

数字逻辑与处理器基础

第二讲 数的表示与布尔代数

李学清

xueqingli@tsinghua.edu.cn

助教：张肇良/尹国栋、黄成宇/陈一鸣

腾讯会议：983-4578-6584

2022年3月1日

通知

- ❖ 后选课的同学
 - 补交第一次作业
- ❖ 微信
 - 实名
- ❖ 期中考试时间冲突问题
 - 问卷调查

助教答疑

❖ 张肇良，尹国栋，黄成宇，陈一鸣

❖ 周日，7pm-9pm

- 罗姆楼5105
- 腾讯会议954-6186-7668

作业

- ❖ 鼓励讨论问题，但请独立完成作业
- ❖ 线下同学下周上课时提交纸版作业，线上同学可上交到网络学堂
- ❖ 作业1
 - 《数字逻辑与处理器基础》第二章课后题：
1, 2, 11

作业

❖ 作业2

- 分别用代数化简法和卡诺图化简法化简逻辑函数：

$$F = A'B'C' + A'B'C + A'BC + ABC + A'BC'$$

- 已知函数： $f(A, B, C) = \sum m(0, 1, 2, 4, 5)$
 - (1)将该函数转化为最大项表示的POS形式
 - (2)使用卡诺图将该函数化简为最简SOP形式
 - (3)使用下表提供的逻辑门，如果要以最小面积实现该函数的计算功能应如何设计电路，此时面积是多少？如果考虑最小延时呢？(注：门名称后的数字代表输入数；电路输入信号仅包括A,B,C)

逻辑门	延时(ps)	面积(μm^2)
NOT	18	8
AND2	50	25
NAND2	30	15
NAND4	70	30

逻辑门	延时(ps)	面积(μm^2)
OR2	55	26
NOR2	35	16
AND4	90	40
OR4	100	42
NOR4	80	32

课程回顾

- 为什么IC和CPU如此重要？有需求
 - 用有限的电路结构解决各式各样的计算
 - 设计约束和指标需求
- 如何应对这些需求
 - 数字化、同步化、结构化
 - 定制硬件和通用软件
 - 使用EDA工具
- 为什么IC发展地如此迅猛
 - 摩尔定律
- 课程培养目标
 - 获取知识，学习技能，领会思想与方法论

布尔代数的定位

❖ The 'big-bang' starts with '0' and '1'

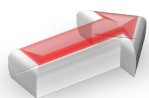
- In *complementary metal-oxide-semiconductor (CMOS)* technology

Week	Date	Main content
1	2/22	概论
2	3/1	布尔代数
3	3/8	组合逻辑
4	3/15	时序逻辑
5	3/22	时序逻辑
6	3/29	计算机指令系统
7	待定	清明节调休(期中考试)
8	4/12	计算机指令系统
9	4/19	微处理器
10	4/26	微处理器
11	5/7	劳动节放假5/3→5/7
12	5/10	微处理器/流水线
13	5/17	流水线
14	5/24	流水线
15	5/31	存储器
16	6/7	存储器/IO与总线/总结

课程目标

- ❖ 如何在数字逻辑中表示数
 - 数的表示
- ❖ 如何用布尔代数计算
 - 算数和函数的布尔代数表示
 - 布尔表达式的化简

主要内容



1

数的编码与二进制表示

2

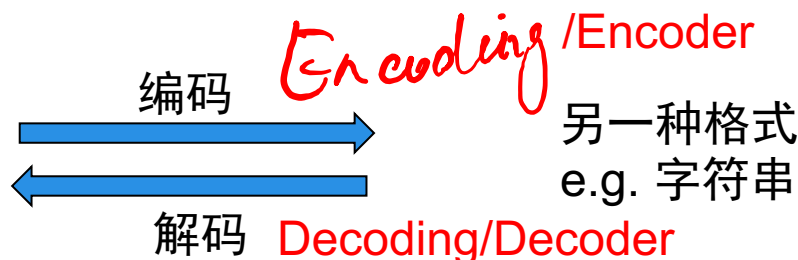
算术与函数的布尔代数表示

3

布尔表达式的化简

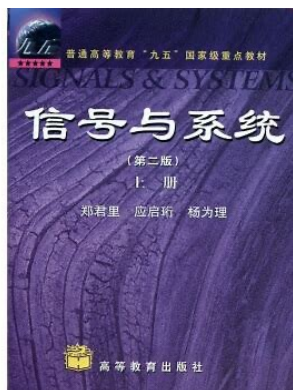
编码和解码

某种信息格式
e.g. 图像、视频、音频



Offset	0	1	2	3	4	5	6	7	-	8	9	A	B	C	D	E	F	ASCII
00000000	FF	D8	FF	E0	00	10	4A	46		49	46	00	01	01	01	00	48	яшяа..JFIF....H
00000010	00	48	00	00	FF	DB	00	43		00	01	01	01	01	01	01	01	.Н..яН.С.....
00000020	01	01	01	01	01	01	01	01		01	01	01	01	01	01	01	01
00000030	01	01	01	01	01	01	01	01		01	01	01	01	01	01	01	01
00000040	01	01	01	01	01	01	01	01		01	01	01	01	01	01	01	01
00000050	01	01	01	01	01	01	01	01		01	FF	DB	00	43	01	01	01яН.С...
00000060	01	01	01	01	01	01	01	01		01	01	01	01	01	01	01	01
00000070	01	01	01	01	01	01	01	01		01	01	01	01	01	01	01	01
00000080	01	01	01	01	01	01	01	01		01	01	01	01	01	01	01	01

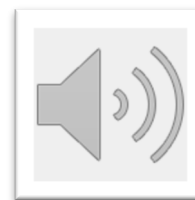
JPEG文件头



SBN 978-7-04-022559-4



Bar code



Hello world



QR code

为什么用二元编码/二进制？

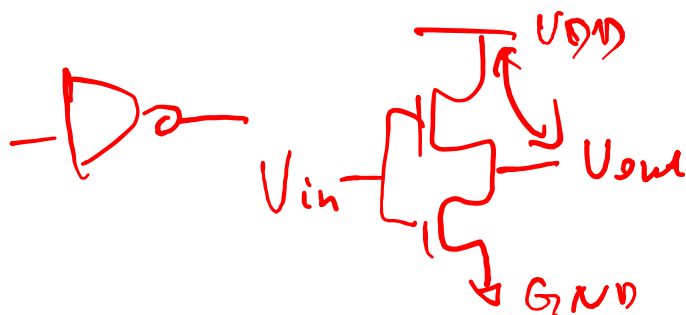
❖ 容易表示和存储

✓ 只用两种状态('0' 和 '1')

✓ 晶体管的导通表示“1”，截止表示“0”

✓ 电位的高表示“1”，低表示“0”

- 材料极化特性
- 磁场、自旋方向
-



✓ 为什么不用十进制/十六进制？

为什么用二元编码/二进制？

❖ 运算简单

✓ 加法和乘法

✓ $0+0=(0)0$, $0+1=(0)1$, $1+1=(1)0$

✓ $0\times 0=0$, $0\times 1=0$, $1\times 1=1$

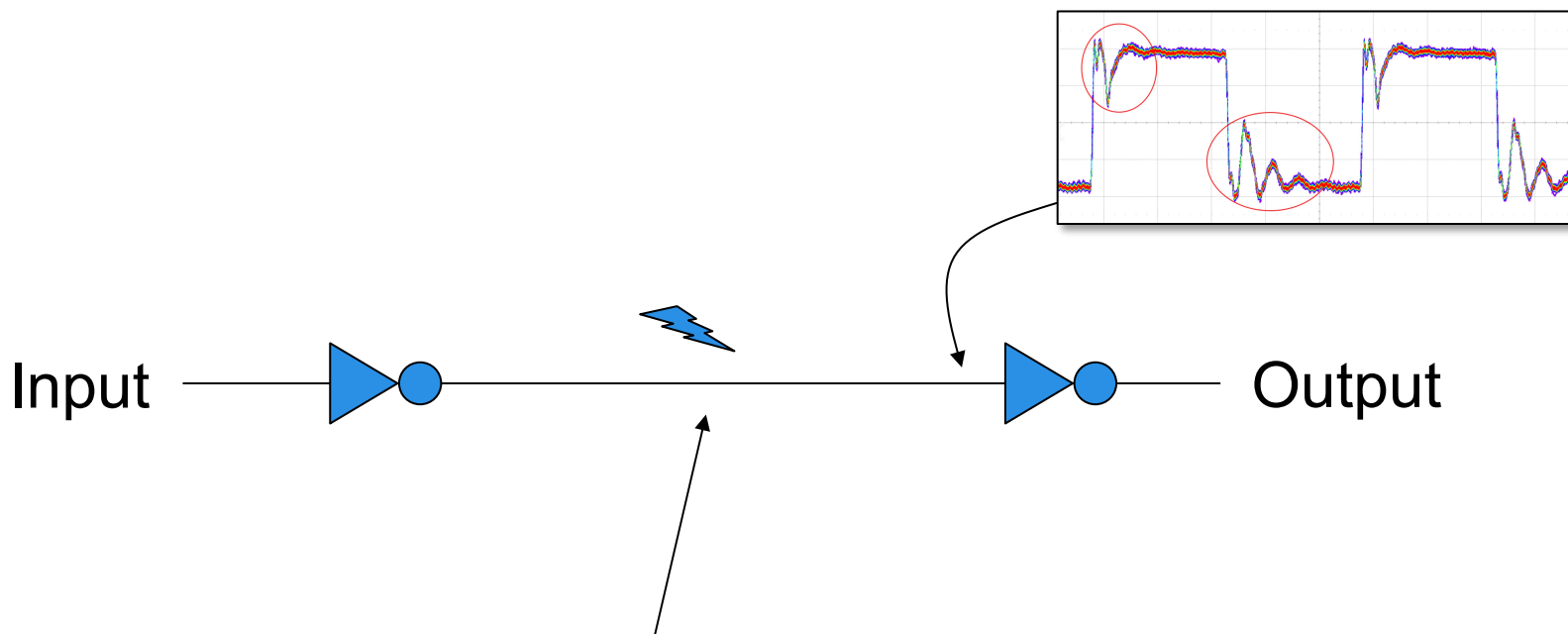
✓ 逻辑对应：

✓ 二进制“1/0”与逻辑值“真/假”

为什么用二元编码/二进制？

❖ 传输简单

✓ 噪音容限高 → 可靠性高

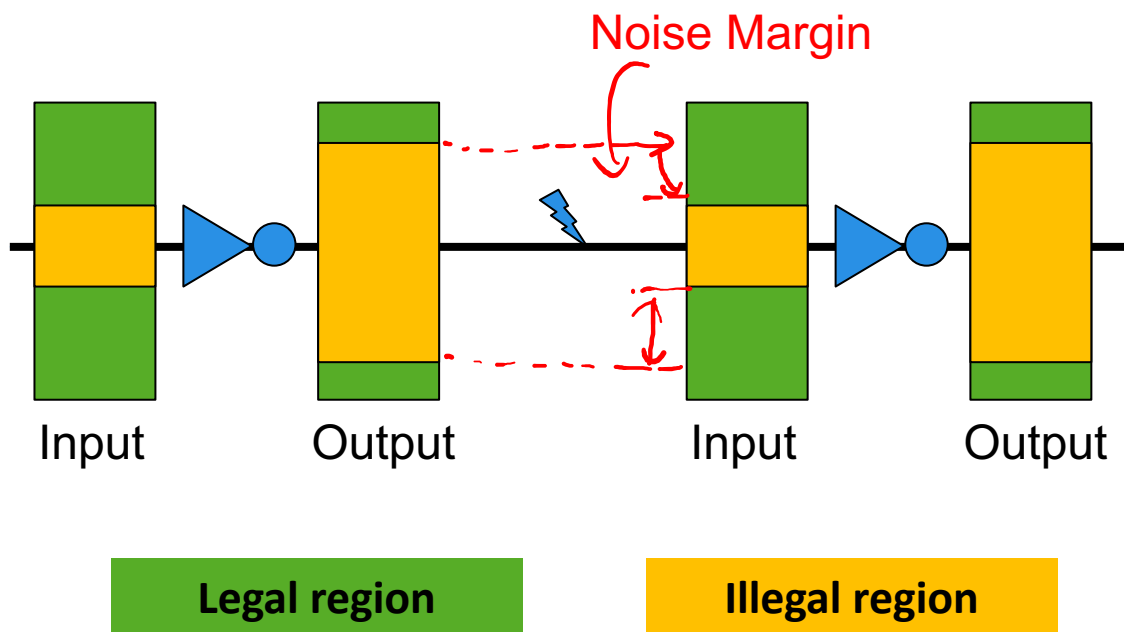


问题：如何处理噪音的干扰？

为什么用二元编码/二进制？

❖ 传输简单

✓ 噪音容限高 → 可靠性高



二进制 Binary System

Radix point

MSB (Most Significant Bit)

