

# 第四章

## 组合电路的逻辑分析与设计

4.1 组合电路的特点及有关问题

4.2 组合电路的逻辑分析

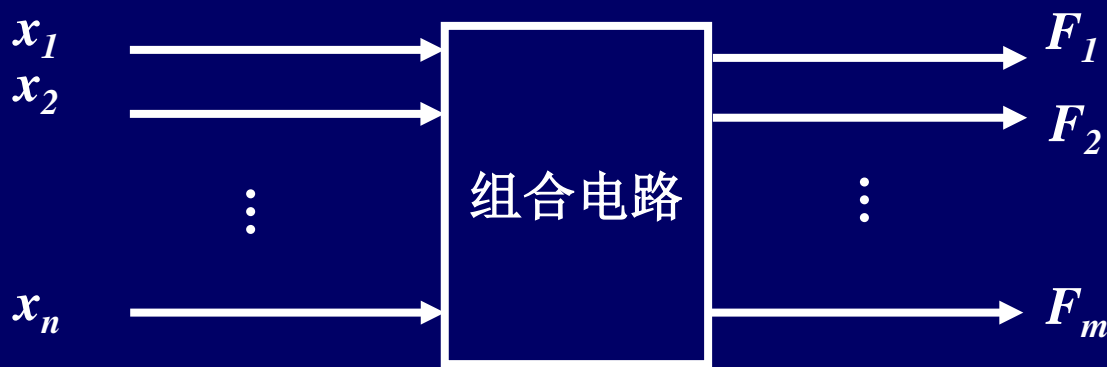
4.3 组合电路的逻辑设计与描述

4.4 组合电路中的竞争与险象

4.5 常用MSI组合电路器件的应用与描述

## 4.1 组合电路的特点及有关问题

组合电路是指电路在任何时刻产生的稳态输出仅仅取决于该时刻输入变量取值组合，而与过去的输入值无关。



$$F_i = f(x_1, x_2, \dots, x_n) \quad (i = 1, 2, \dots, m)$$




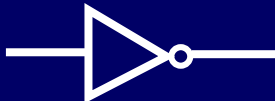


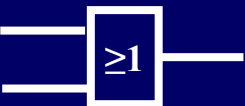





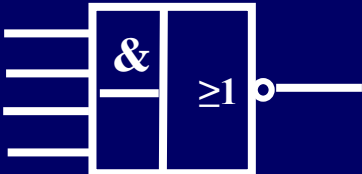
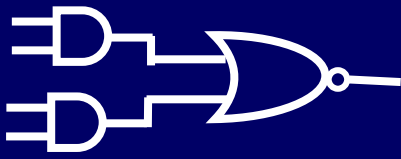
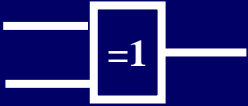

组合电路的特点：

- (1) 由逻辑门电路组成，不含任何记忆元件。
- (2) 信号是单向传输的，不存任何反馈回路。

## 4.1.1 逻辑门符号标准

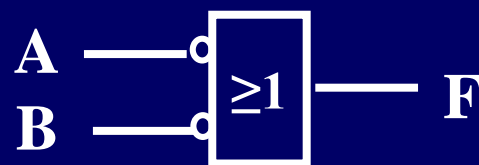
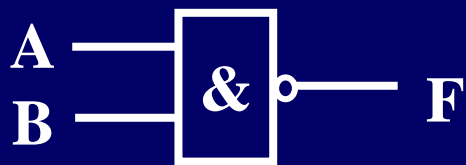
- (1) 长方形符号：所有的门采用相同的长方形形状，用内部标志区分门的类型
- (2) 变形符号：不同类型的门采用不同的形状，用形状区分门的类型。

应避免两种逻辑门符号出现在同一电路图中。

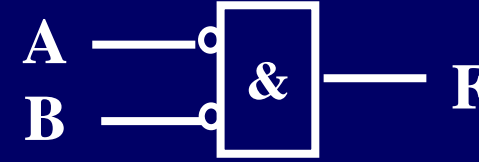
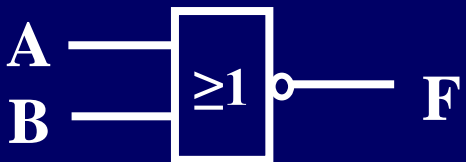
常用逻辑门的两种符号		
名称	长方形符号	变形符号
跟随器		
非门		
与门		
或门		
与非门		
或非门		
与或非门		
异或门		

## 4.1.2 逻辑门的等效符号

$$F = \overline{A \bullet B} = \overline{A} + \overline{B}$$



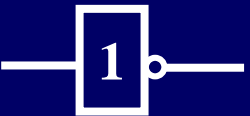
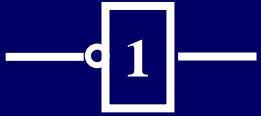

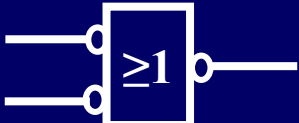
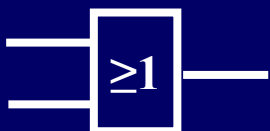

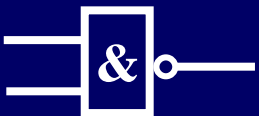
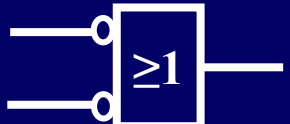
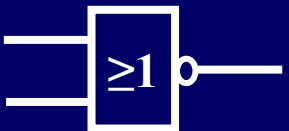
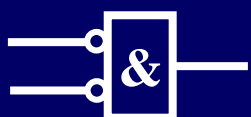


$$F = \overline{A + B} = \overline{A} \bullet \overline{B}$$



门的等效符号主要用于正、负逻辑的表达及有效级的变换。

## 根据摩根定理得到的门的等效符号

名称	原符号	等效符号
跟随器		
非门		
与门		
或门		
与非门		
或非门		

## 4.1.3 信号名和有效级

### 1. 信号命名

逻辑电路的输入、输出信号最好按照其功能和用途来命名。

数据信号用  $D_0$ 、 $D_1$ 、 $D_2 \dots$

地址信号用  $A_0$ 、 $A_1$ 、 $A_2 \dots$

控制信号中的片选信号用  $CS$ ，使能信号用  $EN \dots$

无特殊含义时，输入信号常用  $A$ 、 $B$ 、 $C$ 、 $D \dots$

$X_0$ 、 $X_1$ 、 $X_2 \dots$

$I_0$ 、 $I_1$ 、 $I_2 \dots$

输出信号常用  $F_0$ 、 $F_1$ 、 $F_2 \dots$

$Y_0$ 、 $Y_1$ 、 $Y_2 \dots$

$Z_0$ 、 $Z_1$ 、 $Z_2 \dots$

## 2. 信号的有效级

一个逻辑电路，只有在一定的信号逻辑状态下，才能正确地表现出给定的逻辑功能。即：它的控制条件、测试信号都有一个与之对应的有效级，只有信号处在有效级时，逻辑电路才能正确地执行其功能。

