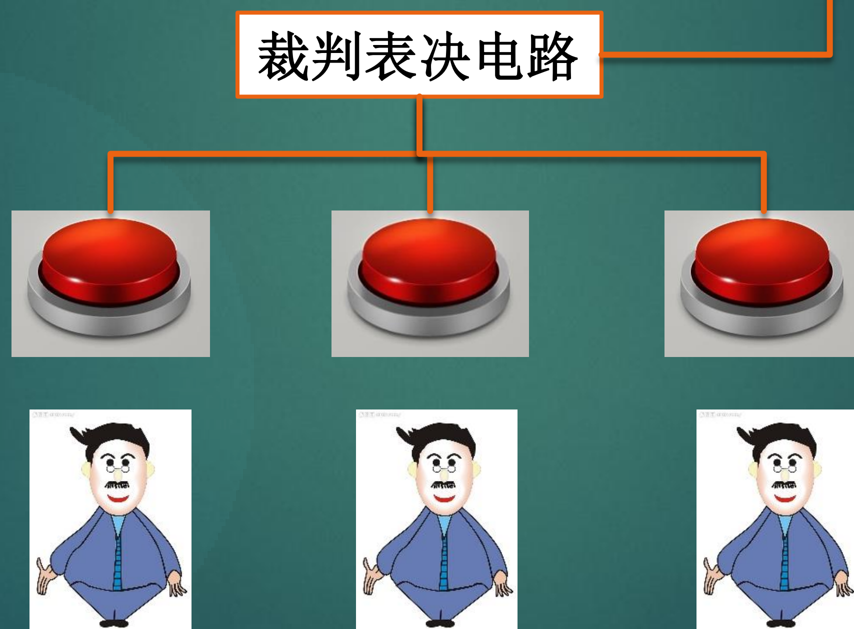
《数字逻辑》
Digital Logic

第二章
逻辑代数基础

北京工业大学软件学院
王晓懿

一个逻辑问题：举重裁判电路



规则：

1. 主裁判具有一票否决权
2. 需多数裁判判定成功

设A表示主裁判判定结果，B表示副裁判判定结果，C表示另一个裁判判定结果；Y表示最终判定结果。

1. 如何分析设计Y与A、B、C之间的关系，使之符合规则要求？

2. 如何设计电路，使之实现规则要求？

3. 如果将裁判人数扩展到5人，7人又如何设计呢？

逻辑代数概述

在数字电路中，主要研究的是电路的输入输出之间的逻辑关系，因此数字电路又称逻辑电路，其研究工具是逻辑代数（布尔代数或开关代数）。

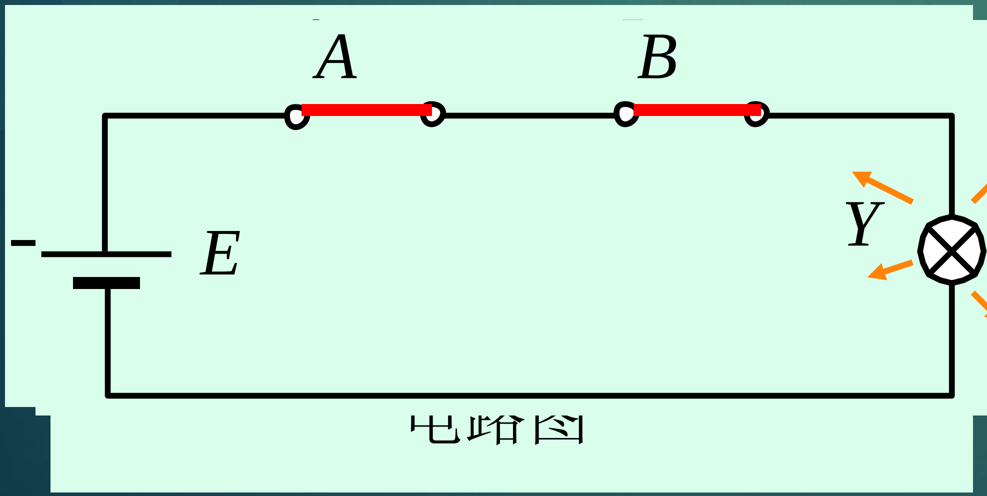
逻辑变量：用字母表示，取值只有0和1。
此时，0和1不再表示数量的大小，只代表两种不同的状态。

逻辑代数中的三种基本运算

一、与逻辑（与运算）

与逻辑：仅当决定事件（Y）发生的所有条件（A，B，C，...）均满足时，事件（Y）才能发生。表达式为： $Y = A\,B\,C\ldots$

例：开关A，B串联控制灯泡Y



A、B都断开，灯不亮。

A、B都接通，灯亮。

A接通、B断开，灯不亮。

A断开、B接通，灯不亮。

将开关接通记作1，断开记作0；灯亮记作1，灯灭记作0。可以作出如下表格来描述与逻辑关系：

功能表

开关 A	开关 B	灯 Y
断开	断开	灭
断开	闭合	灭
闭合	断开	灭
闭合	闭合	亮

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

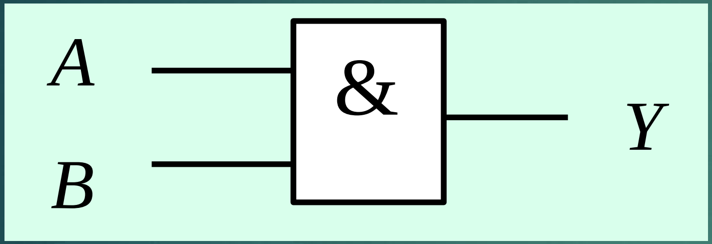
真值表

两个开关均接通时，灯才会亮。逻辑表达式为：

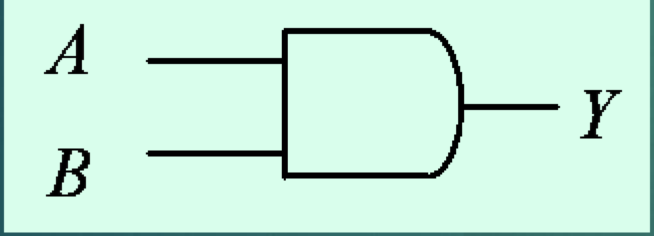
$$Y = A \cdot B$$

实现与逻辑的电路称为与门。

与门的逻辑符号：

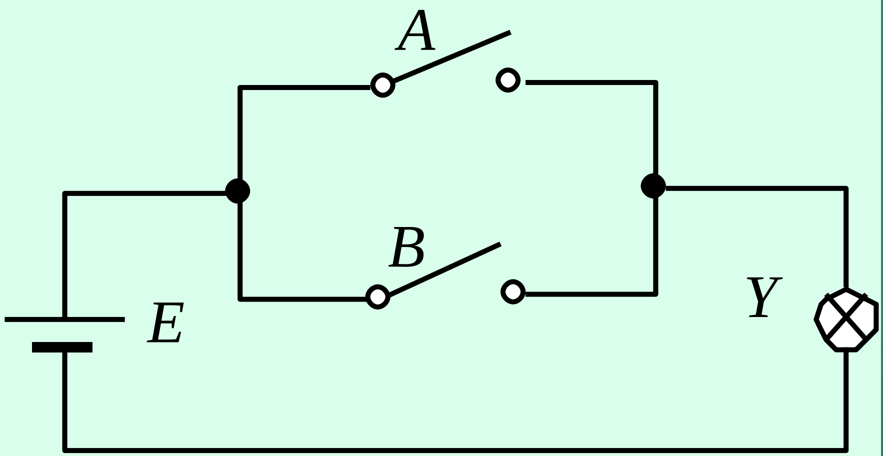


$$Y = A \cdot B$$



二、或逻辑（或运算）

或逻辑： 当决定事件（Y）发生的各种条件A，B，C，...)中，只要有一个或多个条件具备，事件（Y）就发生。表达式为：$Y = A + B + C + \ldots$