LSB (Least Significant Bit)

1 0 1 . 1 1

Weight

2^2

2^1

2^0

2^{-1}

2^{-2}

$$B = \sum_{k_i: 0, 1} k_i \times 2^i$$

$$101.11_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$+ 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$= 5.75_{10}$$

$$= 5 \times 10^1 + 7 \times 10^{-1} + 5 \times 10^{-2}$$

4-bit Binary Integer Number

BIN				DEC	HEX
2^3	2^2	2^1	2^0		
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7

BIN				DEC	HEX
2^3	2^2	2^1	2^0		
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

左移与右移操作

Right shift
右移
对应十进制 $\div 2$

1110. 1-bit right shift=0111.0

$$14 \div 2^1 = 7$$

1110. 2-bit right shift=0011.10

$$14 \div 2^2 = 3.5$$

Left shift
左移
对应十进制 $\times 2$

0110. 1-bit left shift=01100.

$$6 \times 2^1 = 12$$

0110. 2-bit left shift=011000.

$$6 \times 2^2 = 16 + 8 = 24$$

另一种十进制表示法？

$$395_{10} = (256+128+8+2+1)_{10} = (0001\ 1000\ 1011)_2$$

❖ $3_{10} = 0011_2$

❖ $9_{10} = 1001_2$

❖ $5_{10} = 0101_2$

$(\textcolor{red}{3}\textcolor{green}{9}\textcolor{blue}{5})_{10} \rightarrow (\textcolor{red}{0011}\textcolor{green}{1001}\textcolor{blue}{0101})_{\text{NEW}}$

Binary-coded decimal (BCD):
每个十进制数字都用二进制单独表示

BCD码

❖ Binary-coded decimal

- 每个十进制数单独表示
- 编码方式不唯一

❖ 8421 BCD码

- 有权码
- 最常用

十进制数	8.4.2.1BCD			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
权重	8	4	2	1

注意

❖ 8421BCD和二进制数**有区别!**

❖ $(\textcolor{red}{3}\textcolor{green}{9}\textcolor{blue}{5})_{10} = (\textcolor{red}{0011} \textcolor{green}{1001} \textcolor{blue}{0101})_{\text{BCD}} = (0001 \ 1000 \ 1011)_2$

$\neq (\textcolor{red}{0011} \textcolor{green}{1001} \textcolor{blue}{0101})_2$

❖ $(\textcolor{red}{0011} \textcolor{green}{1001} \textcolor{blue}{0101})_2 = (512+256+128+16+4+1)_{10} = 917_{10}$

❖ BCD加法

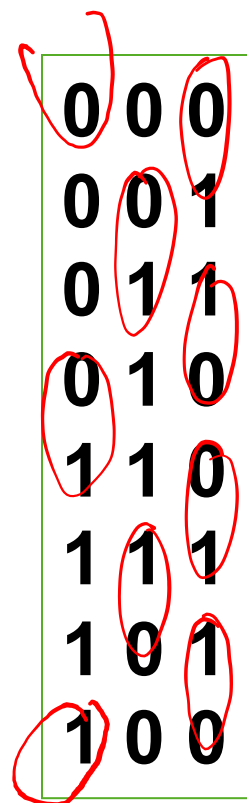
■ $(\textcolor{green}{3}\textcolor{blue}{4})_{10} + (\textcolor{blue}{4}\textcolor{green}{5})_{10} = (\textcolor{green}{0011} \textcolor{blue}{0100})_{\text{BCD}} + (\textcolor{blue}{0100} \textcolor{green}{0101})_{\text{BCD}}$
和: $(\textcolor{red}{0111} \textcolor{red}{1001})_{\text{BCD}}$

■ 进位 (Carry)

• Try $(14)_D + (28)_D = (0001 \ 0100)_{\text{BCD}} + (0010 \ 1000)_{\text{BCD}}$

自己做一遍?

格雷码



0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

Gray

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Binary

- 二进制码的状态转换按照数值的大小不断改变
- 格雷码的状态转换按照仅有一位数字发生变化的原则改变(格雷码相邻两个数之间仅有一位编码不同)
- 格雷码将在卡诺图化简中使用

浮点数

- 浮点数 v 包含两部分，基数 m 和尾数 e

$$v = m2^{e-x}$$

m : 二进制小数

e : 二进制整数

x : 尾数的偏移

本课程不做要求

减法、负数

❖ 加法和减法是最基本的运算

$$\begin{array}{r} 173 \\ +) \quad 44 \\ \hline 217 \end{array}$$

$$\begin{array}{r} 10101101 \\ +) \quad 00101100 \\ \hline 11011001 \end{array}$$

$$\begin{array}{r} 173 \\ -) \quad 46 \\ \hline 127 \end{array}$$

$$\begin{array}{r} 010100 \\ 10101101 \\ -) \quad 00101110 \\ \hline 01111111 \end{array}$$

❖ 数有正有负，如何表示负数？

数值空间的划分

- ❖ n bit 二进制数值空间可表示 2^n 个数值
 - 000, 001, 010, 011,
100, 101, 110, 111
 - 哪些表示正数, 哪些表示负数?
 - 划分标准如何?

负数的表示方式

❖ 有符号数：原码

- $01010101_2 = +85_{10}$
- $11010101_2 = -85_{10}$

❖ 反码，又称1补码

- 一个n比特位反码表示的负数“-N”为
 - 形式上，将该负数对应的无符号n位正数“+N”按位取反
 - 若当作无符号数，其值 $= 2^n - 1 - N$
- 对于n比特反码
 - 其可表示的数的范围： $-(2^{n-1} - 1) \sim +(2^{n-1} - 1)$ ；3 bit：-3 ~ 3
 - 对于0，有两种表示法：00...0和11...1
 - 数值空间利用率低、加减法不兼容

1-补码（反码）数值系统（3bit为例）			
正十进制	正二进制	负十进制	负二进制
0	000	-0	111
1	001	-1	110
2	010	-2	101
3	011	-3	100

负数的表示方式

❖ 补码，又称2补码

- 一个n比特2补码表示的负数(-N)为
 - 将正数N按位取反再加1
 - 无符号数值==反码+1== $2^n - N$
- 其可表示的数的范围： $-2^{n-1} \sim 2^{n-1}-1$
 - MSB是符号位：MSB=1, 负数; MSB=0, 正数;
 - 0仅有一种，便于将减法运算变成加法运算

提问：已知负数的补码表示，值是几？

$$-N = \text{补码无符号值} - 2^n = 5 - 8 \rightarrow -3$$

$$\begin{aligned}
 &= k_n * 2^{n-1} + \dots + k_2 * 2^1 + k_1 * 2^0 - 2^n \\
 &= k_n * (-2^{n-1}) + \dots + k_2 * 2^1 + k_1 * 2^0
 \end{aligned}$$

MSB位权重： -2^{n-1}

正数MSB位为零，所以也适用

反码数值系统（3bit为例）

正十进制	正二进制	负十	负二
0	000	-0	111
1	001	-1	110
2	010	-2	101
3	011	-3	100

2-补码数值系统（3bit为例）

正十进制	正二进制	负十	负二
0	000	-1	111
1	001	-2	110
2	010	-3	101
3	011	-4	100

注意

❖ 反码和补码，均仅针对负数有不同表示，正数一样

❖ 对于正数 $+85_{10}$

- 有符号 01010101_2
- 1补码 01010101_2
- 2补码 01010101_2

❖ 对于负数 -85_{10}

- 有符号数 11010101_2
- 1补码 10101010_2
- 2补码 10101011_2

1-补码数值系统（3bit为例）

正十进制	正二进制	负十进制	负二进制
0	000	-0	111
1	001	-1	110
2	010	-2	101
3	011	-3	100

2-补码数值系统（3bit为例）

正十进制	正二进制	负十进制	负二进制
0	000	-1	111
1	001	-2	110
2	010	-3	101
3	011	-4	100

以下哪组选项对-7的4位1补码和2补码的编码是正确的：

- ☐ A 1000, 1001
- ☐ B 1001, 1000
- ☐ C 1000, 1000
- ☐ D 1001, 1001



⚠ 请在问卷中回复答案

Time: 120s

以下哪组选项对-7的4位1补码和2补码的编码是正确的：

A 1000, 1001

B 1001, 1000

C 1000, 1000

D 1001, 1001

解答： **A**

利用-7对应的正数进行计算
+7 : 0111

1补码：按位取反，即1000
2补码：按位取反加1，即1001

补码的应用：用加法来实现减法！

❖ ‘-N’的补码

- 无符号数值： $N_{us} = 2^n - N$

$$\begin{array}{r} 010 \\ - 001 \\ \hline 001 \end{array} \longleftrightarrow \begin{array}{r} 1010 \\ - 0001 \\ \hline 1001 \end{array}$$

❖ 减法 (3-bit 例子)

- $2 - 1 = 2 + 2^3 - 1 = 2 + (2^3 - 1) = 2 + (111)_{us}$

- $A - N = A + (-N) = A + (2^n - N) = A + N_{us}$

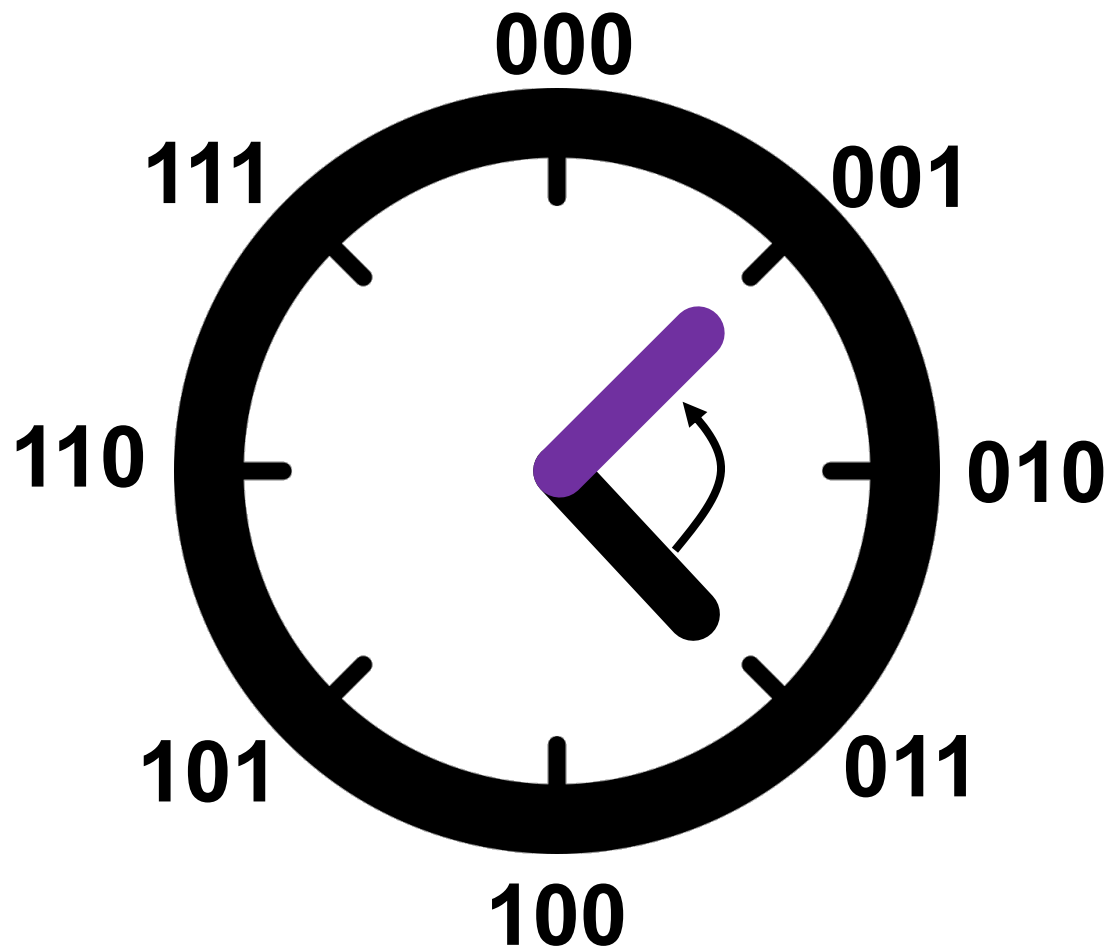
$$\begin{array}{r} 010 \\ + 111 \\ \hline 1001 \end{array}$$

ignored

正数 (Same for 1's and 2's)		负数			
		反码1's complement		补码2's complement	
DEC	BIN	DEC	BIN	DEC	BIN
0	000	-0	111	-1	111
1	001	-1	110	-2	110
2	010	-2	101	-3	101
3	011	-3	100	-4	100