例如，某个译码器只有在使能信号为高电平时，才能表现出译码的功能。

再例如，某个片选信号低有效的存储器芯片，只有在片选信号为低电平时，才能正确实现其读、写功能。

信号的有效级仅分为高有效或低有效：

高有效——信号为高电平或逻辑“1”时有效。

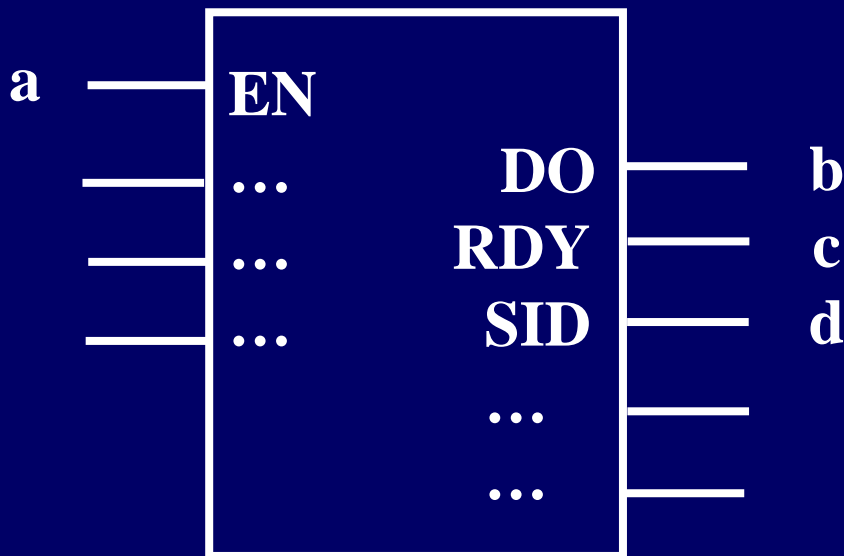
低有效——信号为低电平或逻辑“0”时有效。



按照约定，信号的有效级应反映在信号名上。即：将表示有效级的符号作为信号名的前缀或后缀。

信号有效级的约定		
组号	低电平有效	高电平有效
1	ACK-	ACK+
2	ERROR .L	ERROR .H
3	ACS(L)	ACS(H)
4	CS*	CS
5	/EN	EN
6	RESET#	RESET

## 4.1.4 引端的有效级



引端的有效级是指输入、输出上的物理量（逻辑电平）与其内部逻辑状态的对应关系。

器件的符号框

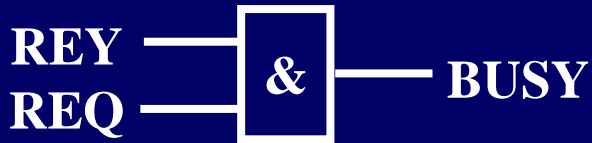
### 逻辑非符号体制

图形符号上有带逻辑非符号（即小圆圈）和不带逻辑非符号的输入、输出。只表示输入、输出的内部逻辑状态和外部逻辑状态之间的关系，而外部逻辑状态和物理量（逻辑电平）之间的关系是用正、负逻辑来规定的。

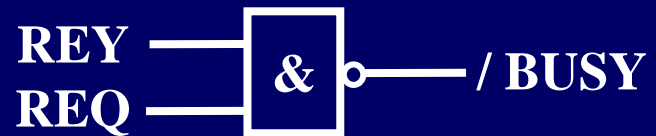
## 4.1.5 引端有效级的变换（混合逻辑变换）

例：设输入信号为REY，当信号REQ到达，则产生输出信号BUSY。  
要求输入均有效时，输出才有效。

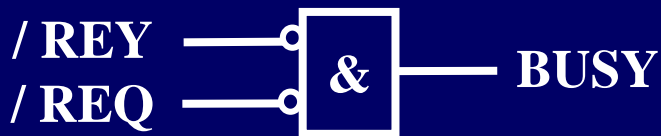
解：这无疑是一个与操作。表达式为  $BUSY = REY \bullet REQ$   
但是，如果考虑输入、输出有效级的要求，可用不同类型的逻辑门完成这个“与”功能。