电路图

真值表

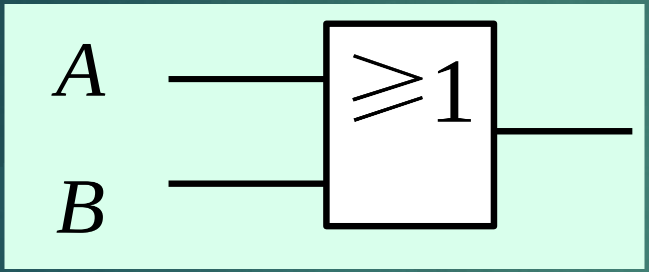
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

两个开关只要有一个接通，灯就会亮。逻辑表达式为：

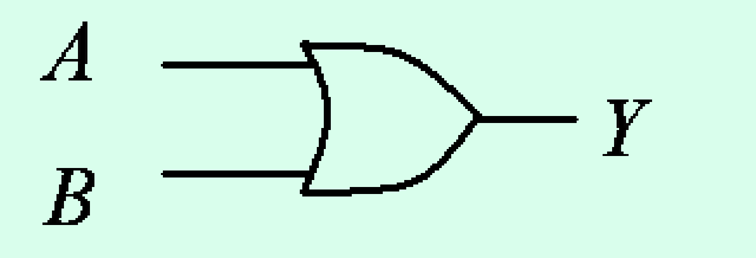
$$Y = A + B$$

实现或逻辑的电路称为**或门**。

或门的逻辑符号：

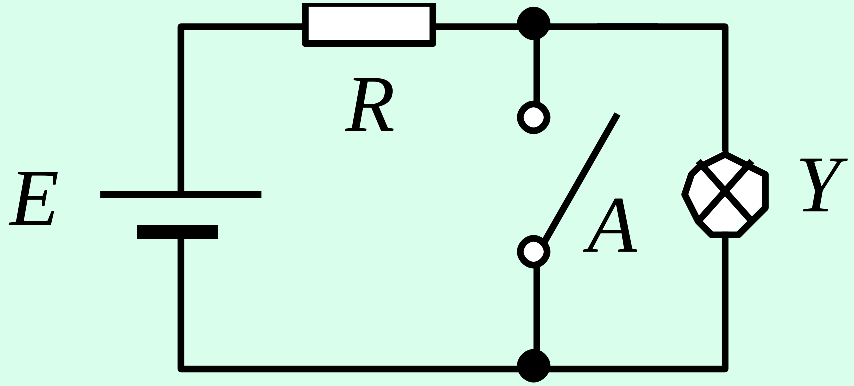


Y=A+B



三、非逻辑（非运算）

非逻辑： 指的是逻辑的否定。当决定事件（Y）发生的条件（A）满足时，事件不发生；条件不满足，事件反而发生。表达式为：$Y=\overline{A}$ 或Y＝A′



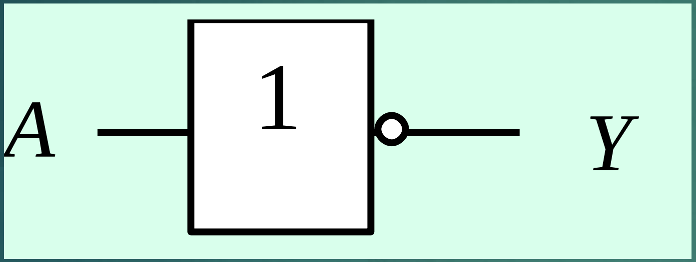
电路图

真值表

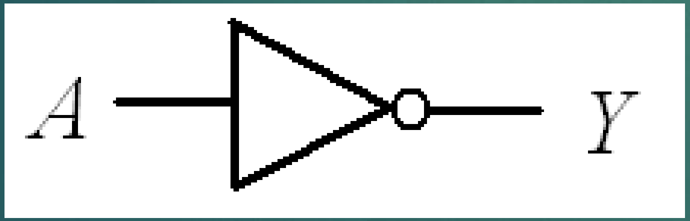
A	Y
0	1
1	0

实现非逻辑的电路称为非门。

非门的逻辑符号：



$Y = A'$



逻辑函数及其表示方法

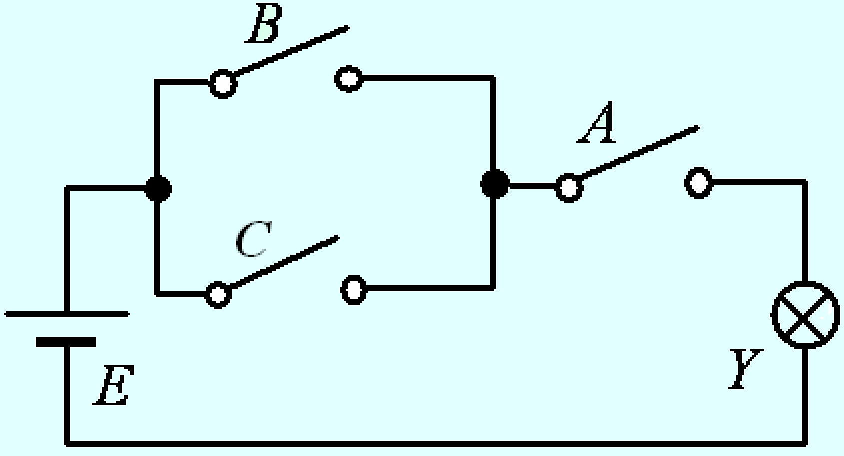
一、逻辑函数

如果以逻辑变量作为输入，以运算结果作为输出，当输入变量的取值确定之后，输出的取值便随之而定。输出与输入之间的函数关系称为逻辑函数。Y=F(A,B,C,...)

