香农加入AT&T Bell 实验室
  - *"The mathematical theory of communication"*, 1948

如何做一个能计算的设备?



1. 数制和编码

内容

- 数制
- 运算
- 数制转换
- 计算机中数的表示
- 计算机编码

1.1 数制

- 数制（如，十进制）
  - 符号的有序集(数字集：0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  - 关系（基本运算：加+、减-、乘×、除÷）
- 书写方法：

$$N = 2312.98$$

  - 多项式表示：

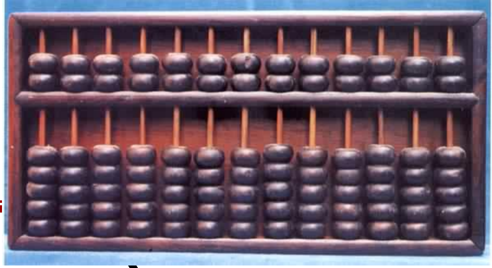
$$N = 2\times10^3+3\times10^2+1\times10^1+2\times10^0+9\times10^{-1}+8\times10^{-2}$$

  - 一般的表示方法（按位记数法）：

$$N = a_{n-1}a_{n-2}\ldots a_1a_0.a_{-1}a_{-2}\ldots a_{-m}$$

$$N = a_{n-1}\times10^{n-1}+\ldots+a_0\times10^0+a_{-1}\times10^{-1}+\ldots+a_{-m}\times10^{-m}$$

按位记数法（*r* 进制）



$$N = (a_{n-1}a_{n-2}\ldots a_1a_0.a_{-1}a_{-2}\ldots a_{-m})_r$$

其中：

. —— 基点（小数点）

r —— 基数

n —— 整数位数

m —— 小数位数

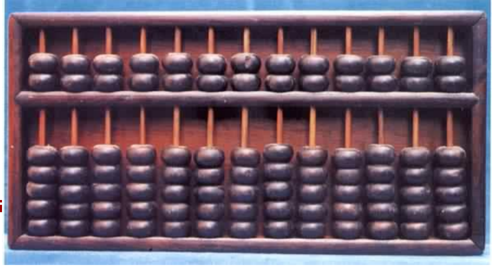
a_i —— 整数部分第 i 位， $n\text{-}1 \geq i \geq 0$

a_{-i} —— 小数部分第 i 位， $-1 \geq i \geq -m$

a_{n-1} —— 最高位

a_{m} —— 最低位

按位记数法（*r* 进制）



$$(a_{n-1}a_{n-2}\ldots a_1a_0.a_{-1}a_{-2}\ldots a_{-m})_r$$

$$N = a_{n-1}\times \underline{r^{\,n-1}}+\ldots+a_0\times r^{\,0}+a_{-1}\times r^{\,-1}+\ldots+a_{-m}\times r^{\,-m}$$

位权重

缩写方式： $N=\sum_{i=-m}^{n-1} a^i r^i$

数字系统中的常用的数制

- 十进制 Decimal
  - $r = 10$, (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- 二进制 Binary
  - $r = 2$, (0, 1)
- 八进制 Octal
  - $r = 8$, (0, 1, 2, 3, 4, 5, 6, 7)
- 十六进制 Hexadecimal
  - $r = 16$
  - (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

$(10, 11, 12, 13, 14, 15)_{10}$

- 特殊数制：混合位权数制
  - 时间（年 ： 月 ： 日 ： 小时 ： 分钟 ： 秒 ： 毫秒）

二进制数

- 基数：$r = 2$
- 数字集：(0, 1)

$$(10\text{1}0110000110100)_2$$

bit 比特

6 901234 567892

八进制数和十六进制数——二进制缩写形式

- 八进制数
  - $r = 8 = 2^3$

$$(100\ 001\ 111\ 000\ 001)_2$$

$$(41701)_8$$

- 十六进制数
  - $r = 16 = 2^4$

$$(0100\ 0011\ 1100\ 0001)_2$$

$$(43C1)_{16}$$

1.2 算数运算

- 基本的四则运算法则
  - 加+
  - 减-
  - 乘×
  - 除÷
- 运算中的进位规则

- 特殊数制：混合位权数制
  - 小时（24）：分钟（60）：秒（60）：毫秒（1000）

十进制四则运算

- 加法运算表
- 乘法运算表
- 加法进位规则
  - 逢十进一
  - 借一当十
- 乘法进位规则

$$\begin{array}{r} 8521 \\ +\,1299 \\ \hline 9820 \end{array} \qquad \begin{array}{r} 8521 \\ -\,1299 \\ \hline 7222 \end{array}$$

二进制运算

- 加法运算表
- 乘法运算表

加法运算
$0+0=\ \ 0$
$0+1=\ \ 1$
$1+0=\ \ 1$
$1+1=10$

乘法运算
$0\times 0=0$
$0\times 1=0$
$1\times 0=0$
$1\times 1=1$

- 加法进位规则：逢二进一、借一当二

R进制的运算规则

- 加法运算表
- 乘法运算表
- 加法进位规则
  - 逢R进一、借一当R
- 十六进制运算(r =16)

$$
\begin{array}{r}
8\ 5\ 2\ F \\
+\ 1\ 2\ 9\ 9 \\
\hline
9\ 7\ C\ 8
\end{array}
$$

*R*进制的转换

- 转换方法
  - 逐步代入法
  - 除数取余法和乘数取整法
- *A*进制转十进制

  将*R*进制数$(a_{n-1}a_{n-2}\ldots a_1a_0.a_{-1}a_{-2}\ldots a_{-m})_A$ 中的参数代入公式：

$$N = a_{n-1} \times R^{n-1} + \ldots + a_0 \times R^0 + a_{-1} \times R^{-1} + \ldots + a_{-m} \times R^{-m}$$

R进制向十进制转换

- 方法：代入法
- 二进制到十进制的转换($r = 2$)

  $N = (1011)_2$

  $N = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = (11)_{10}$
- 八进制到十进制的转换($r = 8$)

  $N = (127)_8$

  $N = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 = (87)_{10}$
- 十六进制到十进制的转换($r = 16$)

  $N = (15F)_{16}$

  $N = 1 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (351)_{10}$

A进制向B进制的转换

- 方法：除数取余法和乘数取整法
- 整数部分采用除数取余法（连除法）

$$(N)_A = b_{n-1}B^{n-1}+\dots+b_1B^1+b_0B^0$$

$$N/B = b_{n-1}B^{n-2} + \dots + b_1B^0 + \underbrace{(b_0B^{-1})}_{\text{余数}}$$

- 小数部分采用乘数取整法（连乘法）

$$(N)_A = b_{-1}B^{-1}+ b_{-2}B^{-2}\dots + b_{-m}B^{-m}$$

$$N\times B = \underbrace{b_{-1}B^0}_{\text{整数}} + b_{-2}B^{-1} + \dots + b_{-m}B^{-(m-1)}$$

通用的进制之间转换算法

- 命题：将一个A进制数转换为B进制
- 算法一：
  - 采用逐步代入法（B=10）；
  - 采用除数取余法和乘数取整法；
- 算法二：
  - 先将A进制数转换成十进制数；
  - 再将十进制数转换成B进制数。

