NJUAI
南京大学
人工智能学院

第3章 组合逻辑电路

第一讲 组合逻辑电路概述

第二讲 典型组合逻辑部件设计

第三讲 组合逻辑电路时序分析

第一讲 组合逻辑电路概述

- 组合逻辑电路构成规则
- 逻辑电路图
- 两级与多级组合逻辑电路
- 组合逻辑电路设计
- 无关项、非法值和高阻态

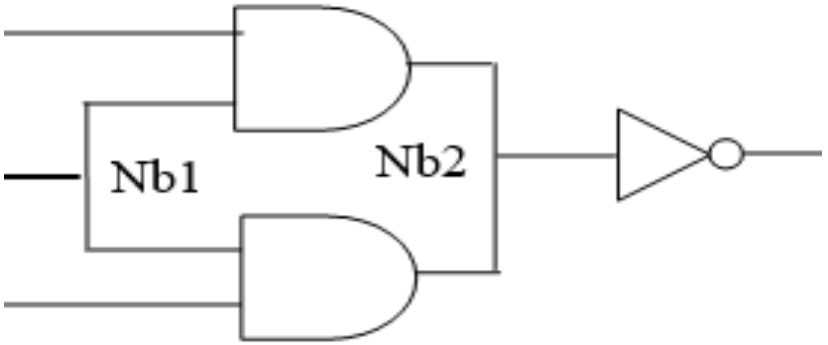
1 组合逻辑电路概述

- 数字逻辑电路可被看成是带有若干输入端和若干输出端的黑盒子，每个输入端和输出端只有高电平、低电平两种状态，对应1或0。
- 分为组合（combinational）逻辑电路和时序（sequential）逻辑电路两种类型。
  - 组合逻辑电路的输出值仅依赖于当前输入值
  - 时序逻辑电路的输出值不仅依赖于输入值，还与当前状态（现态）有关。电路中存在存储部件或反馈结构

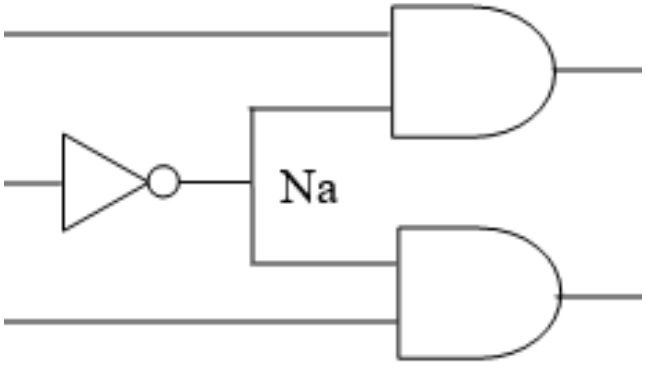
- 黑盒内部可被看成由若干元件和若干连线互连而成。
  - 元件本身又可以是一个数字逻辑电路
  - 连线可以是输入连线(如A1)、内部连线(如N1)和输出连线(如F1)

1.1 组合逻辑电路构成规则

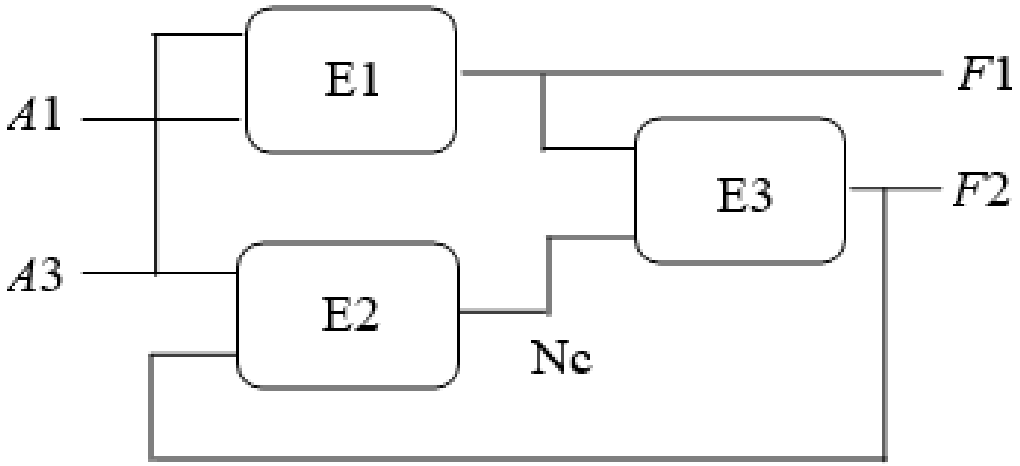
- 最简单的组合逻辑电路是逻辑门电路，实现基本逻辑运算
- 组合逻辑电路构成规则
  - 每个元件本身是组合逻辑电路
  - 输出连线不能互连
  - 输出连线不能反馈到元件输入端



不是组合逻辑电路



是组合逻辑电路



不是组合逻辑电路

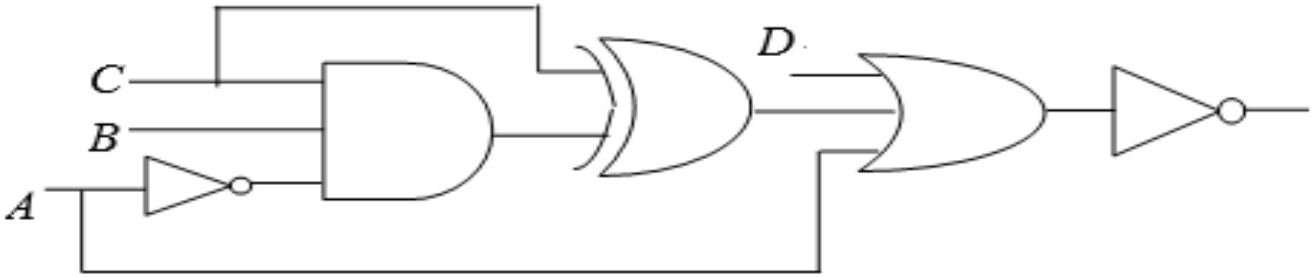
1.2 逻辑电路图

- **逻辑电路图描述数字电路内部元件的结构及其相互连接关系**
  - 每个逻辑电路图的输出信号可以用输入信号的逻辑表达式表示它们之间的逻辑关系
- **逻辑电路图给出了逻辑关系(逻辑表达式)的一种实现方式**
  - 一个真值表可能对应多个不同的逻辑表达式，从而对应多个不同的逻辑电路图，因而可以有多个不同的实现方式
- **任何逻辑表达式都可写成与、或、非三种基本运算的逻辑组合**
  - 任何逻辑表达式都可以用基本逻辑门画出对应的逻辑电路图
- **一个逻辑门的输出可作为另一个逻辑门的输入**
  - 扇入系数：一个逻辑门所允许的输入端的最大数目
  - 扇出系数：一个逻辑门输出端信号所能驱动的下一级输入端的最大数目

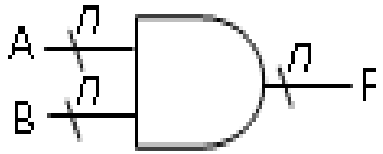
1.2 逻辑电路图

- 画逻辑电路图时，须依据逻辑运算的优先级确定逻辑门间的连接关系
  - 优先级高的运算对应的逻辑门的输出，是优先级低的运算对应逻辑门的输入
  - 优先级顺序如下： 非 > 与和与非 > 异或和同或 > 或和或非

例：画出$\overline{\overline{A}\cdot B\cdot C\oplus C+A+D}$ 对应的逻辑电路图

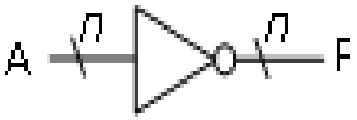


- n位逻辑运算在输入端和输出端标注位数即可

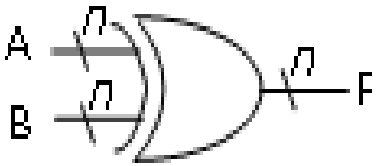


F=A•B

F=A+B



$F=\overline{A}$



F=A⊕B

1.3 两级和多级组合逻辑电路

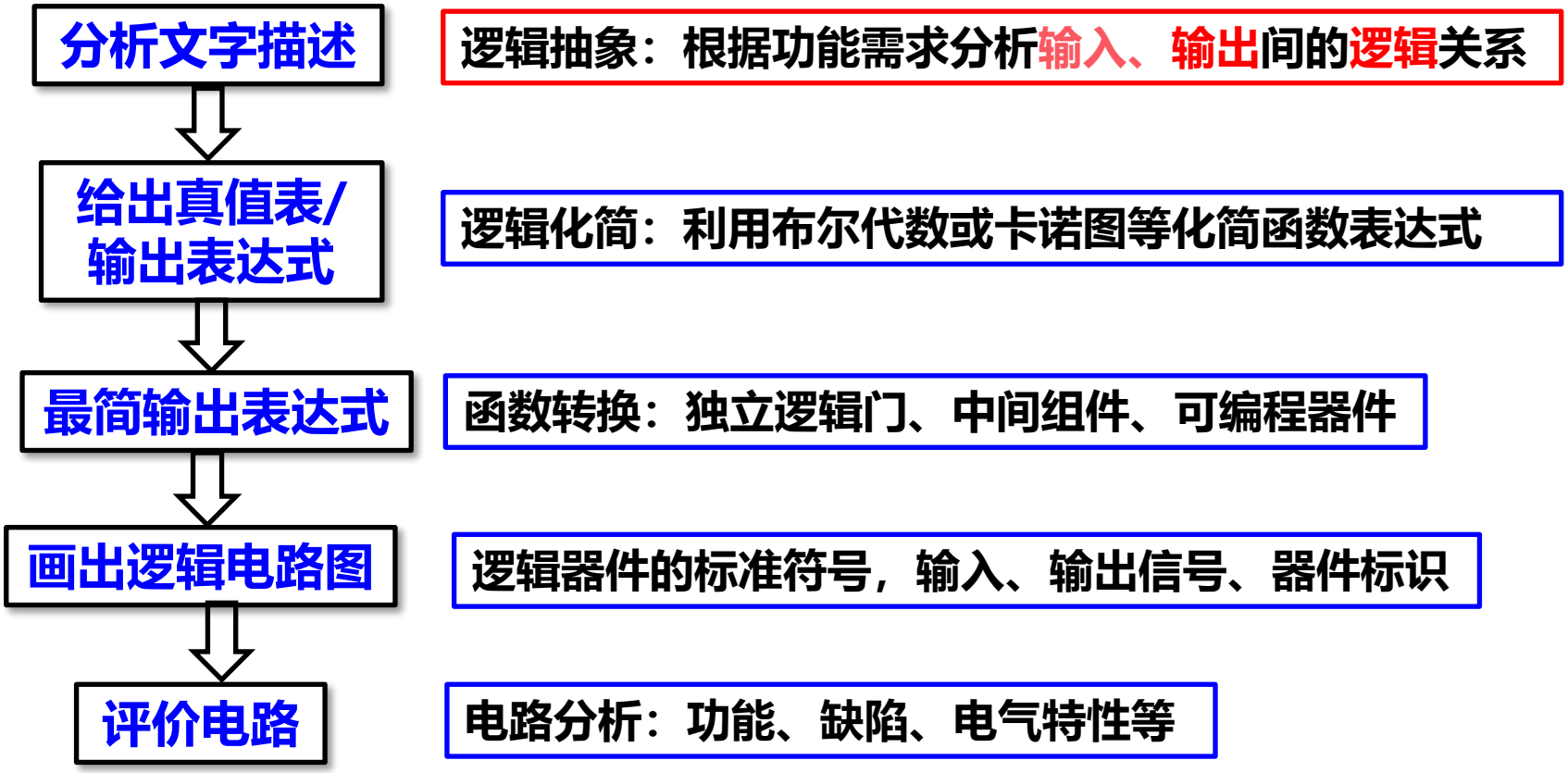
- 任何逻辑表达式都可以转换成与-或表达式和或-与表达式
- 任何组合逻辑电路都可以是一个两级电路
- 与-或表达式对应的电路
  - 第一级是若干个与门
  - 第二级是一个或门，其输入是所有与门的输出
  - 例：$\overline{\overline{A}\cdot B\cdot C\oplus C+A+D}$可转换为 $\overline{A}\cdot B\cdot\overline{D}+\overline{A}\cdot\overline{C}\cdot\overline{D}$

  转换前：最长路径从A输入端到输出经过了非门、与门、异或门、或门和非门

  转换后：最长路径只经过非门、与门和或门
- 两级组合逻辑电路好处：比多级组合逻辑电路的传输时间更短，速度更快；坏处：使用两级组合电路所需的硬件数量可能会增长
- 采用两级还是多级需要在速度和成本之间进行权衡