二、逻辑函数表示方法

常用逻辑函数的表示方法有：逻辑真值表（真值表）、逻辑函数式（逻辑式或函数式）、逻辑图、波形图、卡诺图及硬件描述语言。它们之间可以相互转换。

举重裁判电路



电路图

设A、B、C为1表示开关闭合，0表示开关断开；

Y为1表示灯亮，为0表示灯暗。得到逻辑函数的各种表示形式：

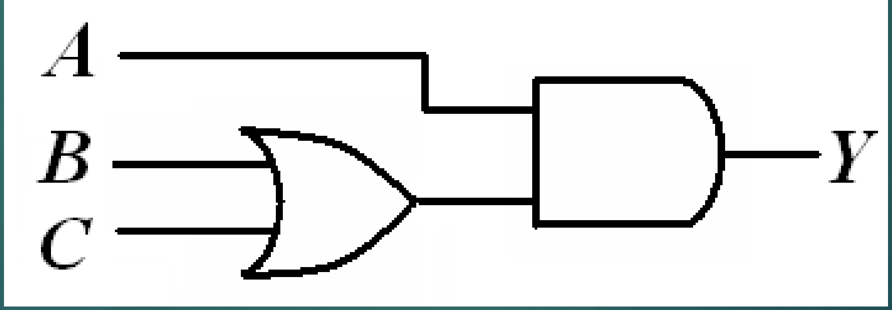
真值表

输	入		输 出
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

函数式

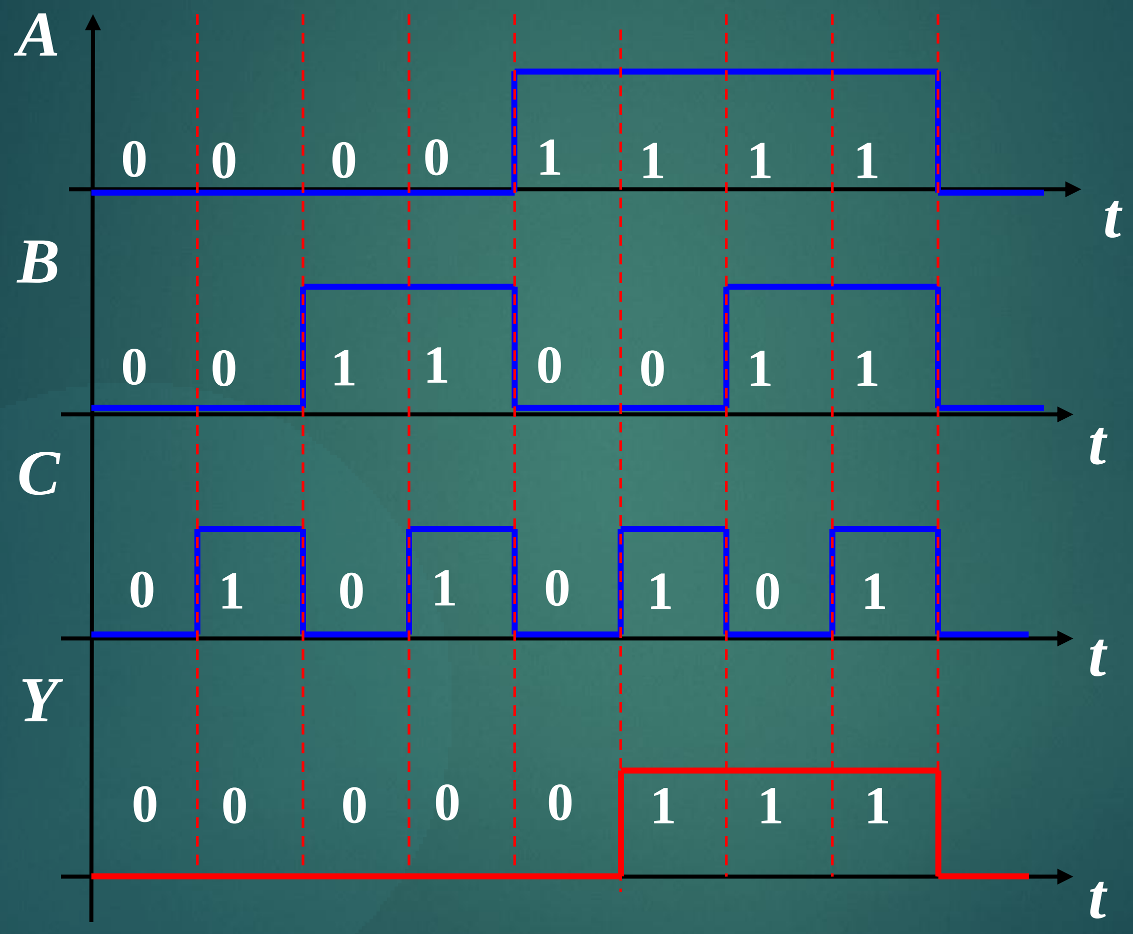
$$
\begin{aligned}
Y &= A(B+C) \\
&= AB'C + ABC' + ABC
\end{aligned}
$$

逻辑图



波形图

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



真值表：将输入、输出的所有可能状态一一对应地列出。

A	Y
0	1
1	0

一输入变量，二种组合

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

二输入变量，四种组合

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

三输入变量，八种组合

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

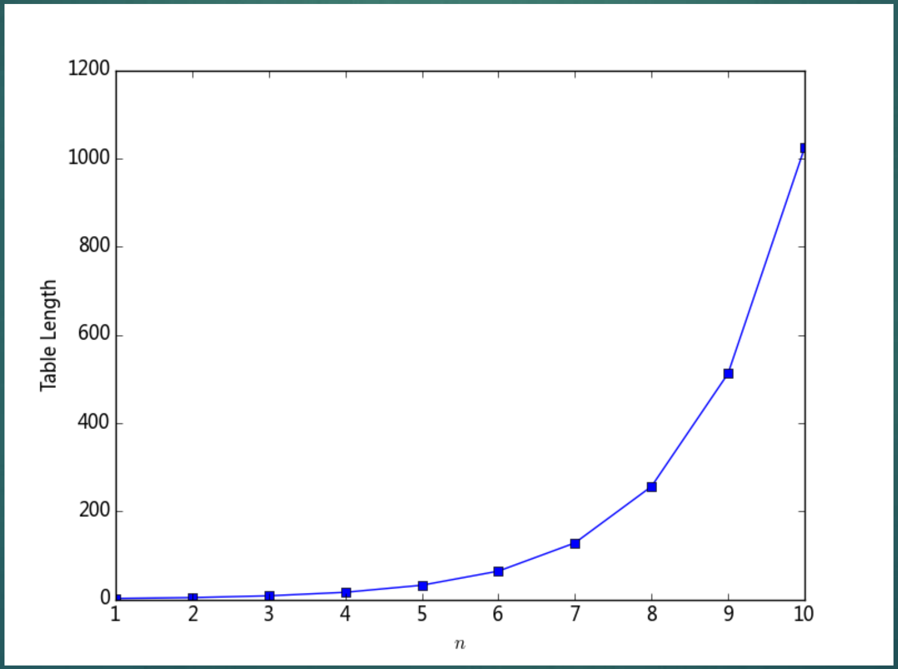
A	B	C	D	Y
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

四输入变量，16种组合

真值表的特点

- 思路直接，表达直观
- 真值表的长度随逻辑变量个数增多迅速增加

n个输入变量的真值表的行数为 2^n



逻辑函数式

把逻辑函数的输入、输出关系写成**与**、**或**、**非**等逻辑运算的组合式，即**逻辑代数式**，又称为**逻辑函数式**，通常采用“**与或**”的形式。

