第3章　组合逻辑电路

组合逻辑电路：电路在任一时刻的输出状态仅由该时刻的输入信号决定，与电路在此信号输入之前的状态无关.

3.2 组合逻辑电路的分析

3.2.1 分析方法

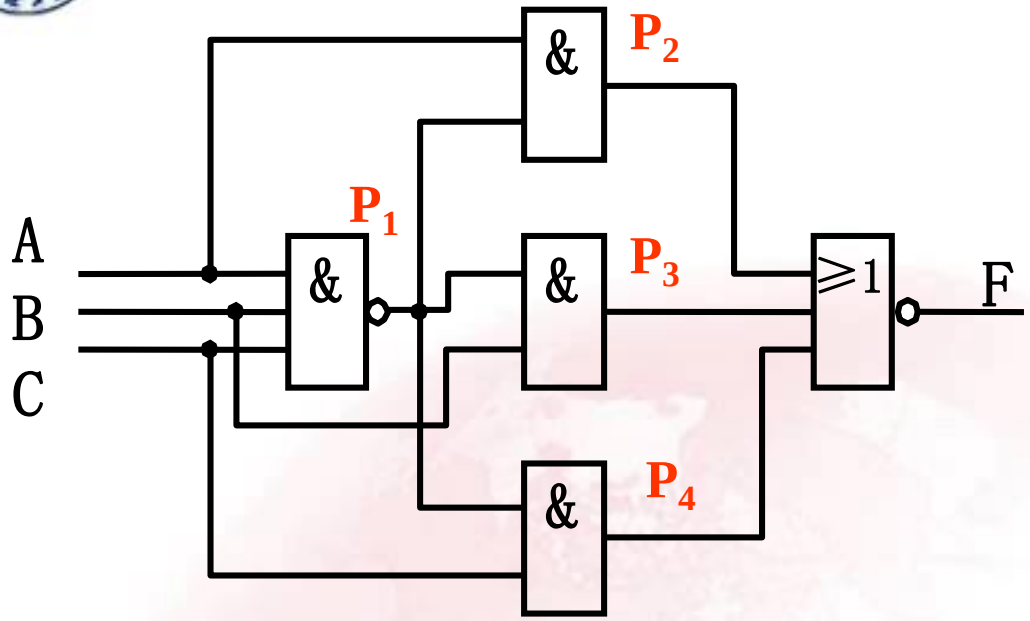
分析步骤:

(1) 根据逻辑电路图, 写出输出逻辑函数表达式;

(2) 根据逻辑表达式, 列出真值表;

(3) 由真值表或表达式分析电路功能.

例： 分析下图所示逻辑电路 真值表:



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

$F=\overline{P_2+P_3+P_4}=\overline{(A+B+C)\cdot\overline{ABC}}$

$P_2=A\cdot P_1$ $=ABC+\overline{A}\,\overline{B}\,\overline{C}$

$P_3=B\cdot P_1$

$P_4=C\cdot P_1$

$P_1=\overline{ABC}$

逻辑功能:

一致电路

3.3 组合逻辑电路设计

一般步骤:

(1) 由实际逻辑问题列出真值表;

(2) 由真值表写出逻辑表达式;

(3) 化简、变换输出逻辑表达式;

(4) 画出逻辑图。

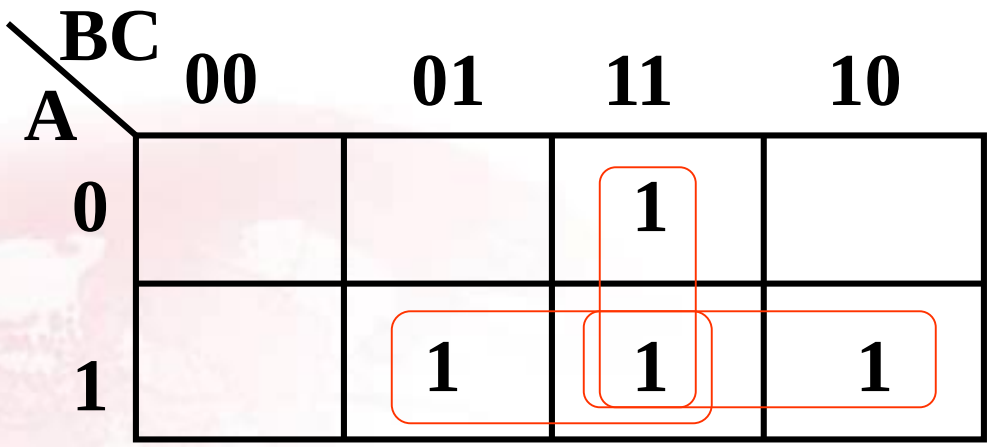
例：试用与非门设计一个三变量表决电路，表决规则为少数服从多数.