1.4 组合逻辑电路设计

从文字描述到逻辑电路或系统设计的整个过程如下：

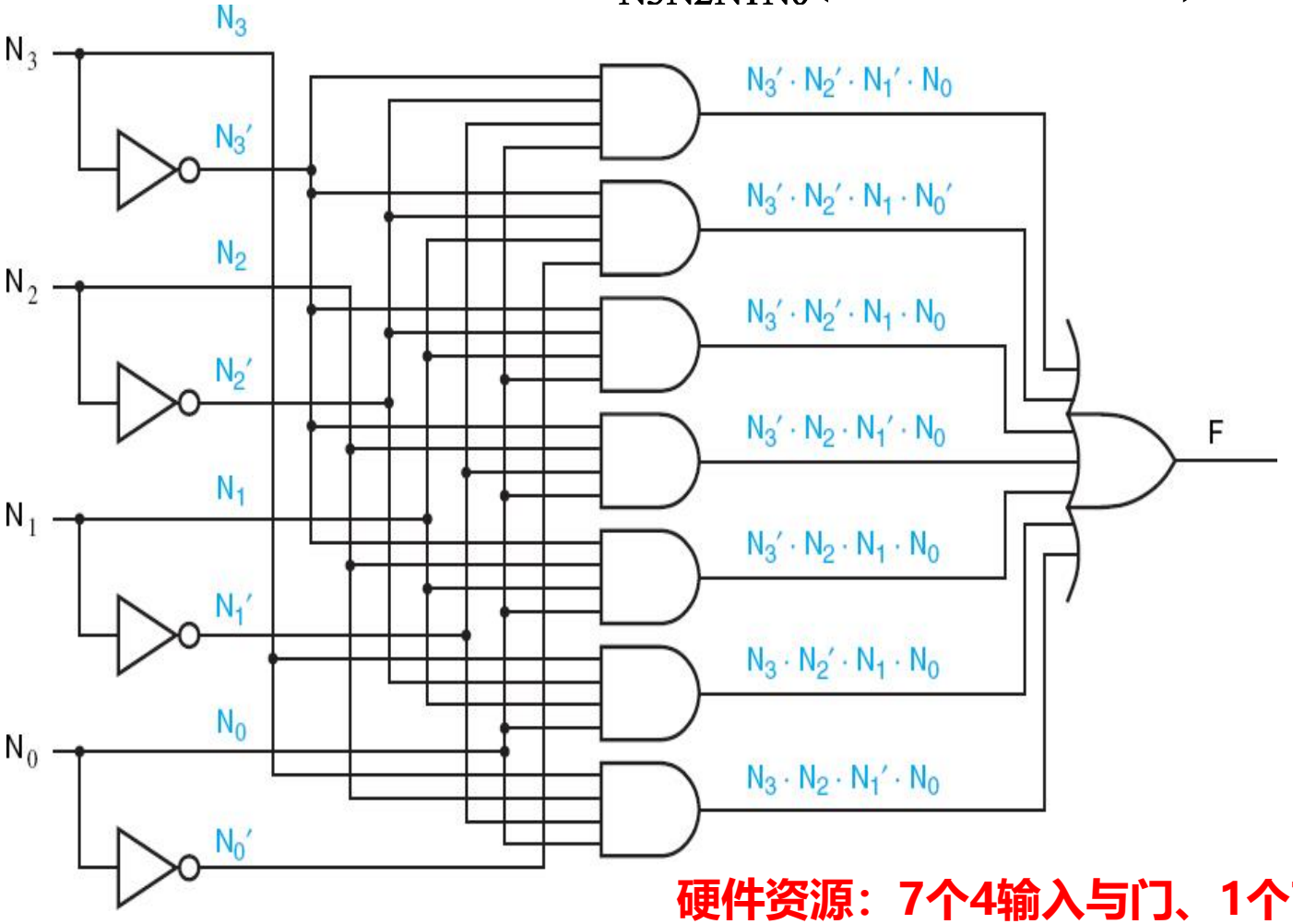


1.4 组合逻辑电路设计

例1：素数检测器的设计

- 4-bit input, $N_3N_2N_1N_0$

写出最小项表达式 $F = \Sigma_{N3N2N1N0}(1,2,3,5,7,11,13)$



列出真值表

row	N_3	N_2	N_1	N_0	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

硬件资源：7个4输入与门、1个7输入或门

1.4 组合逻辑电路设计

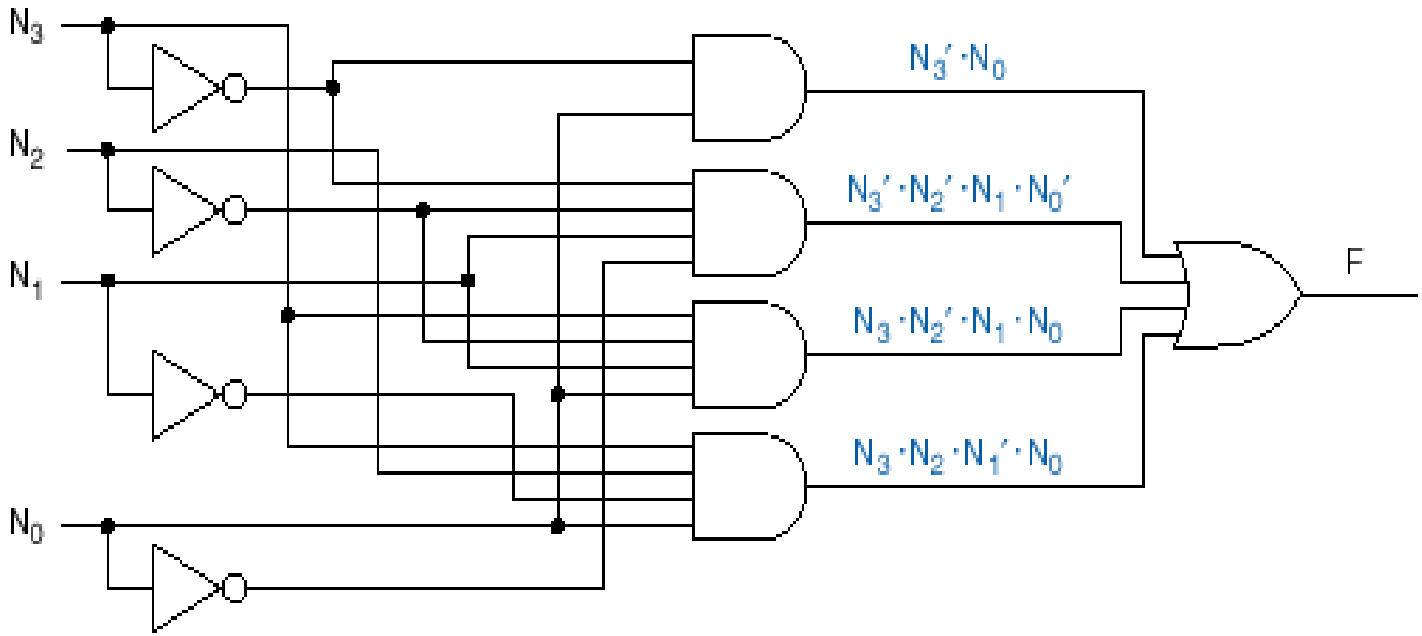
利用布尔代数化简，以减少逻辑门数和输入端数　　$X \cdot Y + X \cdot Y' = X$

$$F = \Sigma_{N3N2N1N0}(1,2,3,5,7,11,13)$$

$$= N_3' \cdot N_2' N_1' N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + \ldots$$

$$= N_3' N_2' \cdot N_0 + N_3' \cdot N_2 \cdot N_0 + \ldots$$

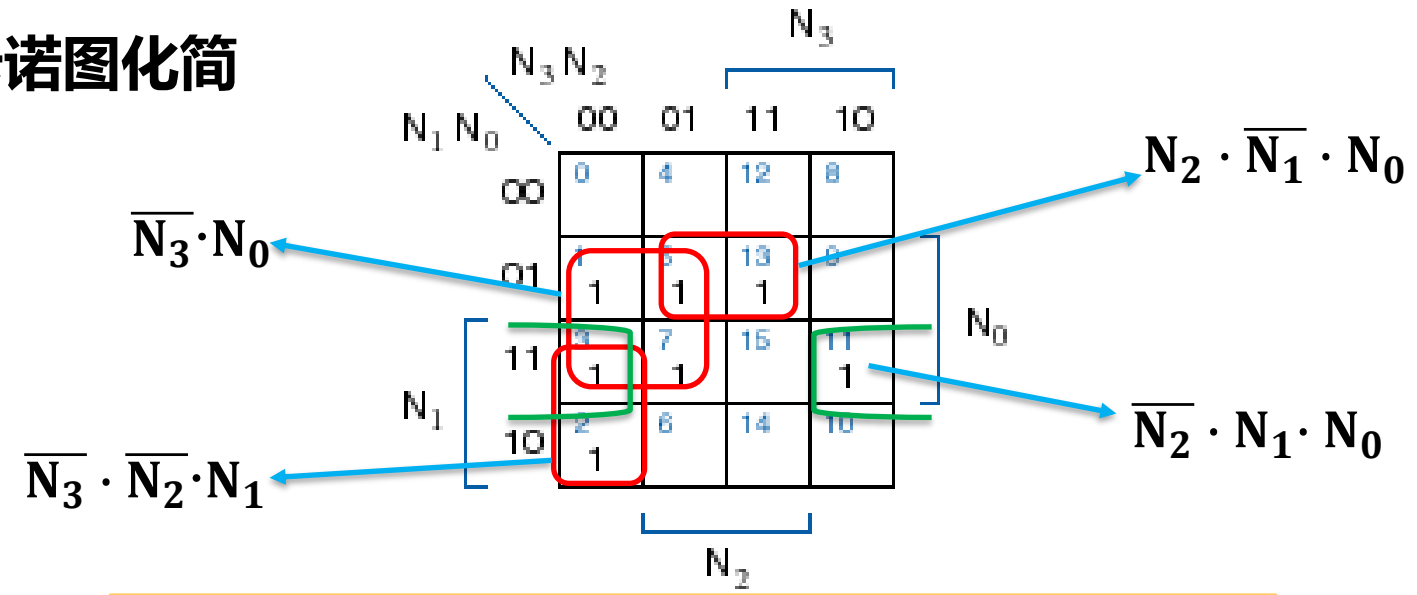
$$= N_3' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0' + N_3 \cdot N_2' \cdot N_1 \cdot N_0 + N_3 \cdot N_2 \cdot N_1' \cdot N_0$$



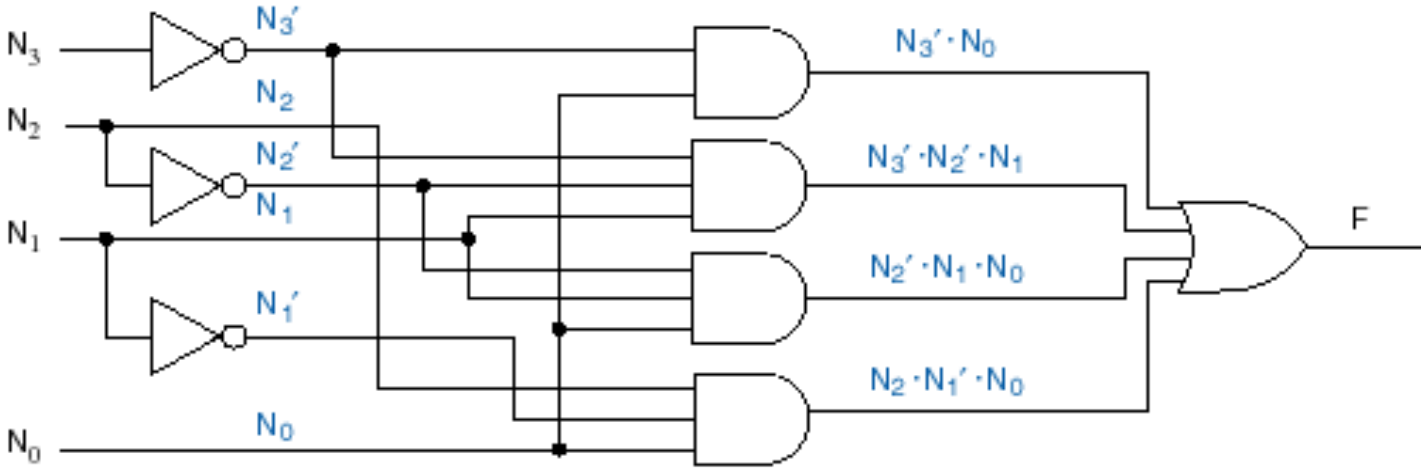
减少：3个与门和17个输入端

1.4 组合逻辑电路设计

按卡诺图化简



$$F=N_3'\cdot N_0+N_3'\cdot N_2'\cdot N_1+N_2'\cdot N_1\cdot N_0+N_2\cdot N_1'\cdot N_0$$



减少：3个与门和20个输入端

1.4 组合逻辑电路设计

例2：设计一个监视交通信号灯工作状态的逻辑电路

1、进行逻辑抽象

输入变量：红R 黄Y 绿G 三盏灯的状态

灯亮为1，不亮为0

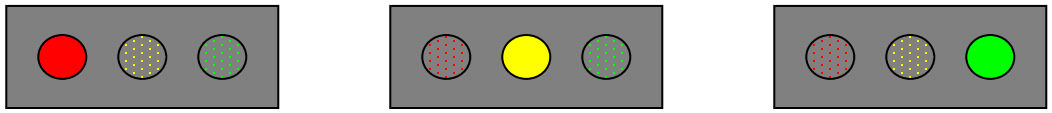
输出变量：故障信号F

正常工作为0，发生故障为1

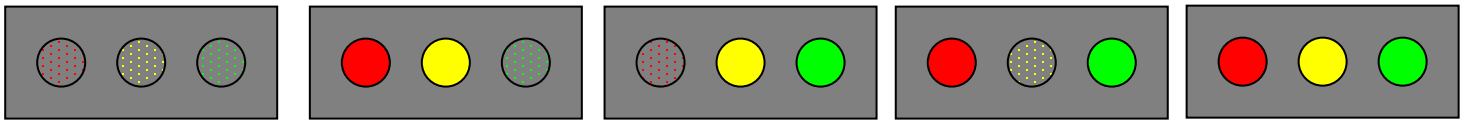
真值表

R	Y	G	F
0	0	0	1
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	1

正常工作状态



故障状态



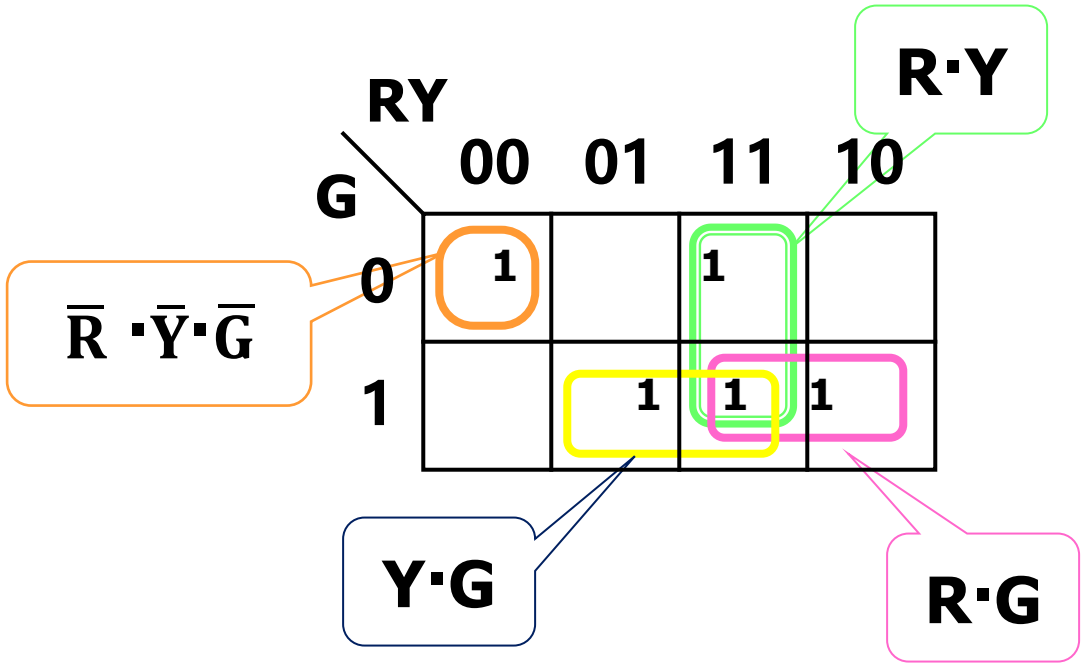
逻辑抽象结果

真值表

R	Y	G	F
0	0	0	1
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	1

2、逻辑化简

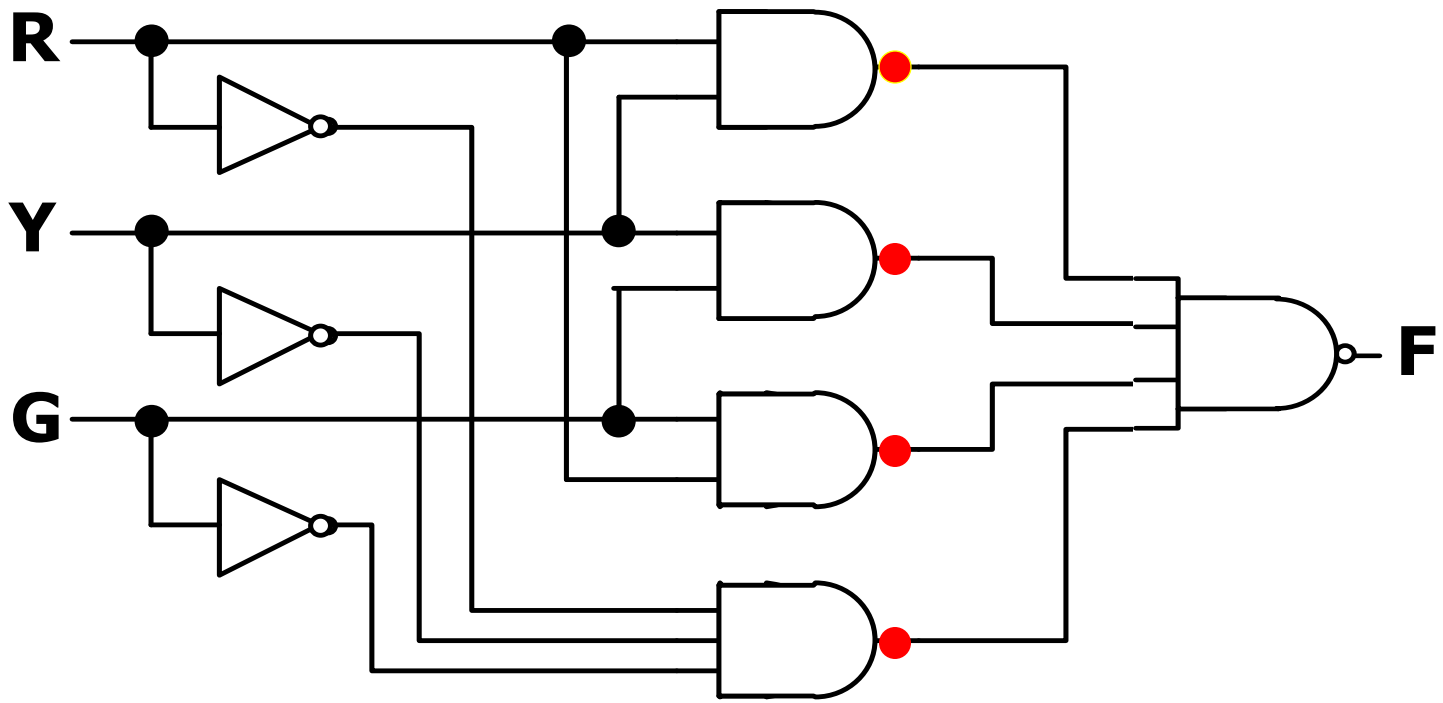
写出逻辑函数式并化简



$$F=\bar{R}\cdot\bar{Y}\cdot\bar{G}+R\cdot Y+R\cdot G+Y\cdot G$$

3、电路设计

$$F = \bar{R} \cdot \bar{Y} \cdot \bar{G} + R \cdot Y + R \cdot G + Y \cdot G$$



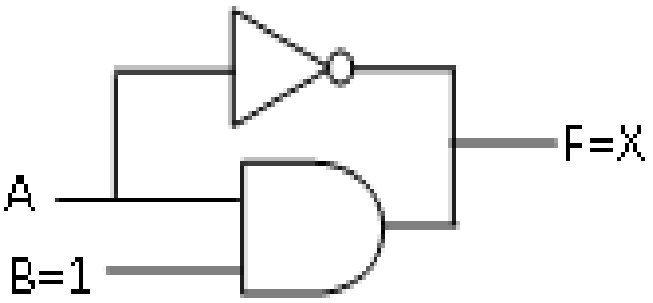
用与非门实现

1.5 无关项、非法值和高阻态

- **无关项**
  - 某些输入组合对应的输出值可以任意值，某些输入组合不应该出现在真值表中
  - 这些输入组合对应的输出值在化简时可标识为d，可以取值0或1，具体数值根据化简的需要而定
  - 可灵活应用以简化电路从而降低成本，但也更易受干扰