十进制向A进制的转换

- 十进制转换成二进制
- 十进制转换成八进制
- 十进制转换成十六进制

Divisor	Dividend	Remainder	
16	315	B	LSD
16	19	3	
16	1	1	MSD
	0		

Divisor	Dividend	Remainder	
8	315	3	LSD
8	39	7	
8	4	4	MSD
	0		

LSD：Least Significant Digit 最低有效位
MSD：Most Significant Digit 最高有效位

A向A^k进制数的转换

- 设N为一个A进制数，将N中的每k个数字分为一组，每组直接用一个A^k进制数代替。
- 整数从基点向高位分组，不够补0。
- 小数从基点向低位分组，不够补0。

- 应用：二进制到八进制和十六进制的转换
  - $(001\ 010\ 111.\ 100)_2 = (127.4)_8$ （3位一组）
  - $(1011\ 0110\ 0101\ 1111)_2 = (B65F)_{16}$ （4位一组）

A^k进制数向A进制数的转换

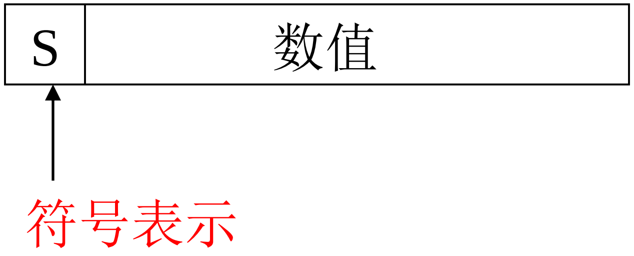
- 设N为一个A^k进制数，将N中的每个数字直接用k个A进制数代替。
- 应用：八进制和十六进制到二进制的转换

$$(127.4)_8 = (1\ 010\ 111.\ 1)_2$$

$$(B65F)_{16} = (1011\ 0110\ 0101\ 1111)_2$$

1.4 有符号数的表示

- 数的符号
  - 正数：+（或省略）
  - 负数：-
- 在数字系统中用数字表示符号称为符号数字
- 数的符号位：数的最高位
- 数的符号在*r*进制数中的表示
  - 正数：0
  - 负数：*r*-1



- 数的符号在二进制数中的表示
  - 正数：0
  - 负数：1

原码(Sign-and-Magnitude)

- r进制数 $N = \pm(a_{n-1}a_{n-2}\ldots a_1a_0.a_{-1}a_{-2}\ldots a_{-m})_r$的原码表示：

$$N = (sa_{n-1}a_{n-2}\ldots a_1a_0.a_{-1}a_{-2}\ldots a_{-m})_{原}$$

其中，符号位$s = 0$表示正数，$s = r-1$表示负数

- 例题：$-(11)_{10}$的2进制原码和10进制原码
  - 二进制$(1, 1011)_2$
  - 十进制$(9, 11)_{10}$
- 在原码中存在+0和-0，0,0…0和r-1,0…0
- 原码是最直接的表示，在数字系统中并不常用
  - 需要更多的电路来实现。
  - 实现电路比其它形式的电路慢。

补运算(Radix Complements)

- r进制数 $(N)_r$ 的补定义为：

$$[N]_r = r^n-(N)_r$$

其中，n是$(N)_r$中数字的个数（位数）

- n位r进制数的补的范围
  - 最大正数 $r^{n-1}-1$
  - 最小负数 $-r^{n-1}$
- 二进制的补(two’s complements)

$$[N]_2 = 2^n-(N)_2$$

- 补码的多项式表示

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

补运算（续）

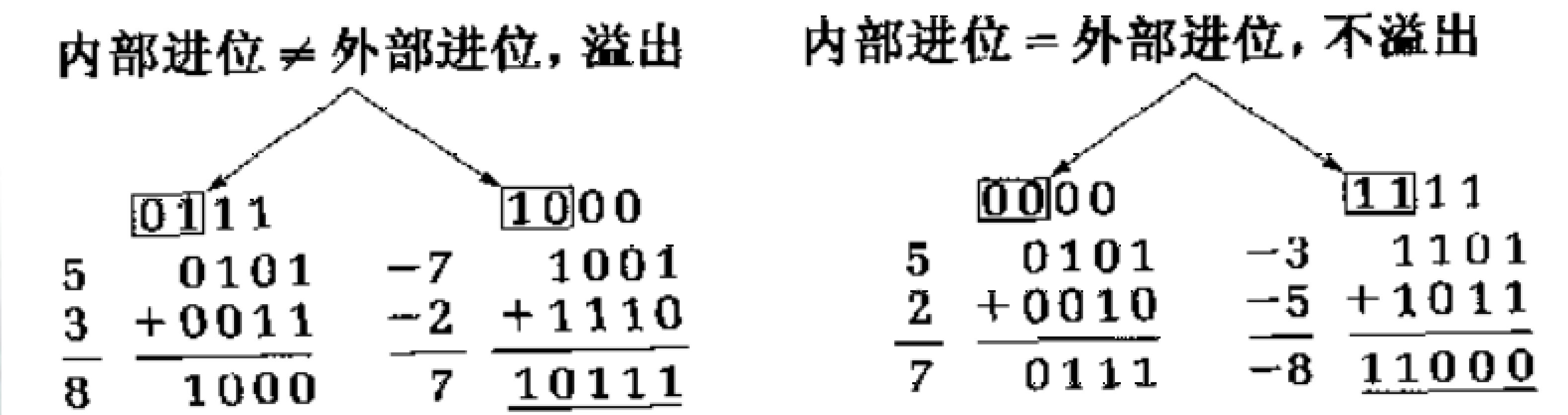
- 求补算法一：$r^n-(N)_r$
- 求补算法二：
  - 命题：求$(N)_r$的补$[N]_r$
  - 对于$(N)_r$中的数字，从最低位向最高位寻找第一个非0的数字a_i，将该数字用$r-a_i$替换，将剩余的每个数字a_j用$(r-1)-a_j$替换。
- 求补算法三：每位数字a_i用$(r-1)-a_i$替换，末位加一。
- 二进制求补：每位取反，末位加1。

补码（Twos-complement, 2补码）

- 正数
  - $N = +(a_{n-2},\ldots\ldots,a_0)_2 = (0, a_{n-2}, \ldots\ldots,a_0)_{补}$
  - $0 \geq N \geq 2^{n-1}-1$
- 负数 $N = (a_{n-1}a_{n-2}\ldots a_1a_0)$
  - -N 的补码用$[a_{n-1}a_{n-2}\ldots a_1a_0]_r$表示
  - $-1 \geq N \geq -2^{n-1}$
- 特别的说明：$(1,0\ldots\ldots0)_{补}$表示-2^{n-1}。
  - 有一定的数学意义
  - 没有重码
- 补码的表示范围： $2^{n-1}-1 \geq N \geq -2^{n-1}$

二进制补码运算

- 补码用来将减法转换成加法，减少机器的硬件。
  - A-B = A + (-B)
- 数的机器表示
  - 用一定有限数量的比特表示数字
  - n比特能表示2^n个数
  - 补码的表示范围：$2^{n-1}-1 \ge N \ge -2^{n-1}$
- 溢出(overflow)
  - 上溢：$N > 2^{n-1}-1$，或者
  - 下溢：$N < -2^{n}-1$