解：（1）列真值表

设：由A、B、C表示三个输入变量，F表示表决结果。并设A、B、C为1表示赞成，为0表示反对；F为1表示表决通过，为0 表示不通过。

A	B	C	F
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	1

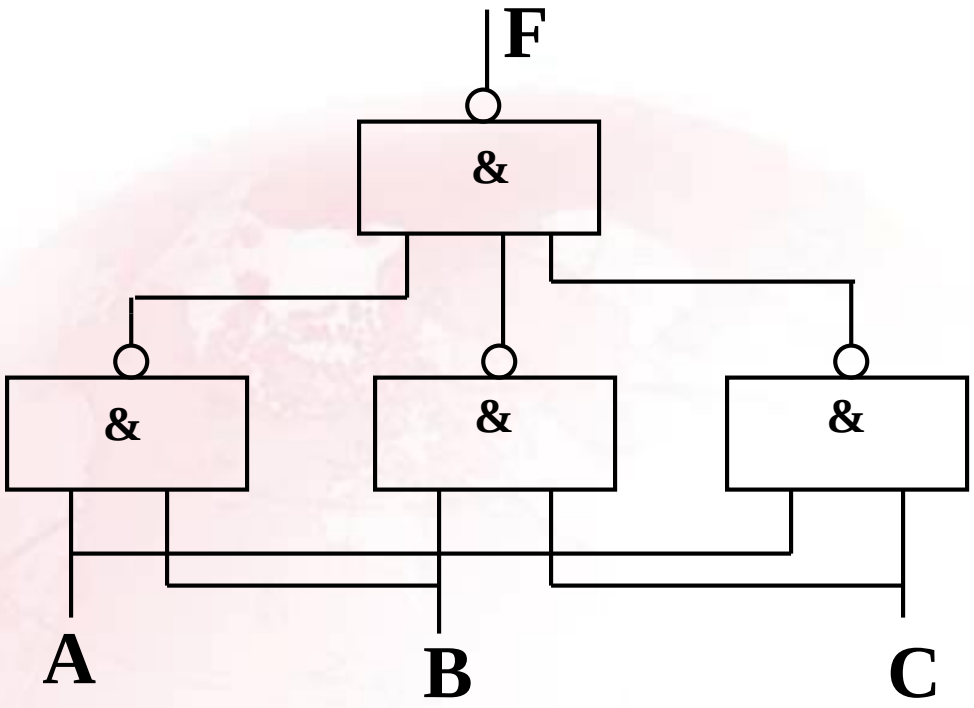
(2) 化简、求最简函数表达式



$$F=AB+AC+BC$$

$$=\overline{\overline{AB}\cdot\overline{AC}\cdot\overline{BC}}$$

(3) 画出电路图



例　设计一个两位二进制数比较器。

解 设被比较的数分别为 $A=A_1A_0$,$B=B_1B_0$;比较的结果为:$A_1A_0>B_1B_0$时，输出$F_1=1$； $A_1A_0=B_1B_0$时，输出$F_2=1$； $A_1A_0<B_1B_0$时，输出$F_3=1$.

列真值表：

A_1	A_0	B_1	B_0	F_1	F_2	F_3	A_1	A_0	B_1	B_0	F_1	F_2	F_3
0	0	0	0	0	1	0	1	0	0	0	1	0	0
0	0	0	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	0	0	1	1	0	1	0	0	1	0
0	0	1	1	0	0	1	1	0	1	1	0	0	1
0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	1	0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0	1	0	0
0	1	1	1	0	0	1	1	1	1	1	0	1	0