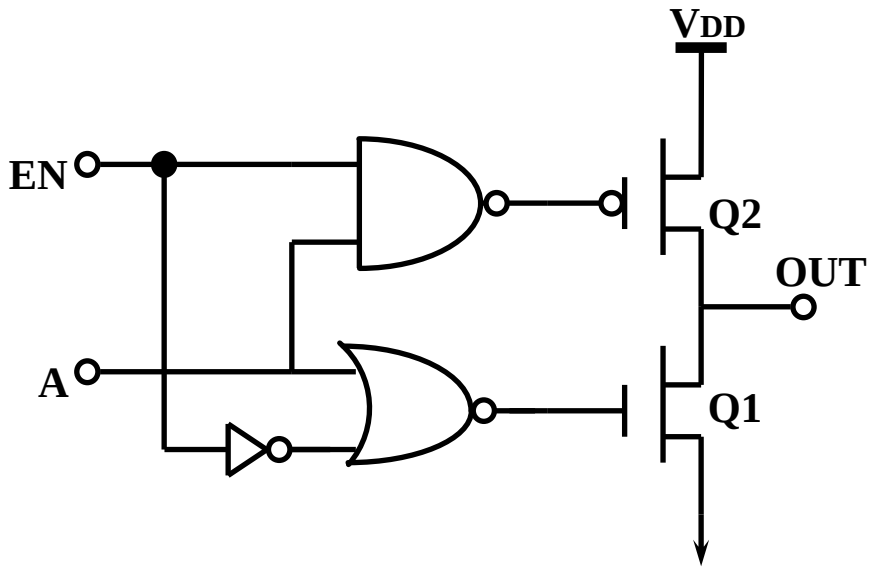
    例如：8421 BCD码输入时，大于1001的编码为无关项
- **非法值**
  - 信号值不能被有效识别为高电平或低电平，处于不确定状态。

    例如：下图中的信号X

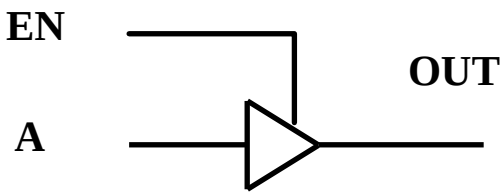


1.5 无关项、非法值和高阻态

- **三态门（three-state gate）是一种重要的总线接口电路，也称三态缓冲器，其输出既可以是通常的逻辑值1 或 0，又可以是高阻态**
- **高阻态Hi-Z：输出处于非正常逻辑态的第三种电气态，好像和电路断开一样**
- **三态门有一个额外的输出使能控制端EN**



EN	A	Q1	Q2	OUT
L	L	off	off	Hi-Z
L	H	off	off	Hi-Z
H	L	on	off	L
H	H	off	on	H



三态门用途：可用于连接总线，多个三态输出连在一起等

第二讲 典型组合逻辑部件

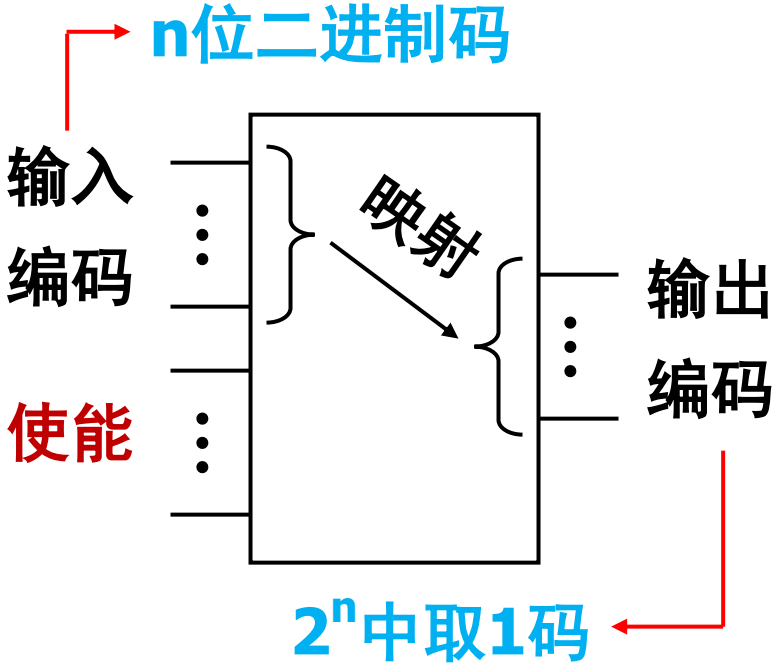
- 译码器和编码器
- 多路选择器和多路分配器
- 半加器和全加器

2 典型组合逻辑部件

- 复杂数字系统通常采用层次化、模块化方式构建
  - 由基本组合逻辑部件和时序逻辑部件相互连接构建
- 基本的组合逻辑部件的功能有：
  - 译码与编码、选择与分配、比较、运算、缓存并传送等
- 组合电路功能的实现方式：
  - 采用分立SSI门电路来实现（逻辑函数）
  - 使用只读存储器ROM来实现，效率较低（真值表）
  - 用提供单一功能的逻辑部件来实现，如译码器、编码器、加法器、比较器等

2.1 译码器和编码器

- **译码/编码功能：**
  - **信号与信号编码之间的转换，如：**n位信号编码←→2^n位信号
- **译码器（decoder）：一种多输入、多输出的组合电路。**
  - **编码→信号的转换，输入端数比输出端数少**
  - **通常输出采用2^n中取1码(单热点,one-hot)编码表征信号**
  - **可以通过使能端EN来控制电路实现映射功能**



- **n-2^n译码器**
  - **输入：n位二进制编码**
  - **输出：2^n中取1码**
- **例如：**
  - **2-4译码器**
  - **3-8译码器**
  - **4-16译码器**

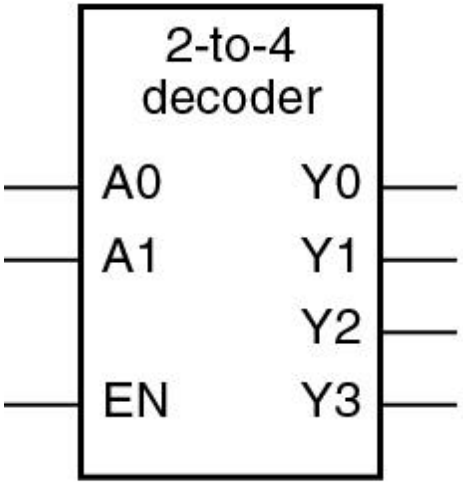
2.1 译码器和编码器

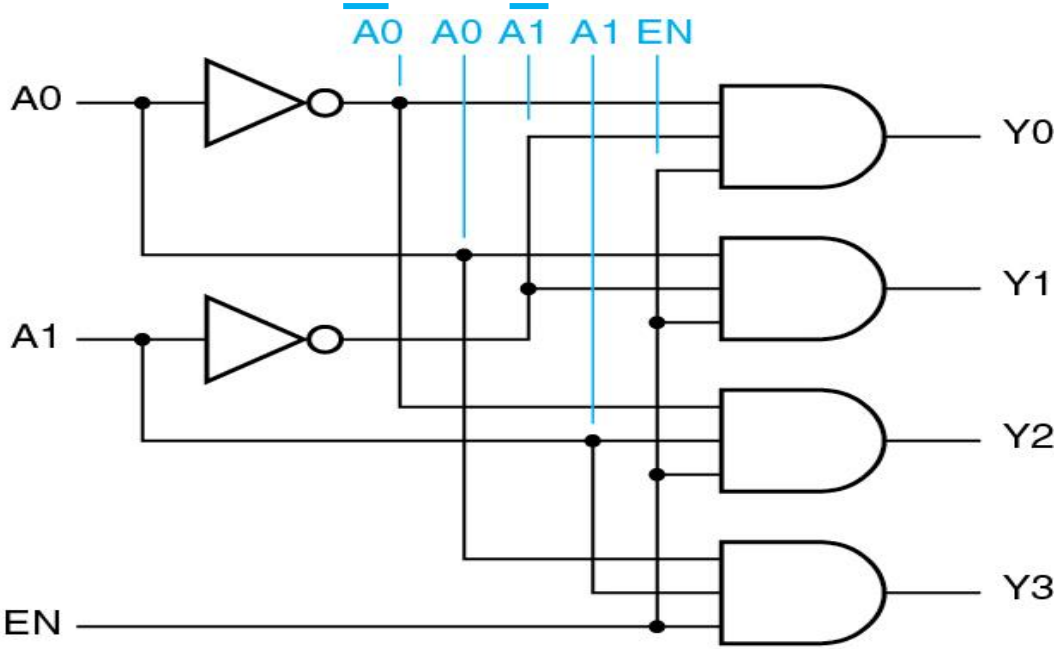
x表示该输入don't care

例1：2-4译码器74X139

- 输出端高电平有效，表示选中输出，对应输入信号的最小项
- 通过使能控制端EN（Enable Control）禁止或实现相应功能
- EN=0时，输出为全0
  - 消除干扰、功能扩展

Inputs			Outputs			
EN	A1	A0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