加减法 vs 拨钟

- $A - N = A + (-N) = A + (2^n - N) = A + N_{us}$



011-010

VS

011+110

用加法来实现减法

❖ $A - B = A + (-B) = A + (0 - B)$

例：六位二进制数的计算：8-20→8+(-20)

传统减法模式：

$$\begin{array}{r} \overset{1\ 0\ 1\ 0}{0\ 0\ 1\ 0\ 0\ 0} \\ 8: \\ -) \ 20: \ 0\ 1\ 0\ 1\ 0\ 0 \\ \hline -12: \ 1\ 1\ 0\ 1\ 0\ 0 \end{array}$$

补码减法模式：

$$\begin{array}{r} \ 0\ 0\ 1\ 0\ 0\ 0 \\ +) \ 1\ 0\ 1\ 1\ 0\ 0 \\ \hline \ 1\ 1\ 0\ 1\ 0\ 0 \end{array}$$

实例验证，补码可以利用加法器实现完整的加减法！

可有问题？

4-bit digits

$$\begin{array}{r}
 +3 \quad 0011 \\
 +) +4 \quad 0100 \\
 \hline
 +7 \quad 0111
 \end{array}$$

$$\begin{array}{r}
 +6 \quad 0110 \\
 +) -3 \quad 1101 \\
 \hline
 +3 \quad 10011
 \end{array}$$

Ignore

$$\begin{array}{r}
 +5 \quad 0101 \\
 +) +6 \quad 0110 \\
 \hline
 +11 \quad 1011
 \end{array}$$

-5

$$\begin{array}{r}
 -6 \quad 1010 \\
 +) -3 \quad 1101 \\
 \hline
 -9 \quad 10111
 \end{array}$$

+7 Ignore

溢出 *overflow*

❖ 什么是溢出

- 计算结果超出了可表示数的范围
 - $-2^{n-1} \sim 2^{n-1}-1$

❖ 如何判断溢出（什么时候可能溢出）

- 两正数相加，结果变负数
- 两负数相加，结果变正数（或零）

❖ 例：(1) -96-33

(2) -96-32

溢出

	1	0	1	0	0	0	0	0
+	1	1	0	1	1	1	1	1
<hr/>								
1	0	1	1	1	1	1	1	1

127

	1	0	1	0	0	0	0	0
+	1	1	1	0	0	0	0	0
<hr/>								
1	1	0	0	0	0	0	0	0

-128

未溢出

处理溢出

❖ 增加位数

				+6		00110	
				+) -3		+) 11101	
				<hr/>		<hr/>	
				+3		100011	
						↑ 忽略	
	+5						
						11010	
+) +6		+) 00110		+) -3		+) 11101	
<hr/>		<hr/>		<hr/>		<hr/>	
+11		01011		-9		110111	
						↑ 忽略	

处理溢出

❖ 增加位数

■ 正数增加的位数补0

- $0110 \rightarrow 00110$
- $2^2 + 2^1 \rightarrow 2^2 + 2^1$

■ 负数增加的位数补1

- $1101 \rightarrow 11101$
- $-2^3 + 2^2 + 2^0 \rightarrow -2^4 + 2^3 + 2^2 + 2^0$
- 负数原来的负权重位变成正权重位

溢出的简单判断

- 判断是否超出数值范围
- 对符号位进位输入(Cin)和输出(Cout)做异或

Cin	Cout	结果
0	0	正确
0	1	溢出, 上溢 (正数相加变负数)
1	0	溢出, 下溢 (负数相加变正数)
1	1	正确

有进位 无进位生成

1 0 1 0 0 0 0 0

溢出 + 1 1 0 1 1 1 1 1

1 0 1 1 1 1 1 1

127

有进位 有进位生成

1 0 1 0 0 0 0 0

+ 1 1 1 0 0 0 0 0

1 1 0 0 0 0 0 0

-128

未溢出

以下2补码运算会溢出的是哪一项？

- A 1001+0110
- B 0011+1110
- C 1101+1011
- D 0101+0111



方法二：

Cin	Cout	结果
0	0	正确
0	1	溢出，上溢（正数相加变负数）
1	0	溢出，下溢（负数相加变正数）
1	1	正确

解答： D

A $1001+0110$

C $1101+1011$

B $0011+1110$

D $0101+0111$

对于A, B两项, 都是一正一负相加, 正数与负数相加一定不会溢出;

对于C项, 1101即-3, 1011即-5, 二者相加为-8, 在4位2补码数的表示范围内 (-8至+7), 因此不溢出。

对于D项, 0101即5, 0111即7, 二者相加为12, 超出4位2补码数的表示范围, 因此会溢出。

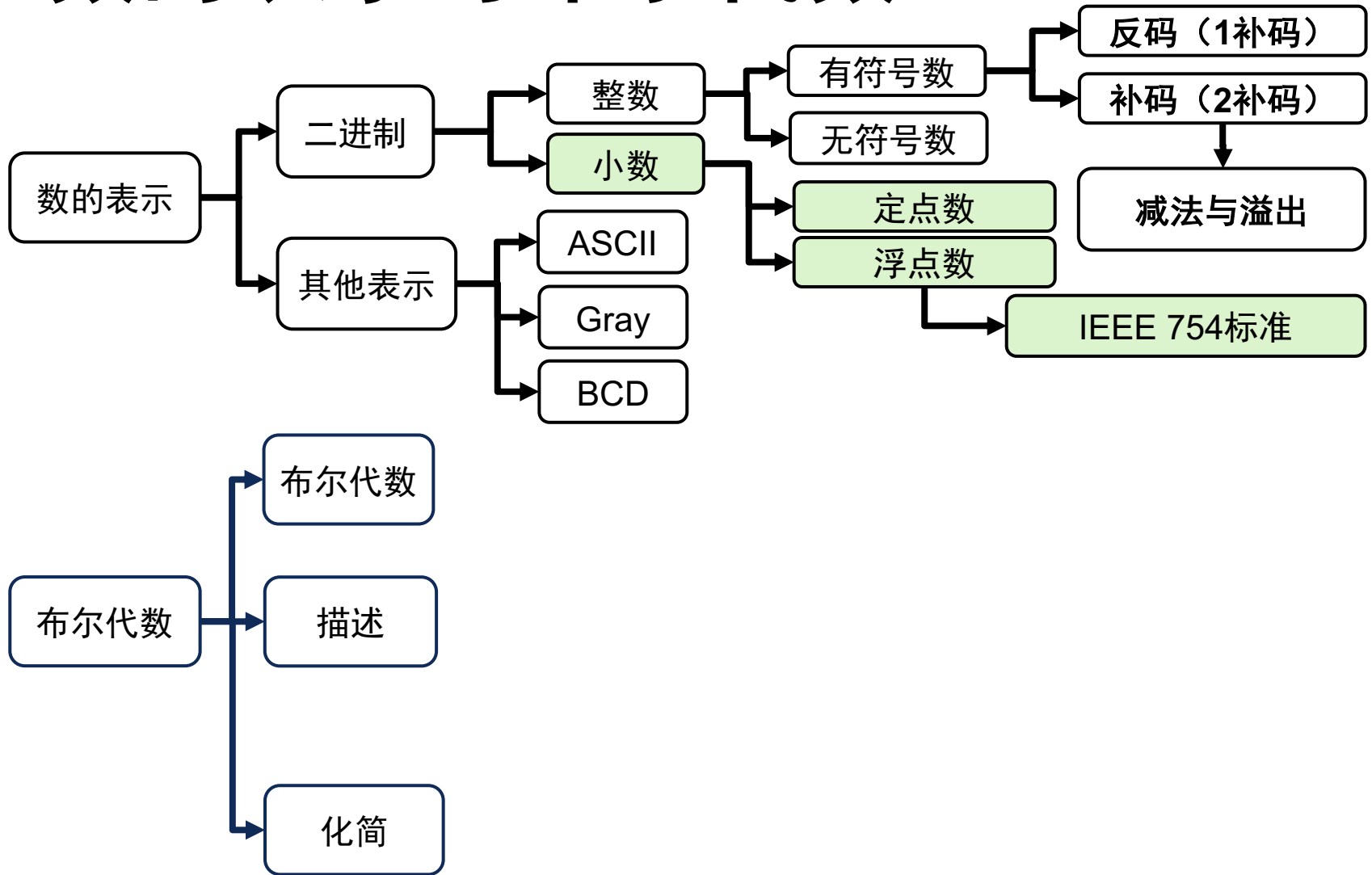
也可以通过技巧判断:

课后思考

已知A和B的4比特2补码分别为A=1011和B=0011, 请问A-B是否会发生溢出?

$\begin{array}{r} 1101 \\ 1011 \\ \hline 11000 \end{array}$	$\begin{array}{r} 0101 \\ 0111 \\ \hline 01100 \end{array}$
符号位最高位 进位都为1	符号位进位0, 最高位进位1

数的表示与布尔代数



主要内容

1

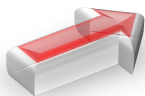
数的编码与二进制表示

2

算术与函数的布尔代数表示

3

布尔表达式的化简



布尔代数的定义

- ❖ 布尔代数是数字系统的数学基础
- ❖ 用逻辑运算符表示输入逻辑变量与输出逻辑变量之间的关系
 - 运算: NOT (非), AND (与), OR (或)
 - 优先权: 非最高, 与次之, 或最低
 - 运算符: + (或/逻辑加), \bullet (与/逻辑乘), ' (-) (非/反)

基本定律 (1)

❖ Commutative Law 交换律

- $X+Y=Y+X$ $XY=YX$

❖ Associative Law 结合律

- $(X+Y)+Z=X+(Y+Z)=X+Y+Z$

- $(XY)Z=X(YZ)=XYZ$

❖ Distributive Law 分配律

- $X+(Y \cdot Z)=(X+Y)(X+Z)$

- $X(Y+Z)=XY+XZ$

❖ Complement 互补律

- $X+X'=1$ $X \cdot X'=0$

基本定律(2)

❖ 与0 & 1的运算

- $A+0=A, A+1=1$
- $A \cdot 1=A, A \cdot 0=0$ ($A+1=1$ 的对偶)

❖ Idempotent 重叠

- $X+X=X; X \cdot X=X$ ($X+X=X$ 的对偶)

❖ Involution 对合

- $(X')'=X$

基本定理(3)

❖ Consensus theorem 一致性定理（多数定理）

- $(X+Y)(Y+Z)(X'+Z)=(X+Y)(X'+Z)$
- $X \cdot Y + Y \cdot Z + X' \cdot Z = XY + X'Z$
- 用于化简逻辑表达式（代数化简法用到）
- 用于消除组合电路的竞争冒险（下一次课用到）

用真值表验证一致性定理

X	Y	Z	X'	Y'	Z'	$(X+Y)(Y+Z)(X'+Z)$	$(X+Y)(X'+Z)$
0	0	0	1	1	1	0	0
0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	1
0	1	1	1	0	0	1	1
1	0	0	0	1	1	0	0
1	0	1	0	1	0	1	1
1	1	0	0	0	1	0	0
1	1	1	0	0	0	1	1