画卡诺图化简：

$$F_2=\overline{A}_1\overline{A}_0\overline{B}_1\overline{B}_0+\overline{A}_1A_0\overline{B}_1B_0+A_1\overline{A}_0B_1\overline{B}_0+A_1A_0B_1B_0$$

$$F_1=A_1\overline{B}_1+A_1A_0\overline{B}_0+A_0\overline{B}_1\overline{B}_0$$

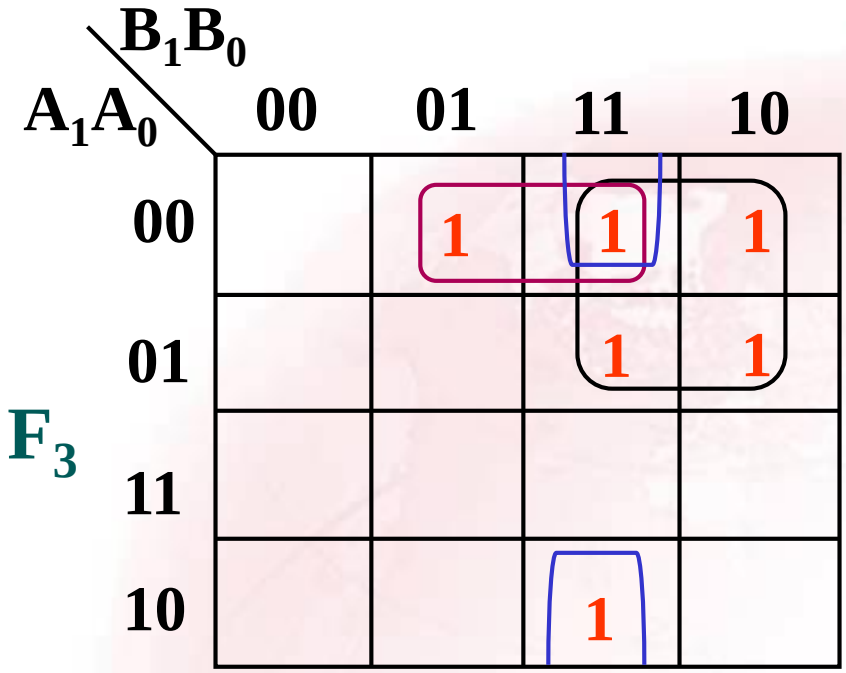
F_1

A_1A_0 \ B_1B_0	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

F_2

A_1A_0 \ B_1B_0	00	01	11	10
00	1			
01		1		
11			1	
10				1

$$F_3=\overline{A}_1B_1+\overline{A}_1\overline{A}_0B_0+\overline{A}_0B_1B_0$$



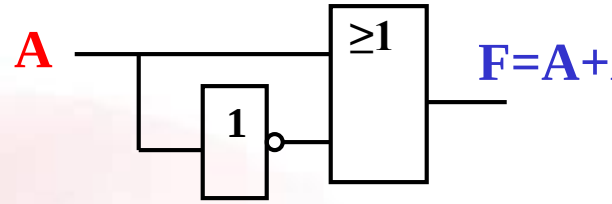
按F_1、F_2和F_3表达式可方便地用门电路实现比较器的逻辑功能。

3.4 组合逻辑电路中的冒险

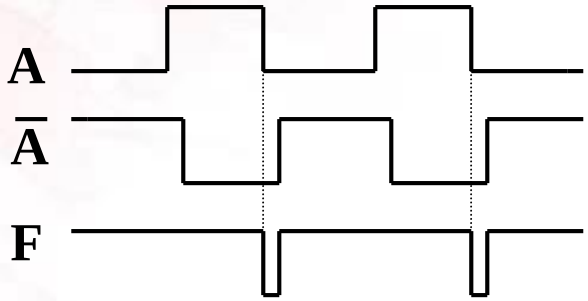
前面分析组合逻辑电路时，没有考虑门电路的延迟时间对电路的影响。实际上，由于门电路延迟时间的关系，可能会使逻辑电路产生错误输出。通常把这种现象称为竞争冒险。