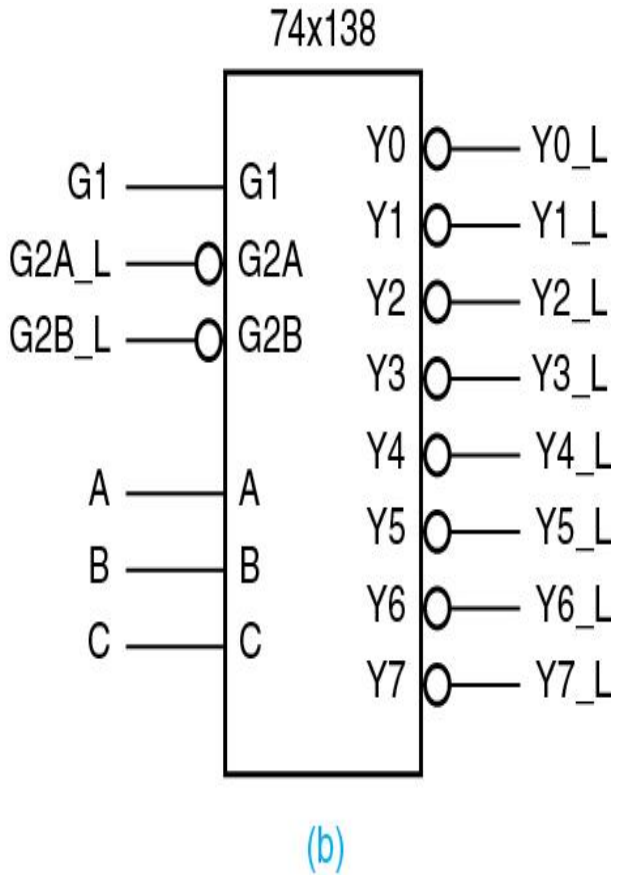


如，$Y2=EN\cdot A1\cdot\overline{A0}=EN\cdot m_2$

2.1 译码器和编码器

例2：3-8译码器 74X138

- 输出端低电平有效，对应输入信号的最大项
- 有3个使能控制端



2.1 译码器和编码器

NjUAI 南京大学 人工智能学院

74X138的功能表

Inputs						Outputs							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

M_6

$EN=\overline{G1}+G2A\_L+G2B\_L$

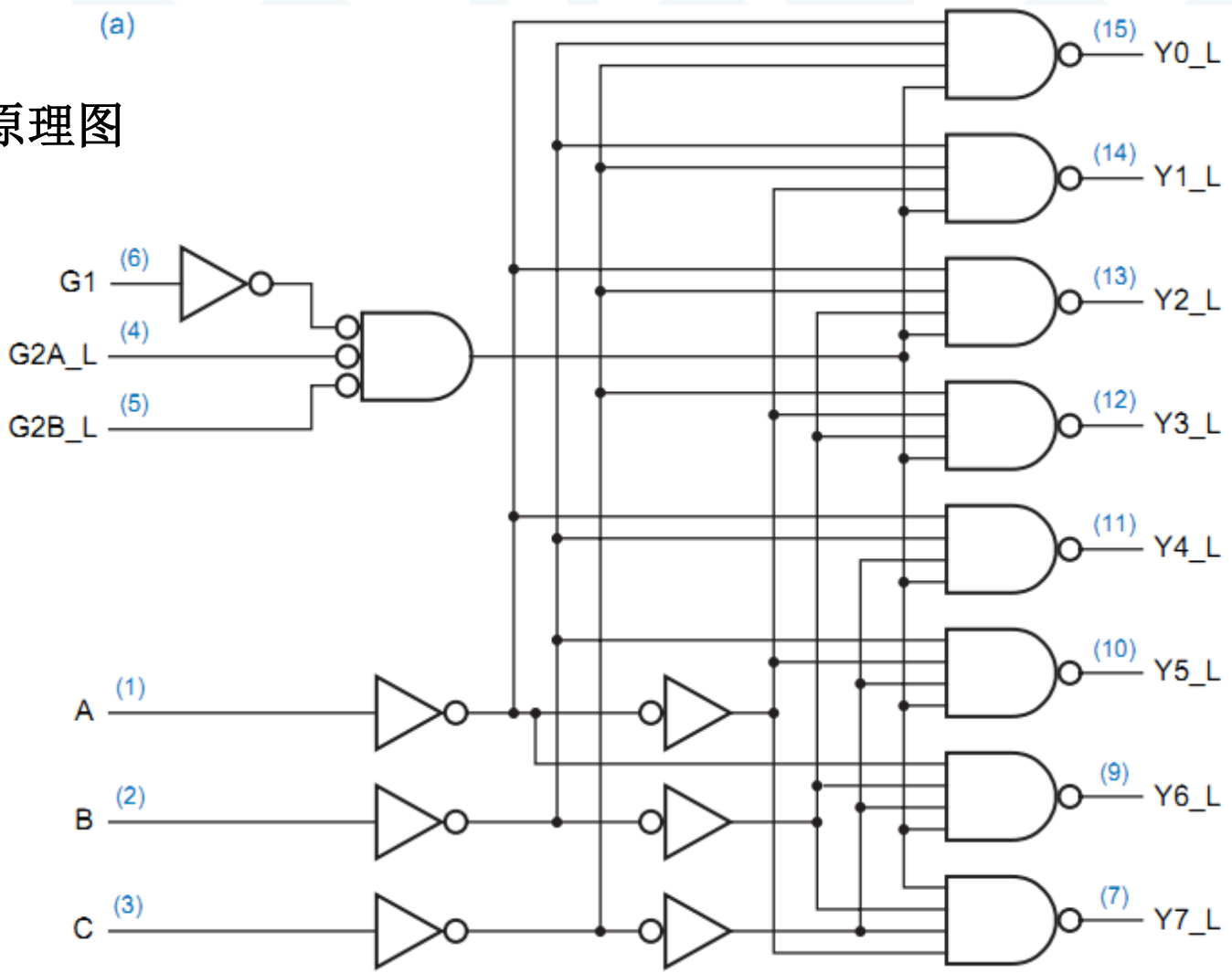
例：$Y6\_L=EN+\bar{C}+\bar{B}+A$

2.1 译码器和编码器

74X138逻辑原理图

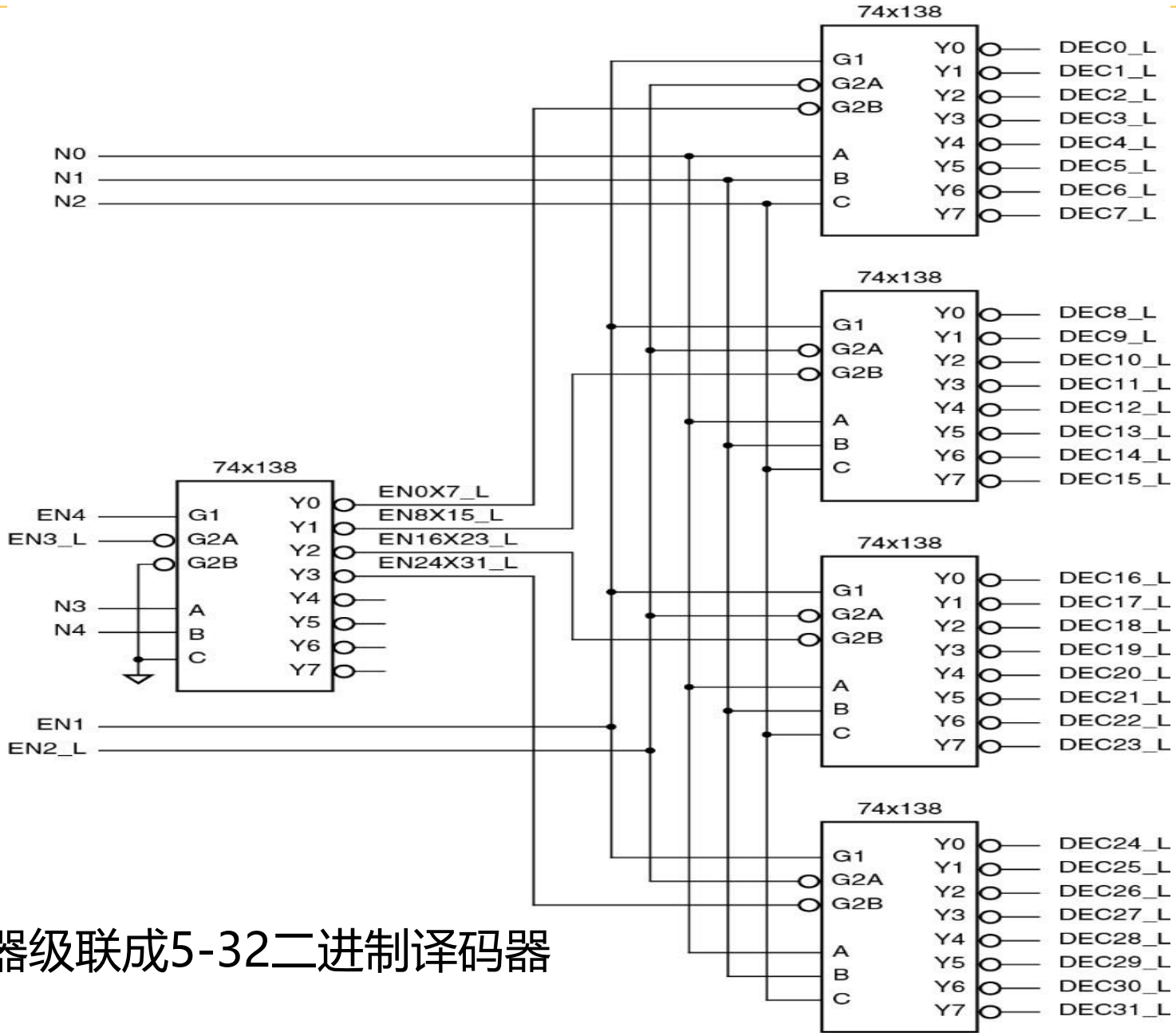
使能控制端

输入端



思考：当输入端同时发生变化时，对输出端的影响。

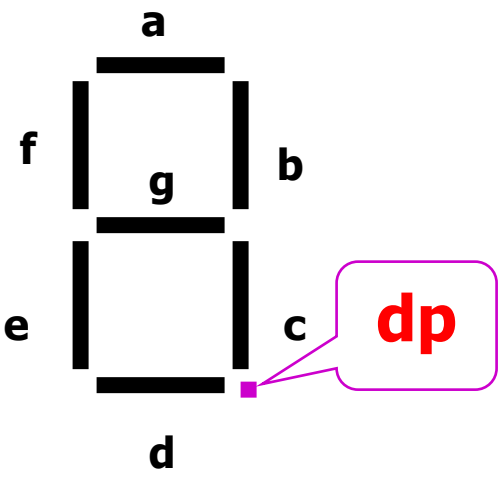
2.1 译码器和编码器

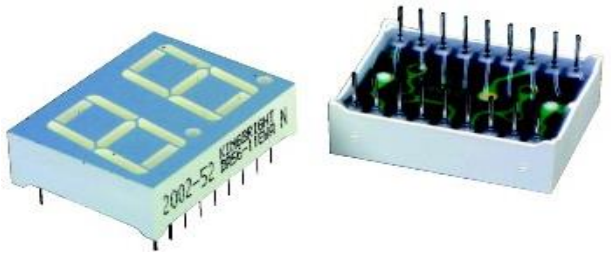


3-8译码器级联成5-32二进制译码器

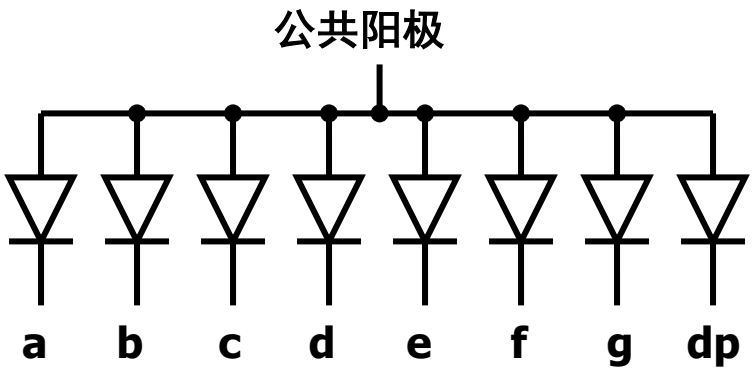
2.1 译码器和编码器

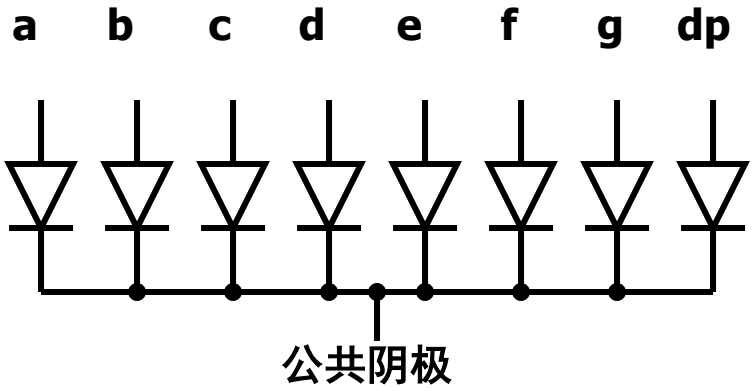
例2：七段显示译码器





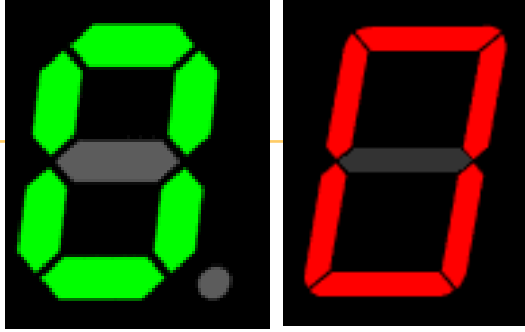
每段是一个LED (发光二极管)，通过控制其亮和灭，可得到数字或字母等符号的形状





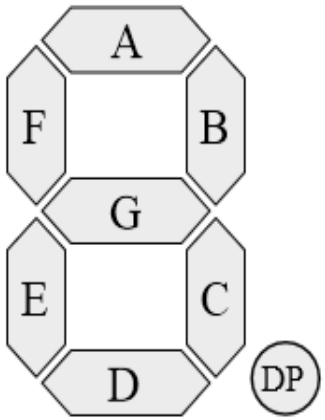
共阳极，输入为低电平二极管导通；
共阴极，输入为高电平二极管导通。

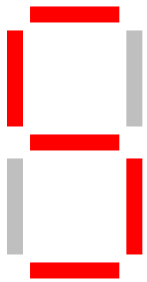
2.1 译码器和编码器



- 输入信号：4位二进制编码
- 输出：七段码（的驱动信号）a ~ g

  假设共阴极，即 1-亮；0-灭





1011011

1111110

0110011

2.1 译码器和编码器

- 七段数字显示译码器真值表

	A3	A2	A1	A0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
A	1	0	1	0	1	1	1	0	1	1	1
b	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
d	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

2.1 译码器和编码器

◆根据显示需要，考虑是否使用A~F输入信号

以下是输出信号a的卡诺图

A~F输入信号作为无关项

A3A2 \ A1A0	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	d	d	d	d
10	1	1	d	d

$a=A3+A1+A2A0+\overline{A2}\ \overline{A0}$

$a=A3+A1+A2\odot A0$

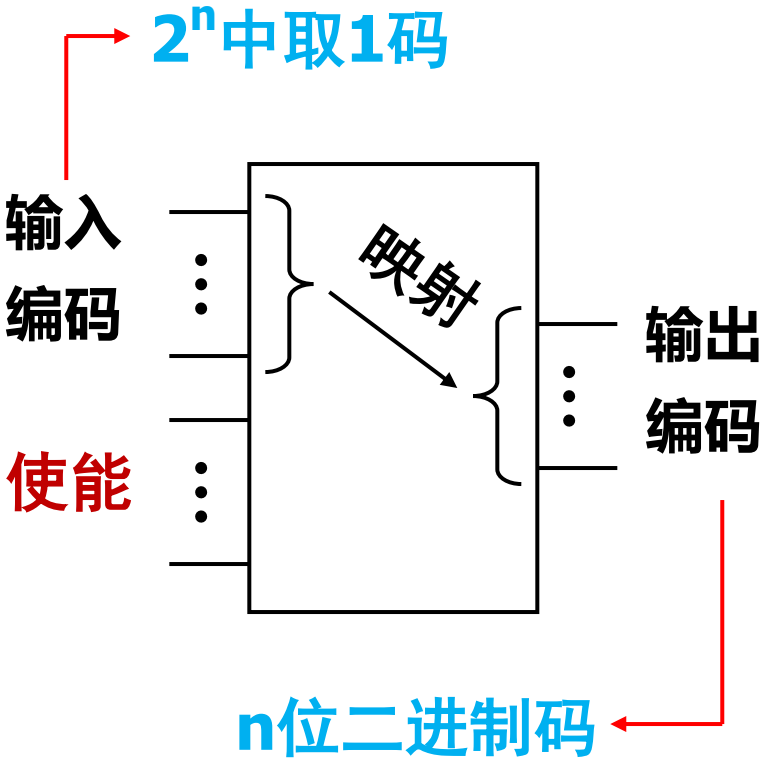
A~F输入信号作为有效项

A3A2 \ A1A0	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	1	1	1
10	1	1	0	1

$a=A2A0+\overline{A2}\ \overline{A0}+A1\cdot\overline{A0}+\overline{A3}\cdot A1+A3\cdot\overline{A2}\cdot\overline{A0}$

2.1 译码器和编码器

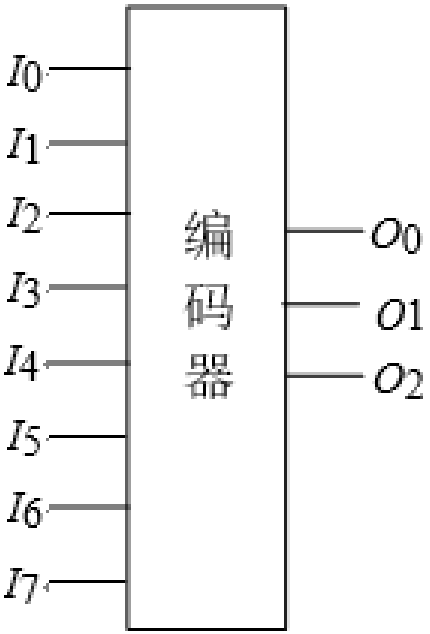
- 编码器encoder：译码器的逆向电路
  即输出是输入信号的二进制编码



- 最常见是2^n-n编码器，也称为二进制编码器。
  - 2^n个输入端
  - n个输出端
- 分类：
  - 互斥(唯一输入)编码器
  - 非互斥编码器
  - 优先级编码器

2.1 译码器和编码器

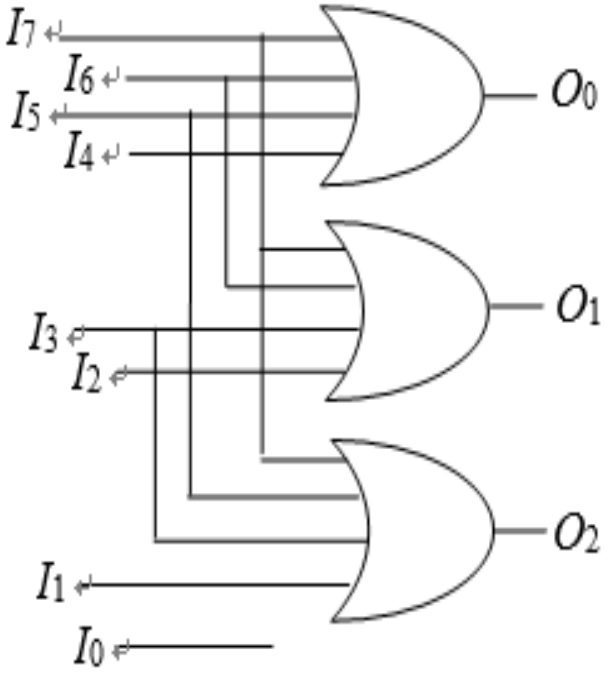
- 3 位二进制编码器（8-3 编码器）
  - 输入I_0~I_7是一组互斥变量，每次只有一个输入端I_i为1，其余都为0，输出为i的二进制编码。



a) 编码器符号

	O_0	O_1	O_2
I_0	0	0	0
I_1	0	0	1
I_2	0	1	0
I_3	0	1	1
I_4	1	0	0
I_5	1	0	1
I_6	1	1	0
I_7	1	1	1

b) 编码器真值表



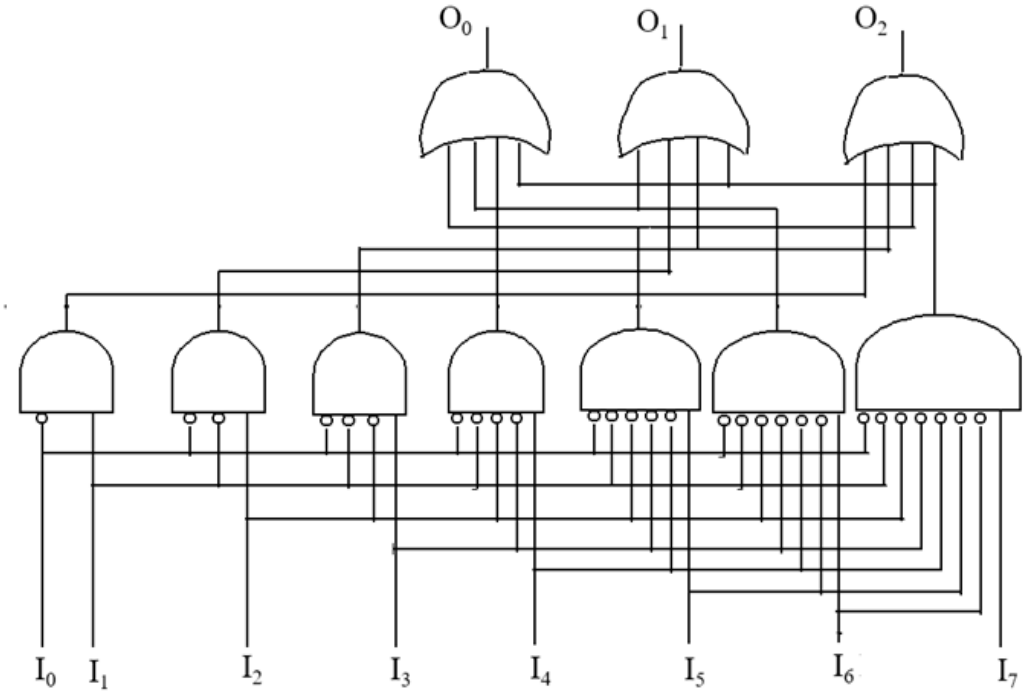
c) 编码器电路图

2.1 译码器和编码器

- 3 位优先权编码器
  - 多个输入可同时为1，但只对优先级最高的输入进行编码输出
  - 假定优先级顺序为I0 > I1 > I2 > I3 > I4 > I5 > I6 > I7，则：

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	O_0	O_1	O_2
1	x	x	x	x	x	x	x	0	0	0
0	1	x	x	x	x	x	x	0	0	1
0	0	1	x	x	x	x	x	0	1	0
0	0	0	1	x	x	x	x	0	1	1
0	0	0	0	1	x	x	x	1	0	0
0	0	0	0	0	1	x	x	1	0	1
0	0	0	0	0	0	1	x	1	1	0
0	0	0	0	0	0	0	1	1	1	1

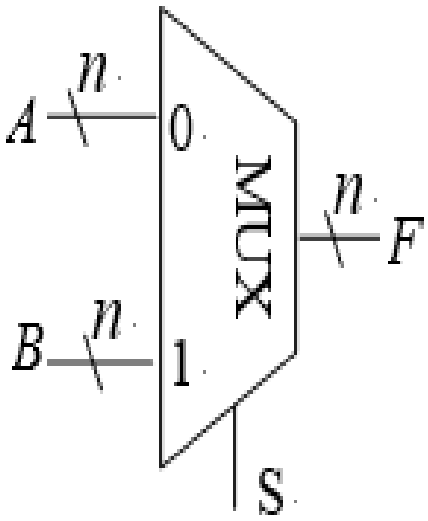
优先权编码器
的功能描述



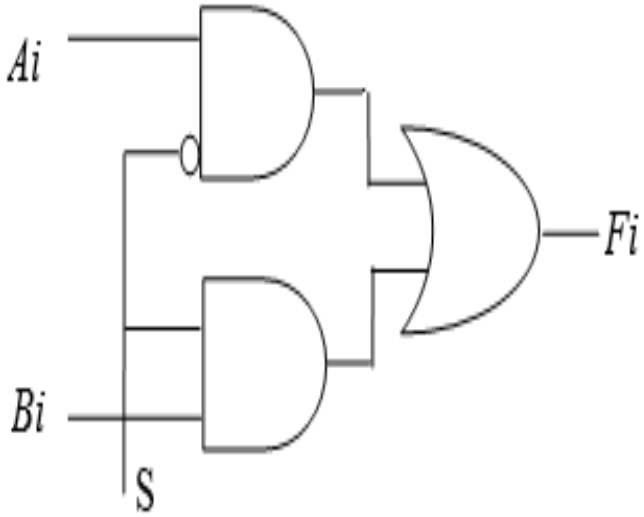
优先权编码器逻辑电路图

2.2 多路选择器和多路分配器

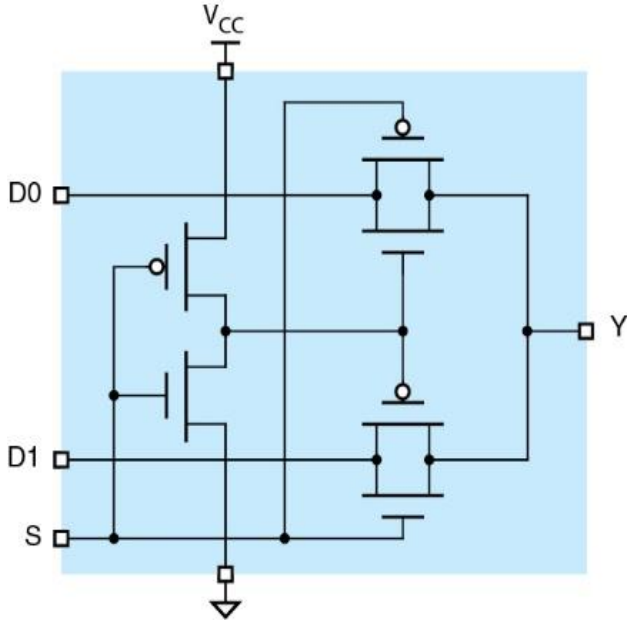
- 2-路选择器有两个输入端和一个输出端，有一个控制端，用于控制选择哪一路输出
- 在计算机中，2-路选择器的每个输入、输出端通常都有n位