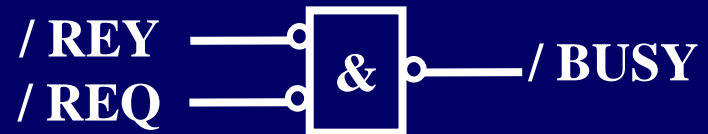
高有效输入、输出（与门）



高有效输入、低有效输出（与非门）



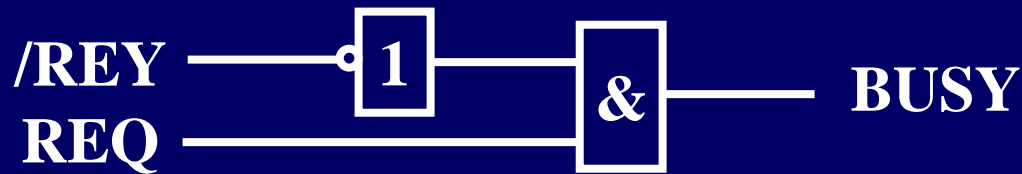
低有效输入、高有效输出（或非门）



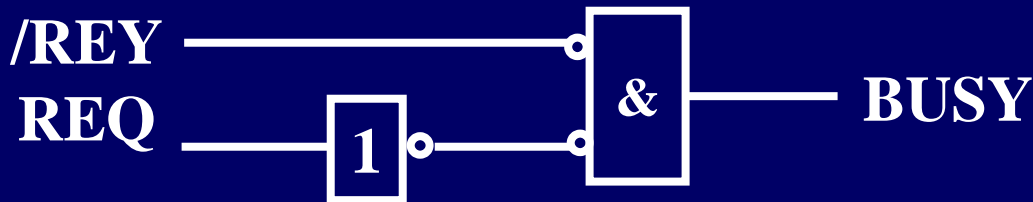
低有效输入、低有效输出（或门）

当输入信号具有不同的有效级，且与可用门的输入端有效级不一致时，可采用非门改变输入信号的有效级。

例如：给定输入信号为 **/REQ**、**REQ**，输出信号为**BUSY**，可用下面的电路产生输出信号。



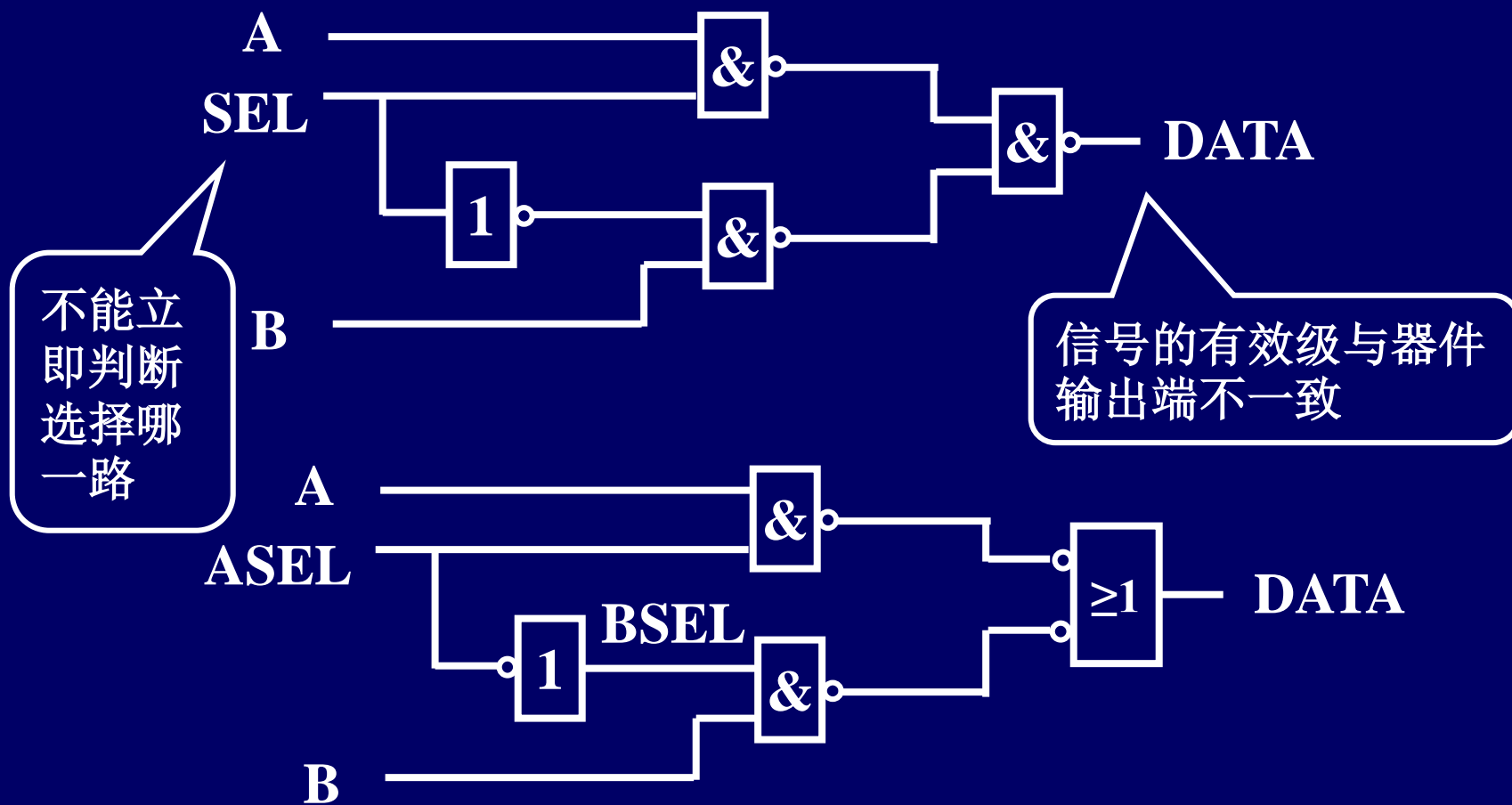
用一个与门



用一个或非门

带逻辑非符号的门（如或非门）比不带逻辑非符号的门（如与门）速度更快。

如果逻辑图上给出的信号有效级与器件对应端的有效级不一致，常使某一部分电路的逻辑功能含糊不清，且难以判断逻辑图中某点的信号极性，为读图及测试带来困难。



将图改画后，逻辑功能一目了然：当ASEL有效， $DATA=A$ ；反之， $DATA=B$ 。

## 4.2 组合电路的逻辑分析

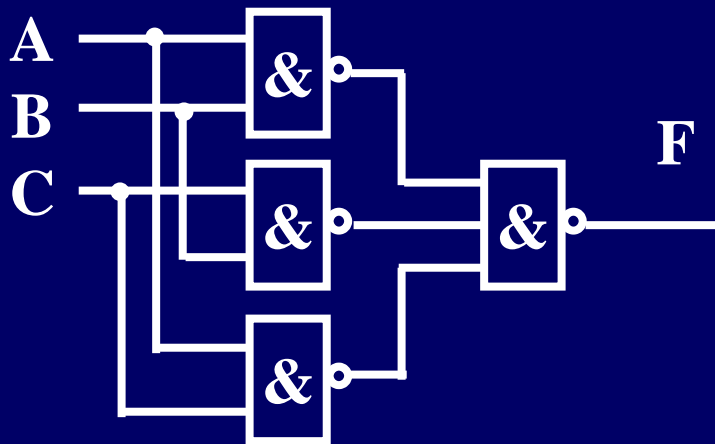
分析目的：

- (1) 确定输入变量的不同取值时，电路的逻辑功能是否满足要求。
- (2) 得到输出函数的标准与或表达式，以便采用可编程逻辑器件实现。
- (3) 利用该电路的逻辑功能描述，建立硬件描述语言模型。

分析方法：

- (1) 根据给定的逻辑图写出输出函数的逻辑表达式；
- (2) 化简输出函数的逻辑表达式；
- (3) 列出输出函数的真值表；
- (4) 电路逻辑功能评述。

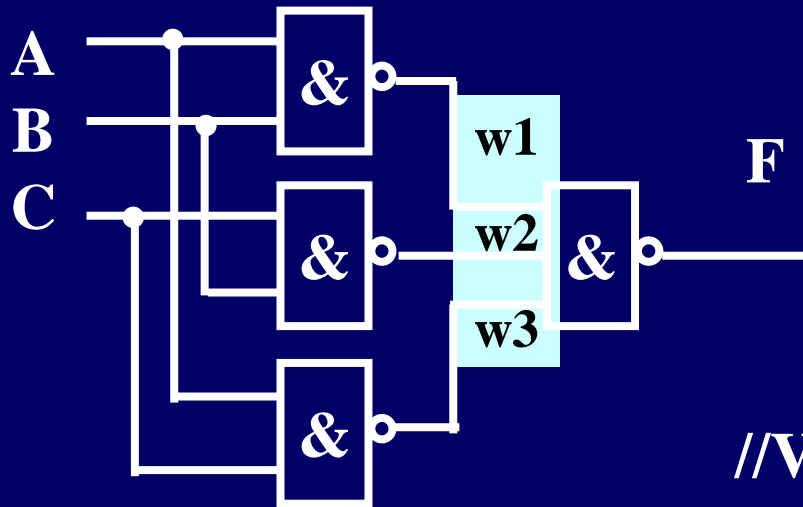
例1：试分析图示逻辑电路的功能， 并建Verilog HDL模型。



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

<i>A</i>	<i>B</i>	<i>C</i>	<i>F</i>
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