产生冒险的原因

以例说明



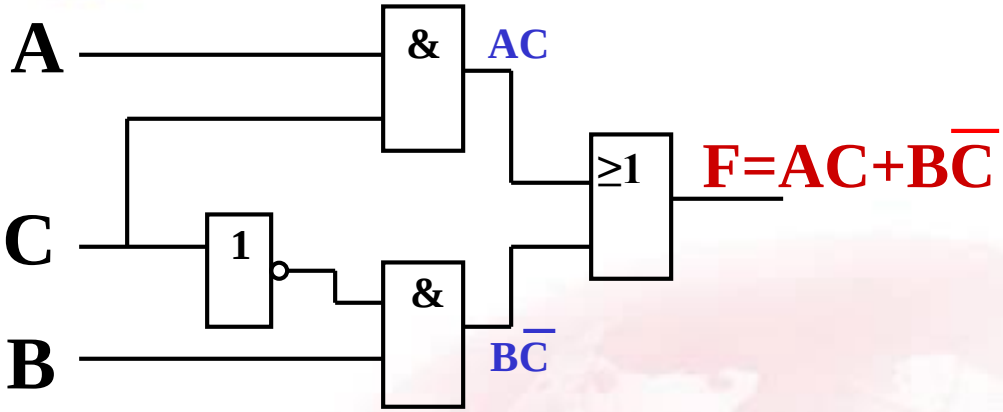
$F=A+\overline{A}=1$ 理想情况

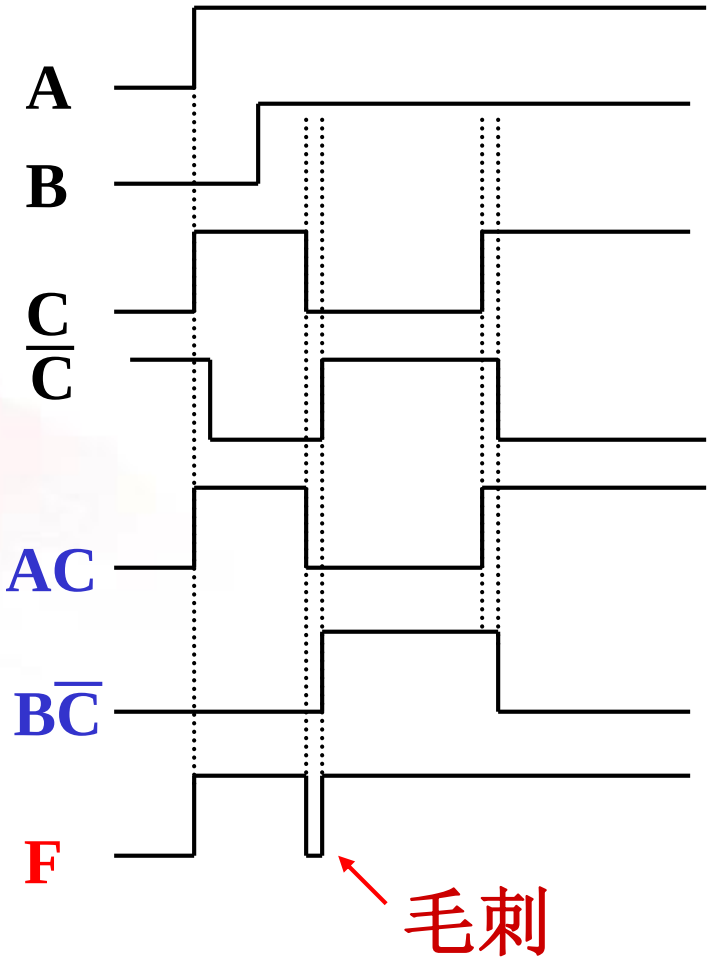


实际情况

造成冒险的原因是由于A和$\overline{A}$到达或门的时间不同。

再举一例





(分析中略去与门和或门的延时）

产生冒险的原因之一：

电路存在由非门产生的互补信号，且互补信号的状态发生变化时有可能出现冒险现象。

消去冒险的方法

1. 发现并消去互补变量

例如：$F=(A+B)(\bar{A}+C)$　在$B=C=0$时，$F=A\bar{A}$. 若直接根据这个逻辑表达式组成电路，就可能出现冒险。

将上式写成：$F=AC+\bar{A}B+BC$, 已将$A\bar{A}$去掉，则不会出现冒险。

2. 增加乘积项

例如：$F=AC+B\bar{C}$, 当$A=B=1$时，$F=C+\bar{C}$. 若直接根据这个逻辑表达式组成电路，就可能出现冒险。

将上式写成： $F=AC+B\bar{C}+AB$, 这样，当A=B=1时，不会出现$F=C+\bar{C}$,所以C状态的变化，不会影响输出。

3. 输出端并联电容器

如果逻辑电路在较慢速度下工作，为了消去冒险，可以在输出端并联一电容，其容量在4~20pF之间，该电容和门的输出电阻构成RC低通网络，对窄脉冲起平滑作用。

3.5 可编程逻辑器件和VHDL概述

利用可编程逻辑器件（PLD，Programmable Logic Device）来实现电路的设计

硬件描述语言（HDL，HardwareDescription Language）就是可以描述硬件电路的功能

VHDL是应用最为广泛的国际标准电子设计语言

3.5.1 VHDL基本结构

硬件描述语言的基本格式包括两个要素

输入、输出的定义（即输入、输出说明）

对输出如何响应输入的定义（工作原理）

对应逻辑符号的描述部分：实体（Entity）

对应逻辑关系的说明部分：结构体(Architecture)

以二输入与门为例:

```
1    ENTITY and_gate IS
2      PORT(a,b:IN BIT;
3           y:  OUT BIT);
4    END and_gate;

5    ARCHITECTURE rtl OF and_gate IS
6      BEGIN
7        y<=a AND b;
8    END rtl;
```

ENTITY 实体名 IS

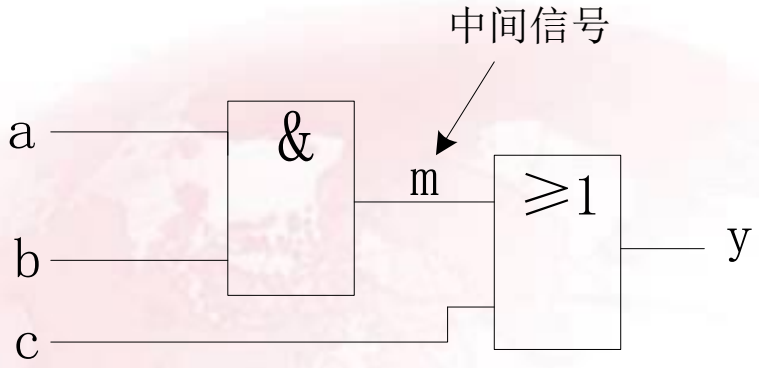
PORT（端口名：端口模式 端口类型）

ARCHITECTURE 结构体名OF 实体名 IS