2-路选择器符号



一位2-路选择器逻辑电路

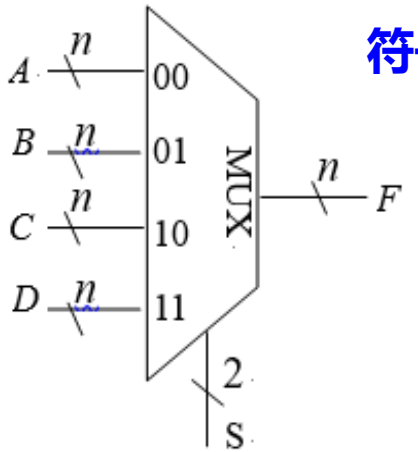


传输门实现2-路选择器

2.2 多路选择器和多路分配器

- 4-路选择器

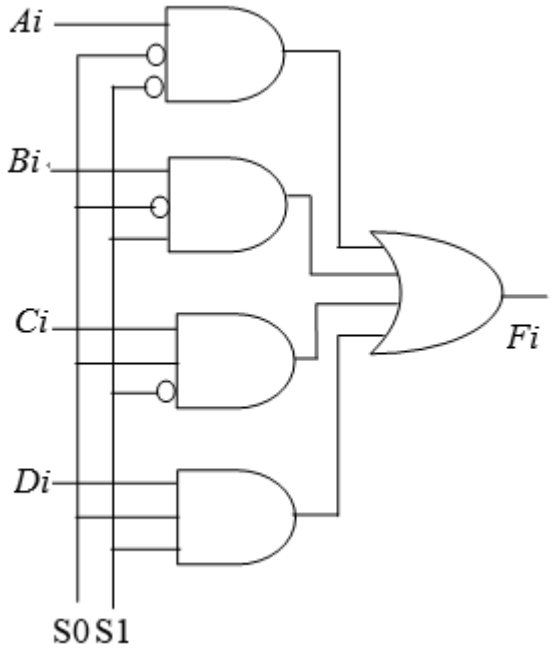
符号



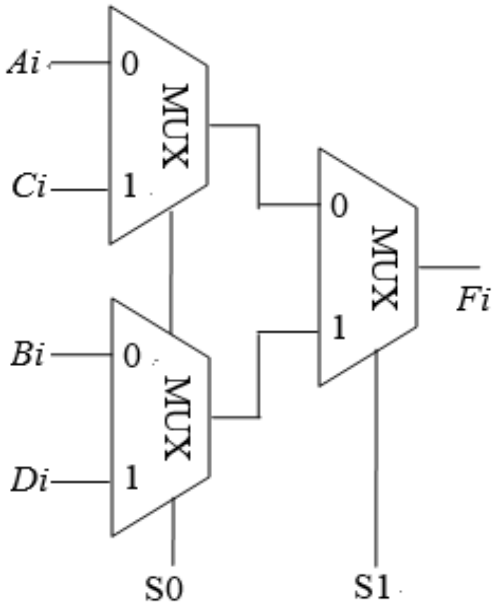
真值表

S0	S1	F
0	0	A
0	1	B
1	0	C
1	1	D

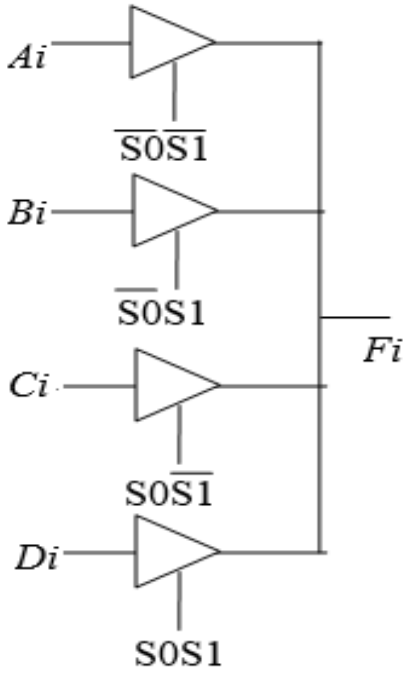
- 一位4-路选择器的实现



两级门电路



多层次级联



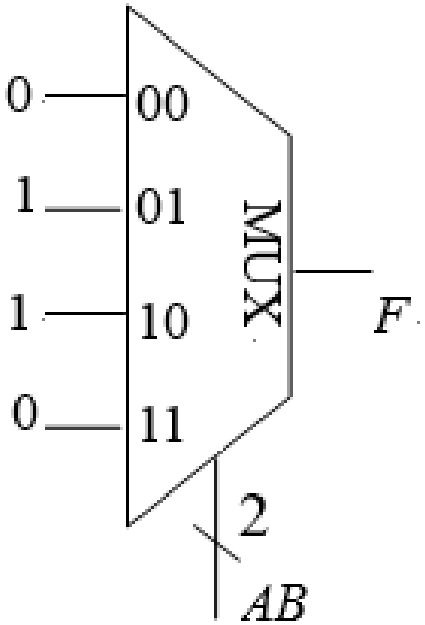
三态门电路

2.2 多路选择器和多路分配器

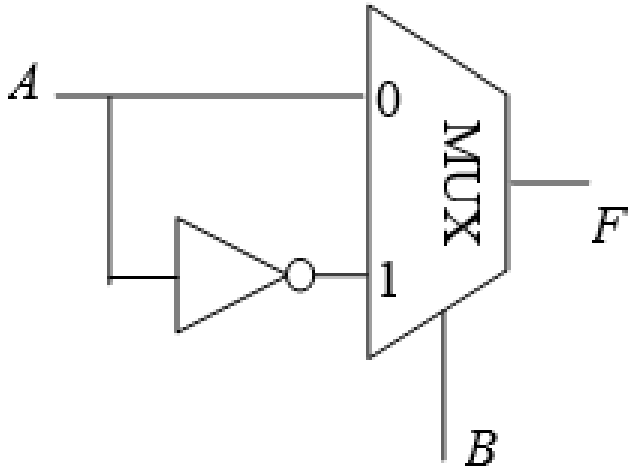
- 可以基于多路选择器实现组合逻辑电路的功能
- 例1：基于多路选择器实现某组合逻辑电路的功能（可用如下真值表描述）

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

真值表



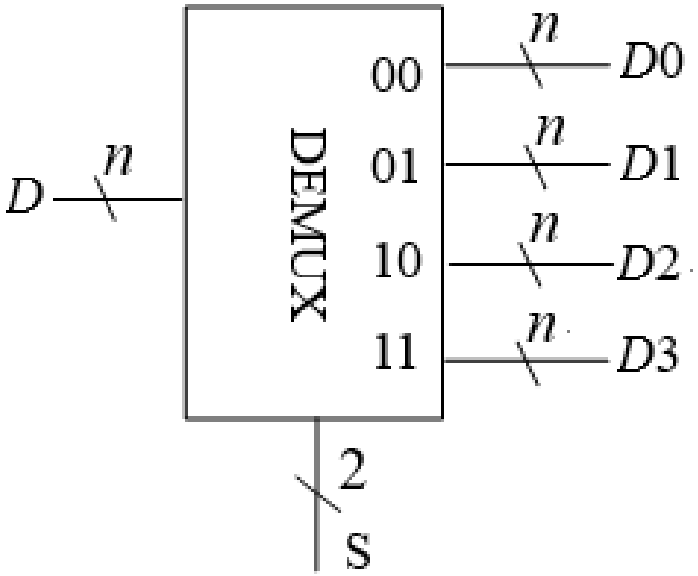
用一个4-路选择器实现



用一个2-路选择器和一个非门实现

2.2 多路选择器和多路分配器

- 多路分配器（demultiplexer）：把唯一的输入信号发送到多个输出端中的一个。从哪一个输出端送出输入信号，取决于控制端。简写为DMUX或DEMUX
- 4-路分配器的符号和真值表



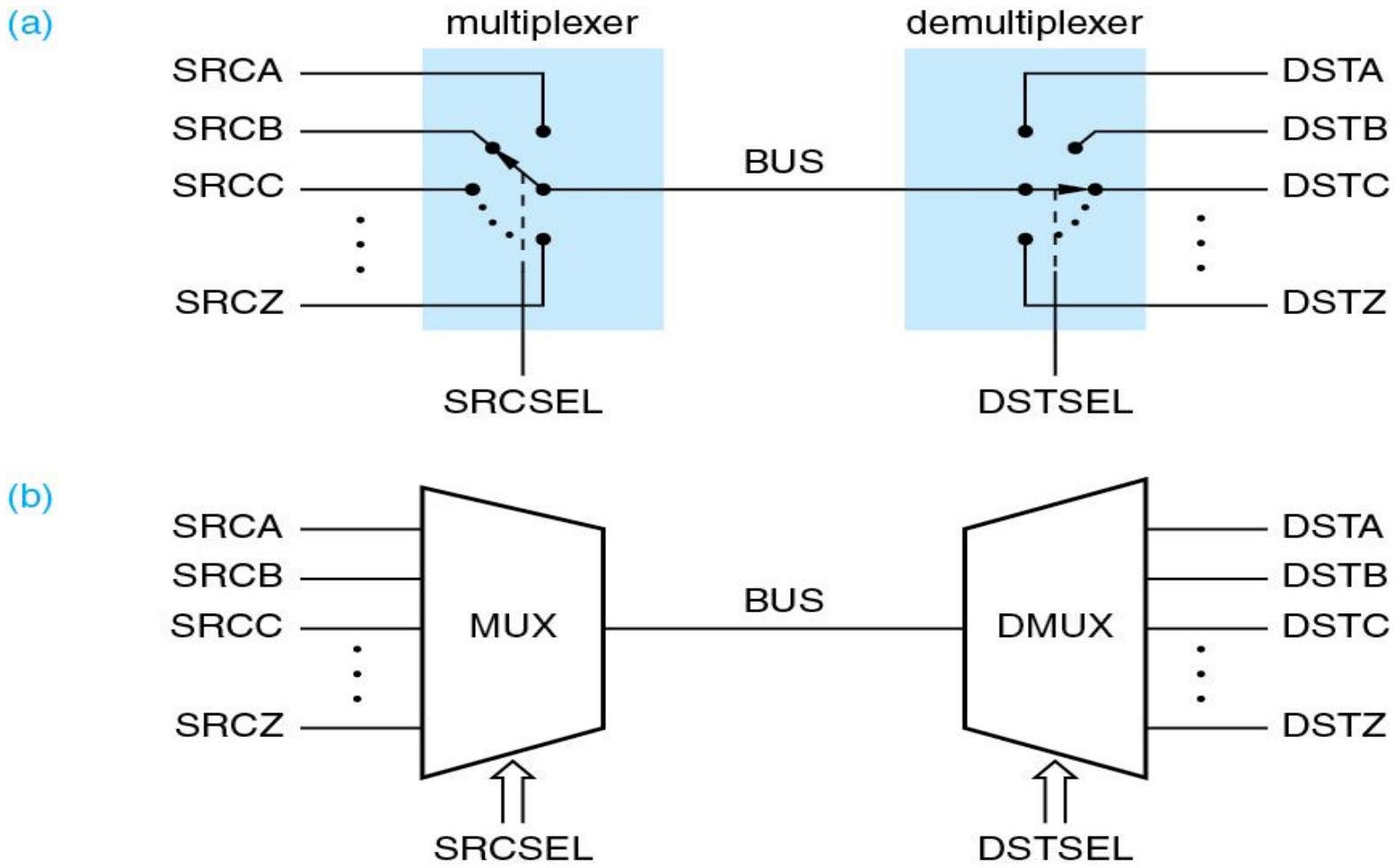
四路分配器的符号

S0	S1	D0	D1	D2	D3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

四路分配器真值表

2.2 多路选择器和多路分配器

- 多路分配器常与多路选择器联用，以实现多通道数据的分时传送。



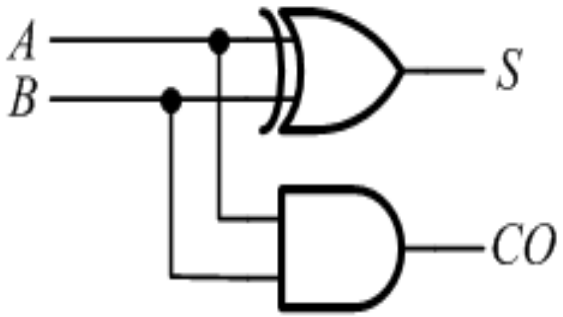
2.3 半加器和全加器

- 半加器（Half Adder，简称HA）：仅考虑加数和被加数，不考虑低位来的进位

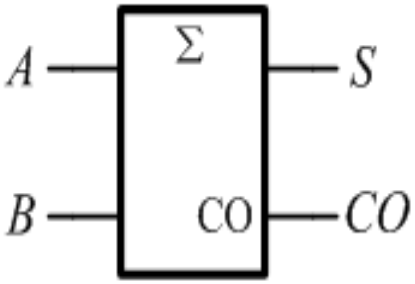
输入		输出	
被加	加数	和数	进位
A	*B*	*S*	*CO*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S=\overline{A}B+A\overline{B}=A\oplus B$$

$$CO=A\cdot B$$



电路图



逻辑符号

2.3 半加器和全加器

- 全加器（Full Adder，简称FA）：输入为加数、被加数和低位进位Cin，输出为和F、进位Cout

A	B	Cin	F	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

真值表

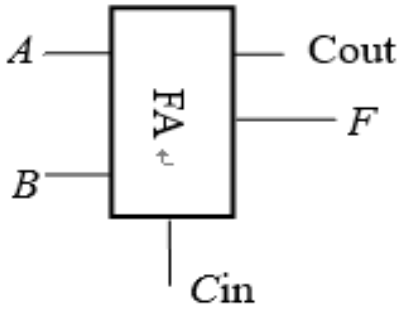
$$F=\overline{A}\cdot\overline{B}\cdot Cin+\overline{A}\cdot B\cdot\overline{Cin}+A\cdot\overline{B}\cdot\overline{Cin}+A\cdot B\cdot Cin$$