即证明： $(X+Y)(Y+Z)(X'+Z)=(X+Y)(X'+Z)$

布尔函数

❖ 多种表现形式

- 布尔代数的表达式： $f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$
 - Variables and operators
- 真值表 (Truth table)
- 电路图 (Circuit diagram)

基本定理(4)

❖ Duality 对偶

- 对偶是布尔代数的一个非常有用的性质
- 一个布尔表达式的对偶可以通过将与和或互换，逻辑0和1互换，同时保持布尔变量不变得到

❖ 对偶函数的一般形式

- $\{f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot)\}^D$
- $=f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$

❖ 如果一个布尔代数定理成立，其对偶亦成立

- 例： $X+0=X \rightarrow (X+0)^D=(X)^D$ 即 $X \cdot 1=X$

❖ 如果两个布尔表达式的对偶函数是相同的，则这两个表达式是相同的： $F_1^D=F_2^D \leftrightarrow F_1=F_2$

❖ 可以用来证明一致性定理/分配律/结合律

德摩根率(De Morgan's Law)

❖ 函数反演的一般形式:

- $\{f(X_1, \dots, X_n, 0, 1, +, \cdot)\}' = f(X_1', \dots, X_n', 1, 0, \cdot, +)$
- $\text{AND} \leftrightarrow \text{OR}, 0 \leftrightarrow 1, X \leftrightarrow X'$

❖ 例子

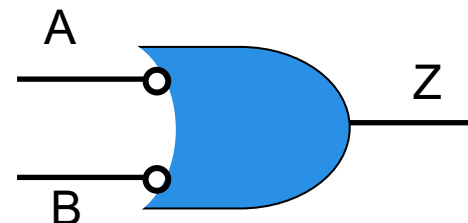
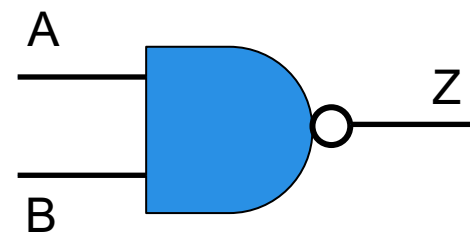
- $\overline{AB} = \overline{A} + \overline{B}$
- $\overline{A + B} = \overline{A} \cdot \overline{B}$
- 其他

德摩根率(De Morgan's Law)

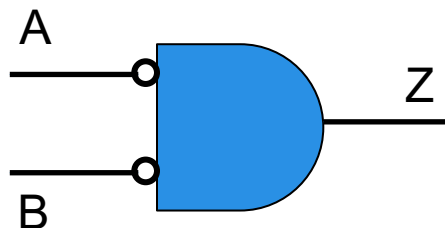
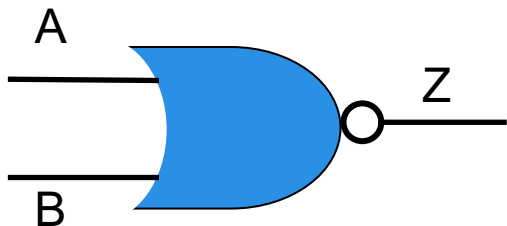
- $(X+Y+Z+\dots)' = X' \cdot Y' \cdot Z' \cdot \dots$
- $(X \cdot Y \cdot Z \cdot \dots)' = X' + Y' + Z' + \dots$

$$\overline{AB} = \overline{A} + \overline{B}$$

NAND gate



$$\overline{A + B} = \overline{A} \cdot \overline{B}$$



NOR gate

用真值表验证德摩根定律

X	Y	X'	Y'	X+Y	XY	$(X+Y)'$	$X'Y'$	$(XY)'$	$X'+Y'$
0	0	1	1	0	0	1	1	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	1	0	0	0	0

求反

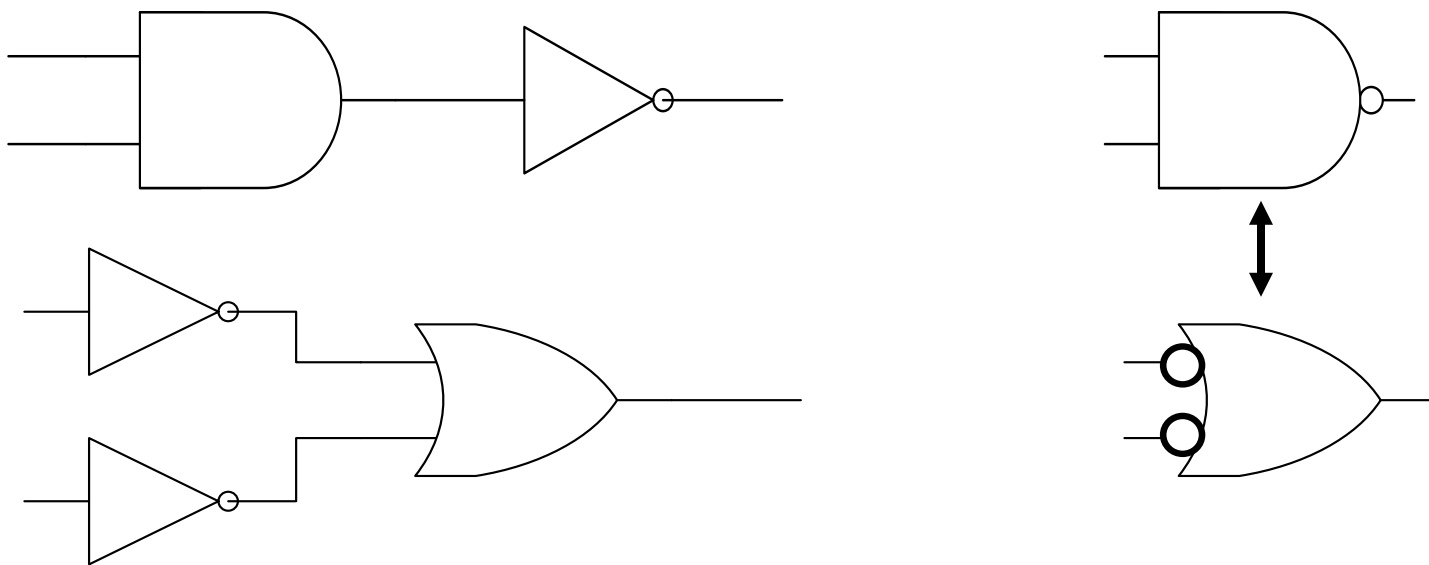
$$Y=A(B+C)+CD$$

与和或互换
逻辑0和1互换
布尔变量求反

$$\begin{aligned} Y' &= [A(B+C) + CD]' \\ &= [A(B+C)]' \bullet (CD)' \\ &= [A' + (B+C)'] \bullet (C' + D') \\ &= (A' + B'C') \bullet (C' + D') \end{aligned}$$

用德摩根定律变换逻辑门电路的形式

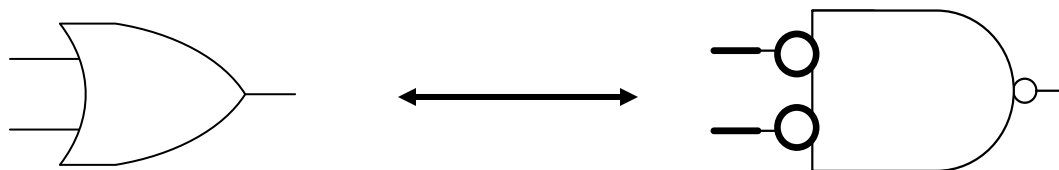
$$\overline{A \cdot B} = \bar{A} + \bar{B}$$



与非门等价于对输入取反的或门

用德摩根定律变换逻辑门电路的形式

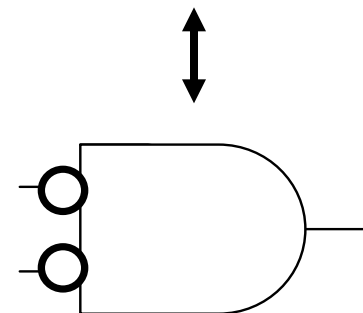
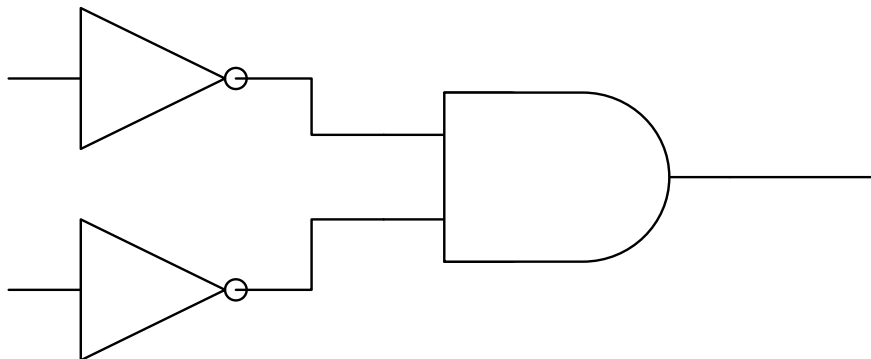
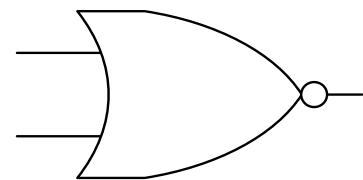
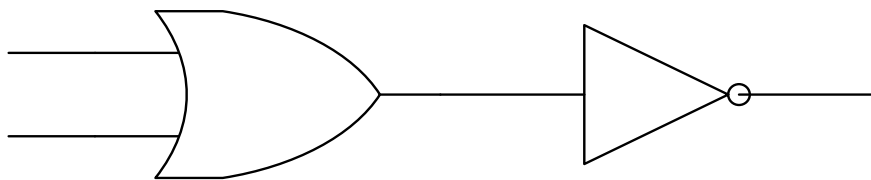
$$A + B = \overline{\overline{A} \cdot \overline{B}}$$



或门等价于对输入取反的与非门

用德摩根定律变换逻辑门电路的形式

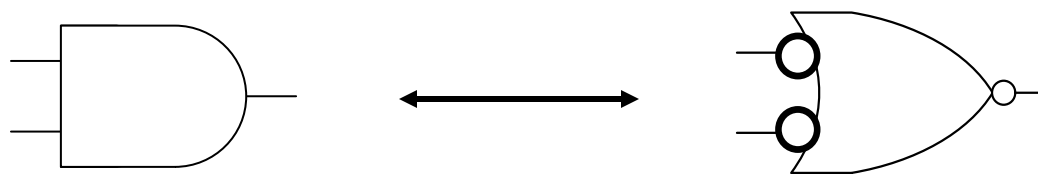
$$\overline{A + B} = \bar{A} \cdot \bar{B}$$



或非门等价于对输入取反的与门

用德摩根定律变换逻辑门电路的形式

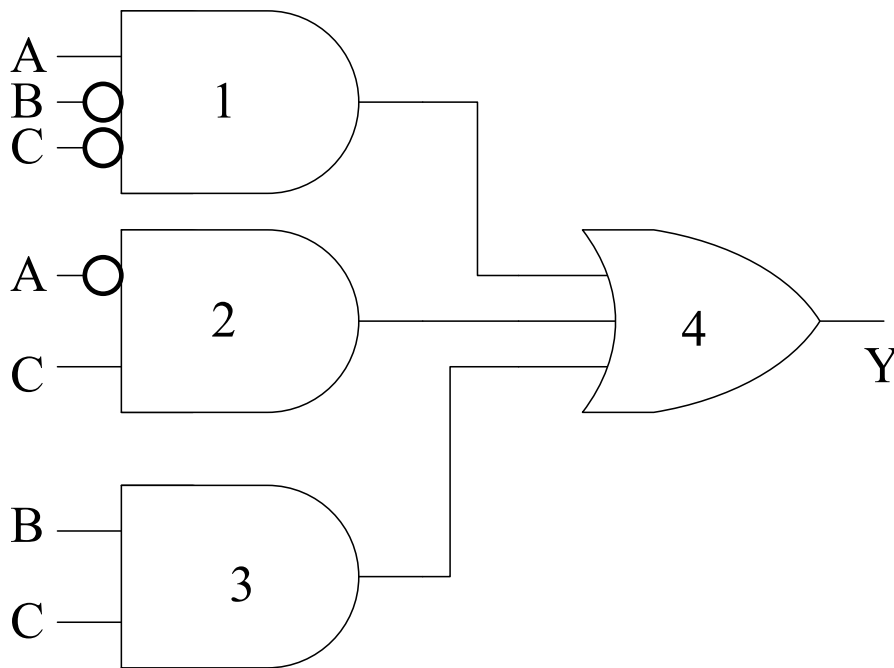
$$A \cdot B = \overline{\overline{A} + \overline{B}}$$