比如：$Y = AB'C + ABC' + ABC$

逻辑函数式的特点

优点：

- 可以较为简洁

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

- 易于变量替换（代入定理）

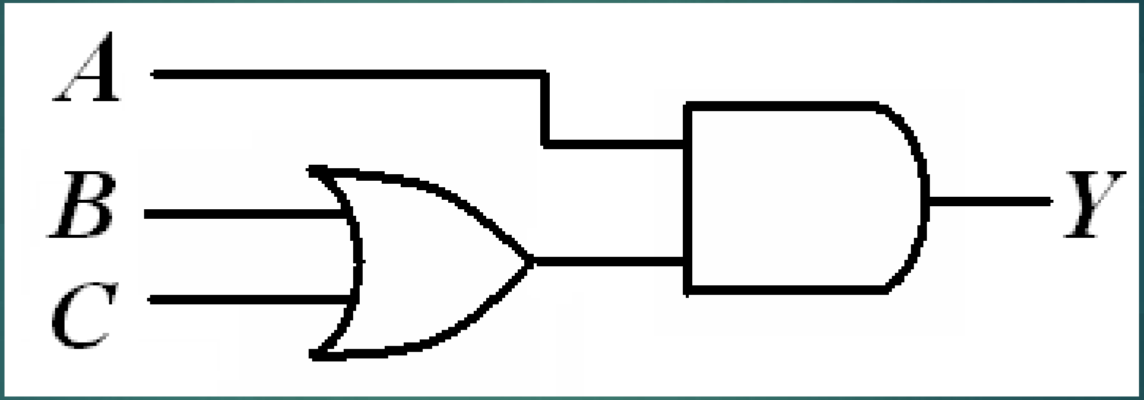
缺点：

- 表达式不唯一
- 较为抽象，难于直接得到

$$Y = A(B+C)$$
$$= AB'C + ABC' + ABC$$

逻辑图：

把相应的逻辑关系用逻辑符号和连线表示出来。



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

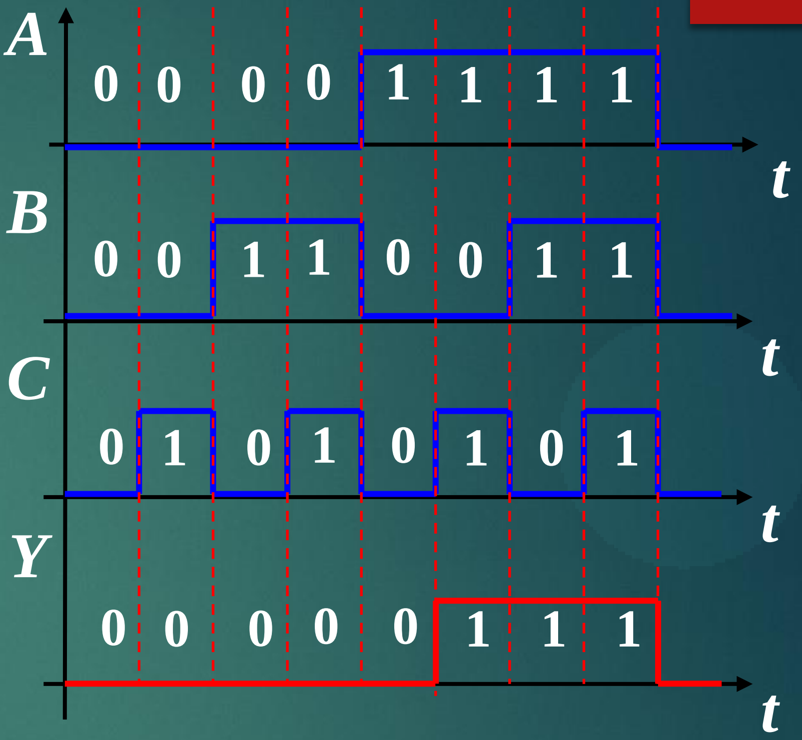
特点：

直接对应于逻辑门电路实现

逻辑表达较为繁琐，用于分析设计逻辑较为不便

波形图：

真值表的“改头换面”？

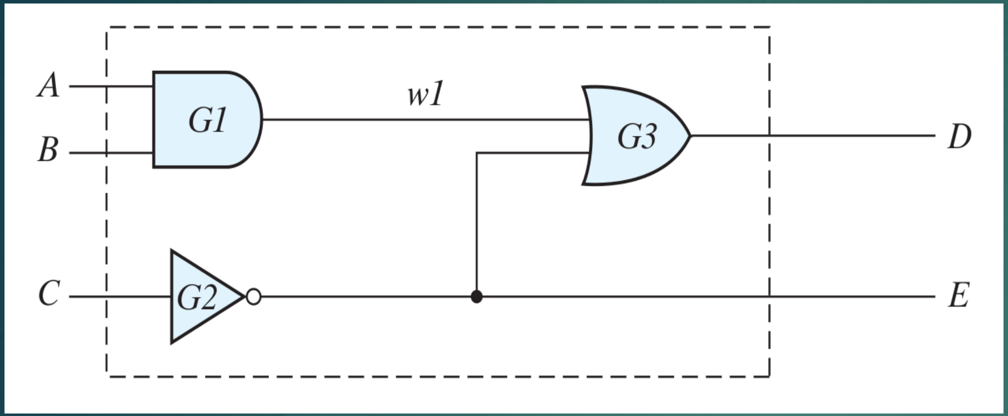


特点：

体现了逻辑变量的电信号表示。

对于分析 “不完美”的电信号所导致问题非常有用。

硬件描述语言（HDL）：




```
module  Simple_Circuit (A, B, C, D, E);
  output          D, E;
  input           A, B, C;
  wire            w1;

  and             G1 (w1, A, B); // Optional gate instance name
  not             G2 (E, C);
  or              G3 (D, w1, E);
endmodule
```