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

化简后：

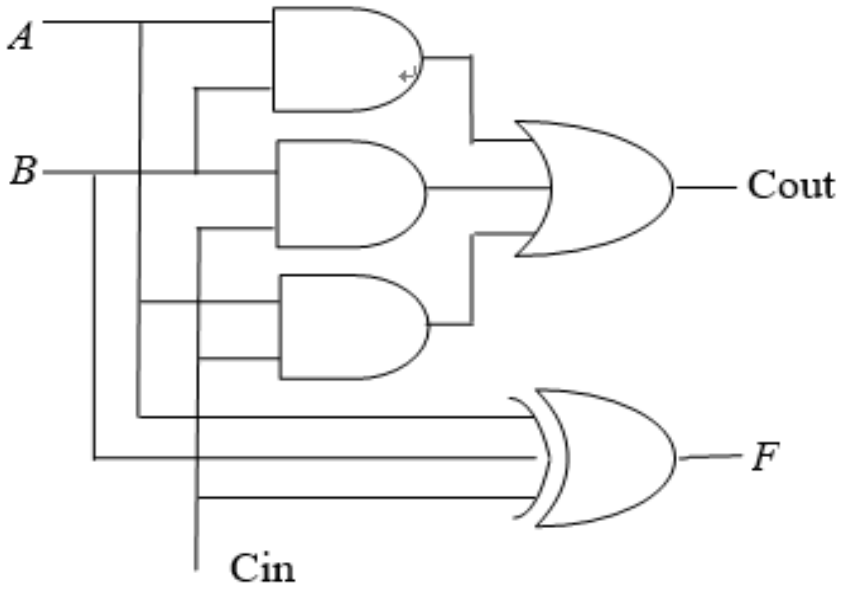
$$F=A\oplus B\oplus Cin$$

$$Cout=A\cdot B+A\cdot Cin+B\cdot Cin$$



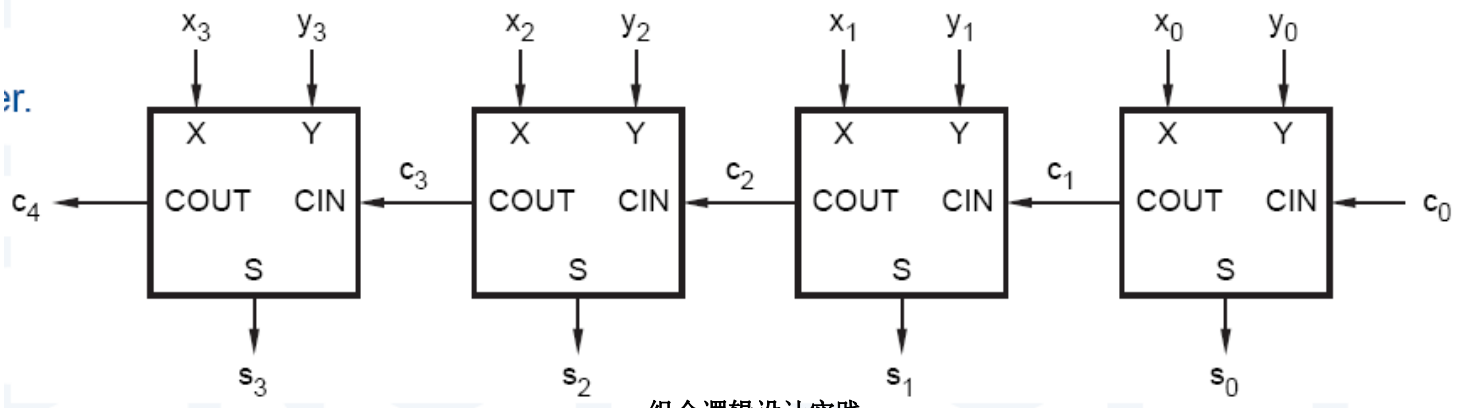
逻辑符号

全加器逻辑电路图



补1： 加法器

- 串行进位加器：行波进位加法器 ripper adder
  - 规格：
    - 两个二进制字，每个n位，相加。
  - 方法：
    - n个全加器的级联，属于迭代电路。
  - 延迟： $t_{ADD} = t_{XYCout} + (n-2) \cdot t_{CinCout} + t_{CinS}$
  - 特点：简单、速度慢



第三讲 组合逻辑部件时序分析

- 传输延迟和最小延迟
- 竞争冒险

3 组合逻辑电路时序分析

- 信号通过连线和电路元件时会有一定时间的延迟（Delay）
- 电路的延迟取决于电路内部的设计及外部特性，影响因素包括但不限于：
  - 连线的长短、元件的数量
  - 电路制造工艺、工作电压
  - 环境噪声、温度等外在条件
  - 高低电平的转换过渡时间

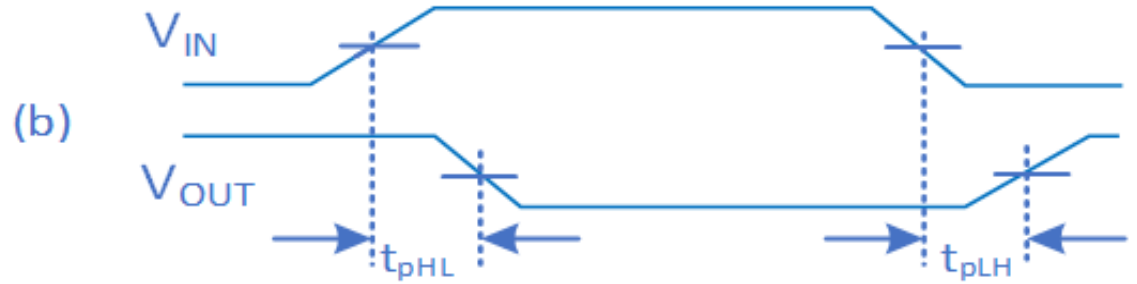
因此，任何组合逻辑电路从输入信号的改变，到随之引起的输出信号的改变，都有一定时间的延迟

3.1 传输延迟和最小延迟

- 通常用时序图反映电路的延迟
  - 下降沿延迟t_{pHL}：输入变化引起相应输出从高到低变化的时间
  - 上升沿延迟t_{pLH}：输入变化引起相应输出从低到高变化的时间
- 通常取信号转换时间中间点来测量延迟时间

忽略上升时间和下降时间

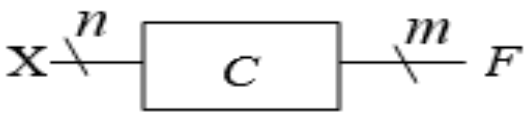
在转换中间点测量



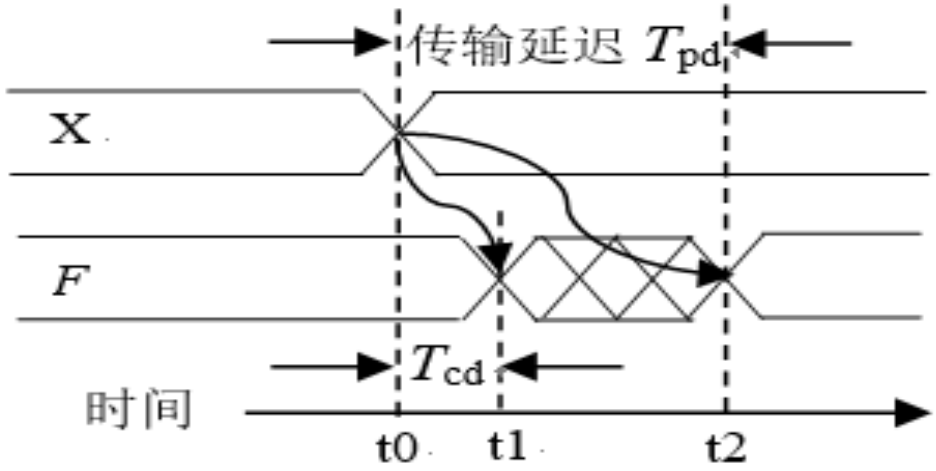
反相器电路的时序图反映了其电路延迟情况

3.1 传输延迟和最小延迟

- 逻辑门电路具有最大延迟（传播延迟）和最小延迟时序特征
- 组合逻辑电路的时序特征主要包括传输延迟（propagation delay）和最小延迟（contamination delay）
  - 传输延迟Tpd：从输入端的变化开始到所有输出端得到最终稳定的信号所需的最长时间
  - 最小延迟Tcd：从输入端的变化开始到任何一个输出开始发生改变所需的最短时间



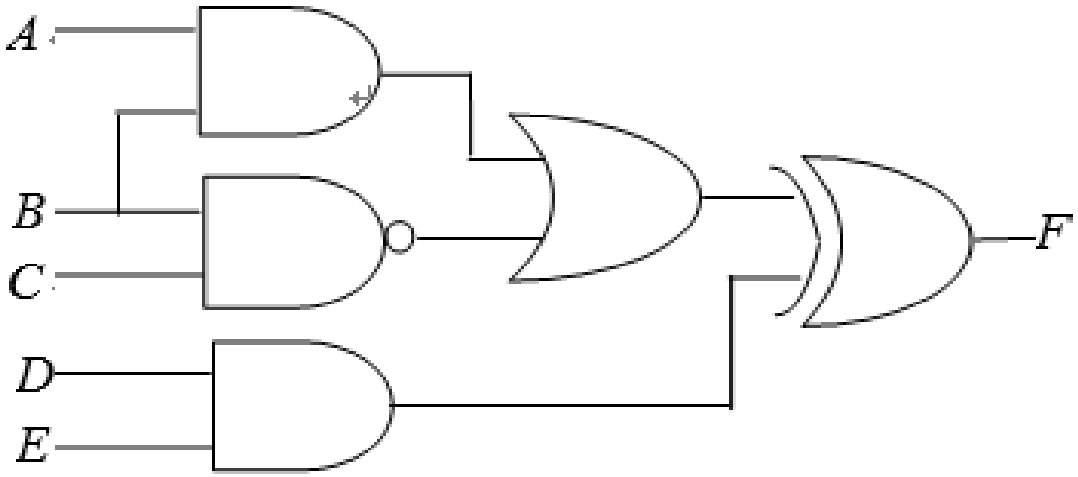
组合逻辑电路C的传输延迟 T_{pd} 和最小延迟 T_{cd}



3.1 传输延迟和最小延迟

- **关键路径：一个组合逻辑电路在输入和输出之间经过的最长路径**
  - **传输延迟就是关键路径上所有元件的传输延迟之和**
  - **最小延迟就是最短路径上所有元件的最小延迟之和**

例： 假定所有逻辑门电路的传输延迟和最小延迟分别为90ps和60ps，计算下图中组合逻辑电路的传输延迟和最小延迟。

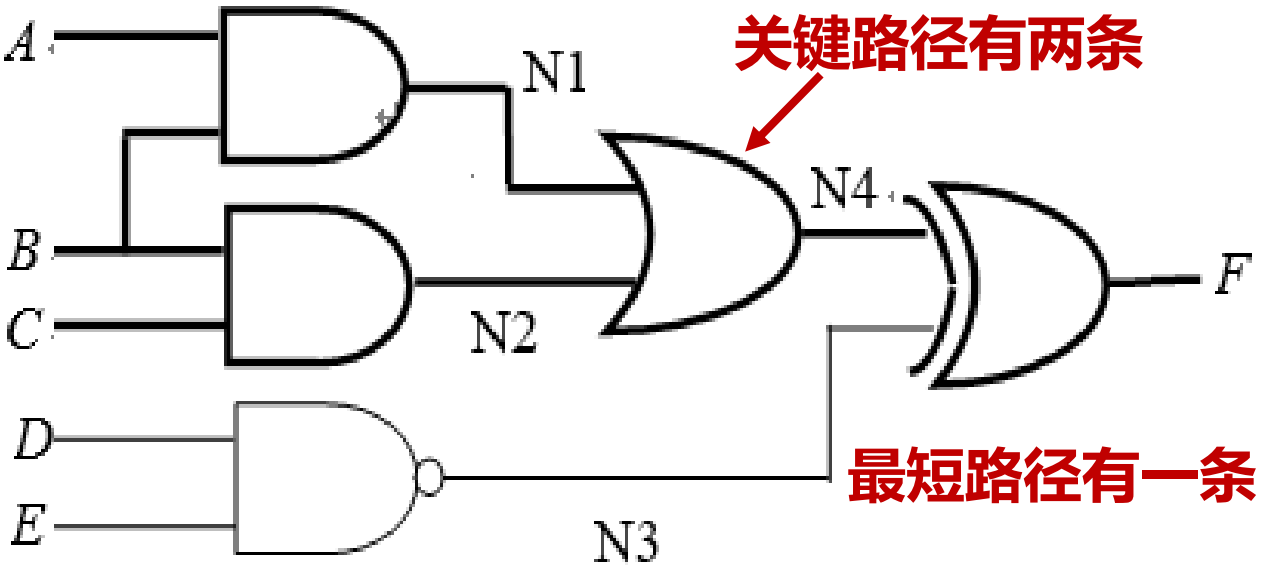


关键路径由哪些连线和逻辑门组成？最短路径呢？

3.1 传输延迟和最小延迟

- **关键路径：一个组合逻辑电路在输入和输出之间经过的最长路径**
  - **传输延迟就是关键路径上所有元件的传输延迟之和**
  - **最小延迟就是最短路径上所有元件的最小延迟之和**

例： 假定所有逻辑门电路的传输延迟和最小延迟分别为90ps和60ps，计算下图中组合逻辑电路的传输延迟和最小延迟。

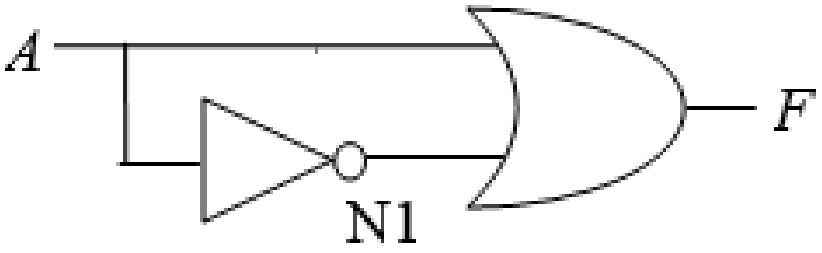


传输延迟 T_{pd} 为3级门传播延迟之和，即90psx3=270ps

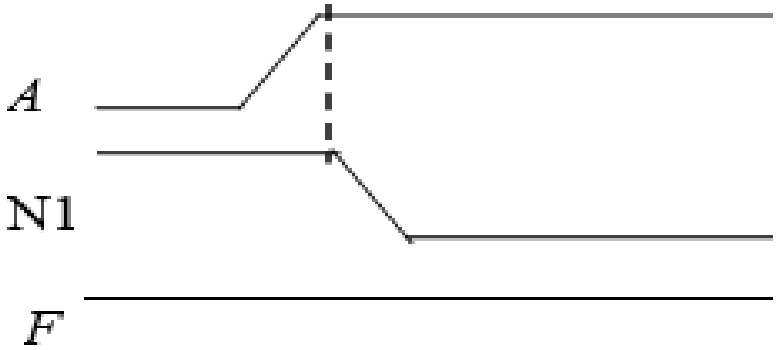
最小延迟 T_{cd} 为2级门最小延迟之和，即60psx2=120ps

3.2 竞争冒险

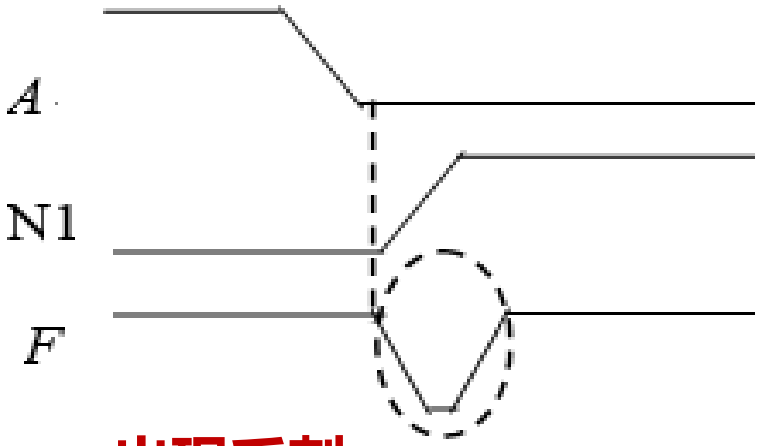
- 如果存在某个输入信号经过两条或两条以上的路径作用到输出端，由于各路径延迟不同，因而该输入信号对输出端会发生先后不同的影响，该现象称为**竞争（race）**
- 由于竞争的存在，在输入信号变化的瞬间，输出端可能会出现不正确的尖峰信号，这种信号称为**毛刺（glitch）**
- 出现毛刺的电路称为存在**冒险（hazard）**或**竞争冒险**或**险象**
- 可通过低通滤波或增加冗余项来修改逻辑设计等方式避免毛刺