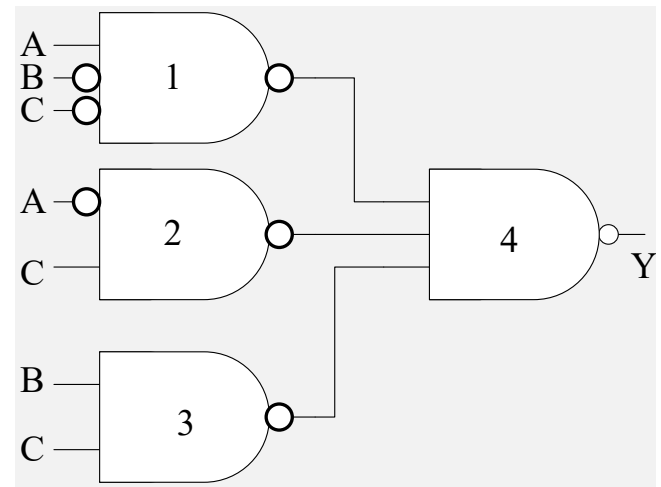
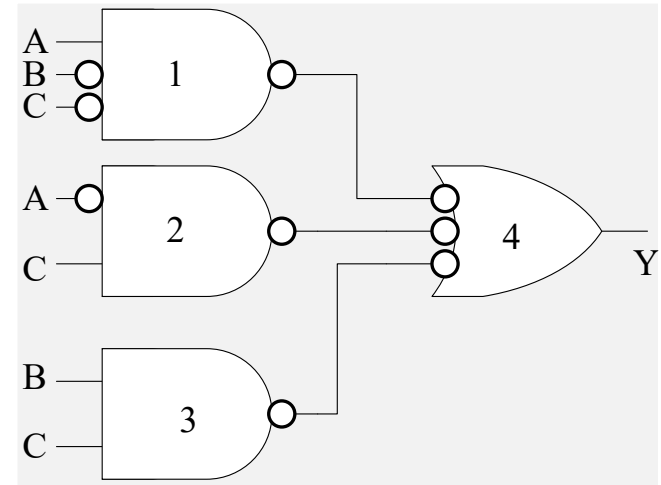
与门等价于对输入取反的或非门

将与或表达式变换成仅用与非门实现

$$Y = AB'C' + A'C + BC$$



$$Y = [(AB'C')'(A'C)'(BC)']'$$



对偶和反的区别

❖ General form of inversion

$$\{f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot)\}' = f(X_1', X_2', \dots, X_n', 1, 0, \cdot, +)$$

❖ General form of duality

$$\{f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot)\}^D = f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$$

$$Y = A(B + C) + \overline{CD}$$

❖ Inversion of Y

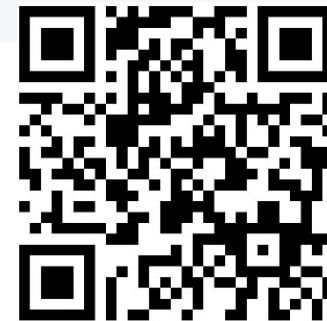
布尔变量取反

$$\begin{aligned} Y' &= (\bar{A} + \bar{B}\bar{C})(\overline{\bar{C} + \bar{D}}) = (\bar{A} + \bar{B}\bar{C})CD \\ &= \bar{A}CD \end{aligned}$$

❖ Duality of Y

布尔变量不取反

$$\begin{aligned} Y^D &= (A + BC)\overline{\bar{C} + \bar{D}} = (A + BC)\bar{C}\bar{D} \\ &= A\bar{C}\bar{D} + BC\bar{C}\bar{D} \\ &= A\bar{C}\bar{D} + 0 = A\bar{C}\bar{D} \end{aligned}$$



如何仅用或非门实现 $Y=AB'C'+A'C+BC$?

- A $Y=[(A+B'+C')'+(A'+C)'+(B+C)']'$
- B $Y=\{[(A'+B+C)'+(A+C')'+(B'+C')']'+0\}'$
- C $Y=(A'+B+C)'+(A+C')'+(B'+C')'$

Time: 120s

解答： **B**

利用德摩根律，可以得到：

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

由于只使用或非门，故转换为：

$$Y = \{ [(A' + B + C)' + (A + C')' + (B' + C')']' + 0 \}'$$

延伸思考：仅用与门、异或门等单独的一个门实现呢？
有哪些门可以通过组合和输入限制实现任意逻辑
(**Universal Gate**)?

其他布尔函数

❖ 与

$0 \cap 0 = 0$
 $0 \cap 1 = 0$
 $1 \cap 0 = 0$
 $1 \cap 1 = 1$

❖ 或

$0 \cup 0 = 0$
 $0 \cup 1 = 1$
 $1 \cup 0 = 1$
 $1 \cup 1 = 1$

❖ 非

$0' = 1$
 $1' = 0$

与=乘

与=进位

异或=和
异或: $a \oplus b = (\sim a \cap b) \cup (a \cap \sim b)$

• 乘法 $a \times b = f$

$0 \times 0 = 0$
 $0 \times 1 = 0$
 $1 \times 0 = 0$
 $1 \times 1 = 1$

• 加法

$a + b = cs$

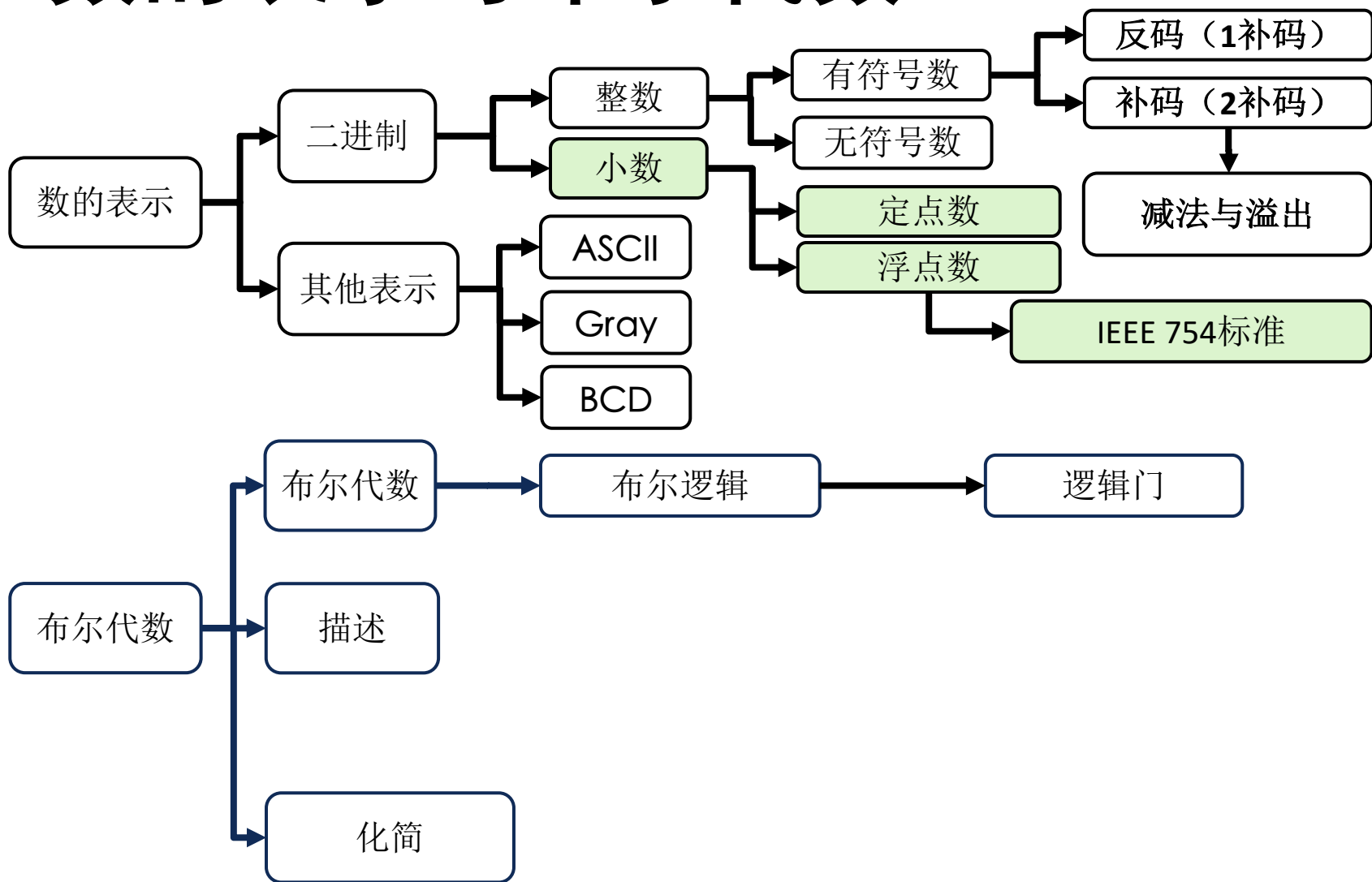
$0 + 0 = 00$
 $0 + 1 = 01$
 $1 + 0 = 01$
 $1 + 1 = 10$

• $f = a \bullet b$

• $c = a \bullet b$

• $s = ab' + a'b = a \oplus b$

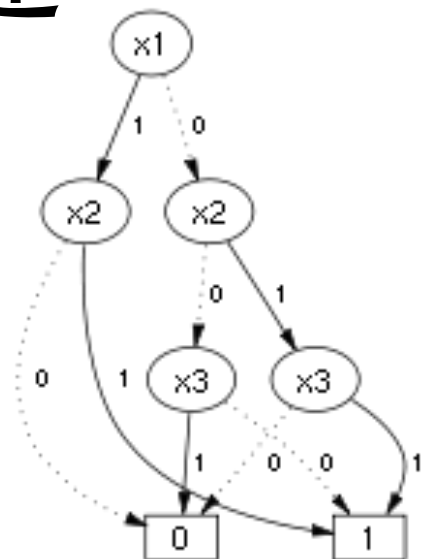
数的表示与布尔代数



函数的布尔代数描述

❖ 二元域上函数的表示方法：

- 真值表（及卡诺图）
- 表达式
- 其他方式（程序、
Binary decision diagram树）



❖ 例：日历子系统

- 根据月份和是否闰年的标志，
给出这个月份的天数

日历子系统的C程序

```
❖ Integer number_of_days (month, leap_flag) {  
    ■ switch (month){  
        • case 'january': return (31);  
        • case 'february': if (leap_flag ==1) then return (29) else return (28);  
        • case 'march': return (31);  
        • case 'april': return (30);  
        • case 'may': return (31);  
        • case 'june': return (30);  
        • case 'july': return (31);  
        • case 'august': return (31);  
        • case 'september': return (30);  
        • case 'october': return (31);  
        • case 'november': return (30);  
        • case 'december': return (31);  
        • default: return (0);  
    ■ }  
❖ }
```


日历子系统的逻辑实现

❖ 组合逻辑实现过程

■ 逻辑抽象

- 输入: $\text{month}(a_8, a_4, a_2, a_1)$, leap_flag
- 输出: d28, d29, d30, d31

■ 逻辑函数：真值表

■ 逻辑表达式与化简

■ 逻辑电路

真值表—输出与输入之间的逻辑关系

$a_8a_4a_2a_1$

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-	-	-	-	-	-

无关项

- 输出不随输入变化
- 实际情况中不存在
- 不影响功能

逻辑函数

❖ **d28** = a_8' and a_4' and a_2 and a_1' and leap'