几种表示方法之间的相互转换

1、真值表→逻辑函数式

方法:将真值表中为1的项相加,写成 “与或式”。

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

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

2、逻辑式→真值表

方法:将输入变量取值的所有组合状态逐一带入逻辑式求函数值，列成表即得真值表。

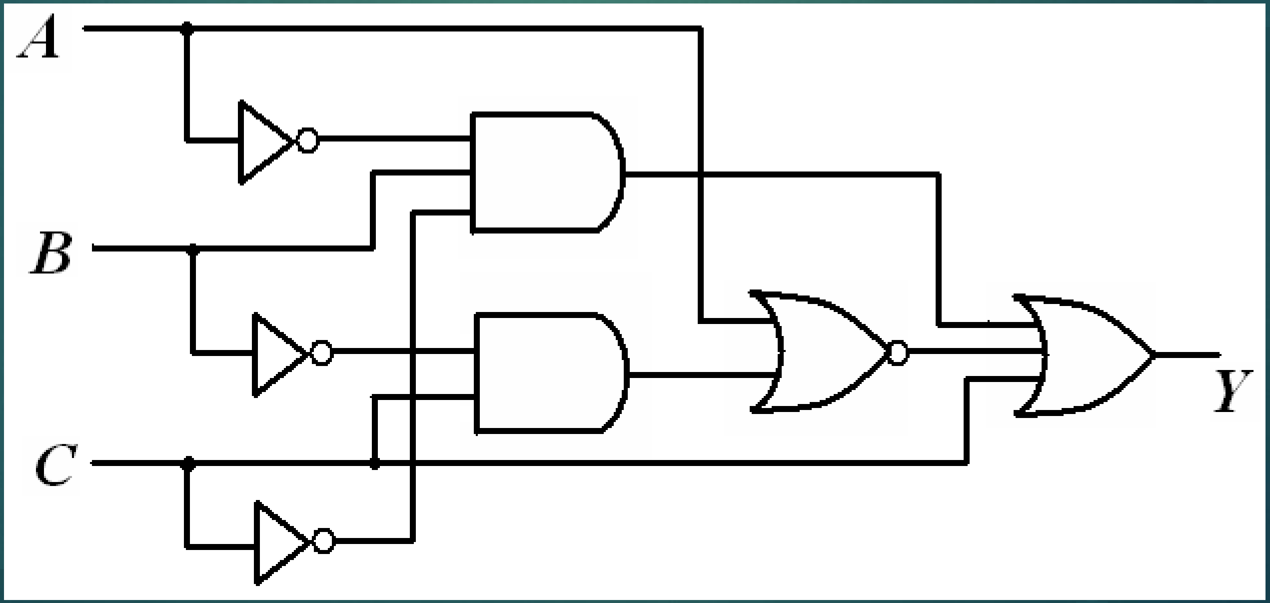
$$Y = A + B'C + A'BC'$$

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

3、逻辑式→逻辑图

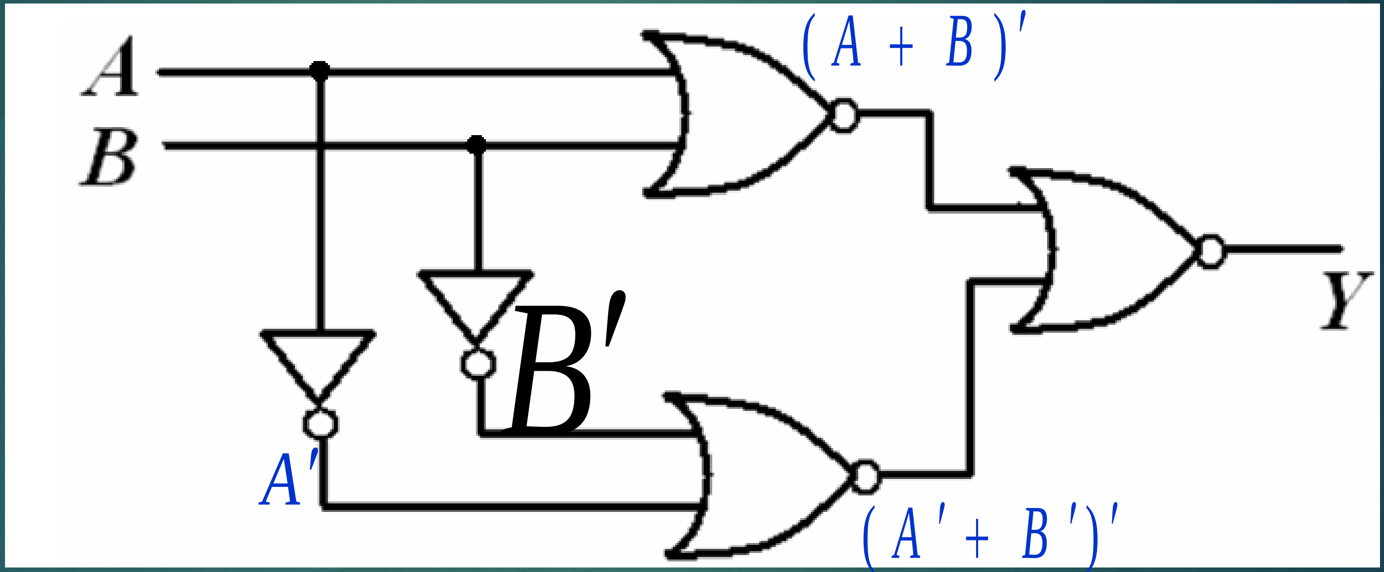
方法:用图形符号代替逻辑式中的运算符号，就可以画出逻辑图.

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



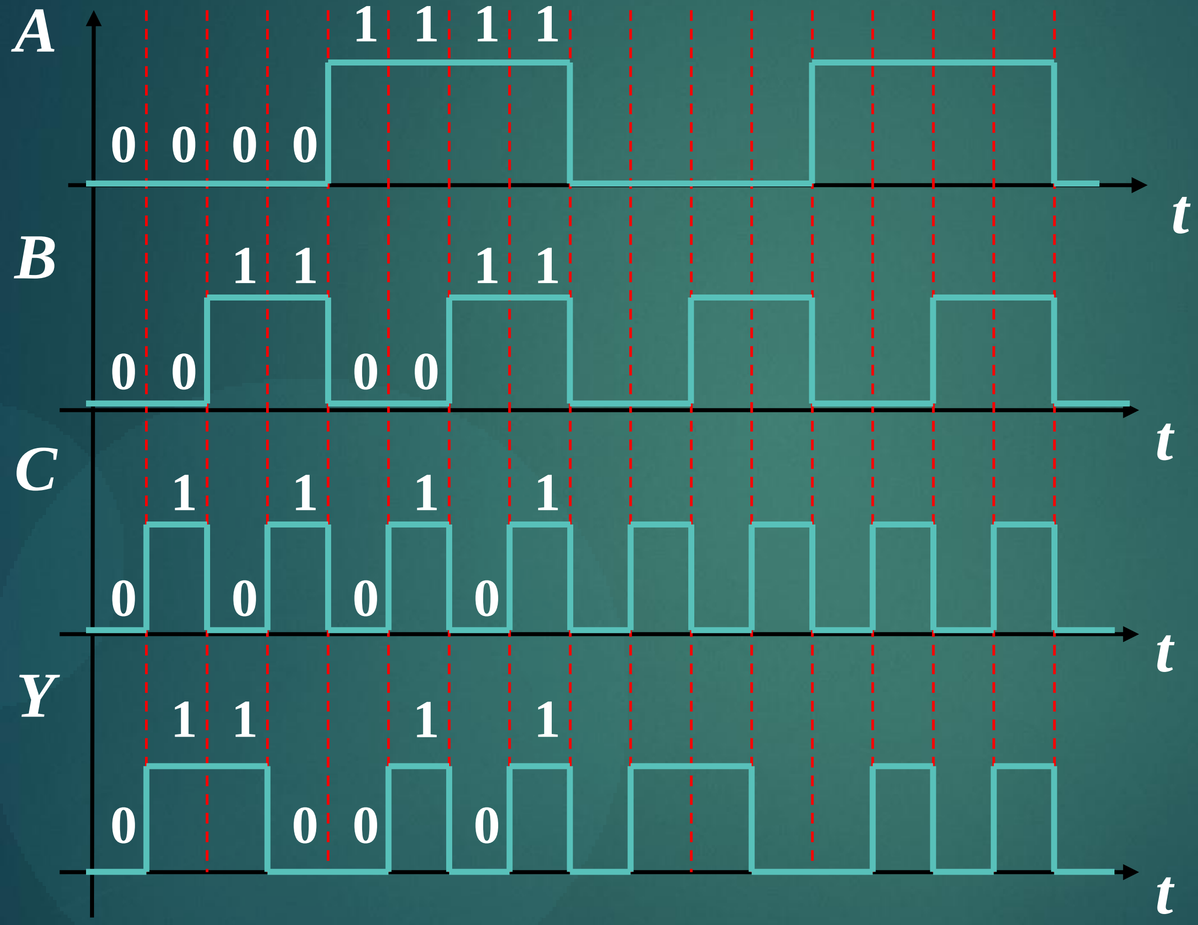
4、逻辑图→逻辑式

方法:从输入端到输出端逐级写出每个图形符号对应的逻辑式，即得到对应的逻辑函数式.



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

5、波形图→真值表



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

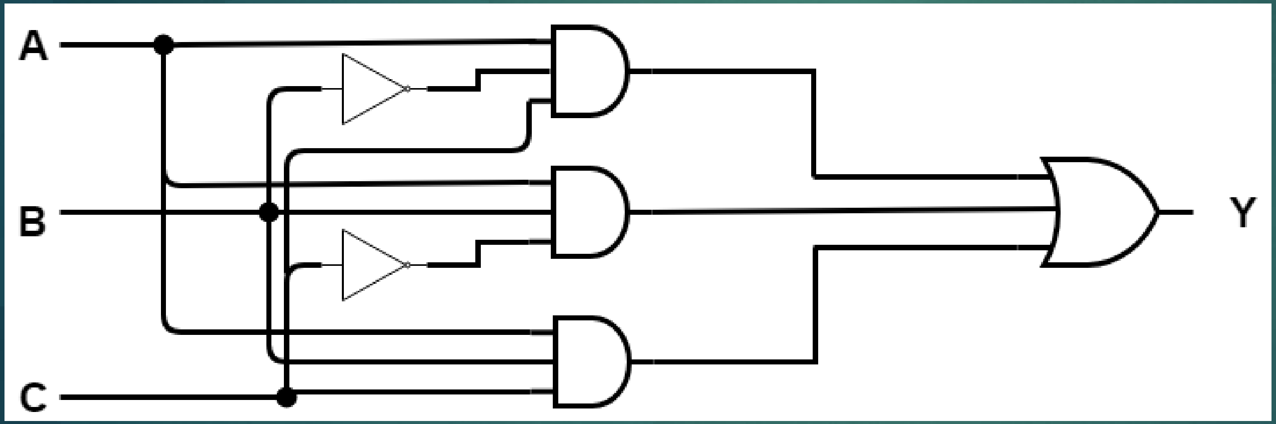
逻辑函数化简

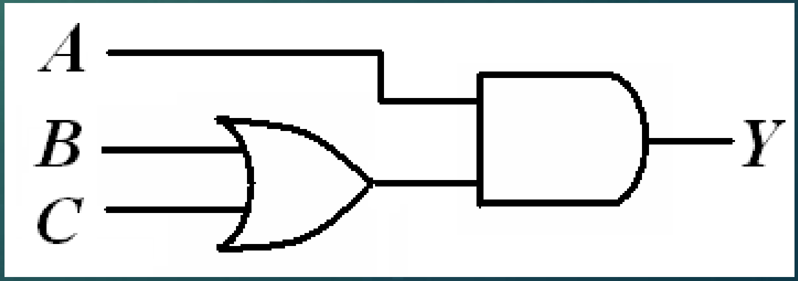
举重裁判电路：

$$Y = AB'C + ABC' + ABC$$

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

为什么要化简逻辑函数？





公式法化简逻辑函数

逻辑函数化简的目的: 省器件！用最少的门实现相同的逻辑功能，每个门的输入也最少。

主要掌握与或表达式的化简：

(1) 乘积的个数最少(用门电路实现，所用与门的个数最少)