存在竞争冒险的电路

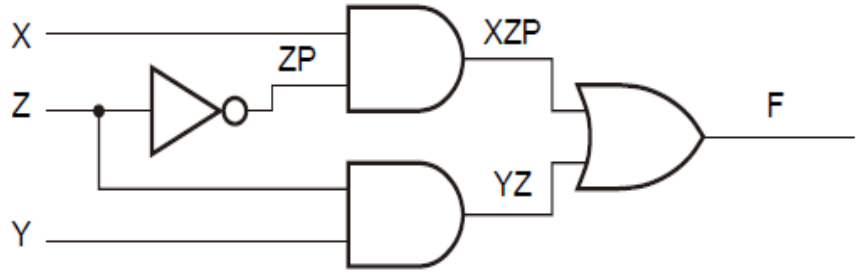


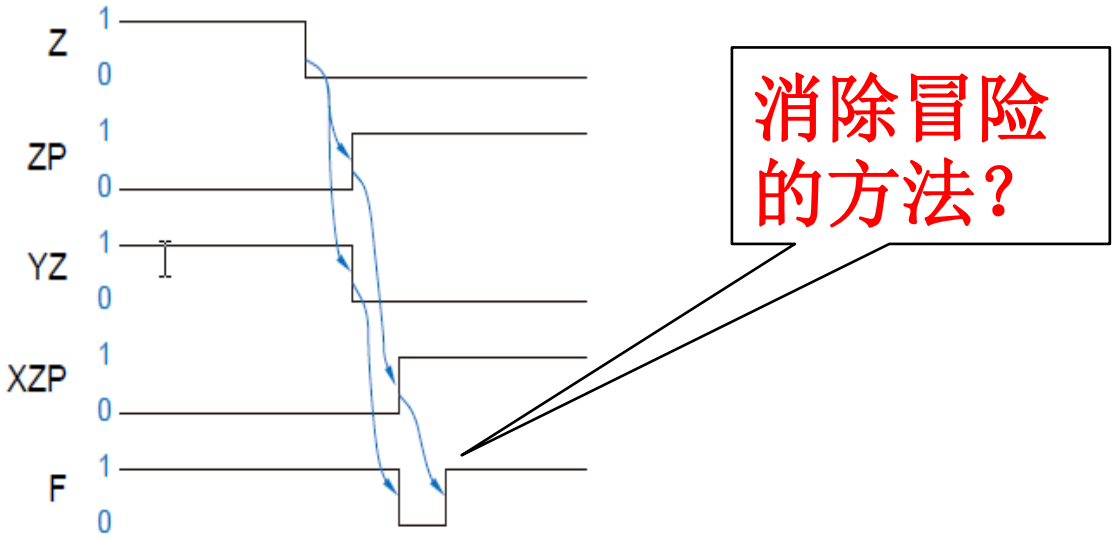
未发生毛刺



出现毛刺

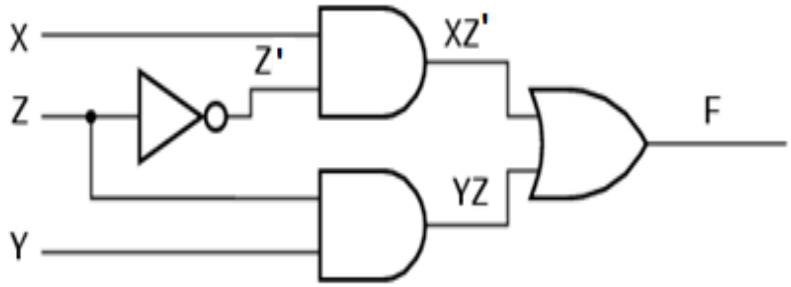
冒险举例





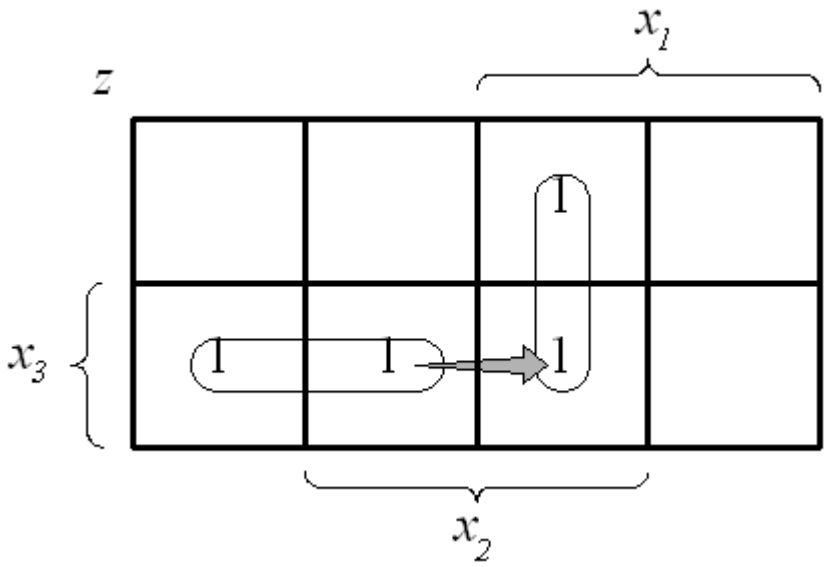
静态冒险的检测

- 卡诺图检测：在卡诺图中存在两个质主蕴涵项相切，当从一个质主蕴涵项向另一个转换时，一旦有传递延迟，则产生险态。



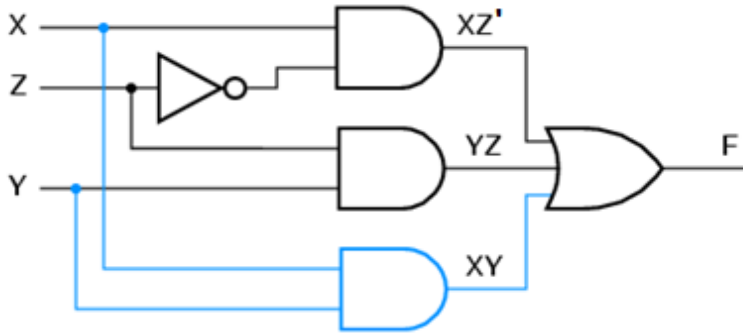
Z \ XY	00	01	11	10	
0	0	0	1	1	XZ’
1	0	1	1	0	YZ