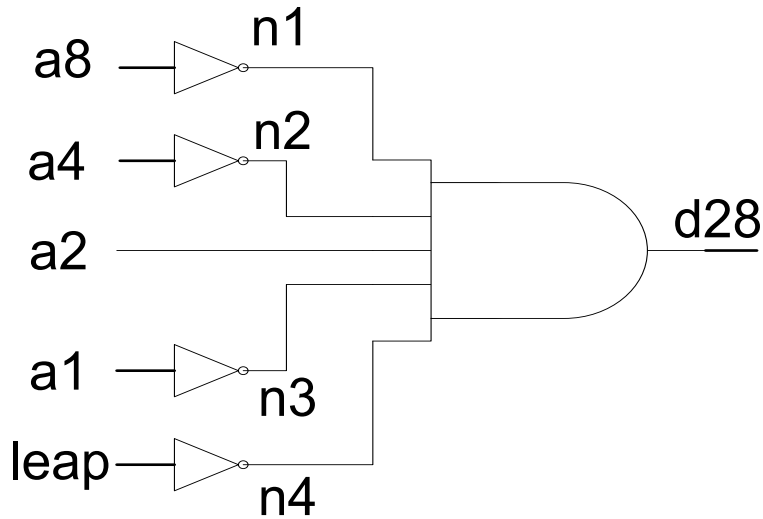
❖ **d29** = a_8' and a_4' and a_2 and a_1' and leap

$a_8a_4a_2a_1$

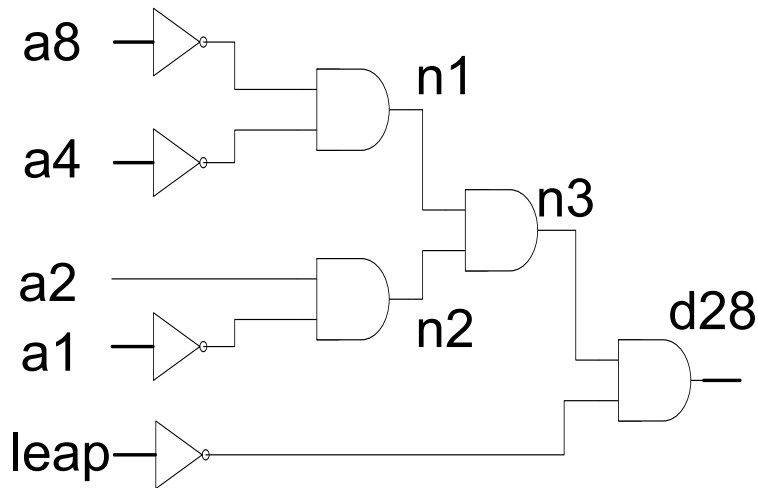
Month	Leap	d28	d29
0000	-	-	-
0001	-	0	0
0010	0	1	0
0010	1	0	1

d28的逻辑电路

Verilog: 一种硬件描述语言



```
module CalD28 (a8,a4,a2,a1,leap,d28);  
  input  a8,a4,a2,a1,leap;  
  output d28;  
  not u1(n1, a8);  
  not u2(n2, a4);  
  not u3(n3, a1);  
  not u4(n4, leap);  
  and u5(d28, n1, n2, a2, n3, n4);  
endmodule
```

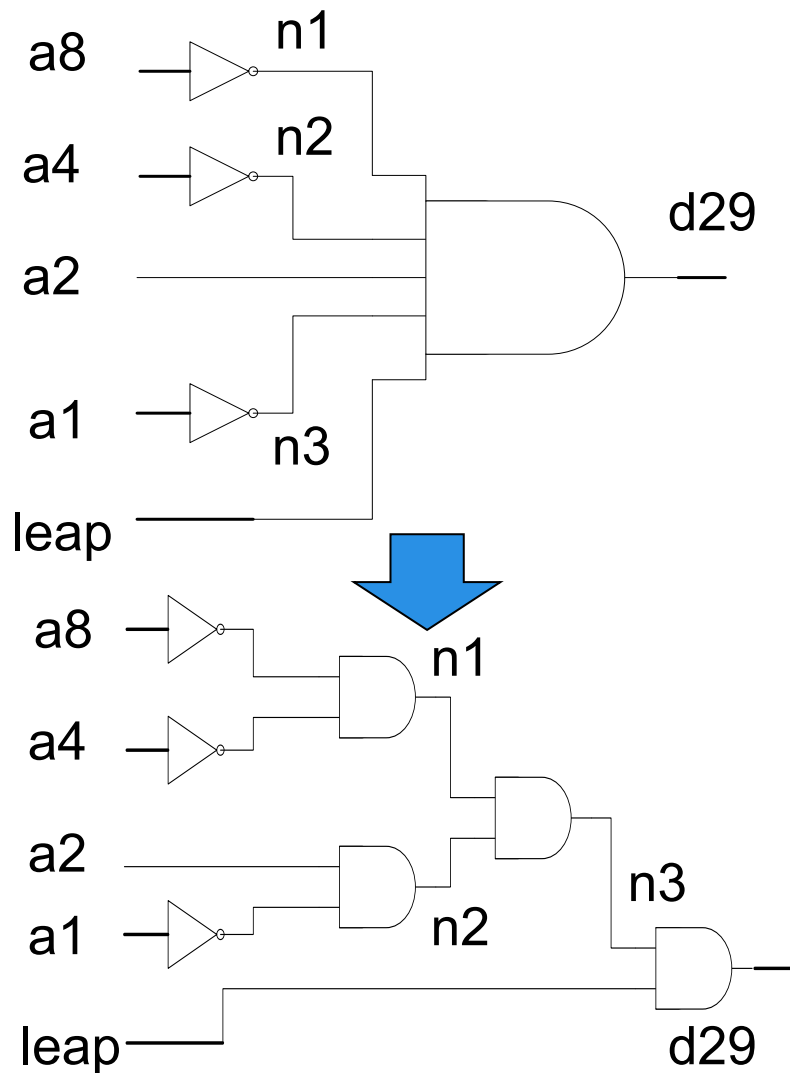


```
module CalD28 (a8,a4,a2,a1,leap,d28);  
  input  a8,a4,a2,a1,leap;  
  output d28;  
  wire n1,n2,n3;  
  assign n1=(~a8)&(~a4);  
  assign n2=a2&(~a1);  
  assign n3=n1&n2;  
  assign d28=n3&(~leap);  
endmodule
```

d29的逻辑电路

Verilog代码

```
module CalD29 (a8,a4,a2,a1,leap,d29);  
  input  a8,a4,a2,a1,leap;  
  output d29;  
  not u1(n1, a8);  
  not u2(n2, a4);  
  not u3(n3, a1);  
  and u4(d29, n1, n2, a2, n3, leap);  
endmodule
```



```
module CalD29 (a8,a4,a2,a1,leap,d29);  
  input  a8,a4,a2,a1,leap;  
  output d29;  
  wire n1,n2,n3;  
  assign n1=(~a8)&(~a4);  
  assign n2=a2&(~a1);  
  assign n3=n1&n2;  
  assign d28=n3&leap;  
endmodule
```

d30和d31呢？

- ❖ $d30 = a_8' \cdot a_4 \cdot a_2' \cdot a_1' \text{ or } a_8' \cdot a_4 \cdot a_2 \cdot a_1' \text{ or } a_8 \cdot a_4' \cdot a_2' \cdot a_1 \text{ or } a_8 \cdot a_4' \cdot a_2 \cdot a_1$
- ❖ $d31 = a_8' \cdot a_4' \cdot a_2' \cdot a_1 \text{ or } a_8' \cdot a_4' \cdot a_2 \cdot a_1 \text{ or } a_8' \cdot a_4 \cdot a_2' \cdot a_1 \text{ or } a_8' \cdot a_4 \cdot a_2 \cdot a_1 \text{ or } a_8 \cdot a_4' \cdot a_2' \cdot a_1' \text{ or } a_8 \cdot a_4' \cdot a_2 \cdot a_1' \text{ or } a_8 \cdot a_4 \cdot a_2' \cdot a_1'$

考虑电路，

如何表示它们对应的布尔代数式？

——逻辑化简

Month	d30	d31
0000	-	-
0001	0	1
0010	0	0
0010	0	0
0011	0	1
0100	1	0
0101	0	1
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	0	1
1011	1	0
1100	0	1
1101	-	-
111-	-	-

布尔代数式的化简

❖ 两级逻辑

- 规范形式
- 卡诺图

❖ 两级逻辑的化简

- 卡诺图化简法
- Q-M 法

标准形式：两级逻辑

❖ SOP 和 POS

- Sum of Products 与或表达式

$$f(A, B, C, D) = AB + A\bar{C} + C\bar{D}$$

- Product of Sums 或与表达式

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

$$\begin{aligned} d31 = & a_8' \cdot a_4' \cdot a_2' \cdot a_1 \text{ or } a_8' \cdot a_4' \cdot a_2 \cdot a_1 \\ & \text{or } a_8' \cdot a_4 \cdot a_2' \cdot a_1 \text{ or } a_8' \cdot a_4 \cdot a_2 \cdot a_1 \text{ or } \\ & a_8 \cdot a_4' \cdot a_2' \cdot a_1' \text{ or } a_8 \cdot a_4' \cdot a_2 \cdot a_1' \text{ or } \\ & a_8 \cdot a_4 \cdot a_2' \cdot a_1' \end{aligned} \rightarrow \Sigma(ABCD)$$

Month	d30	d31
0000	-	-
0001	0	1
0010	0	0
0011	0	1
0100	1	0
0101	0	1
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	0	1
1011	1	0
1100	0	1
1101	-	-
111-	-	-

逻辑函数的最小项，记为 m

在 n 变量逻辑函数中，包含全部 n 个变量的与项，其中每个变量必须且只以原变量或反变量的形式出现一次

最小项	使最小项取值为1的输入变量值			最小项下标 i	m_i
	A	B	C		
$A'B'C'$	0	0	0	0	m_0
$A'B'C$	0	0	1	1	m_1
$A'BC'$	0	1	0	2	m_2
$A'BC$	0	1	1	3	m_3
$AB'C'$	1	0	0	4	m_4
$AB'C$	1	0	1	5	m_5
ABC'	1	1	0	6	m_6
ABC	1	1	1	7	m_7
权重	2^2	2^1	2^0		

MSB



LSB

用最小组表示逻辑函数(SOP)

$$Y = f(A, B, C) = \bar{A}B + BC + A\bar{B}\bar{C}$$

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

$$= \bar{A}BC + \bar{A}B\bar{C} + ABC + \bar{A}BC + A\bar{B}\bar{C}$$

MSB  $\bar{A}BC$ $\bar{A}B\bar{C}$ ABC $\bar{A}BC$ $A\bar{B}\bar{C}$  LSB

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

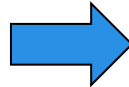
$011_2 = 3_{10} \quad 010_2 = 2_{10} \quad 111_2 = 7_{10} \quad 100_2 = 4_{10}$

$$= m_3 + m_2 + m_7 + m_3 + m_4$$

$$= \sum m(2, 3, 4, 7)$$

Using Min-term in truth table

$a_8a_4a_2a_1$	d30
0000	-
0001	0
0010	0
0011	0
0100	1
0101	0
0110	1
0111	0
1000	0
1001	1
1010	0
1011	1
1100	0
1101	-
111-	-



$a_8a_4a_2a_1$	Min-term	d30
0000	m0	-
0001	m1	0
0010	m2	0
0011	m3	0
0100	m4	1
0101	m5	0
0110	m6	1
0111	m7	0
1000	m8	0
1001	m9	1
1010	m10	0
1011	m11	1
1100	m12	0
1101	m13	-
111-	m14-m15	-

POS逻辑表示

$$\begin{aligned} Y &= (A + B) \cdot (\bar{A} + B + C) \\ &= (A + B + C \cdot \bar{C}) \cdot (\bar{A} + B + C) \\ &= (A + B + C)(A + B + \bar{C})(\bar{A} + B + C) \\ &= M_0 M_1 M_4 \\ &= \prod M(0, 1, 4) \end{aligned}$$

