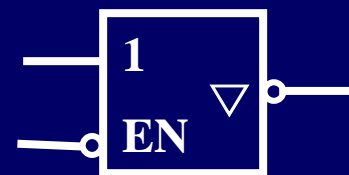
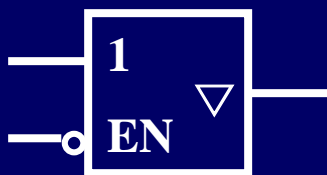
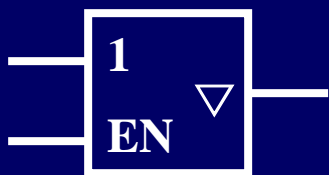


## 5 三态缓冲器

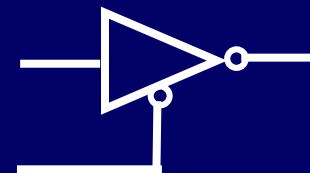
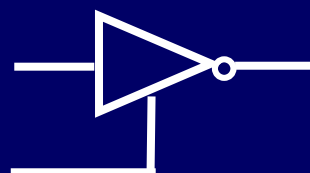
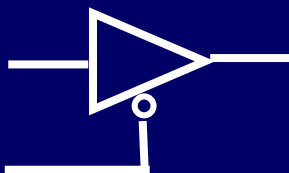
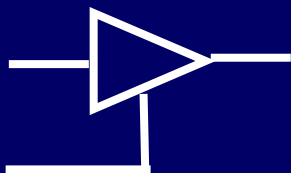
三态缓冲器（Three – state Buffer）又称为三态门、三态驱动器，其三态输出受使能输入端的控制，当使能输入有效时，器件实现正常逻辑状态输出（逻辑0、逻辑1）；当使能输入无效时，输出处于高阻状态，即等效于与所连接的电路断开。