F = XZ’ + YZ



静态冒险的消除

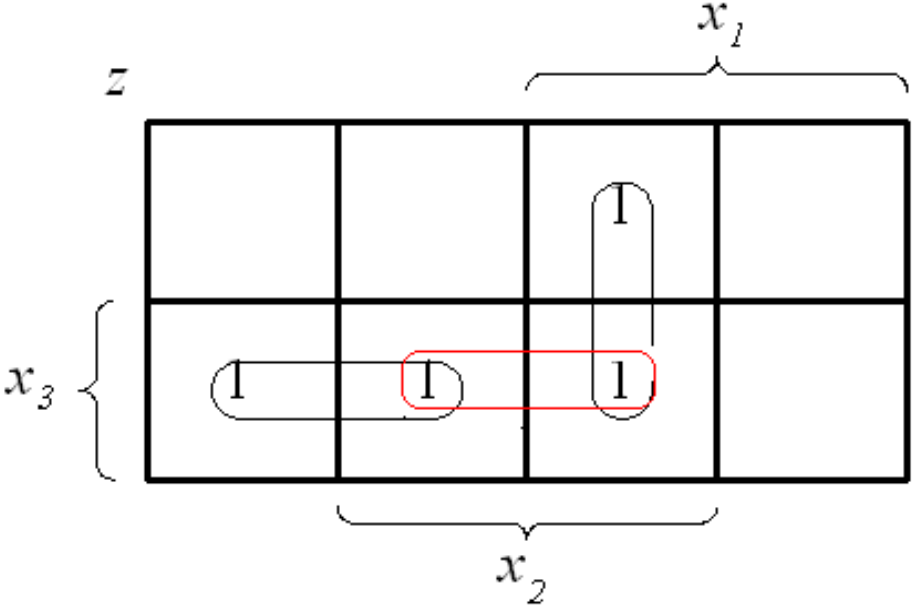
- 添加一致项consensus:增加新的主蕴涵项，覆盖相切的两个质主蕴涵。



Z \ XY	00	01	11	10	
0	0	0	1	1	XZ'
1	0	1	1	0	YZ

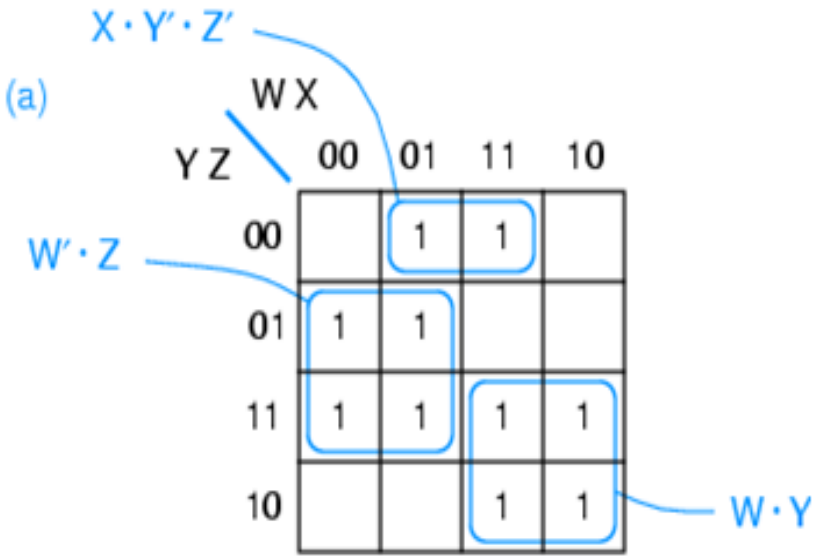
XY

F = XZ' + YZ + XY



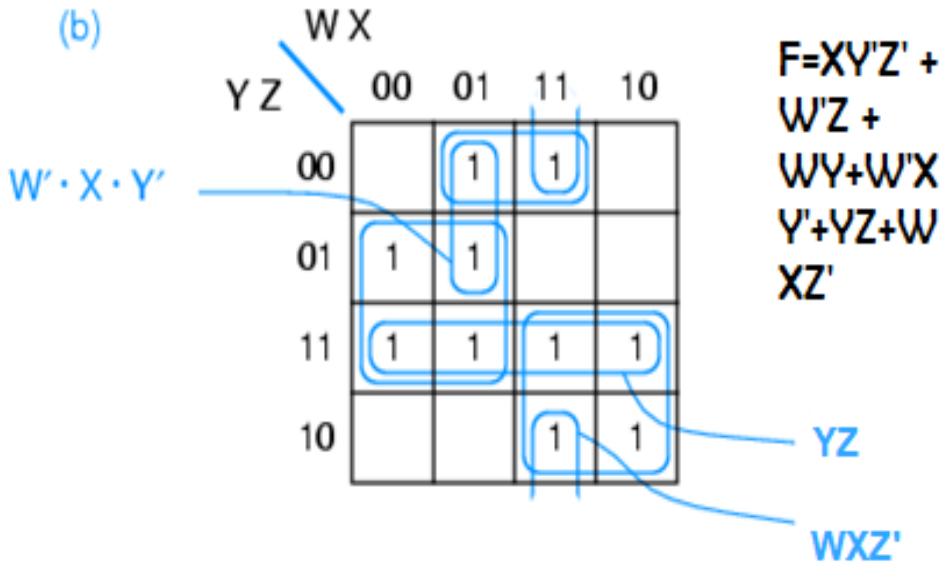
$Z=x'_1x_2+x_1x_3+x_2x_3$

另一个例子



$$F = X \cdot Y' \cdot Z' + W' \cdot Z + W \cdot Y$$

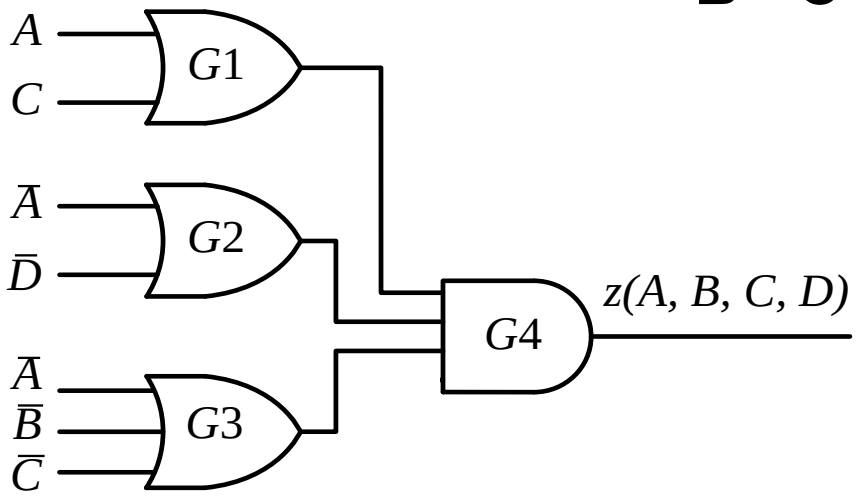
消除冒险之前

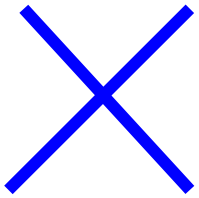


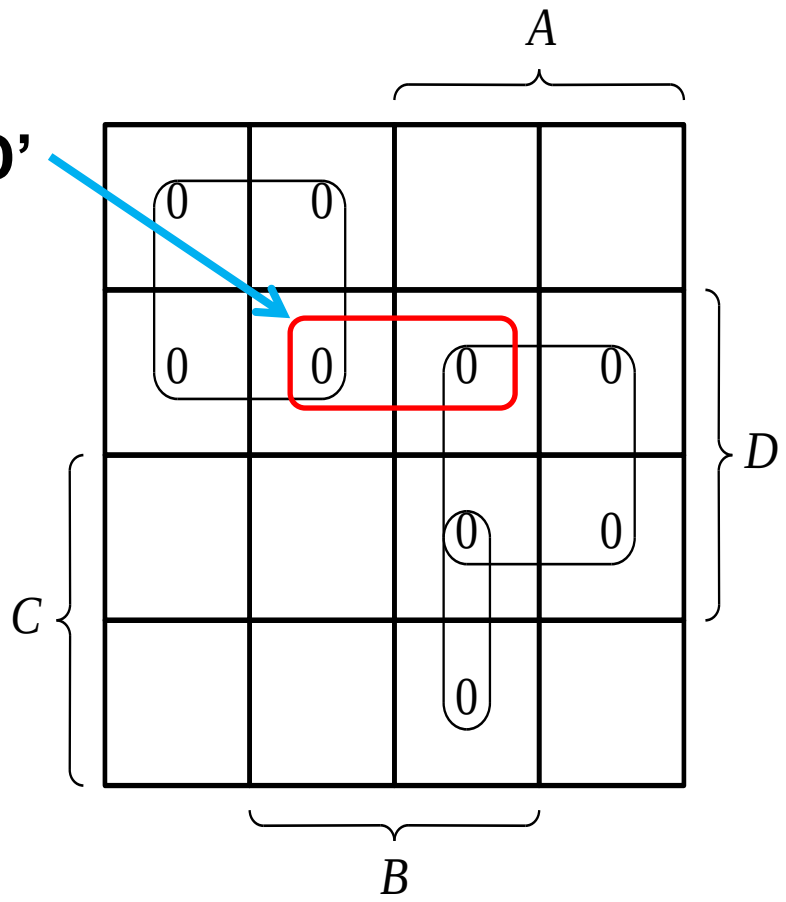
消除冒险之后

静态冒险的消除

◆静0冒险消除



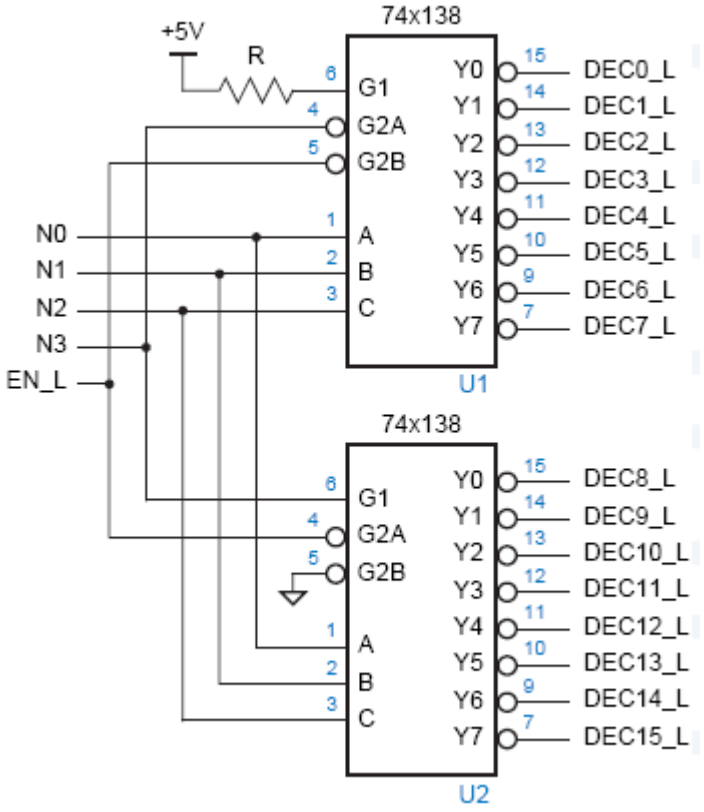


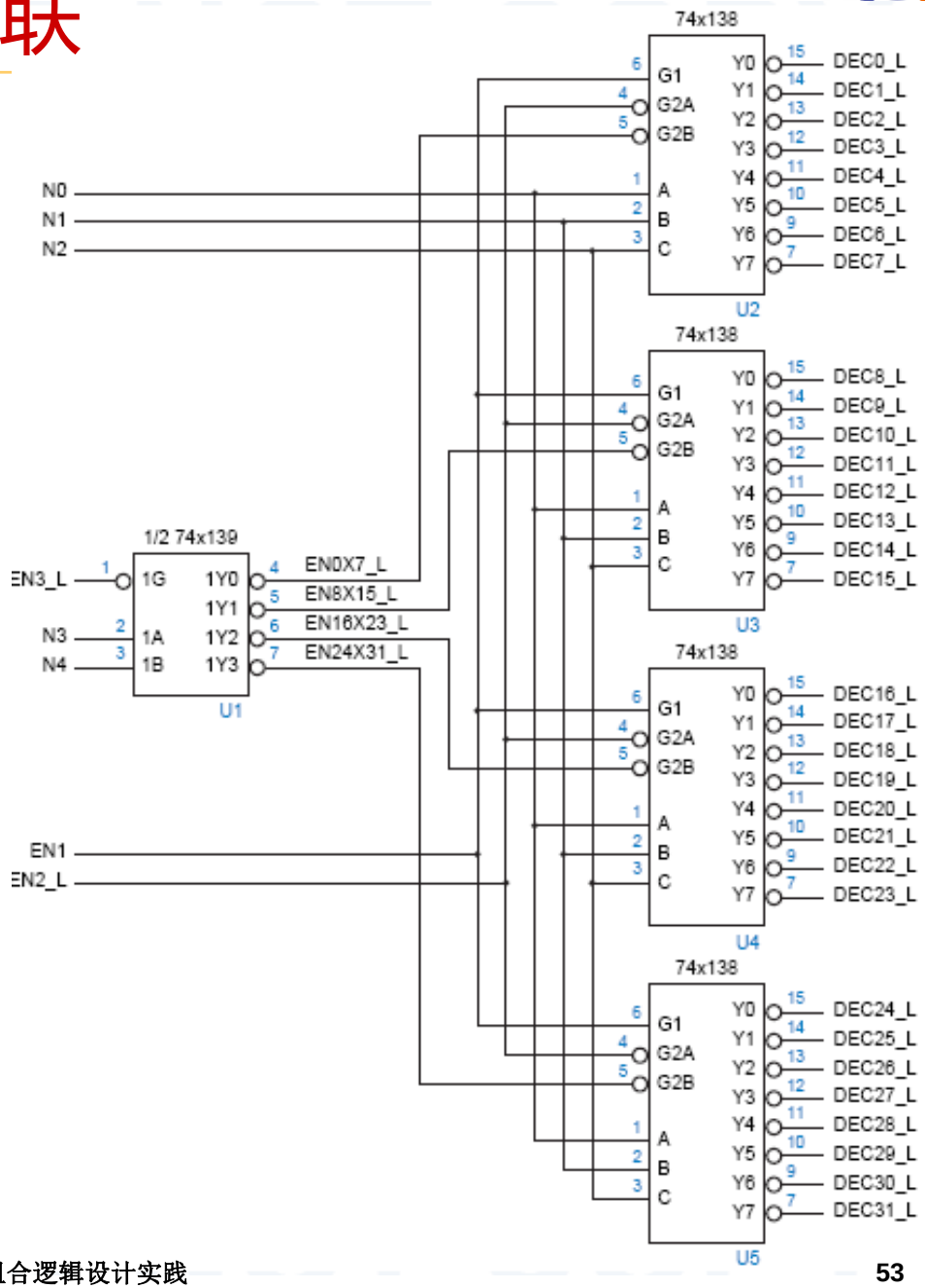


补2：译码器的级联

◆级联二进制译码器：

- 4-16译码器
- 5-32译码器



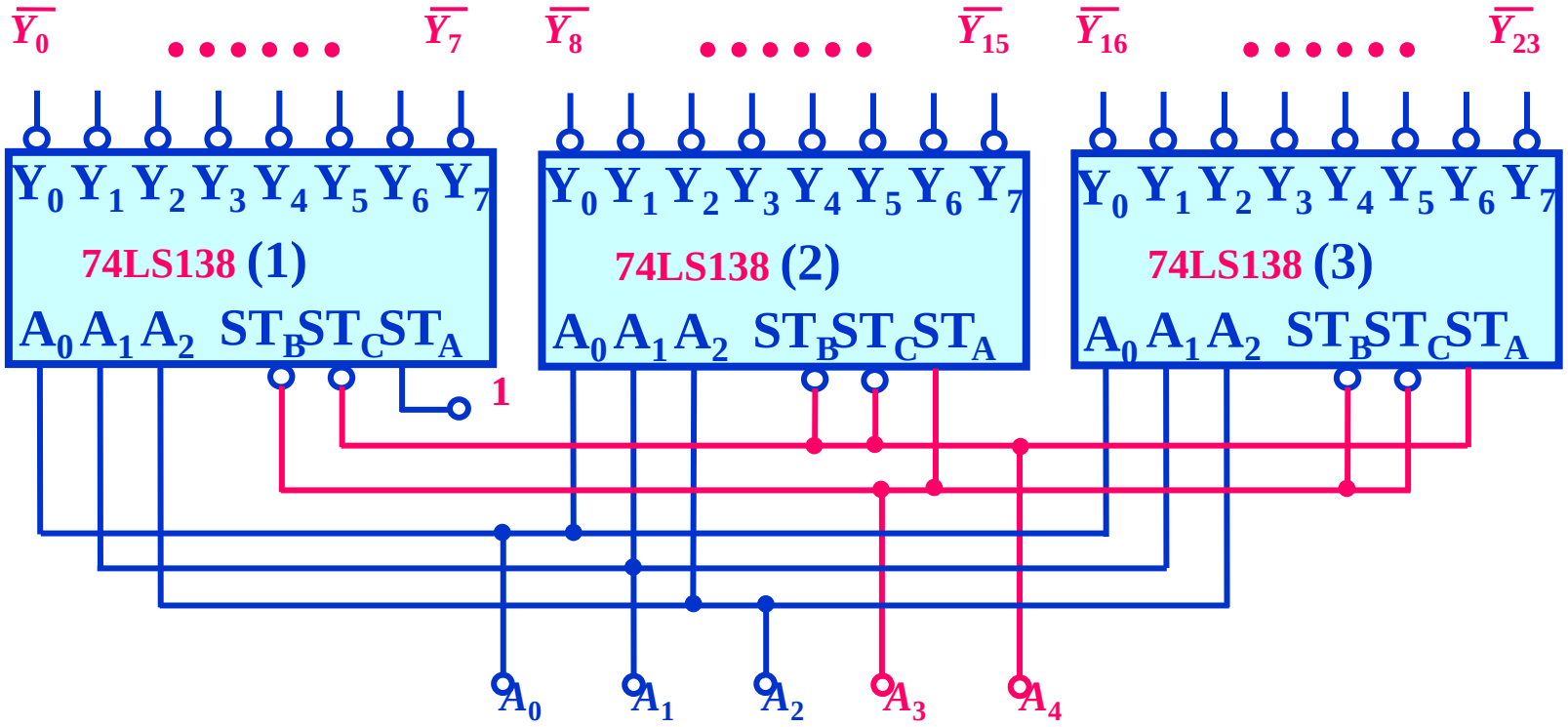


三片 3 线- 8 线

↓

5 线 - 24 线

A_4	A_3	(1)	(2)	(3)	输 出
0	0	工	禁	禁	$\overline{Y}_0 \sim \overline{Y}_7$
0	1	禁	工	禁	$\overline{Y}_8 \sim \overline{Y}_{15}$
1	0	禁	禁	工	$\overline{Y}_{16} \sim \overline{Y}_{23}$
1	1	禁	禁	禁	全为 1



补3：多路选择器的推广应用

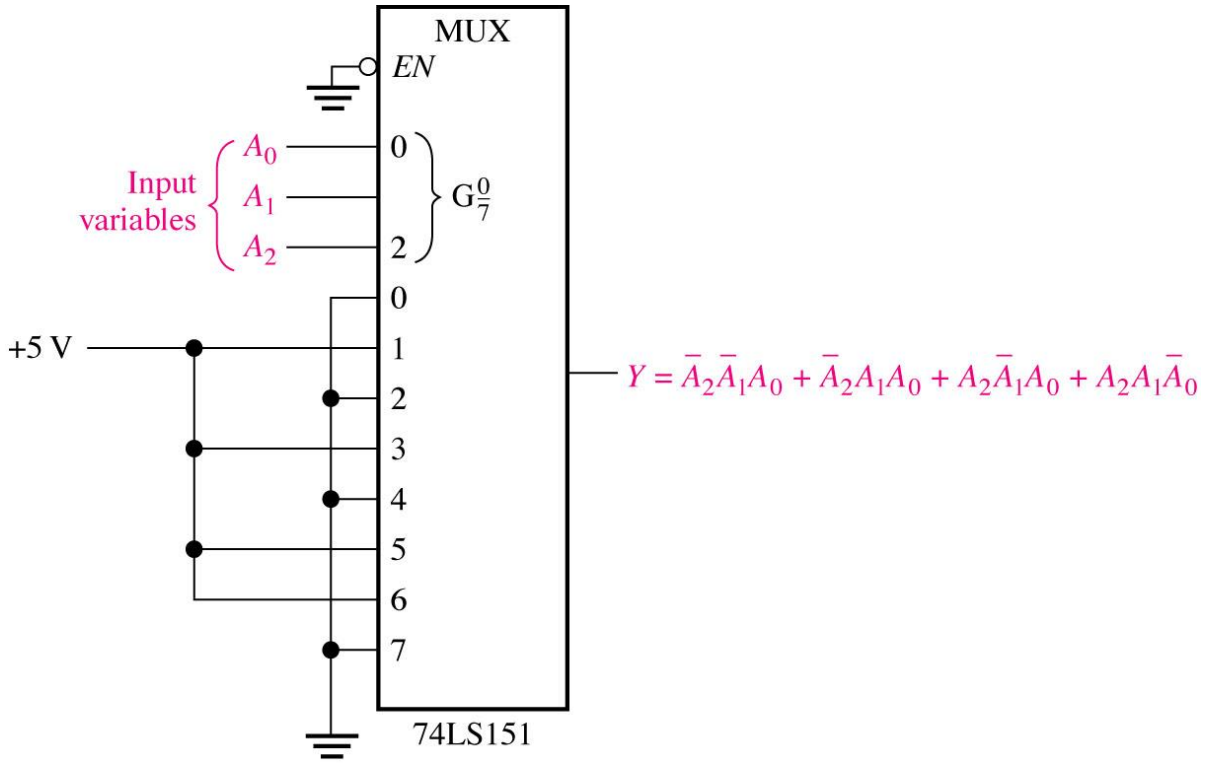
- 多路选择器除完成对多路数据进行选择的基本功能外，
- 在逻辑设计中主要用来实现各种逻辑函数功能。
- 用多路选择器实现分时多路转换电路。(将并行输入的数据转换成串行输出)。

Use MUX to Realize Function

- 方法I：用具有n个选择变量的MUX实现n个变量的函数。
  - 将函数的n个变量依次连接到MUX的n个选择变量端，并将函数表示成最小项之和的形式。若函数表达式中包含最小项m_i，则相应MUX的D_i接1，否则D_i接0。

- **Example1 Use 74151(8 to 1MUX) to Realize function: *f(A0,A1,A2)* = ∑*m*(1,3,5,6)**

m	$A_2A_1A_0$	D_i
0	0 0 0	0
1	0 0 1	1
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	0



方法简单，但并不经济，因为MUX的数据输入端未能得到充分利用。对于具有n个变量的逻辑函数，完全可以用少于n个选择变量的MUX实现。利用数据输入线定义变量。

- 方法II：用具有n-1个选择控制变量的MUX实现n个变量函数功能
  - 即从函数的n个变量中任n-1个作为MUX的选择控制变量，并根据所选定的选择控制变量将函数变换成$F=\sum m_i D_i$的形式，以确定各数据输入D_i。假定剩余变量为X，则D_i的取值只可能是0、1、X或X'四者之一。

N-1 selector value

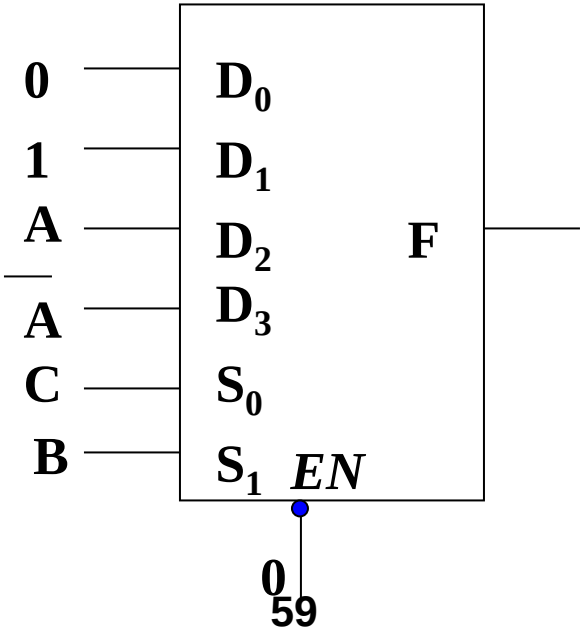
- **Example：Use 4 to 1 MUX to realize F（A，B，C）=∑m(1,3,5,6）**

  选择B、C为选择变量，A为数据输入。将函数表达式展开成选择变量最小项表示形式。列出选择器的功能表，确定每条数据输入线的值。画出逻辑电路图

$$f = \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}C + AB\overline{C}$$
$$= \overline{B}\,\overline{C} \bullet 0 + \overline{B}C(A + \overline{A}) + B\overline{C}A + BC\overline{A}$$

选择 BC	输入 A	输出 F
00	0	0
01	1	1
10	A	1
11	$\overline{A}$	1

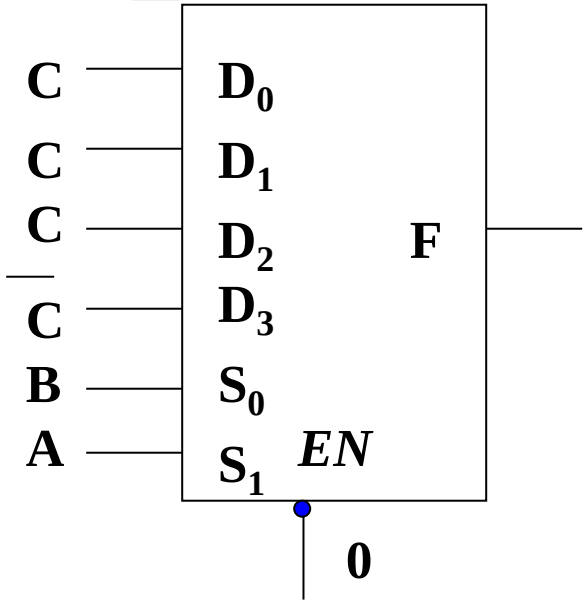
功能表



2 选择AB为选择变量，C为数据输入，画出卡诺图，标出实现函数的最小项，确定每种选择情况下，数据输入线的值（0,1，原变量，反变量）。F（A，B，C）=∑m(1,3,5,6）

输入 C	选择 AB 00	01	10	11
0	0	2	4	6
1	1	3	5	7
D_i				

卡诺图实现表



思考：1什么情况下，数据输入线值为0，什么时候为1？

2在增加逻辑门电路控制的情况下，能否用更少的选择变量来实现布尔函数？

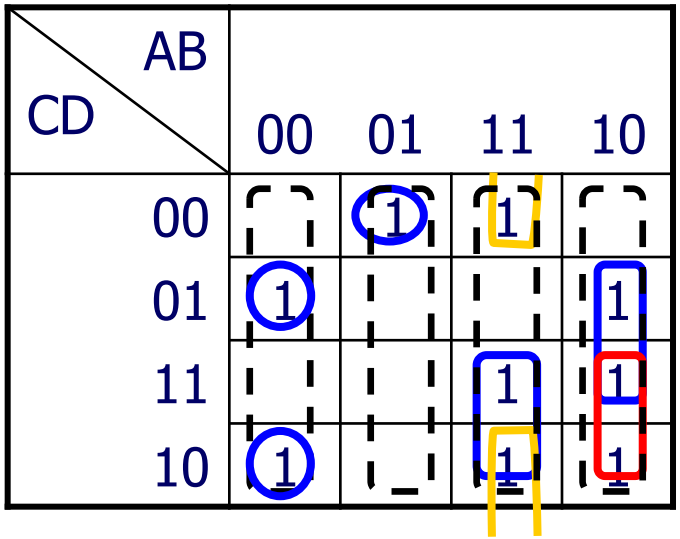
◆例： 用4路选择器实现4变量逻辑函数的功能，函数式为

$F(A，B，C，D)=\sum m(1，2，4，9，10，11，12，14，15)$

1、选择2个变量作为选择变量，其余两个变量用为数据输入值，以选择变量为基准，画出函数的卡诺图。

2、选择变量的每一列作为子卡诺图，对输入变量进行化简，确定每一种选择情况下的输入值D_i。

3、用基本逻辑门电路实现数据输入值。

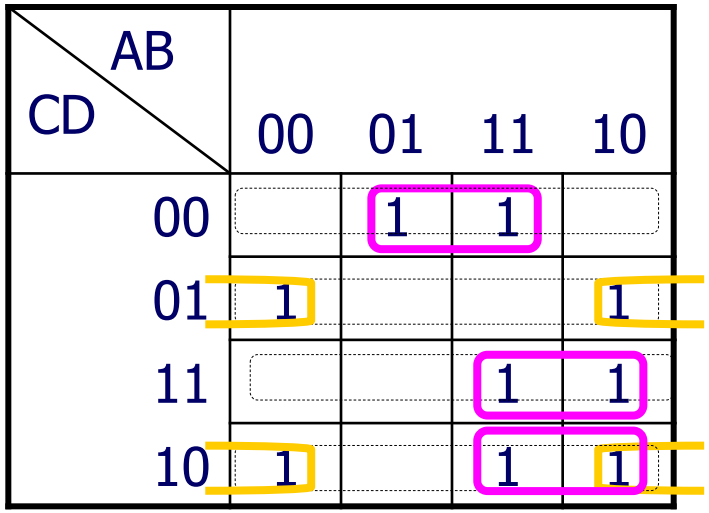


1假设AB为选择输入，则数据输入线的值为：

D0=C'D+CD'=C⊕D　　D1=C'·D'

D2=C+D　　D3=C+D'

2假设CD为选择输入，AB为数据输入，则数据线的值为：



D0=B

D1=B’

D2=A+B’

D3=A

设定不同的选择变量可取得不同导致化简的结果。

更多变量的函数，先对函数表达式化简，消去变量，得到最简式后，再设置选择变量和数据输入变量。