“<=”为赋值符

3.5.2 VHDL中的中间信号

电路模块内部的信号点，不是模块的输入也不是输出



与输入输出端口分开定义，在逻辑功能描述部分定义

仅在一个模块内部有效

```
1       ENTITY fig2 IS
2          PORT(a,b,c : IN BIT;
3             y: OUT BIT);
 4       END fig2;

5       ARCHITECTURE ckt OF fig2 IS
6          SIGNAL m :BIT;
 7         BEGIN
8          m<=a AND b;
9          y<=m OR c;
10        END ckt;
```

SIGNAL是关键字，定义m为中间信号

并行赋值语句

3.5.3 VHDL描述逻辑电路的进程形式

进程语句（PROCESS）是VHDL常用的子结构描述语句

以2输入与非门为例：

```
1    LIBRARY IEEE;                                    库说明语句
2    USE IEEE.STD_LOGIC _1164.ALL;                    使用包集合的说明语句

3    ENTITY nand2 IS
4      PORT(a,b: IN STD_LOGIC;                        实体描述部分
5                y: OUT STD_LOGIC);
6    END nand2;
```

```
7    ARCHITECTURE nand2_l OF nand2 IS
8    BEGIN
9      PROCESS (a,b)
10   VARIABLE tmp:STD_LOGIC_VECTOR(1 DOWNTO 0);
11   BEGIN
12     tmp:=a&b;
13   CASE tmp IS
```

结构体描述部分

PROCESS (敏感信号表)

变量定义语句，定义tmp为新的变量

“: =”为变量赋值符号。“&”为并置运算符

条件选择语句

```
14          WHEN”00”=>y<=’1’;
15          WHEN”01”=>y<=’1’;
16          WHEN”10”=>y<=’1’;
17          WHEN”11”=>y<=’0’;
18           WHEN OTHERS=>y<=’X’;   输出状态不定
19        END CASE;
20       END PROCESS;                进程结束语句
21      END nand2_l;
```