### 三态缓冲器的逻辑符号

#### 矩形符号



#### 变形符号



高有效使能  
原码输出

低有效使能  
原码输出

高有效使能  
反码输出

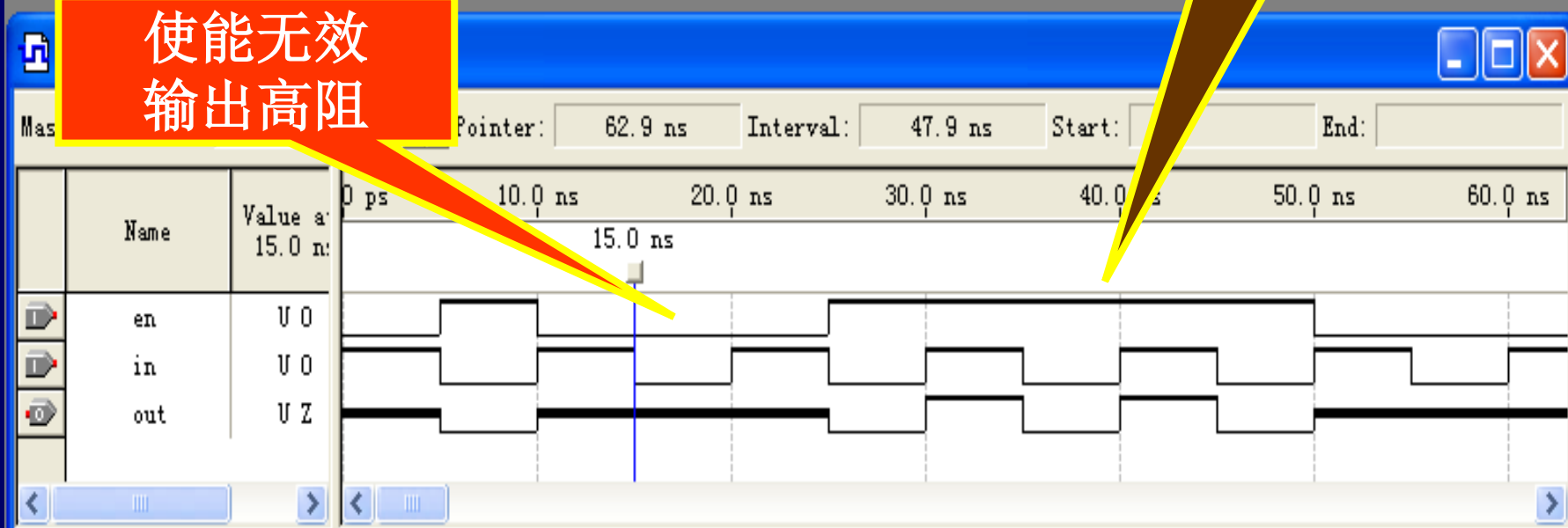
低有效使能  
反码输出

## 三态缓冲器的Verilog HDL模型及仿真

```
abc three_t.v
1 module three_t(in,en,out);
2   input in;
3   input en;
4   output out;
5   assign out=(en==1)? in : 1'bz;
6 endmodule
7
```

使能有效  
正常逻辑输出

使能无效  
输出高阻

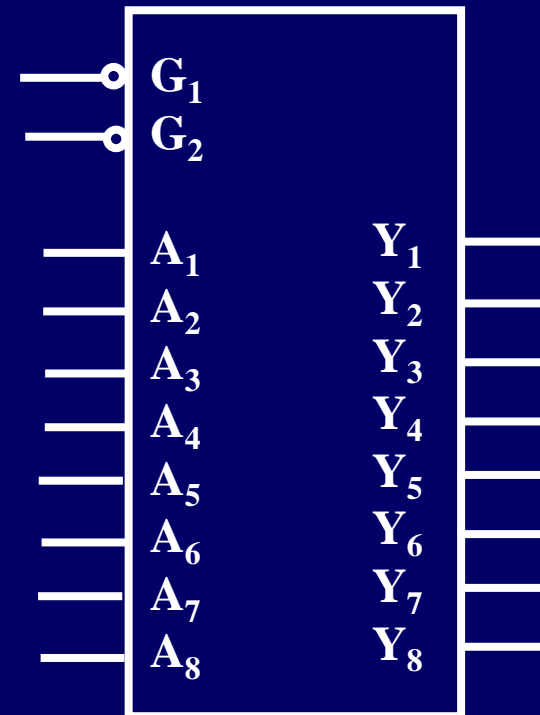
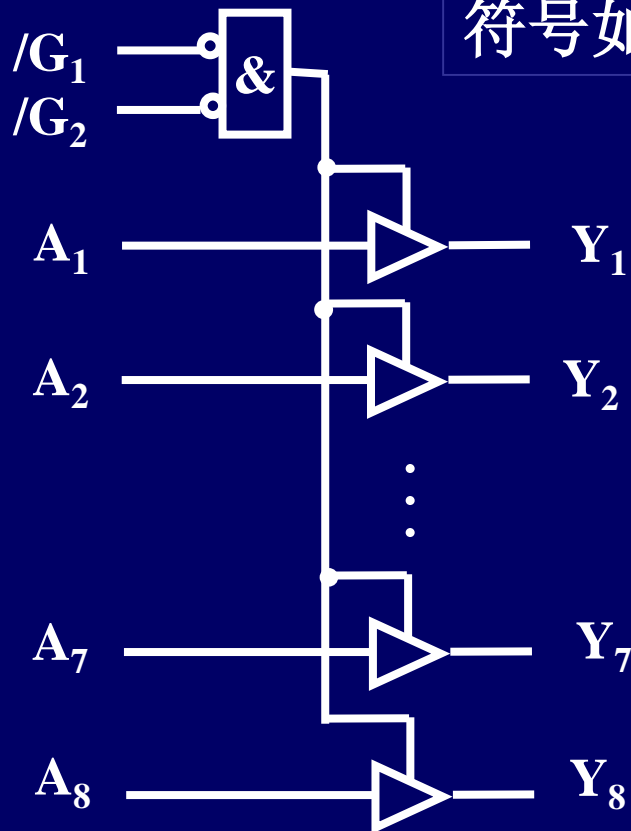


三态缓冲器常用于多个数据源  
共享一根（组）公用线（总线）



当译码器的所有使能端有效时，SS2~SS0的组合使/SELP~/SELW在同一时刻只有一个有效，从而使8个数据源P~W中的一个驱动SDATA；当使能端无效时，则没有一个三态门被使能，输出均为高阻态。