(2) 在满足(1)的条件下，乘积项中的变量最少(与门的输入端最少)

最简的目标不同，达到的效果也不同。如果功耗最小或者可靠性最高是目标，化简的结果完全不同！

逻辑代数的基本公式和定律

一、基本公式

1.常量之间的关系

与运算： $0 \cdot 0 = 0$　　$0 \cdot 1 = 0$　　$1 \cdot 0 = 0$　　$1 \cdot 1 = 1$

或运算： $0 + 0 = 0$　　$0 + 1 = 1$　　$1 + 0 = 1$　　$1 + 1 = 1$

非运算： $\overline{1} = 0$　　$\overline{0} = 1$

请特别注意与普通代数不同之处

2.基本公式

0-1 律： $\begin{cases} A+0=A \\ A\cdot 1=A \end{cases}$ $\begin{cases} A+1=1 \\ A\cdot 0=0 \end{cases}$

分别令A=0及A=1代入这些公式，即可证明它们的正确性。

互补律： $A+\bar{A}=1 \quad A\cdot\bar{A}=0$

重叠律： $A+A=A \quad A\cdot A=A$

还原律（双重否定律）： $\bar{\bar{A}}=A$

亦称 非非律

3.基本定理

交换律：$\begin{cases} A\cdot B = B\cdot A \\ A+B = B+A \end{cases}$

结合律：$\begin{cases} (A\cdot B)\cdot C = A\cdot(B\cdot C) \\ (A+B)+C = A+(B+C) \end{cases}$

分配律：$\begin{cases} A\cdot(B+C) = A\cdot B + A\cdot C \\ A+B\cdot C = (A+B)\cdot(A+C) \end{cases}$

反演律（摩根定律）：$\begin{cases} \overline{(A\cdot B)} = \overline{A}+\overline{B} \\ \overline{(A+B)} = \overline{A}\cdot\overline{B} \end{cases}$

利用真值表很容易证明这些公式的正确性。如证明A·B=B·A：

A	B	AB	BA
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

二、常用公式

1. $A+AB = A$

2. $A+\overline{A}B= A+B$　　$A(\overline{A}+B)= AB$

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

注：红色变量被吸收掉！统称 吸收律

冗余定律或多余项定理的其他形式

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

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

同理：此多余项可以扩展成其他形式

逻辑代数的基本定理

一、代入定理

任何一个含有变量A的等式，如果将所有出现A的位置都用同一个逻辑函数代替，则等式仍然成立。这个规则称为代入定理。

例如，已知等式 $\overline{(A\cdot B)}=\bar{A}+\bar{B}$，用函数*Y*=*BC*代替等式中的*B*，根据代入定理，等式仍然成立，即有：

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

二、 反演定理

对于任何一个逻辑表达式Y，如果将表达式中的所有“·”换成“＋”，“＋”换成“·”，“0”换成“1”，“1”换成“0”，原变量换成反变量，反变量换成原变量，那么所得到的表达式就是函数Y的反函数$\overline{Y}$（或称补函数）。这个规则称为反演定理。

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

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

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

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

应用反演定理应注意两点：

1、保持原来的运算优先顺序，即如果在原函数表达式中，AB之间先运算，再和其它变量进行运算， 那么非函数的表达式中，仍然是AB之间先运算。

2、不属于单个变量上的反号应保留不变。

三、 对偶定理

对于任何一个逻辑表达式*Y*，如果将表达式中的所有“·”换成“＋”，“＋”换成“·”，“0”换成“1”，“1”换成“0”，而变量保持不变，则可得到的一个新的函数表达式 Y^D, Y^D称为*Y*的对偶式。

对偶定理：如果两个逻辑式相等，则它们的对偶式也相等。

利用对偶规则,可以使要证明及要记忆的公式数目减少一半。

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

$$Y^D = A + B \cdot C$$

$$Y = \overline{(AB + CD)}$$

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

（2）式 $1 \cdot A = A$

（12）式 $0 + A = A$

$$A(B + C) = AB + AC$$

$$A + BC = (A + B)(A + C)$$

逻辑函数的标准形式

逻辑函数可以表示为最小项之和的形式（与或表达式）或者最大项之积的形式（或与表达式）

应用最多的是最小项之和的形式，也叫最小项标准式。

最小项也是卡诺图化简的基础。

BACK

最小项(MinTerm)

逻辑函数有n个变量，由它们组成的具有n个变量的乘积项中，每个变量以原变量或反变量的形式出现且仅出现一次，这个乘积项为最小项。N个变量有2^n个最小项。

例如：n=3，对A、B、C，有8个最小项

$$\overline{A}\,\overline{B}\,\overline{C} \quad A\overline{B}\,\overline{C} \quad \overline{A}B\overline{C} \quad AB\overline{C}$$

$$\overline{A}\,\overline{B}C \quad A\overline{B}C \quad \overline{A}BC \quad ABC$$

最小项(续)

- 对任意最小项，只有一组变量取值使它的值为1，其他取值使该最小项为0
- 为方便起见，将最小项表示为m_i

n=3的8个最小项为：

$$m_0=\overline{A}\,\overline{B}\,\overline{C}\quad m_1=A\overline{B}\,\overline{C}\quad m_2=\overline{A}B\overline{C}\quad m_3=AB\overline{C}$$

$$m_4=\overline{A}\,\overline{B}C\quad m_5=A\overline{B}C\quad m_6=\overline{A}BC\quad m_7=ABC$$

最小项(续)

- 任何逻辑函数均可表示为唯一的一组最小项之和的形式，称为标准的与或表达式
- 某一最小项不是包含在**F**的原函数中，就是包含在**F**的反函数中
- 例： $F = \overline{A}B + BC + A\overline{B}\overline{C}$

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

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

$$= m_6 + m_2 + m_7 + m_1$$

$$= \sum m^3(1,2,6,7)$$

BACK

最大项(MaxTerm)

- n个变量组成的或项，每个变量以原变量或反变量的形式出现且仅出现一次，则称这个或项为最大项

例如：n=3的最大项为

$$M_0 = A + B + C \quad M_1 = \overline{A} + B + C$$

$$M_2 = A + \overline{B} + C \quad M_3 = \overline{A} + \overline{B} + C$$

$$M_4 = A + B + \overline{C} \quad M_5 = \overline{A} + B + \overline{C}$$

$$M_6 = A + \overline{B} + \overline{C} \quad M_7 = \overline{A} + \overline{B} + \overline{C}$$

最大项(续)

- 对任意一个最大项，只有一组变量取值使它的值为0，而变量的其他取值使该项为1
- 将最大项记作M_i
- 任何一个逻辑函数均可表示为唯一的一组最大项之积，称为标准的或与表达式
- n个变量全体最大项之积必为“0”
- 某个最大项不是含在F的原函数中，就是在F的反函数中

最大项(续)