电路功能：多数变量取值为1，则F为1；否则，F为0。  
该电路为少数服从多数电路——表决电路。



## Verilog HDL门级建模

自行构造了一个3人多数  
表决“逻辑器件”。

```
//Verilog HDL门级建模
module example( a,b,c,f );
    input a,b,c;
    output f;
    wire w1,w2,w3;
    nand u1 (w1,a,b),
           u2 (w2,b,c),
           u3 (w3,a,c),
           u4 (f,w1,w2,w3);
endmodule
```



## 建立Verilog HDL的数据流描述模型

$$F = AB + BC + AC$$

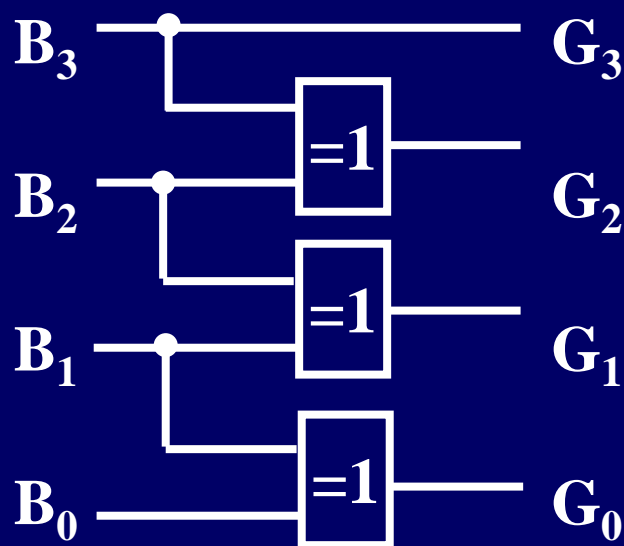
```
//Verilog HDL数据流模型
module example( a,b,c,f );
    input a,b,c;
    output f;
    wire w1,w2,w3;
    assign w1=a&b;
    assign w2=b&c;
    assign w3=a&c;
    assign f = w1|w2|w3;
endmodule
```

## 建立Verilog HDL的行为模型

<i>A</i>	<i>B</i>	<i>C</i>	<i>F</i>
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

```
module example( a,b,c,f );  
    input  a,b,c;  
    output f ;  
    reg    f ;  
    always @(a or b or c)  
        begin  
            case ({a,b,c})  
                3'b011 : f=1;  
                3'b101 : f=1;  
                3'b110 : f=1;  
                3'b111 : f=1;  
                default : f=0;  
            endcase  
        end  
endmodule
```

## 例2：试分析图示逻辑电路的功能



$$G_3 = B_3$$

$$G_2 = B_3 \oplus B_2$$

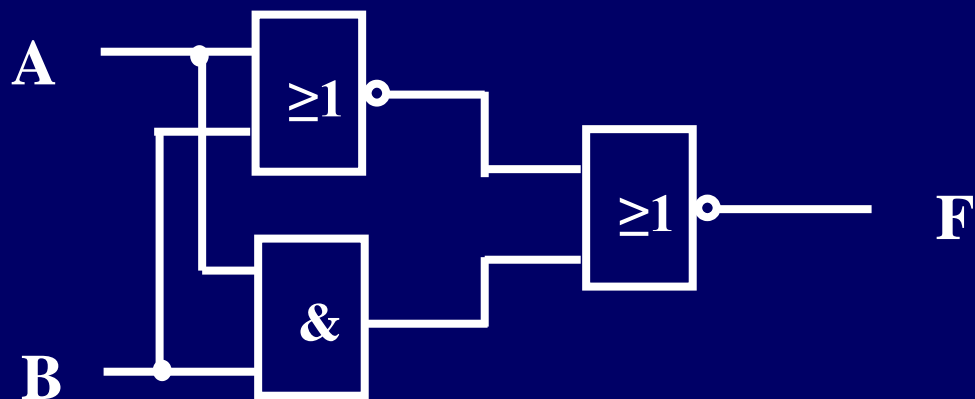
$$G_1 = B_2 \oplus B_1$$

$$G_0 = B_1 \oplus B_0$$

功能描述：自然二进制码转换成格雷码（循环码）

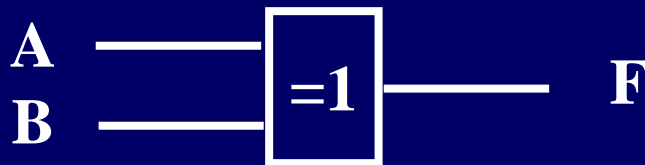
$B_3 B_2 B_1 B_0$	$G_3 G_2 G_1 G_0$
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