补码运算（续）

- 如果， $B \ge 0, C \ge 0$
- $A = B + C; (A)_2 = (B)_2 + (C)_2$
- $A = B - C; (A)_2 = (B)_2 + [C]_2$；忽略进位
- $A = - B - C; (A)_2 = [B]_2 + [C]_2$；

Case	Carry	Sign Bit	Condition	Overflow ?
B + C	0	0	$B + C \le 2^{n-1} - 1$	No
	0	1	$B + C > 2^{n-1} - 1$	Yes
B - C	1	0	$B \le C$	No
	0	1	$B > C$	No
-B - C	1	1	$-(B + C) \ge -2^{n-1}$	No
	1	0	$-(B + C) < -2^{n-1}$	Yes

补码运算（续）

- 在用补码表示计算中
  - 加法：两个数相加
  - 减法：加上减数的补
  - 忽略最高位进位，判断溢出
    - 同号相加结果异号
    - 最高位进位和次高位进位不同

  - 以上的计算机方法可以延伸到任何*r*进制运算中。

1.5 计算机编码

- 编码(Code)
  - 利用给定符号集合对信息的一种系统的、标准化的表示。
  - 采用二进制编码: $N \le 2^n$，n=log2(N)
  - 编码的可计算性
  - 计算机中的编码
    - 数据编码
    - 指令编码
    - 字符编码
    - 检错码和纠错码
    - 其他编码（汉字、多媒体）
  - 如何用二进制表示年、月、日、小时、分钟、秒?

数字编码

- 表示计算机中的数
- 定点表示和浮点表示
- 定点表示
  - 表示有符号的整数或有符号小数
  - 可用原码，**补码**和反码表示负数
  - 定点整数：小数点位置默认在最低位右面
  - 定点小数：小数点位置默认在符号位和最高数位之间
- 余K码（移码）
  - 经常用于浮点的指数表示（余8码）

浮点表示

- 科学记数法

$$N = M \times r^{E}$$

M —— 尾数；E —— 指数

- *IEEE* 754标准

S_M	Exponent E	Mantissa M

↑ Sign of mantissa

单精浮点

S_M	Exponent E	Mantissa M (most significant part)

Mantissa M (least significant part)

双精浮点

RISC-V 指令编码

imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

RISC-V 指令编码（续）

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW

字符和其它编码

- 二进制表示数字、字母、字符串等等
- 十进制的二进制编码（Binary-coded Decimal, BCD码）
  - 表示十进制数字0-9
  - 用4位二进制表示
  - 加权码(weighted code)
  - 8421码
  - 一个8421码 ($a_3a_2a_1a_0$)表示的数字是：

$$a_3 \times 8 + a_2 \times 4 + a_1 \times 2 + a_0 \times 1$$

8-4-2-1码（BCD码）例子

- 8-4-2-1码的$(0111)_{8421}$表示7

$$0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 7$$

- 8-4-2-1码的对应表

0: 0000 1: 0001 2: 0010 3: 0011 4: 0100

5: 0101 6: 0110 7: 0111 8: 1000 9: 1001

- 十进制数表示

$$(5120)_{10}$$

$$(0101\ 0001\ 0010\ 0000)_{BCD}$$

BCD码的加法

- 考虑两个4位BCD码的加法
  - 考虑直接采用二进制加法
  - 如果二进制加法结果小于9，没有溢出，就是和的BCD码本身
  - 如果二进制加法结果大于9，溢出了，怎么办?
  - 将结果再加上6（0110），分两种情况要考虑
    - 结果小于16，大于9
    - 结果大于16

$$\begin{array}{r} 5 = 0101 \\ 3 = 0011 \\ \hline 1000 = 8_{10} \end{array}$$

$$\begin{array}{r} 5 = 0101 \\ 8 = 1000 \\ \hline 1101 = 13_{10}\ \text{溢出} \\ +\ 0110 \\ \hline 1\ 0011 = 13_{BCD} \end{array}$$

$$\begin{array}{r} 9 = 1001 \\ 8 = 1000 \\ \hline 1\ 0001 = 17_{10}\ \text{溢出} \\ +\ 0110 \\ \hline 1\ 0111 = 17_{BCD} \end{array}$$

二进制与BCD码的转换

- 采用如下图的数据结构
  - 1. 二进制数左移1位
  - 2. 如果8位移位完成，BCD码就在各列中，结束。
  - 3. 否则，如果任何列中的数值大于或等于5，该列加3
  - 4. 跳到步骤1

操作	百位	十位	个位	二进制	
十六进制				A	B
开始状态				1 0 1 0	1 0 1 1

8位二进制到BCD码的转换

操作	百位	十位	个位	二进制	
十六进制				A	B
开始				1 0 1 0	1 0 1 1
移位3次			1 0 1	0 1 0 1	1
加3			1 0 0 0	0 1 0 1	1
移位4		1	0 0 0 0	1 0 1 1	
移位5		1 0	0 0 0 1	0 1 1	
移位6		1 0 0	0 0 1 0	1 1	
移位		1 0 0 0	0 1 0 1	1	
加3		1 0 1 1	1 0 0 0	1	
移位	1	0 1 1 1	0 0 0 1		
结束 BCD	1	7	1		

思考：BCD码如何转二进制？其它特殊进制的算法呢？

ASCII编码

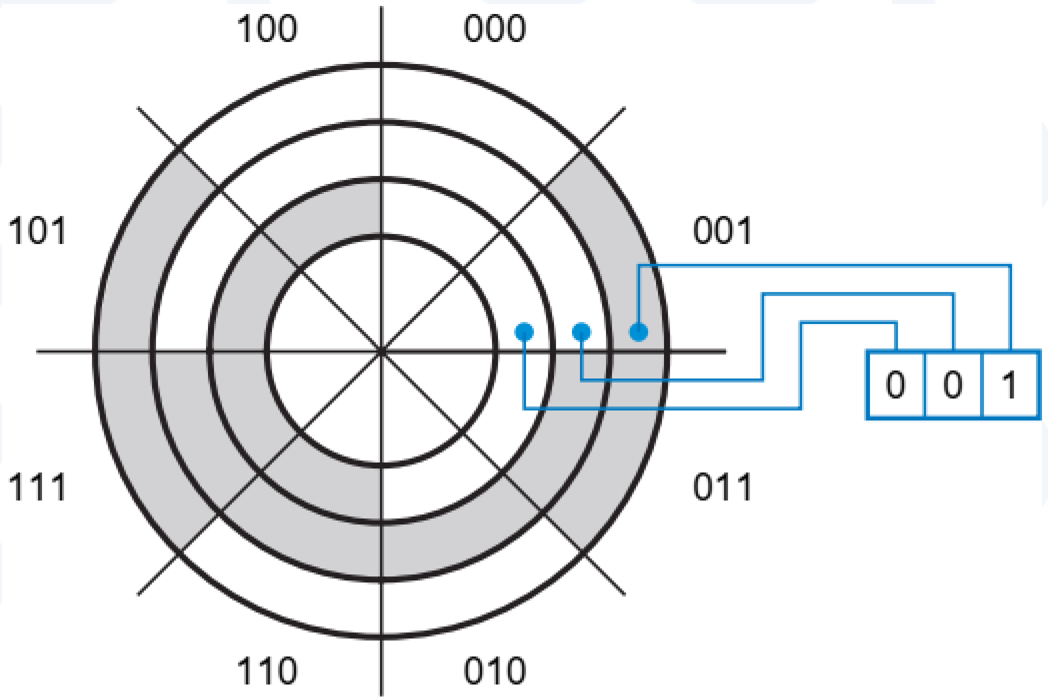
- 时间上使用最广泛的符号编码
- 7-bit ASCII 编码
- 第8位经常用来检错（校验位）
- 例子：***Digital*** 的ASCII表示