例如：

$$
\begin{aligned}
F &= (A+B)\bullet(\overline{A}+B+C) \\
&= [A+B+(C\bullet\overline{C})]\bullet(\overline{A}+B+C) \\
&= (A+B+C)\bullet(A+B+\overline{C})\bullet(\overline{A}+B+C) \\
&= M_0+M_4+M_1 \\
&= \prod M^4(0,1,4)
\end{aligned}
$$

BACK

卡诺图化简逻辑函数

卡诺图(Karnaugh Map): 逻辑函数的图示表示，把最小项填入卡诺图，利用相邻最小项的互补性，消去一个变量，实现化简。

卡诺图的构成

(1)、由矩形或正方形组成的图形

(2)、将矩形分成若干小方块，每个小方块对应一个最小项

BACK

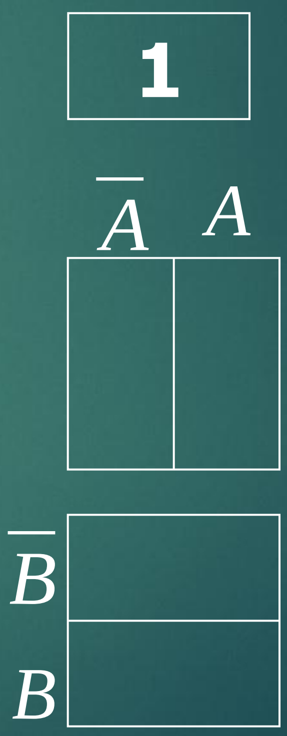
2变量卡诺图(Karnaugh Map)

- 2变量卡诺图

整体为1

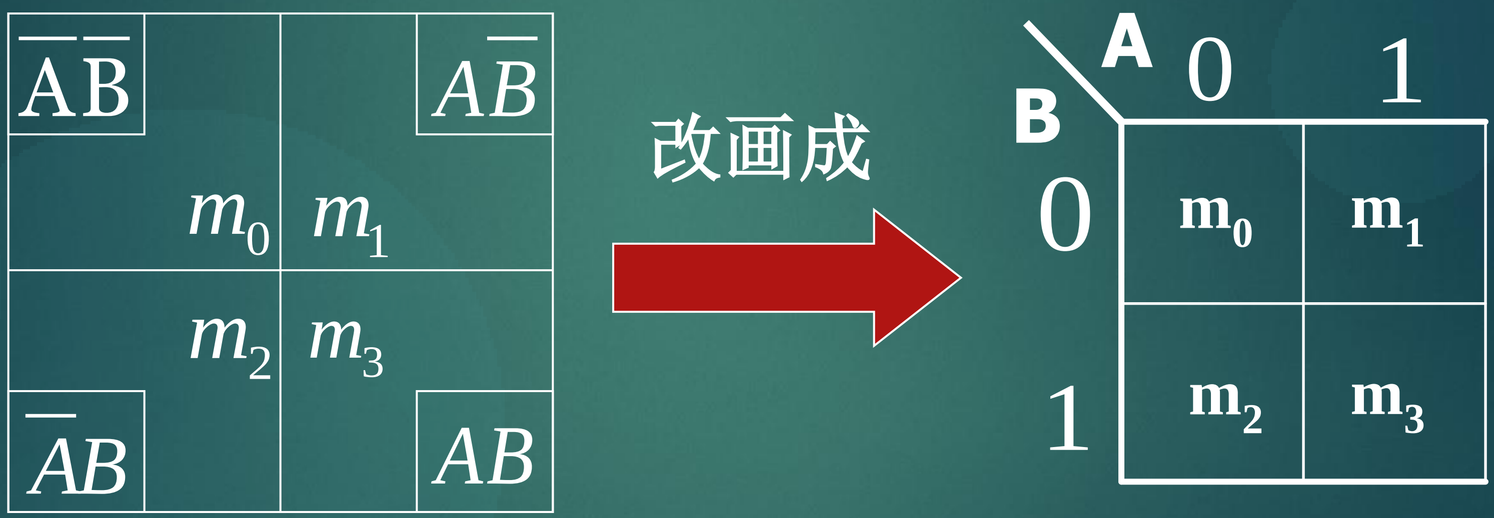
左、右部分表示 $\bar{A}$ A

上、下部分表示 $\bar{B}$ B



2变量卡诺图(Karnaugh Map)

2变量卡诺图可由代表4个最小项的四个小方格组成



2变量卡诺图

3变量Karnaugh Map

3变量卡诺图由**8**个最小项组成，对应图中**8**个小方格

C \ BA	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

注意：表中最小项编码按**00**－**01—11**－**10**循环码顺序排列，而不是**00**－**01**－**10**－**11**（二进制计数的顺序）

4变量Karnaugh Map

DC \ BA	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

卡诺图化简的步骤

1 按照循环码规律指定卡诺图变量取值；

2 在函数最小项对应的小方块填“1”，其他方块填“0”；

3 合并相邻填“1”的小方块，两个方块合并消去一个变量（一维块）；4个方块合并消去两个变量（二维块）；

4 合并过程中先找大圈合并，圈越大消去的变量越多；

5 使每一最小项至少被合并包含过一次；每个合并的圈中，至少要有一个“1”没有被圈过，否则这个圈就是多余的。

“与或”式化简：例1

将表达式$F=AB+\bar{A}C$ 填入卡诺图

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

“与或”式化简：例2

$$F = \overline{B}CD + B\overline{C} + \overline{A}\,\overline{C}D + A\overline{B}C \quad = \sum m^4(2,3,5,8,10,11,12,13)$$

DC \ BA	00	01	11	10
00				
01				
11	1	1		
10				

“与或”式化简：例2（续）

$$F=\overline{B}CD+B\overline{C}+\overline{A}\,\overline{C}D+A\overline{B}C \quad =\sum m^4(2,3,5,8,10,11,12,13)$$

DC \ BA	00	01	11	10
00			1	1
01				
11	1	1		
10			1	1

“与或”式化简：例2（续）

$$F = \overline{B}CD + B\overline{C} + \overline{A}\,\overline{C}D + A\overline{B}C \quad = \sum m^4(2,3,5,8,10,11,12,13)$$

DC \ BA	00	01	11	10
00			1	1
01				
11	1	1		
10	1		1	1

“与或”式化简：例2（续）

$$F=\overline{B}CD+B\overline{C}+\overline{A}\,\overline{C}D+A\overline{B}C \quad =\sum m^4(2,3,5,8,10,11,12,13)$$

DC \ BA	00	01	11	10
00			1	1
01		1		
11	1	1		
10	1		1	1

“与或”式化简：例2（续）

$$F = \overline{B}CD + B\overline{C} + \overline{A}\,\overline{C}D + A\overline{B}C \quad = \sum m^4(2,3,5,8,10,11,12,13)$$

BA / DC	00	01	11	10
00			1	1
01		1		
11	1	1		
10	1		1	1

“与或”式化简：例2（续）

$$F = \overline{B}CD + B\overline{C} + \overline{A}\,\overline{C}D + A\overline{B}C = \sum m^4(2,3,5,8,10,11,12,13)$$

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	1	0	0
11	1	1	0	0
10	1	0	1	1

$B\overline{C}$

$\overline{A}\,\overline{B}D$

$A\overline{B}C$

$$F = B\overline{C} + \overline{A}\,\overline{B}D + A\overline{B}C$$

“与或”式化简：例3

$$F=\overline{A}\,\overline{B}\,\overline{C}+\overline{A}CD+\overline{A}B\overline{D}+AD+AC$$