### 例3：试分析图示逻辑电路的功能

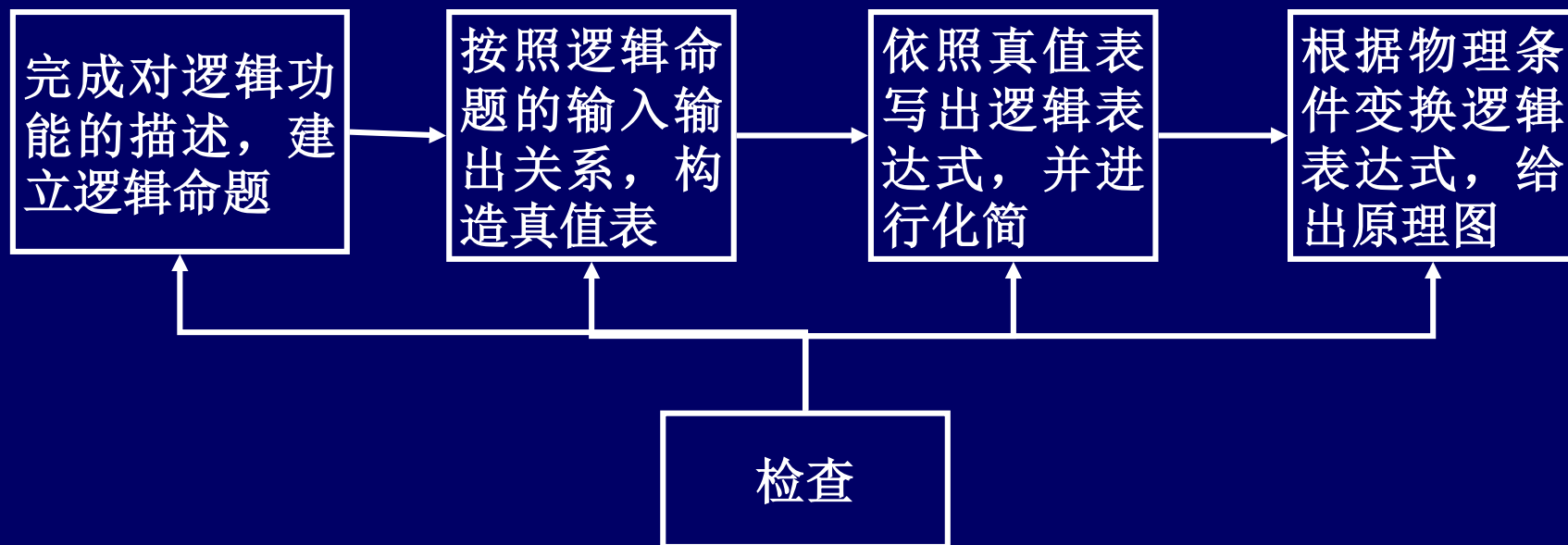


$$F = \overline{\overline{A+B} + AB} = (A+B)\overline{AB} = (A+B)(\overline{A} + \overline{B}) = \overline{A}B + A\overline{B} = A \oplus B$$

功能描述：由化简后的输出函数表达式可看出，原电路实现 A、B 的异或功能，可用一个异或门实现。



## 4.3 组合电路的逻辑设计



## 4.3.1 根据逻辑问题的描述写出逻辑表达式

### 一. 逻辑问题描述——真值表——逻辑表达式

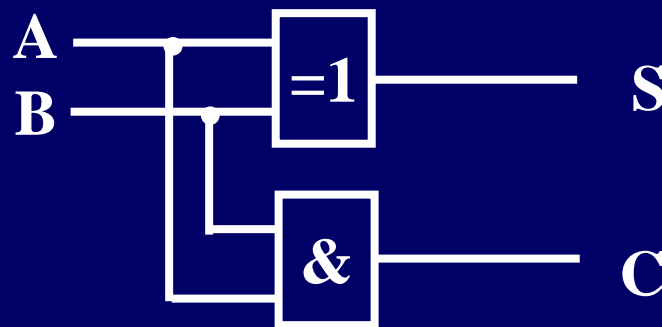
例1：半加器的设计。

半加器是能实现两个一位二进制数相加，求得和数及向高位进位的逻辑电路。

输入		输出	
加数A	加数B	和数S	进位C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

半加器真值表

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



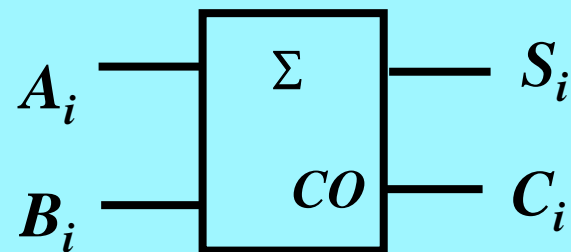
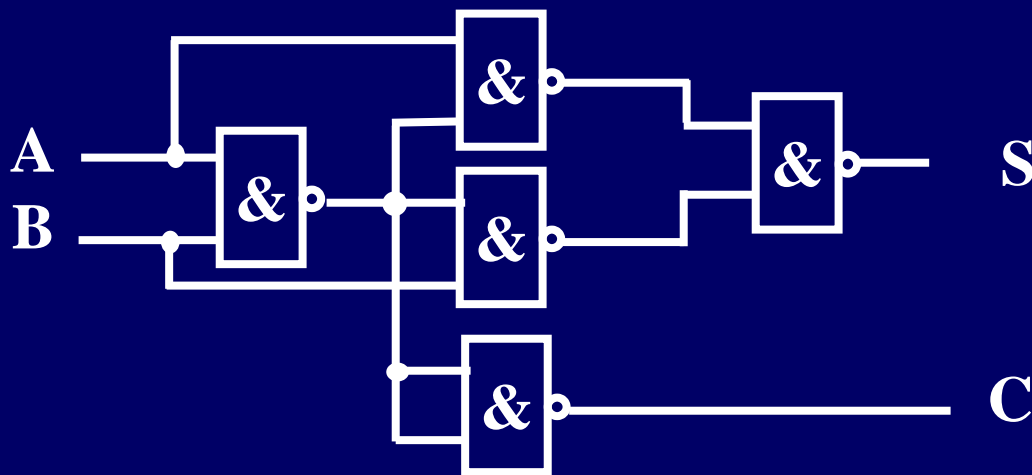
如果用与非门实现

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

$$= \overline{\overline{A}B \bullet \overline{A\overline{B}}} = \overline{\overline{A}BB \bullet A\overline{A}B}$$

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

替代尾因子，消除反变量

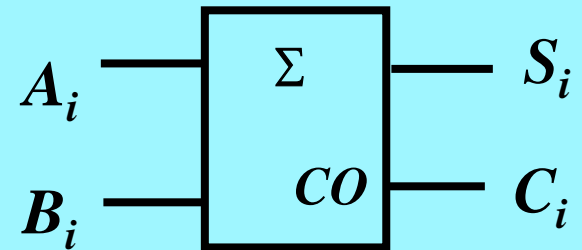
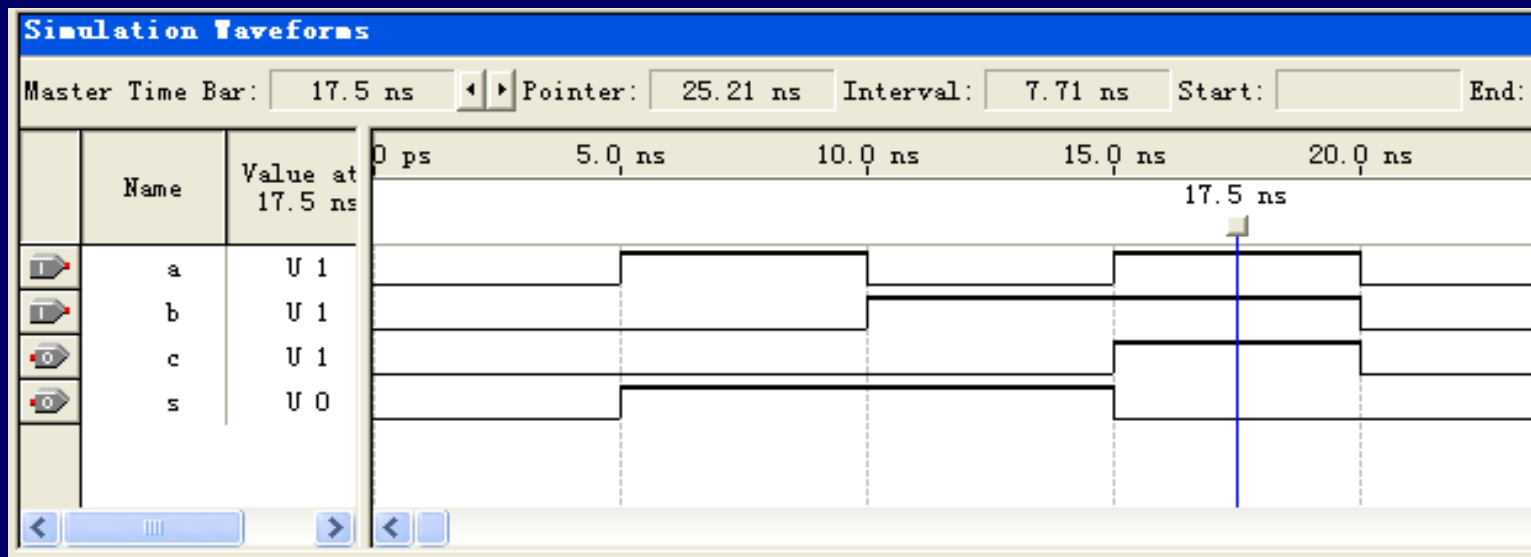