字母	二进制编码	十六进制编码
D	1000100	44
i	1101001	69
g	1100111	67
i	1101001	69
t	1110100	74
a	1100001	61
l	1101100	6C

格雷码(Gray Code)

- 格雷码(Gray Code)
  - 问题的提出：
  - 是一种循环码
  - 相邻数字的编码只有一位不同（距离为1）

二进制	格雷码
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 1
0 1 1	0 1 0
1 0 0	1 1 0
1 0 1	1 1 1
1 1 0	1 0 1
1 1 1	1 0 0

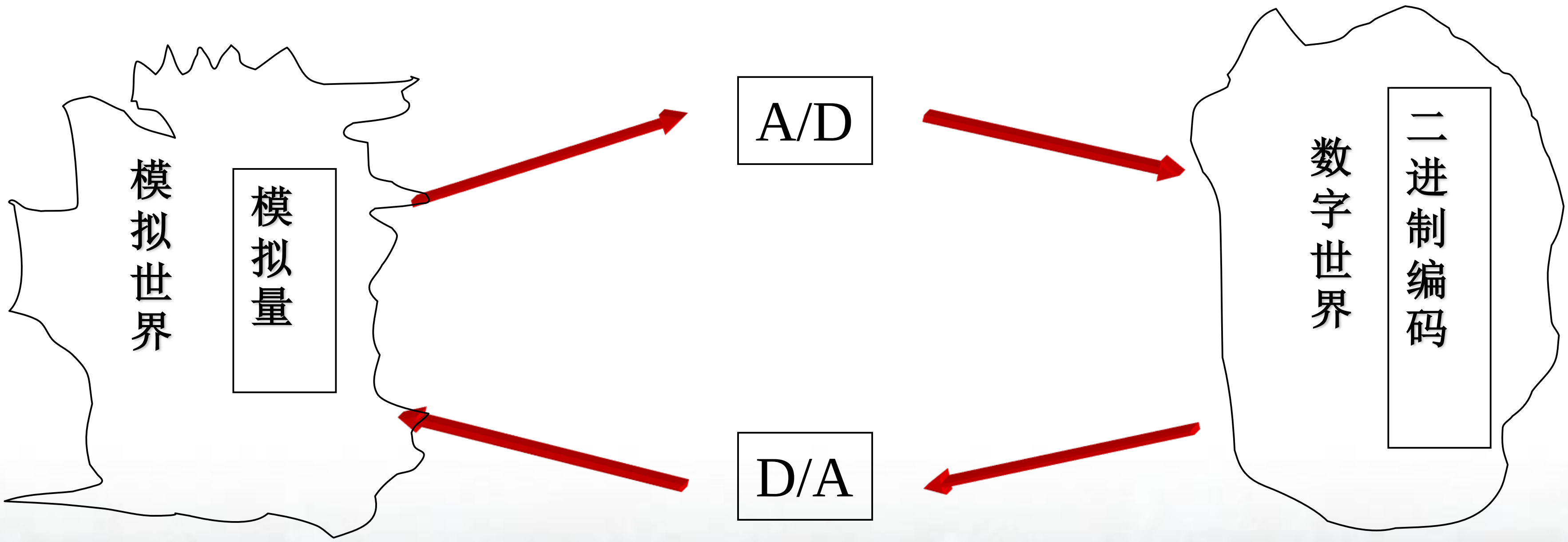


常用编码

- 汉字编码
  - 16位二进制表示一个汉字
  - 国家标准 GB2312-80 7445个编码
  - 图形符号：682个；汉字6763个
  - 一级字库（常用汉字） 3755个
  - 二级字库 （次常用汉字）3008个
  - 例：“啊” 0110000 0100001

其它数字编码：数字媒体编码

- 多媒体
  - 通讯信号
  - 图像
  - 声音



纠错码和检错码的两个概念

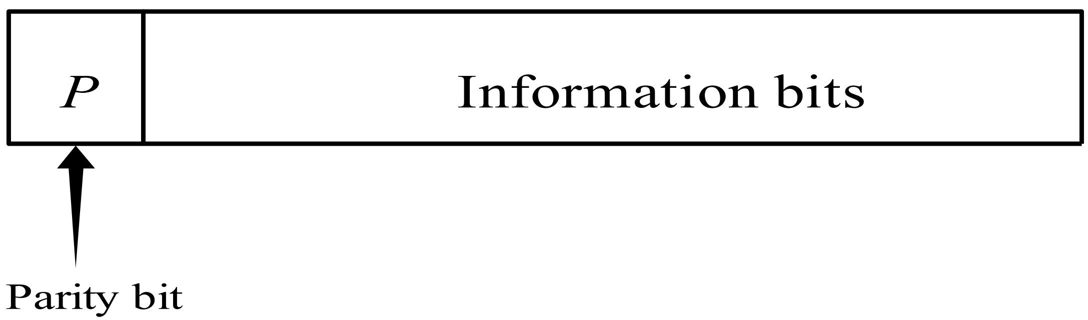
- 模拟信道中的噪音
- 设*I* 和 *J* 是n位信息编码。
  - $w(I)$: *I*中1的个数 (重量，*weight*)
  - $d(I, J)$: *I* 和 *J* 不同位的个数(距离，*distance*)
- 例: I = (01101100) and J = (11000100)
  - $w(I) = 4$ and $w(J) = 3$
  - $d(I, J) = 3$.
- 最小距离$d_{\min}$
- 如果信息编码提供*t* 位的纠错能力和附加的*s* 位的检错能力，一般的：$2t + s + 1 \le d_{\min}$

校验码和纠错码

- 错误: 一或多个比特的值不正确
- 产生错误的原因:
  - 硬件错误
  - 外部接口（噪声）
  - 其它事件
- 校验码/纠错码:可以检测或者修正某这特定类型错误的信息编码。