逻辑函数的最大项，记为M

在n变量逻辑函数中，包含全部n个变量的或项，其中每个变量必须且只以原变量或反变量的形式出现一次

最大项	使最大项取值为0的输入变量值			最大项下标i	M_i
	A	B	C		
$A+B+C$	0	0	0	0	M_0
$A+B+C'$	0	0	1	1	M_1
$A+B'+C$	0	1	0	2	M_2
$A+B'+C'$	0	1	1	3	M_3
$A'+B+C$	1	0	0	4	M_4
$A'+B+C'$	1	0	1	5	M_5
$A'+B'+C$	1	1	0	6	M_6
$A'+B'+C'$	1	1	1	7	M_7
权重	2^2	2^1	2^0		

m_i & M_i

规律: $m_6 = ABC\bar{C} = \overline{\overline{ABC}} = \overline{\overline{A} + \overline{B} + C} = \overline{M_6}$

$$\begin{aligned} Y = \sum m(2, 3, 4, 7) &\rightarrow \bar{Y} = \overline{m_2 + m_3 + m_4 + m_7} \\ &= \overline{\overline{M_2} + \overline{M_3} + \overline{M_4} + \overline{M_7}} \\ &= \overline{\overline{M_2} \cdot \overline{M_3} \cdot \overline{M_4} \cdot \overline{M_7}} \\ &= M_2 \cdot M_3 \cdot M_4 \cdot M_7 \\ &= M_2 \cdot M_3 \cdot M_4 \cdot M_7 \\ &= \prod M(2, 3, 4, 7) \end{aligned}$$

$$\begin{aligned} \sum m(2, 3, 4, 7) &= \prod M(0, 1, 5, 6) \end{aligned}$$

m_i & M_i

规律: $m_6 = ABC = \overline{\overline{ABC}} = \overline{\overline{A} + \overline{B} + \overline{C}} = \overline{M_6}$

$$Y = \sum m(2, 3, 7)$$



$$\bar{Y} = \overline{m_2 + m_3 + m_7}$$

$$= \overline{\overline{M_2} + \overline{M_3} + \overline{M_7}}$$

$$= \overline{\overline{M_2}} \cdot \overline{\overline{M_3}} \cdot \overline{\overline{M_7}}$$

$$= M_2 \cdot M_3 \cdot M_7$$

$$= \prod M(2, 3, 7)$$



$$\bar{Y} = \overline{m_2 + m_3 + m_7}$$

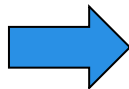
$$= m_0 + m_1 + m_4 + m_5 + m_6$$

$$= \sum m(0, 1, 4, 5, 6)$$

$$\sum m(0, 1, 4, 5, 6) = \prod M(2, 3, 7)$$

在真值表中使用最小项

$a_8a_4a_2a_1$	d30
0000	-
0001	0
0010	0
0011	0
0100	1
0101	0
0110	1
0111	0
1000	0
1001	1
1010	0
1011	1
1100	0
1101	-
111-	-

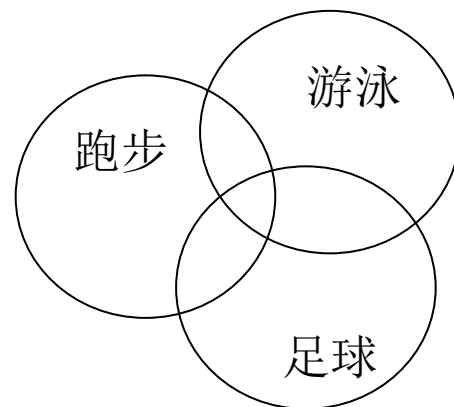


$a_8a_4a_2a_1$	最小项	d30
0000	m0	-
0001	m1	0
0010	m2	0
0011	m3	0
0100	m4	1
0101	m5	0
0110	m6	1
0111	m7	0
1000	m8	0
1001	m9	1
1010	m10	0
1011	m11	1
1100	m12	0
1101	m13	-
111-	m14-m15	-

卡诺图

❖ 逻辑表示图的发展

- 文氏图，1881年，表示集合关系
- 维奇图(Veitch Diagram)，1952年，爱德华·维奇提出后用以描述并最小化布尔表达式
- 卡诺图，1953年，美国贝尔实验室的电信工程师莫里斯·卡诺(Maurice Karnaugh)在1953年根据维奇图改进。



	A		\bar{A}		
B	12	13	5	4	\bar{C}
	14	1	1	7	
\bar{B}	10	1	1	3	C
	1	1	1	0	
	\bar{D}		D		

$AB \setminus CD$	00	01	11	10
00	1	0	0	1
01	1	0	*	*
11	1	0	1	1
10	1	0	1	1

$$Z = \bar{D} + A \cdot C$$


两变量的卡诺图

	B	0	1
A			
0			
1			

	B	0	1
A			
0		$m_0(A'B')$	$m_1(A'B)$
1		$m_2(AB')$	$m_3(AB)$

$$f(A, B)$$

三变量的卡诺图

BC	00	01	11	10
A				
0				
1				

$$f(A, B, C)$$

格雷码的应用

BC	00	01	11	10
A				
0	$m_0(A'B'C')$	$m_1(A'B'C)$	$m_3(A'BC)$	$m_2(A'BC')$
1	$m_4(AB'C')$	$m_5(AB'C)$	$m_7(ABC)$	$m_6(ABC')$

四变量的卡诺图

$$f(A, B, C, D)$$

CD AB	00	01	11	10
00	$m_0(A'B'C'D')$	$m_1(A'B'C'D)$	$m_3(A'B'CD)$	$m_2(A'B'CD')$
01	$m_4(A'BC'D')$	$m_5(A'BC'D)$	$m_7(A'BCD)$	$m_6(A'BCD')$
11	$m_{12}(ABC'D')$	$m_{13}(ABC'D)$	$m_{15}(ABCD)$	$m_{14}(ABCD')$
10	$m_8(AB'C'D')$	$m_9(AB'C'D)$	$m_{11}(AB'CD)$	$m_{10}(AB'CD')$

四变量的卡诺图的另一种形式

$$f(A, B, C, D)$$

AB CD	00	01	11	10
00	$m_0(A'B'C'D')$	$m_4(A'BC'D')$	$m_{12}(ABC'D')$	$m_8(AB'C'D')$
01	$m_1(A'B'C'D)$	$m_5(A'BC'D)$	$m_{13}(ABC'D)$	$m_9(AB'C'D)$
11	$m_3(A'B'CD)$	$m_7(A'BCD)$	$m_{15}(ABCD)$	$m_{11}(AB'CD)$
10	$m_2(A'B'CD')$	$m_6(A'BCD')$	$m_{14}(ABCD')$	$m_{10}(AB'CD')$

五变量的卡诺图

$f(E, A, B, C, D)$

E A B C D:

$2^4 2^3 2^2 2^1 2^0$

CD AB	00	01	11	10
	E=0			
00	$m_0(A'B'C'D')$	$m_1(A'B'C'D)$	$m_3(A'B'CD)$	$m_2(A'B'CD')$
01	$m_4(A'BC'D')$	$m_5(A'BC'D)$	$m_7(A'BCD)$	$m_6(A'BCD')$
11	$m_{12}(ABC'D')$	$m_{13}(ABC'D)$	$m_{15}(ABCD)$	$m_{14}(ABCD')$
10	$m_8(AB'C'D')$	$m_9(AB'C'D)$	$m_{11}(AB'CD)$	$m_{10}(AB'CD')$

CD AB	00	01	11	10
	E=1			
00	$m_{16}(A'B'C'D')$	$m_{17}(A'B'C'D)$	$m_{19}(A'B'CD)$	$m_{18}(A'B'CD')$
01	$m_{20}(A'BC'D')$	$m_{21}(A'BC'D)$	$m_{23}(A'BCD)$	$m_{22}(A'BCD')$
11	$m_{28}(ABC'D')$	$m_{29}(ABC'D)$	$m_{31}(ABCD)$	$m_{30}(ABCD')$
10	$m_{24}(AB'C'D')$	$m_{25}(AB'C'D)$	$m_{27}(AB'CD)$	$m_{26}(AB'CD')$

五变量的卡诺图的另一种形式

BCD EA	000	001	011	010	110	111	101	100
00	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
01	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
11	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
10	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}

用卡诺图表示逻辑函数

$$Y = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{D} + ACD + A\bar{B}$$

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

$$= \bar{A}\bar{B}\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D} + ABCD + A\bar{B}CD + A\bar{B}CD + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D}$$

$$= m_1 + m_6 + m_4 + m_{15} + m_{11} + m_{11} + m_{10} + m_9 + m_8$$

$$= \sum m_i (i = 1, 4, 6, 8, 9, 10, 11, 15)$$

将函数表示成最小项相或的形式

$$Y = \sum m_i (i = 1, 4, 6, 8, 9, 10, 11, 15)$$

CD	00	01	11	10
AB				
00		1		
01	1			1
11			1	
10	1	1	1	1

CD	00	01	11	10
AB				
00	0	1	0	0
01	1	0	0	1
11	0	0	1	0
10	1	1	1	1

布尔函数的描述方式包括：（多选题）

- ☐ A 逻辑表达式
- ☐ B 真值表
- ☐ C 卡诺图
- ☐ D 二元决策图



布尔函数的描述方式包括：（多选题）

- ☐ A 逻辑表达式
- ☐ B 真值表
- ☐ C 卡诺图
- ☐ D 二元决策图

主要内容

1

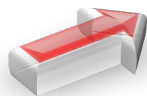
数的编码与二进制表示

2

算术与函数的布尔代数表示

3

布尔表达式的化简



逻辑化简举例

$$Y = AC + \bar{B}C + B\bar{D} + \textcolor{red}{C}\bar{D} + A(B + \bar{C}) + \textcolor{red}{\bar{A}BC\bar{D}} + A\bar{B}DE$$

$$A + AB = A$$

$$= AC + \textcolor{red}{\bar{B}C} + B\bar{D} + C\bar{D} + \textcolor{red}{A\bar{B}\bar{C}} + A\bar{B}DE$$

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

$$= \textcolor{red}{AC} + \bar{B}C + B\bar{D} + C\bar{D} + \textcolor{red}{A} + \textcolor{red}{A\bar{B}DE}$$

$$A + AB = A$$

$$= A + \textcolor{red}{\bar{B}C} + \textcolor{red}{B\bar{D}} + \textcolor{red}{C\bar{D}}$$

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

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

化简动机

❖ 规范形式使得逻辑函数复杂化，其对应门电路也非常复杂

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

$$F = AB' + AB = A(B' + B) = A$$

$$F = A'B' + AB' = (A' + A)B' = B'$$

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

代数化简法

$$AB + A\bar{B} = A$$

$$A + AB = A$$

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

$$A + A = A; A + \bar{A} = 1$$

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

多数定理

两级逻辑化简

有其他方法加速化简过程吗？

- ❖ 卡诺图

- ❖ CAD工具

 - Q-M算法

卡诺图化简法

❖ **相邻**两个最小项仅有1个输入逻辑变量的取值不同，互补

BC	00	01	11	10
A				
0	0	0	1	1
1	0	0	1	1

只有C这一输入逻辑
变量取值不同

只有A这一输入逻辑变量取值不同

卡诺图化简法

- ❖ **相邻**两个最小项仅有1个输入逻辑变量的取值不同，互补
- ❖ 如果这两个相邻最小项都使输出为1，则这两个最小项可合并，消除一个输入逻辑变量

BC	00	01	11	10
A				
0	0	0	1	1
1	0	0	1	1

$$ABC + ABC' = AB \rightarrow$$

可以合并为AB，与C的值无关

卡诺图化简法

- ❖ **相邻**两个最小项仅有1个输入逻辑变量的取值不同，互补
- ❖ 如果这两个相邻最小项都使输出为1，则这两个最小项可合并，消除一个输入逻辑变量
- ❖ 如果有四个相邻的最小项都使得输出函数取值为1，则这四个最小项可以合并，消除两个输入逻辑变量

BC	00	01	11	10
A				
0	0	0	1	1
1	0	0	1	1

$A'B$

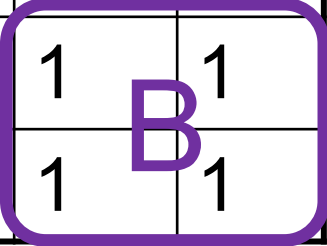
AB

可以合并为B，
与A、C值无关

卡诺图化简法

❖ 如何保证卡诺图化简结果是最简的？ \Rightarrow “蕴含” 的概念

BC	00	01	11	10
A				
0	0	0	1	1
1	0	0	1	1



蕴含项

❖ 如果 $F=1$ 则 $G=1$ ，那么 F 是 G 的蕴含项（Implicant）： $F \Rightarrow G$

■ 助记：【 F 有效】“implies”【 G 有效】

x	y	$x'y$	xy'	xy	F1	F2
0	0	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	1

F1的蕴含项有哪些？ xy' , $x'y$
F2的蕴含项有哪些？

Prime Implicant 本原蕴含项

❖ 函数F的蕴含项P，如果不存在蕴含项 $Q \neq P$ ，使得 $P \Rightarrow Q \Rightarrow F$ ，则称P为本原蕴含项

BC	00	01	11	10
A				
0	0	0	0	1
1	0	0	1	1

①

②

③

②/③: 本原蕴含项

①: 蕴含项, 非本原蕴含项

Essential Prime Implicant本质本原蕴含项

❖ 如果一个函数中某个元素仅由一个本原蕴含项覆盖，则该本原蕴含项为本质本原蕴含项

$$Y = BC' + AB$$

②/③: 本原蕴含项

BC	00	01	11	10
A				
0	0	0	0	1
1	0	0	1	1

②/③: 本质本原蕴含项，因为
 $A'BC'$ 和 ABC 分别只能由②和③覆盖

寻找本质本原蕴含项举例1

AB \ CD	00	01	11	10
00	0	1	1	0
01	1	1	1	0
11	1	0	1	1
10	0	0	1	1

$$Y = BC' + AC + A'B'D$$

不能被其它本原蕴含项覆盖

本原蕴含项: BC' , AC , AB , $A'C'D$, $A'B'D$, $B'CD$

谁是本质本原蕴含项?