半加器逻辑符号

## 半加器的Verilog HDL描述

```
module subadd(a,b,s,c);  
    input a,b;  
    output s,c;  
    assign {c,s}=a+b;  
endmodule
```

## 仿真验证



半加器逻辑符号

其他描述方式?



## 例2：全加器的设计。

全加器是能够实现三个一位二进制数相加（两个本位加数，一个低位向本位的进位），求得和数及向高位进位的逻辑电路。

解： 设

输入端  $A_i$ ——被加数；

$B_i$ ——加数；

$C_{i-1}$ ——低位向本位的进位；

输出端  $S_i$ ——本位和；

$C_i$ ——本位向高位的进位。

输入			输出	
$A_i$	$B_i$	$C_{i-1}$	$S_i$	$C_i$
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

全加器真值表

		$A_i B_i$			
		00	01	11	10
$C_{i-1}$	0		1		1
	1	1		1	

$S_i$

		$A_i B_i$			
		00	01	11	10
$C_{i-1}$	0			1	
	1		1	1	1

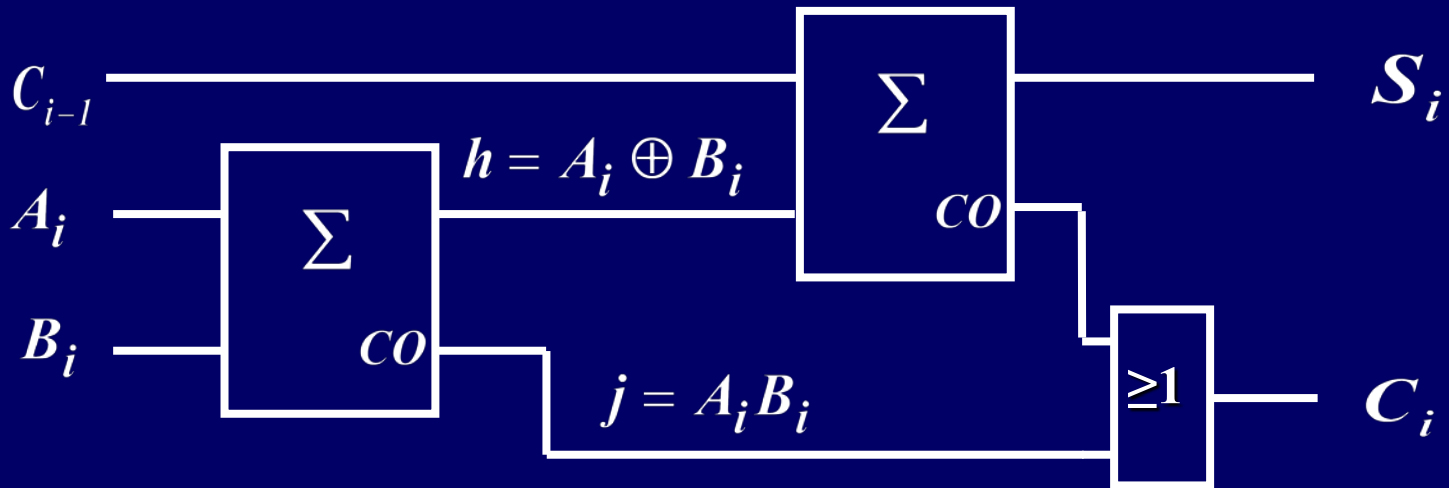
$C_i$

最简与或式

$$S_i = \overline{A_i} \overline{B_i} C_{i-1} + \overline{A_i} B_i \overline{C_{i-1}} + A_i \overline{B_i} \overline{C_{i-1}} + A_i B_i C_{i-1}$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

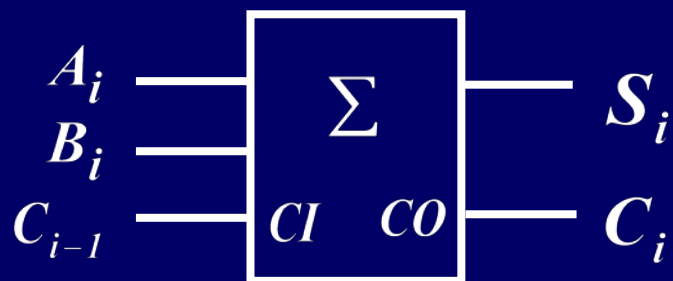
## 用两个半加器



写出逻辑表达式:

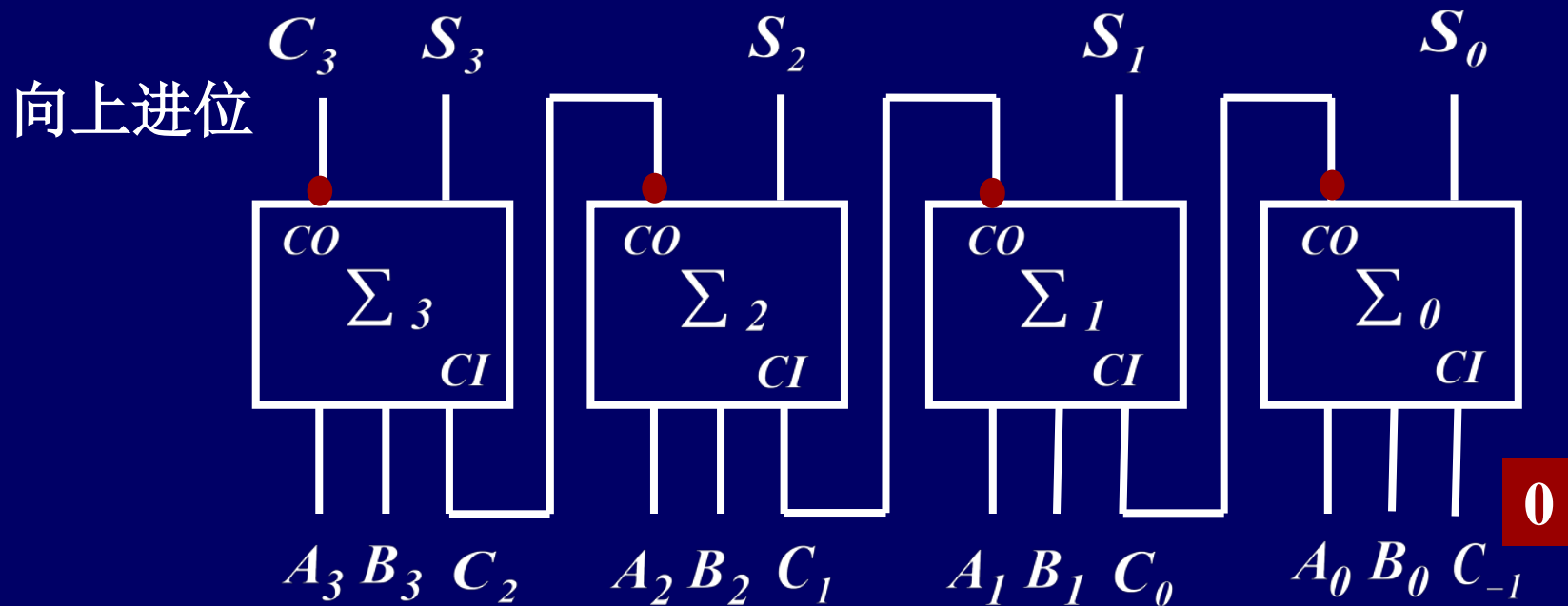
$$S_i = A_i \oplus B_i \oplus C_{i-1} \quad (S_i = \overline{\overline{h}hC_{i-1}} \bullet \overline{\overline{C_{i-1}}hC_{i-1}})$$

$$\begin{aligned} C_i &= A_i B_i + (A_i \oplus B_i) C_{i-1} \\ &= A_i B_i + A_i \overline{B_i} C_{i-1} + \overline{A_i} B_i C_{i-1} \\ &= A_i B_i + A_i C_{i-1} + B_i C_{i-1} \end{aligned}$$



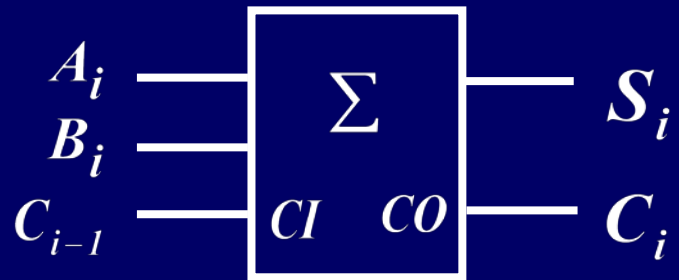
全加器的逻辑符号

用全加器构成四位二进制数加法器：



## 全加器的Verilog HDL描述

```
module add(a,b,cin,s,cout);  
    input a,b,cin;  
    output s,cout;  
    assign {cout,s}=a+b+cin;  
endmodule
```



不用拼接，该如何？

例3：代码转换电路的设计。试设计8421码转换成余3码的逻辑电路。

8421码	余3码
$B_3B_2B_1B_0$	$Y_3Y_2Y_1Y_0$
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	d d d d
1 0 1 1	d d d d
1 1 0 0	d d d d
1 1 0 1	d d d d
1 1 1 0	d d d d
1 1 1 1	d d d d

解：设输入的8421码为 $B_3B_2B_1B_0$ ，  
输出的余3码为 $Y_3Y_2Y_1Y_0$ 。

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00			<i>d</i>	<i>1</i>
01		<i>1</i>	<i>d</i>	<i>1</i>
11		<i>1</i>	<i>d</i>	<i>d</i>
10		<i>1</i>	<i>d</i>	<i>d</i>

$$Y_3 = B_3 + B_2B_0 + B_2B_1$$

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00		<i>1</i>	<i>d</i>	
01	<i>1</i>		<i>d</i>	<i>1</i>
11	<i>1</i>		<i>d</i>	<i>d</i>
10	<i>1</i>		<i>d</i>	<i>d</i>

$$Y_2 = \overline{B_2}B_0 + \overline{B_2}B_1 + B_2\overline{B_1}\overline{B_0}$$

8421码	余3码
$B_3B_2B_1B_0$	$Y_3Y_2Y_1Y_0$
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	d d d d
1 0 1 1	d d d d
1 1 0 0	d d d d
1 1 0 1	d d d d
1 1 1 0	d d d d
1 1 1 1	d d d d

$B_3B_2$		00	01	11	10
$B_1B_0$	00	1	1	d	1
	01			d	
	11	1	1	d	d
	10			d	d

$$Y_1 = B_1B_0 + \overline{B_1}\overline{B_0}$$

$B_3B_2$		00	01	11	10
$B_1B_0$	00	1	1	d	1
	01			d	
	11			d	d
	10	1	1	d	d

$$Y_0 = \overline{B_0}$$

# 建立8421码转换成余3码的Verilog HDL行为模型。

8421码	余3码
B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub>
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	d d d d
1 0 1 1	d d d d
1 1 0 0	d d d d
1 1 0 1	d d d d
1 1 1 0	d d d d
1 1 1 1	d d d d

思路：余3码 = 8421码 + 0011  
控制输入伪码

```

module ch8421_yu3(Bin,Yout);
  input  [3:0] Bin;
  output [3:0] Yout;
  reg     [3:0] Yout;
  always@(Bin)
    if (Bin<=4'b1001)
      Yout=Bin+4'b0011;
    else Yout=0;
endmodule
    
```