奇偶校验码

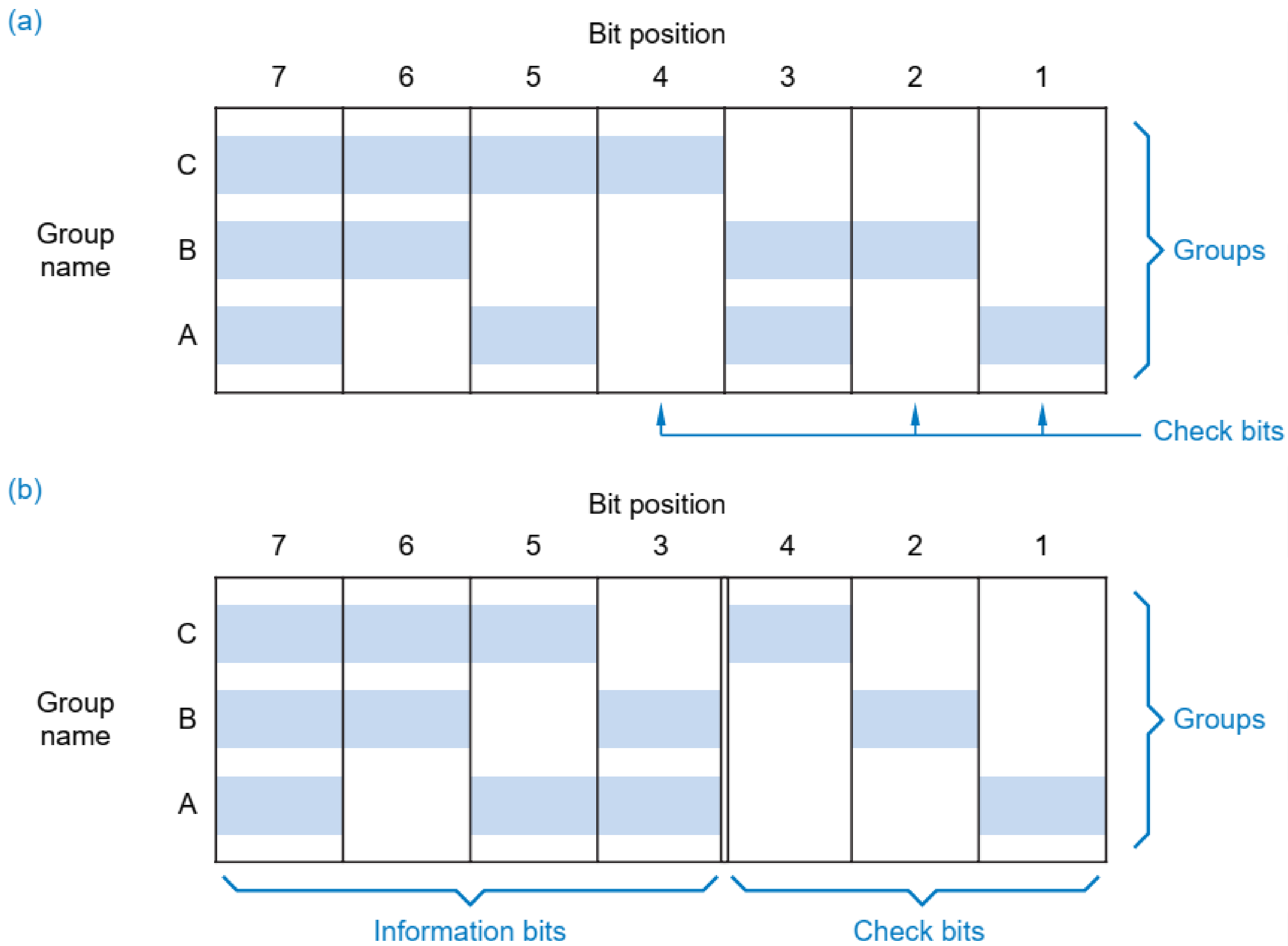
- 一位奇偶校验码
  - 在编码C最前或最后加一位校验位*P*。
  - 形成的编码记为P|C或C|P。
  - 奇校验：W(P|C) 是奇数
  - 偶校验： W(P|C) 是偶数



汉明码

- 理查德·汉明，1968年的ACM图灵奖
- 采用多个校验位，每位根据一个信息位子集决定
- 子集之间相互重叠，保证每个信息位至少在两个子集中出现。
- 具有检错和一定的纠错能力。
- 例子：[7,4]二进制汉明码(x7,x6,x5,c4,x3,c2,c1)
  - 校验位1：1,3,5,7的偶校验, (c1,x3,x5,x7)偶校验
  - 校验位2：2,3,6,7的偶校验, (c2,x3,x6,x7)偶校验
  - 校验位4：4,5,6,7的偶校验, (c4,x5,x6,x7)偶校验
  - c4:c2:c1组成的二进制数为出错的二进制位
  - C_i为第*i* 组是否通过校验, i ＝ 1,2,4

汉明码


(a)
Bit position
7
6
5
4
3
2
1
Group name
C
B
A
Groups
Check bits
(b)
Bit position
7
6
5
3
4
2
1
Group name
C
B
A
Groups
Information bits
Check bits

如何做一个能计算的设备?

人的计算方法

计算方法描述（输入、输出、功能）

二进制数据编码

布尔代数

?

开关电路

计算设备

2. 布尔代数

公理(1)

- 公理1：（封闭性）布尔代数是一个封闭的代数系统，它包含一个集合*B*，*B*中包括两个以上的元素；在集合*B*上定义两个二元操作{•}和{+}，一个一元操作{’}分别称为与(AND)、或(OR)和非(NOT)；对集合*B*中的任意两个元素*a*和*b*，有：
  - $a \bullet b$ 是*B*中的元素
  - $a + b$ 是*B*中的元素

- 类比
  - AND和乘法相类比
  - OR和加法相类比

公理(2)

- 公理2：（交换律）对于任何B中的元素 a 和 b：
  - (a) $a + b = b + a$
  - (b) $a \bullet b = b \bullet a$

- 公理3：（结合律）对于任何B中的元素 a, b, c：
  - (a) $a + (b + c) = (a + b) + c$
  - (b) $a \bullet (b \bullet c) = (a \bullet b) \bullet c$

公理(3)

- 公理4：（恒等性）在集合B中存在唯一的元素1和0，对集合B中的每个元素有：
  – (a) $a + 0 = a$
  – (b) $a \bullet 1 = a$

- 0对于+操作是恒等单元
- 1对于•操作是恒等单元

公理(4)

- 公理5：（分配律）对于任何B中的元素 a, b, c：
  - (a) $a + (b \bullet c) = (a + b) \bullet (a + c)$;
  - (b) $a \bullet (b + c) = (a \bullet b) + (a \bullet c)$;
- 公理6：（互补）对于任何B中的元素 a，在B中存在唯一的元素a'：
  - (a) $a + a' = 1$
  - (b) $a \bullet a' = 0$
- 为了书写方便，可以将•省略：$ab = a \bullet b$
- a' 一般也可以表示成 $\bar{a}$

布尔代数——定理(1)

- 对偶原理
- 定理1（幂等性）
  - (a) $X + X = X$
  - (b) $X \bullet X = X$
- 定理2（空单元）
  - (a) $X + 1 = 1$
  - (b) $X \bullet 0 = 0$
- 定理3（自反率）
  - $(X')' = X$

0和1元素的性质

0和1元素的性质

OR	AND	Complement
$X + 0 = X$	$X \bullet 0 = 0$	$0' = 1$
$X + 1 = 1$	$X \bullet 1 = X$	$1' = 0$

二进制加法运算	二进制乘法运算	布尔代数或运算	布尔代数与运算
$0 + 0 = 0$	$0 \times 0 = 0$	$0 + 0 = 0$	$0 \bullet 0 = 0$
$0 + 1 = 1$	$0 \times 1 = 0$	$0 + 1 = 1$	$0 \bullet 1 = 0$
$1 + 0 = 1$	$1 \times 0 = 0$	$1 + 0 = 1$	$1 \bullet 0 = 0$
$1 + 1 = 10$	$1 \times 1 = 1$	$1 + 1 = 1$	$1 \bullet 1 = 1$