本质本原蕴含项: BC' , AC

寻找本质本原蕴含项举例2

AB \ CD	00	01	11	10
00	0	0	1	0
01	1	1	1	0
11	0	1	1	1
10	0	1	0	0

$$Y = ACD + ABC' + A'C'D + A'BC$$

○ 只能被一个本原蕴含项覆盖

本原蕴含项: BD , ACD , ABC' , $A'C'D$, $A'BC$

谁是本质本原蕴含项?

非本质本原蕴含项: BD

寻找本质本原蕴含项举例3

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

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

$$= AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C + A\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + \bar{A}B\bar{C}$$

$$= m_6 + m_4 + m_3 + m_1 + m_5 + m_1 + m_6 + m_2$$

$$= \sum m(1, 2, 3, 4, 5, 6)$$

谁是本质本原蕴含项？ 无！

BC	00	01	11	10
A				
0	0	1	1	1
1	1	1	0	1

本原蕴含项: $A'C$, $A'B$, AB' , AC' , $B'C$, BC'

寻找本质本原蕴含项举例3

$$Y = A\bar{C} + \bar{A}C + \bar{B}C + B\bar{C} = \sum m(1,2,3,4,5,6)$$

找到覆盖所有最小项的本质本原蕴含项

BC	00	01	11	10
A				
0	0	1	1	1
1	1	1	0	1

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

BC	00	01	11	10
A				
0	0	1	1	1
1	1	1	0	1

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

寻找本质本原蕴含项举例4

$$Y = ABC + ABD + A\bar{C}\bar{D} + \bar{C}\bar{D} + A\bar{B}C + \bar{A}C\bar{D}$$

		<u>D</u>				
		CD	00	01	11	10
AB						
A	00	1				1
	01	1				1
	11	1	1, 1	1, 1		1
	10	1	1	1		1
		<u>C</u>				
		B				

寻找本质本原蕴含项举例4

$$Y = A + \bar{D}$$

CD \ AB	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	1	1	1
10	1	1	1	1

\bar{D}

寻找本质本原蕴含项举例4

$$Y = A + \bar{D}$$

CD	00	01	11	10
AB				
00	1	0	0	1
01	1	0	0	1
11	1	1	1	1
10	1	1	1	1

A

$$\bar{Y} = \bar{A}D$$

$$Y = \bar{\bar{Y}} = \overline{\bar{A}D} = A + \bar{D}$$

\bar{D}

用卡诺图化简举例

$$\begin{aligned} Y &= A\bar{C} + \bar{A}C + \bar{B}C + B\bar{C} \\ &= A(B + \bar{B})\bar{C} + \bar{A}(B + \bar{B})C + (A + \bar{A})\bar{B}C + (A + \bar{A})B\bar{C} \\ &= AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C + A\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + \bar{A}B\bar{C} \\ &= m_6 + m_4 + m_3 + m_1 + m_5 + m_1 + m_6 + m_2 \\ &= \sum m(1, 2, 3, 4, 5, 6) \end{aligned}$$



观察卡诺图，以下哪些选项不是本原蕴含项？

- ☐ A ac'
- ☐ B $a'bc'$
- ☐ C ab'
- ☐ D $a'b'd'$

cd	00	01	11	10
ab				
00	1	0	0	1
01	1	1	0	0
11	1	1	0	0
10	1	1	1	1

D

B

A

C

解答: **BD**

可以看到

B蕴含E ($a'bc' \Rightarrow bc'$)

D蕴含I ($a'b'd' \Rightarrow b'd'$)

因此根据本原蕴含项的定义, BD项都不是本原蕴含项

函数F的蕴含项P, 如果不存在蕴含项 $l \neq P$, 使得 $P \Rightarrow l \Rightarrow F$, 则称P为本原蕴含项

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	1	1	0	0
10	1	1	1	1

Diagram illustrating the Karnaugh map for function F with prime implicants highlighted:

- E** (Essential Prime Implicant): A blue rectangle covering the 1s in the first two columns (ab=00, 01).
- B** (Non-essential Prime Implicant): A green rectangle covering the 1s in the first two columns of the first two rows (cd=00, 01).
- D** (Non-essential Prime Implicant): A green rectangle covering the 1s in the first and fourth columns of the first row (cd=00).
- I** (Non-essential Prime Implicant): A red rectangle covering the 1s in the first column of the first two rows (cd=00, 01).
- J** (Non-essential Prime Implicant): A red rectangle covering the 1s in the fourth column of the first two rows (cd=00, 01).

Don't care 无关项

❖ 不完全确定的函数

- BCD累加电路
- 十进制数字到七段显示译码电路
- 输入取值组合1010~1111在实际电路中不出现

❖ D代表无关项表示的最大项

❖ d代表无关项表示的最小项

BCD累加电路的真值表

Function: $Y_8Y_4Y_2Y_1=ABCD+1$ (循环表示)

Input				Output				Input				Output			
A	B	C	D	Y_8	Y_4	Y_2	Y_1	A	B	C	D	Y_8	Y_4	Y_2	Y_1
0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0
0	0	1	0	0	0	1	1	1	0	1	0	x	x	x	x
0	0	1	1	0	1	0	0	1	0	1	1	x	x	x	x
0	1	0	0	0	1	0	1	1	1	0	0	x	x	x	x
0	1	0	1	0	1	1	0	1	1	0	1	x	x	x	x
0	1	1	0	0	1	1	1	1	1	1	0	x	x	x	x
0	1	1	1	1	0	0	0	1	1	1	1	x	x	x	x

BCD累加电路的布尔表达式

❖ $Y_1 = D'$

- ❖ 电路的任何实际实现输入都会产生一定的输出（包括无关项）

CD \ AB	00	01	11	10
00	000 1	0010	0100	001 1
01	010 1	0110	1000	011 1
11	xxx 1	xxxx	xxx x	xxx 1
10	100 1	0000	xxx x	xxx 1

无关项在逻辑化简时，可以任意假设其为0或1

用卡诺图得到最简或与表达式

CD	00	01	11	10
AB				
00	0	0	1	1
01	1	1	0	1
11	×	×	×	×
10	1	1	×	×

此例
与前面的BCD
累加器的例子
无关

$$\bar{Y} = \bar{A}\bar{B}\bar{C} + BCD$$

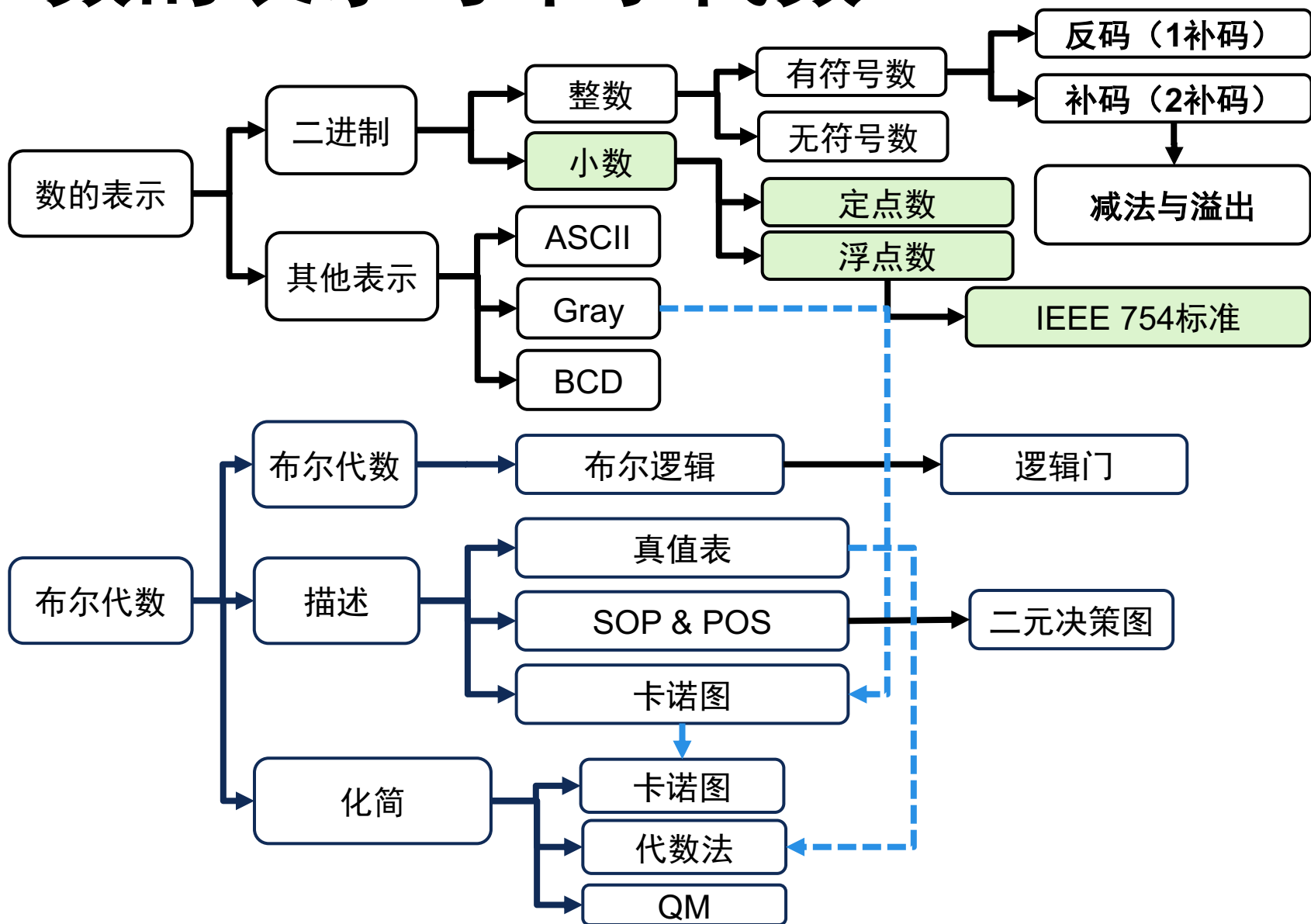
$$Y = \bar{\bar{Y}} = (A + B + C) \cdot (\bar{B} + \bar{C} + \bar{D})$$

POS Form

Quine-McCluskey法

- ❖ 由W.V. Quine和Edward J. McCluskey提出的表格形式的化简方法
- ❖ 可用计算机辅助设计
- ❖ 与卡诺图方法一致，找出相邻最小项，消除变量因子
- ❖ 用于多输入变量的函数化简

数的表示与布尔代数



总结

- ❖ To know
 - Number system
- ❖ To design
 - Simplified Boolean logic and circuits
- ❖ To celebrate
 - The beauty of Boolean methodology