DC \ BA	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	1	1	1	1
10	1	1	1	0

$\overline{A}\,\overline{C}\,\overline{D}$

BC

AC

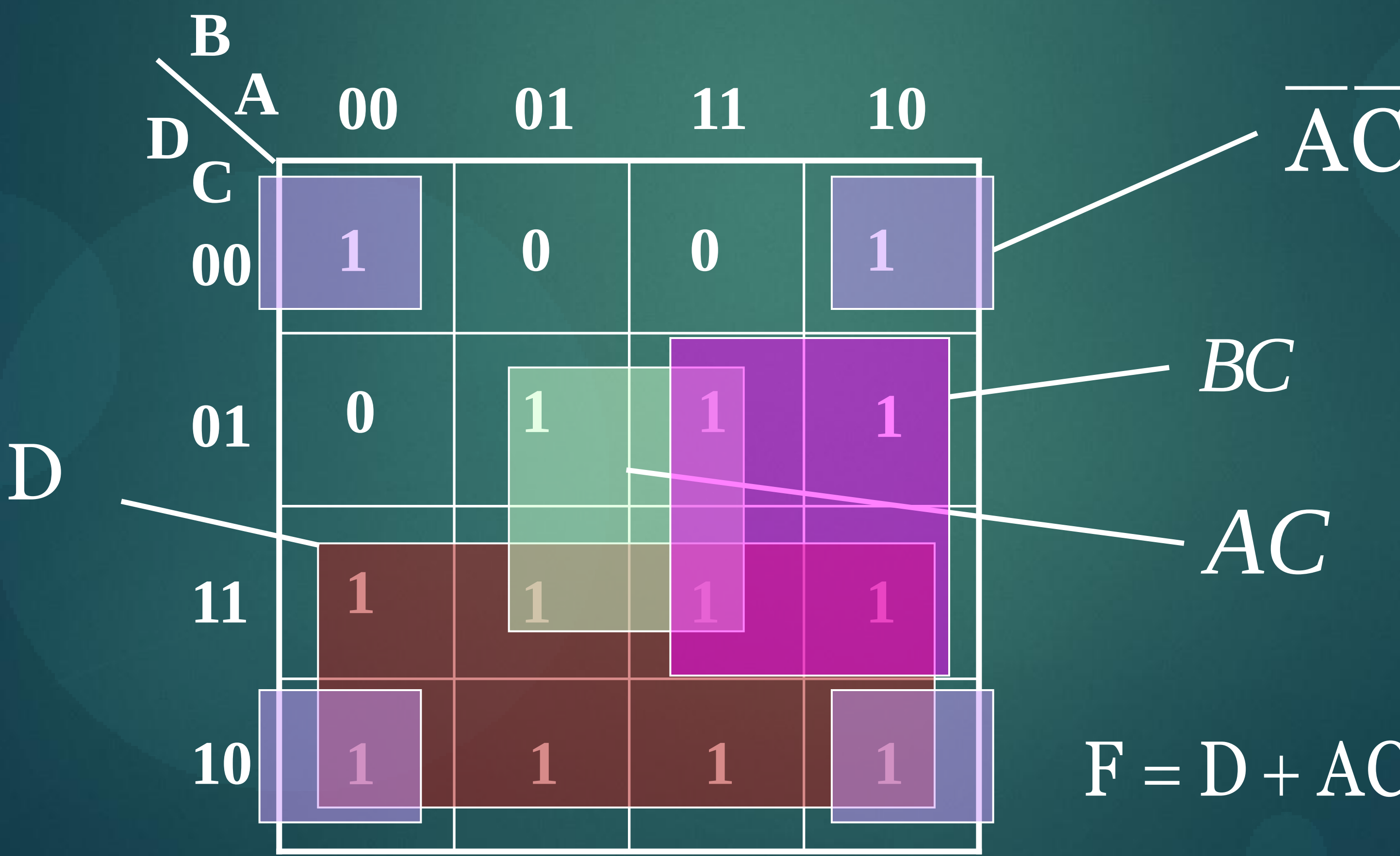
$\overline{B}D$

AD

$$F=AC+AD+\overline{B}D+BC+\overline{A}\,\overline{C}\,\overline{D}$$

“与或”表达式化简：

如果在上图中的m_{10}=1,可以出现8个1相连，消去3个变量。



$$F = D + AC + BC + \overline{A}\,\overline{C}$$

“与或”表达式化简：

此时，图上有13个最小项为1, 只有3个最小项为0，写$\overline{F}$的表达式更简单。

DC \ BA	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	1	1	1	1
10	1	1	1	1

$A\overline{C}\,\overline{D}$

$\overline{A}\,\overline{B}C\overline{D}$

$$\overline{F} = A\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D}$$

$$F = \overline{A\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D}}$$

注意：经常是写$\overline{F}$比直接写F简单。

卡诺图化简总结

卡诺图化简的核心是找到并且合并相邻最小项。

相邻三种情况：相接，相对，相重。5变量卡诺图才会出现相重的情况。

合并过程中先找大圈合并，圈越大消去的变量越多；使每一最小项至少被合并包含过一次；每个合并的圈中，至少要有一个“1”没有被圈过，否则这个圈就是冗余的。

特殊形式的逻辑函数化简

- 基本形式的逻辑函数：
  单输出逻辑函数，F＝f(A,B,C…)
- 特殊形式的逻辑函数：
  1. 多输出逻辑函数
  2. 包含不管项的逻辑函数
- 只要求掌握卡诺图化简法

(1)多输出逻辑函数的化简

多输出逻辑函数：同一组输入变量，有两个以上的输出。

$$F_1 = f(A,B,C\ldots)$$

$$F_2 = f(A,B,C\ldots)$$

化简时，在“与或”表达式中要尽量寻找公共的“或”项，使公共项为多个函数共享，这时从单个输出看可能不是最简，但总体是最简。

$$\begin{cases} F_1 = A\overline{B}\,\overline{C} + A\overline{B}C + ABC \\ F_2 = AB\overline{C} + \overline{A}B\overline{C} + ABC \end{cases}$$

C \ BA	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$F_1 = A\overline{B} + AC$

C \ BA	00	01	11	10
0	0	0	1	1
1	0	0	1	0

$F_2 = AB + B\overline{C}$

单独看F_1, F_2都是最简，但没有公共项。

C \ BA	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$F_1 = A\overline{B} + ABC$

C \ BA	00	01	11	10
0	0	0	1	1
1	0	0	1	0

$F_2 = B\overline{C} + ABC$

ABC是公共项

(2)有“不管项”的逻辑函数化简

包含不管项（**Don’t Care**）的逻辑函数：函数**F**的取值只和一部分最小项有关，另一部分最小项既可以取“**0**”，也可以取“**1**”，这些最小项称“不管项”或“任意项”。

“不管项”的两种情况：

1. 这些输入组合不可能出现
2. 其输入组合虽能出现，但最小项的值是“**1**”还是“**0**”，人们不关心。

例：设计一位十进制数的数值范围判断器，当x>=5，F=1；否则，F=0。
(ABCD表示一位十进制数,A是低位,D是高位)

	A	B	C	D	F
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	1	1	0	0	0
4	0	0	1	0	0
5	1	0	1	0	1
6	0	1	1	0	1
7	1	1	1	0	1

	A	B	C	D	F
8	0	0	0	1	1
9	1	0	0	1	1
	0	1	0	1	φ
	1	1	0	1	φ
	0	0	1	1	φ
	1	0	1	1	φ
	0	1	1	1	φ
	1	1	1	1	φ

有“不管项”的逻辑函数化简（续）

F的卡诺图

把φ项当作“1”

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	φ	φ	φ	φ
10	1	1	φ	φ

AC

BC

D

F=D+AC+BC

本章重点：

逻辑代数与逻辑函数的概念

逻辑函数的化简，重点是4变量卡诺图

课后扩展：

搜索Expresso逻辑化简工具并使用它