定理(2)

- 定理4（吸收率）
  - (a) $X + XY = X$
  - (b) $X(X + Y) = X$
- 定理5
  - (a) $X + X'Y = X + Y$
  - (b) $X \bullet (X' + Y) = XY$
- 定理6
  - (a) $XY+XY'=X$
  - (b) $(X+Y)(X+Y')=X$

定理(3)

- 证明：T4(b)

$$
\begin{aligned}
X(X+Y) &= (X+0)(X+Y) \\
&= X + 0 \bullet Y \\
&= X + Y \bullet 0 \\
&= X + 0 \\
&= X
\end{aligned}
$$

- T5(a)例：

$$B + AB'C'D = B + AC'D$$

定理(4)—摩根定理

- 定理7（摩根定理）
  - (a) $(X+Y)'=X'Y'$
  - (b) $(XY)' = X'+Y'$
- 一般的摩根定理
  - (a) $(X+Y+Z+\ldots)'=X'Y'Z'\ldots$
  - (b) $(XYZ\ldots)'=X'+Y'+Z'\ldots$
- 摩根定理的使用：求布尔表达式的反

摩根定理举例

- 例：$\overline{a(b+c)+\overline{a}b}$

$$
\begin{aligned}
\overline{a(b+c)+\overline{a}b} &= \overline{a(b+c)}\cdot\overline{\overline{a}b} \\
&= (\overline{a}+\overline{(b+c)})\cdot(\overline{\overline{a}}+\overline{b}) \\
&= (\overline{a}+\overline{b}\cdot\overline{c})\cdot(a+\overline{b}) \\
&= (\overline{a}+\overline{b}\cdot\overline{c})\cdot a+(\overline{a}+\overline{b}\cdot\overline{c})\cdot\overline{b} \\
&= \overline{a}a+\overline{b}\,\overline{c}a+\overline{a}\overline{b}+\overline{b}\,\overline{c}\overline{b} \\
&= a\overline{b}\,\overline{c}+\overline{a}\overline{b}+\overline{b}\,\overline{c} \\
&= (a\overline{c}+\overline{a})\overline{b}+\overline{b}\,\overline{c} \\
&= \overline{b}(\overline{a}+\overline{c})
\end{aligned}
$$

定理(5)

- 定理8：对偶原理
  - 将一个布尔等式中的 • 和 + 互换，0和1互换，所得结果仍然是一个布尔等式
  - 例：$X + 0 = X$

    $Y \bullet 1 = Y$

定理(6)

- 定理9
  - (a) $(X+Y)(X'+Z)=XZ+X'Y$
  - (b) $XY+X'Z=(X+Z)(X'+Y)$

- 定理10
  - (a) $XY+YZ+X'Z=XY+X'Z$
  - (b) $(X+Y)(Y+Z)(X'+Z)=(X+Y)(X'+Z)$

定理证明

- 证明T10
  - $(X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = X \cdot Y + X' \cdot Z$

	$(X \cdot Y) + (Y \cdot Z) + (X' \cdot Z)$
恒等律	$(X \cdot Y) + (1) \cdot (Y \cdot Z) + (X' \cdot Z)$
互补律	$(X \cdot Y) + (X' + X) \cdot (Y \cdot Z) + (X' \cdot Z)$
分配律	$(X \cdot Y) + (X' \cdot Y \cdot Z) + (X \cdot Y \cdot Z) + (X' \cdot Z)$
交换律	$(X \cdot Y) + (X \cdot Y \cdot Z) + (X' \cdot Y \cdot Z) + (X' \cdot Z)$
分配律	$(X \cdot Y) \cdot (1 + Z) + (X' \cdot Z) \cdot (1 + Y)$
空单元	$(X \cdot Y) \cdot (1) + (X' \cdot Z) \cdot (1)$
恒等律	$(X \cdot Y) + (X' \cdot Z)$

布尔函数（开关函数）

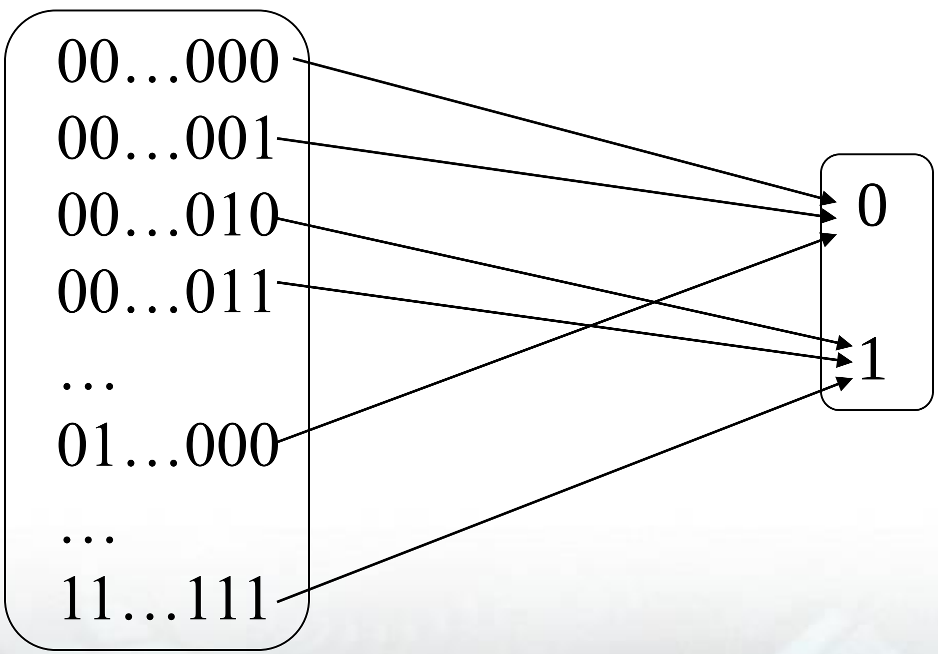
开关函数定义

- 开关函数：将一组在{0, 1}上取值的输入变量唯一映射到取值集合{0, 1}的输出变量的。
- 开关函数：设$X_1, X_2, \ldots, X_n$是n个变量，每个变量取值 0 或者取值 1，令$f(X_1, X_2, \ldots, X_n)$ 是$X_1, X_2, \ldots, X_n$的一个开关函数，f的取值 0 或 1 由$X_1, X_2, \ldots, X_n$的取值决定。
- 例：$f(A,B,C) = AB + A'C + AC'$
- 当A=0, B=1, C=1时，$f(0,1,1) = 0 \cdot 1 + 1' \cdot 0 + 1 \cdot 0' = 1$
- 布尔表达式：包含布尔变量和运算符的代数表示，不包含对输出变量的赋值。