用case语句实现？



## 二. 逻辑问题描述——简化真值表——逻辑表达式

当输入变量较多时，可根据实际需求，仅列出输出函数为1或输出函数为0的简化真值表。

例：已知  $X=x_2x_1$  和  $Y=y_2y_1$  是两个二进制正整数，设计一个X与Y的比较电路。

解：该电路有四个输入端，三个输出端。



设：

当  $x_2x_1 > y_2y_1$  时， $F_1=1$ ；否则 $F_1=0$ 。

当  $x_2x_1 < y_2y_1$  时， $F_2=1$ ；否则 $F_2=0$ 。

当  $x_2x_1 = y_2y_1$  时， $F_3=1$ ；否则 $F_3=0$ 。

根据先比较高位，再比较低位的原则，可仅列出输出函数为1的简化真值表。

简化真值表

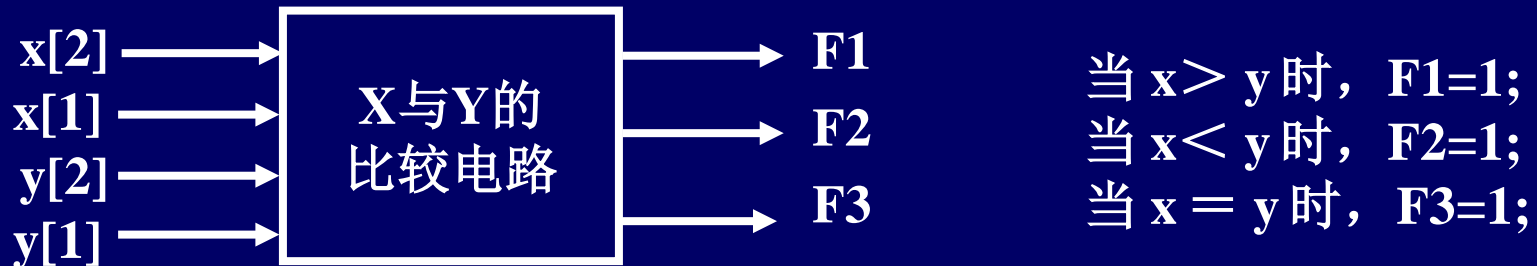
$x_2$	$y_2$	$x_1$	$y_1$	$F_1$	$F_2$	$F_3$
1	0	d	d	1	0	0
0	1	d	d	0	1	0
0	0	1	0	1	0	0
		0	1	0	1	0
		0	0	0	0	1
		1	1	0	0	1
1	1	1	0	1	0	0
		0	1	0	1	0
		0	0	0	0	1
		1	1	0	0	1

$$F_1 = \overline{x_2} \overline{y_2} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1}$$

$$F_2 = \overline{x_2} \overline{y_2} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1}$$

$$F_3 = \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1} + \overline{x_2} \overline{y_2} \overline{x_1} \overline{y_1}$$

## 建立该比较电路的Verilog HDL模型



//行为描述, 比较器, 有参数定义

```
module compare_1 (F1, F2, F3, x, y);
```

```
    parameter      size = 2;
```

```
    input  [size:1] x, y;
```

```
    output  F1, F2, F3;
```

```
    reg     F1, F2, F3;
```

```
    always @ (x or y)
```

```
        if (x>y) begin F1=1 ; F2=0 ; F3=0 ; end
```

```
        else if (x==y) begin F1=0 ; F2=0 ; F3=1 ; end
```

```
        else if (x<y) begin F1=0 ; F2=1 ; F3=0 ; end
```

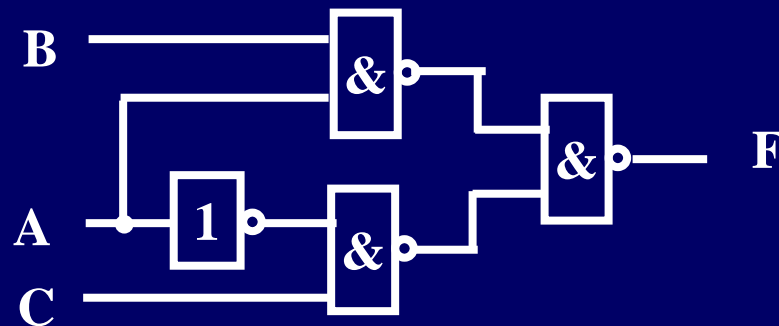
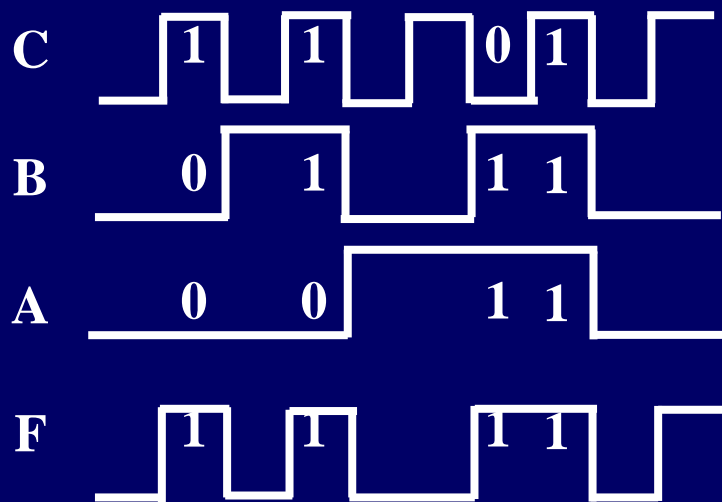
```
endmodule
```

修改, 任意位宽,  
体现优势!

### 三. 逻辑问题描述——逻辑表达式

有时，可由逻辑问题的描述直接写出表达式。

例：测得某电路输出F与输入A、B、C的波形关系如下，试写出逻辑表达式并画出与非门实现的逻辑图。



$$\begin{aligned} F(A, B, C) &= m_1 + m_3 + m_6 + m_7 \\ &= \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}\overline{C} + ABC \\ &= \overline{A}C + AB = \overline{\overline{A}C} \cdot \overline{AB} \end{aligned}$$

# 基于Verilog HDL的组合电路行为建模方法

- 1.门级建模
- 2.数据流描述
- 3.行为描述

重点

**module** 模块名 （端口列表）；

端口定义

**input** 输入端口

**output** 输出端口

数据类型说明

**reg**

**parameter**

逻辑功能定义

**always** @(敏感事件列表)

**begin**

阻塞、非阻塞、

**if-else**、**case**、**for**语句

**end**

**endmodule**

## 用always过程语句描述组合逻辑功能——要点

- always过程块监视敏感事件列表中的每一个事件，只要有一个事件发生，立即对输出进行更新。
- 敏感事件列表中不应包含posedge和negedge关键词，因为组合电路的输出是由输入电平决定的。
- 组合逻辑中所有影响输出的输入信号（变量）都要列入敏感事件列表，保证组合电路的输入输出关系在每个时刻都严格成立。
- always过程块中所有被赋值信号（变量）都应定义为寄存器型。
- 应使用阻塞赋值语句。

## 作业8:

4.1

4.3

4.4

4.9

4.11 (1)

4.12