开关函数

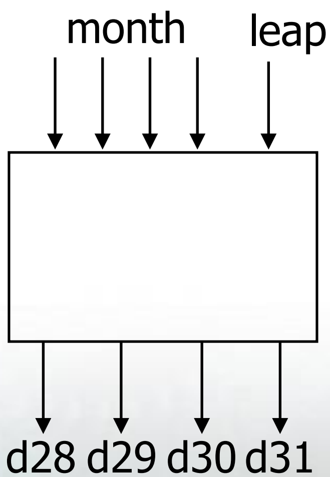
- 一个开关函数$F(X_1, X_2, \ldots, X_n)$

$(X_1, X_2, \ldots, X_n)$　　　　$F(X_1, X_2, \ldots, X_n)$



布尔函数的功能描述——真值表

- 规格说明书Specification
- 问题描述:
  - 输入输出的数量?
  - 输入输出的定义?
  - 二进制编码
- 功能行为:
  - 真值表
  - 布尔函数
  - 波形图
  - HDL



month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

真值表的实例

- 输入值相等判断
- 计算输入二进制中1的个数
- 判断输入二进制是否能被2、3、5整除
- 驱动7段译码显示器
- 二进制加法器
  - 半加器
  - 全加器
- 真值表：最根本的表示方式
  - 优点：最准确
  - 缺点：庞大

三个基本开关函数的真值表

- 将一个开关函数 *f* 对于其变量每种可能取值的结果用表的形式表示。
  - 1对应逻辑“真”；0对应逻辑“假”
- 三个基本函数：与(AND)、或(OR)、非(NOT)的真值表

$a\ b$	$f(a, b) = a+b$
0 0	0
0 1	1
1 0	1
1 1	1

OR

$a\ b$	$f(a, b) = ab$
0 0	0
0 1	0
1 0	0
1 1	1

AND

a	$f(a) = \overline{a}$
0	1
1	0

NOT

两变量函数

- 16种可能的函数

X	Y	16 种可能的函数 (F0–F15)															
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0

X and Y

X

Y

X xor Y

X or Y

X nor Y
not (X or Y)

X = Y

not Y

not X

X nand Y
not (X and Y)

1

真值表与函数

- $f(A, B, C) = AB + A'C + AC'$

ABC	$f(A, B, C)$
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	1

0 ⟶ F(假)

1 ⟶ T(真)

$A\ B\ C$	$f(A, B, C)$
F F F	F
F F T	T
F T F	F
F T T	T
T F F	T
T F T	F
T T F	T
T T T	T

开关函数的规范代数形式

- 积之和(Sum of product, SOP)

$$f(A,B,C)=AB+A'C+AC'$$

与项 或

之

积 和

- 和之积(Product of sum, POS）

$$f(A,B,C)=(A'+B+C)(B'+C+D')(A+C'+D)$$

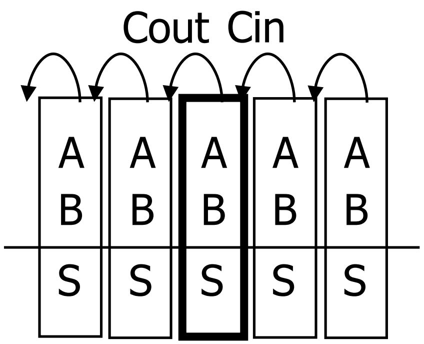
或项 与

之

和 积

例子：1位二进制加法器

- 输入: A, B, Carry-in
- 输出: Sum, Carry-out



A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$$

$$Cout = A' B Cin + A B' Cin + A B Cin' + A B Cin$$

用定理化简表达式

- 布尔代数的定理化简布尔表达式
  - 例：全加器的进位函数

$$
\begin{aligned}
\text{Cout} &= A' B\,Cin + A B' Cin + A B\,Cin' + A B\,Cin \\
&= A' B\,Cin + A B' Cin + A B\,Cin' + A B\,Cin + A B\,Cin \\
&= A' B\,Cin + A B\,Cin + A B' Cin + A B\,Cin' + A B\,Cin \\
&= (A' + A) B\,Cin + A B' Cin + A B\,Cin' + A B\,Cin \\
&= (1) B\,Cin + A B' Cin + A B\,Cin' + A B\,Cin \\
&= B\,Cin + A B' Cin + A B\,Cin' + A B\,Cin + A B\,Cin \\
&= B\,Cin + A B' Cin + A B\,Cin + A B\,Cin' + A B\,Cin \\
&= B\,Cin + A (B' + B) Cin + A B\,Cin' + A B\,Cin \\
&= B\,Cin + A (1) Cin + A B\,Cin' + A B\,Cin \\
&= B\,Cin + A\,Cin + A B (Cin' + Cin) \\
&= B\,Cin + A\,Cin + A B (1) \\
&= B\,Cin + A\,Cin + A B
\end{aligned}
$$

例题

- 完成真值表，实现电路功能：判别4位输入是否可以被2，3，5整除？

X8	X4	X2	X1	By2	By3	By5
0	0	0	0	1	1	1
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	1	0

- 写出By2、By3和By5的布尔表达式

开关函数的标准表示形式（范式）

- 最小项：如果一个函数有*n*个变量，如果一个积项中每个变量以补或非补的形式出现并且只出现一次，这个积项称为最小项。
- 如果一个积之和中的每个积项都是最小项，则称为最小项范式（析取范式，Disjunctive Normal Form, DNF）
- 每个最小项可以用一个二进制数表示
  - 变量用 1 表示
  - 补变量用 0 表示

最小项范式和最小项列表表示

$$f(A,B,C)=A'B'C'+A'BC'+ABC'+A'BC+ABC$$

最小项	编码	编号
$A'B'C'$	000	m_0
$A'BC'$	010	m_2
ABC'	110	m_6
$A'BC$	011	m_3
ABC	111	m_7

$$=m_0+m_2+m_6+m_3+m_7$$

$$=\sum m(0,2,3,6,7)$$

- 在开关函数的最小项表形式中，编码是十分重要的。
- 变量的顺序

真值表和最小项范式的关系

Row No. (i)	Inputs ABC	**Outputs** $f(A,B,C)=\Sigma m(2,3,6,7)$	**Complement** $f'(A,B,C)=\Sigma m(0,1,4,5)$
0	000	0	1 $\leftarrow$ m_0
1	001	0	1 $\leftarrow$ m_1
2	010	1 $\leftarrow$ m_2	0
3	011	1 $\leftarrow$ m_3	0
4	100	0	1 $\leftarrow$ m_4
5	101	0	1 $\leftarrow$ m_5
6	110	1 $\leftarrow$ m_6	0
7	111	1 $\leftarrow$ m_7	0

$$f(A,B,C)=A'BC'+A'BC+ABC'+ABC$$

$$f'(A,B,C)=A'B'C'+A'B'C+AB'C'+AB'C$$

补充

- 开关代数中操作的优先级
  - 与，或，非操作有优先级
  - 优先级顺序为：非，与，或
  - 突出问题：或操作对与操作也存在分配率
- $\bar{f}$ 称为*f* 的非函数或补函数
  - 注意：不能称为反函数
- 非操作相当于求负元素

例子

- $f(A,B,Q,Z) = A'B'Q'Z' + A'B'Q'Z + A'BQZ' + A'BQZ$, 用最小项表的形式表示$f(A,B,Q,Z)$ 和 $f'(A,B,Q,Z)$
  - $f(A,B,Q,Z) = A'\underbrace{B'Q'Z'}_{0000} + A'\underbrace{B'Q'Z}_{0001} + A'\underbrace{BQZ'}_{0110} + A'\underbrace{BQZ}_{0111}$

    $= m_0 + m_1 + m_6 + m_7$

    $= \Sigma\, m(0, 1, 6, 7)$
  - $f'(A,B,Q,Z) = m_2 + m_3 + m_4 + m_5 + m_8 + m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

    $= \Sigma\, m(2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15)$
- $f(A,B,Q,Z) + f'(A,B,Q,Z) = 1$

开关函数最小项范式表示

- 优点：
  - 开关函数的最接近真值表的表示形式
  - 能很容易地判断在一个输入组合下，开关函数的值是否为1。
- 不足：
  - 不是最简单的函数形式
  - 相关的电路复杂
- 需要化简

最大项和最大项范式

- 定义：如果一个函数有*n*个变量，如果一个和项中每个变量以补或非补的形式出现并且只出现一次，这个和项称为最大项。
- 如果一个函数用和之积的形式表示，其中的和项都是最大项，则该函数称为最大项范式（合取范式，Conjunctive Normal Form, CNF）。
- 最大项可以用二进制数表示
  - 非补变量用 0 表示
  - 补变量用 1表示

最大项范式的列表表示

$$f(A,B,C)=(\underbrace{A+B+C}_{000})(\underbrace{A+B+C'}_{001})(\underbrace{A'+B+C}_{100})(\underbrace{A'+B+C'}_{101})$$

最大项	编码	编号
$A+B+C$	000	M_0
$A+B+C'$	001	M_1
$A'+B+C$	100	M_4
$A'+B+C'$	101	M_5

$$=M_0 \bullet M_1 \bullet M_4 \bullet M_5$$

$$=\Pi M(0,1,4,5)$$

- 和最小项范式同理，变量的顺序是重要的。

真值表和最大项范式的关系

$$f(A,B,C) = (A+B+C')(A+B'+C')(A'+B+C')(A'+B'+C')$$
$$= A'B'C' + A'BC' + AB'C' + ABC'$$

Row No. (i)	**Inputs** ABC	**Outputs** $f(A,B,C) = \Pi M(1,3,5,7) = \Sigma m(0,2,4,6)$		
0	000	1		m_0
1	001	0	$\leftarrow$ M_1	
2	010	1		m_2
3	011	0	$\leftarrow$ M_3	
4	100	1		m_4
5	101	0	$\leftarrow$ M_5	
6	110	1		m_6
7	111	0	$\leftarrow$ M_7	

最小项范式和最大项范式的关系

- 对于函数 $f(A,B,C)$
- 一般的 $\overline{(m_1)} = \overline{(\overline{A}\,\overline{B}C)} = A + B + \overline{C} = M_1$
  - $\overline{m_i} = M_i$
  - $\overline{M_i} = \overline{\overline{m_i}} = m_i$
- 一般的

$$f(A,B,C) = \Sigma m_i = \Pi M_j \quad j \neq i$$

$$\overline{f(A,B,C)} = \Pi M_i = \Sigma m_j \quad j \neq i$$

例子

$$\overline{f(A,B,C)} = \overline{\sum m(0,2,4)}$$

$$= \overline{\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C}}$$

$$= (A+B+C)(A+\bar{B}+C)(\bar{A}+B+C)$$

$$= \prod M(0,2,4)$$

最小项范式和最大项范式的关系

- 最小项范式——正逻辑
- 最大项范式——负逻辑
- 相互转换
  - 例：$f(A,B,C)=\Sigma m(2,3,6,7)$

$$=\Pi \mathrm{M}(0,1,4,5)$$

- 应用：
  - 积之和与和之积之间的转换
  - 求负函数

开关函数的最大范式表示

- 优点：
  – 和函数的真值表有直接的关系
  – 能很容易地判断出对于一个输入组合，函数的值是否为0
- 不足：
  – 不是函数的最简单形式
  – 相应的电路比较复杂
- 需要化简

范式的推导

- 定理11：（香农的扩展定理）
  - (a) $f(x_1, x_2, ...x_n) = x_1 \cdot f(1, x_2, ...x_n) + \overline{x_1} \cdot f(0, x_2, ...x_n)$
  - (b) $f(x_1, x_2, ...x_n) = [x_1 + f(0, x_2, ...x_n)] \cdot [\overline{x_1} + f(1, x_2, ...x_n)]$
- 例子：

$$
\begin{aligned}
f(A, B, C) &= AB + A\overline{C} + \overline{A}C \\
&= Af(1, B, C) + \overline{A}f(0, B, C) \\
&= A(1 \cdot B + 1 \cdot \overline{C} + \overline{1} \cdot C) + \overline{A}(0B + 0\overline{C} + \overline{0}C) \\
&= A(B + \overline{C}) + \overline{A}C \\
&= B[A(1 + \overline{C}) + \overline{A}C] + \overline{B}[A(0 + \overline{C}) + \overline{A}C] \\
&= AB + AB\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + \overline{A}\overline{B}C \\
&= ABC + \overline{A}BC + \overline{A}\overline{B}C + AB\overline{C} + A\overline{B}\overline{C}
\end{aligned}
$$

范式的推导（续）

- 应用定理11
- 例子

$$
\begin{aligned}
f(A,B,C) &= AB + A\overline{C} + \overline{A}C \\
&= AB(C+\overline{C}) + A(B+\overline{B})\overline{C} + \overline{A}(B+\overline{B})C \\
&= ABC + AB\overline{C} + A\overline{B}\,\overline{C} + \overline{A}BC + \overline{A}\,\overline{B}C \\
&= m_7 + m_6 + m_4 + m_3 + m_1 \\
&= \Sigma m(1,3,4,6,7)
\end{aligned}
$$

- 同理可处理最大范式的推导

非确定函数

- 布尔函数非完全确定的原因
  - 一些确定的输入组合不会产生
  - 一些输出为0或1只对特定的输入组合成立
- 无关(don’t care)最小项：忽略一些最小项
- 无关最大项：忽略一些最大项
- 表示
  - 无关最小项：d_i
  - 无关最大项：D_i
- 例子：
  - 无关最小项范式：$f(A,B,C)=\Sigma m(0,3,7)+d(4,5)$
  - 无关最大项范式：$f(A,B,C)=\Pi M(1,2,6)\cdot D(4,5)$

非确定性函数

例：BCD码累加器

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

W为0

W为1

W位无关项

这些输入组合在实际中不可能出现
所以输出不用关心具体的值

布尔函数的例子

- 输入值相等判断
- 计算输入二进制中1的个数
- 判断输入二进制是否能被2、3、5整除
- 驱动7段译码显示器
- 二进制加法器
  - 半加器
  - 全加器
- 真值表：最根本的表示方式
  - 优点：最准确
  - 缺点：庞大

如何做一个能计算的设备?

人的计算方法

计算方法描述（输入、输出、行为）

布尔函数

数据编码 | 真值表

积之和SOP | 和之积POS

?

开关电路

组合电路

计算设备

本讲小结

- 数字系统中的数字表示的基本理论和概念
  - 数制
  - 有符号数
  - 计算机编码
- 数字系统的数学基础：布尔代数
  - 6个公理
  - 10个定理（要求只用公理证明）
- 布尔函数
  - 真值表
  - 布尔表达式：或与式（积之和）、或与式（和之积）
  - 最小项、最大项、无关项