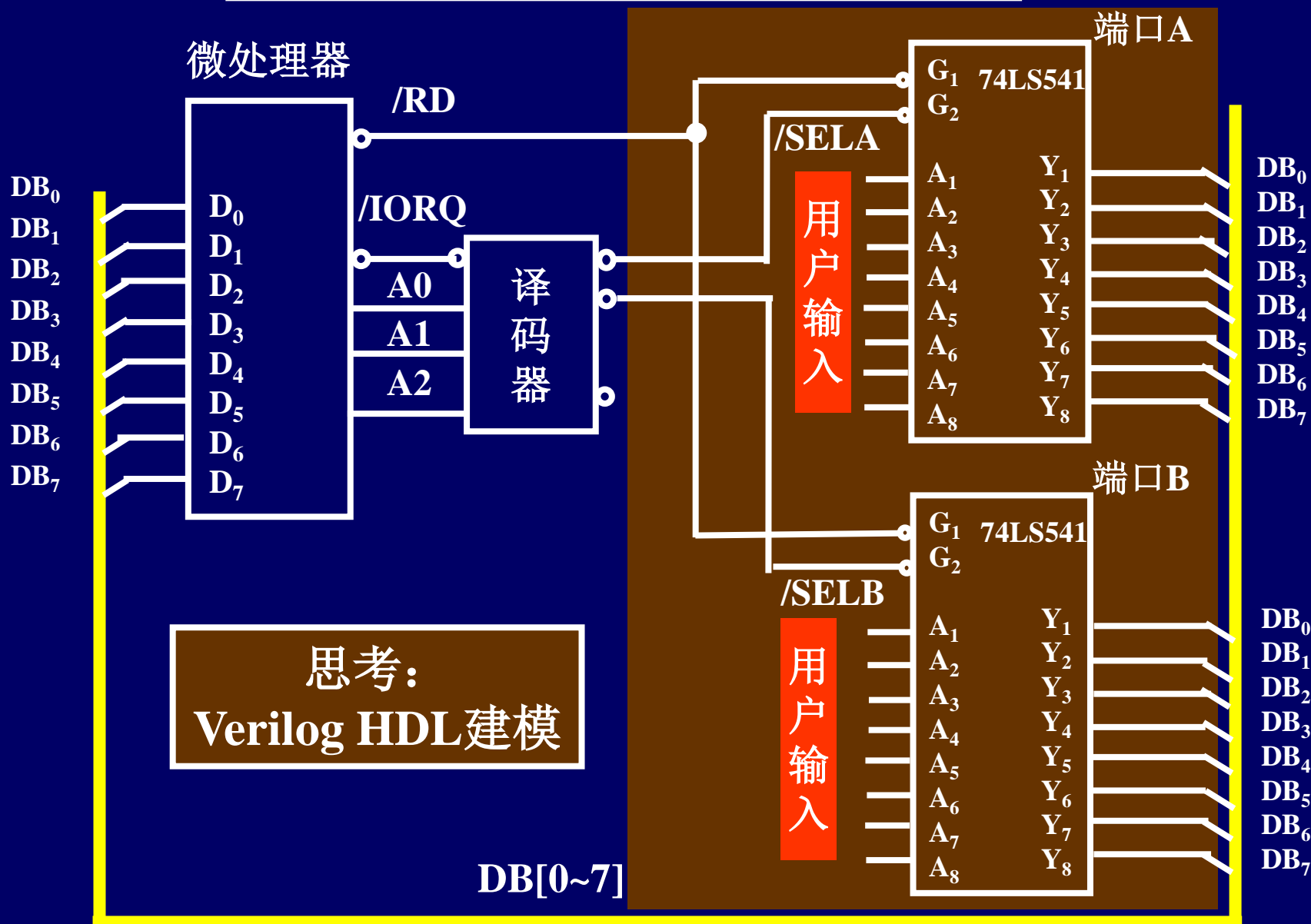
MSI器件74LS541中包含8个独立的三态门，并共用两个使能输入，逻辑图及逻辑符号如下：



74LS541逻辑符号

思考：Verilog HDL建模？

## 74LS541 作为输入端口的应用举例



## 6 奇偶校验电路 (Parity Checking Circuits)

### 一. 奇偶校验码

奇偶校验码 = 数据位 + 奇偶校验位



n 位二进制代码

1位代码

奇偶校验位的取值原则：

采用奇校验时，使整个代码组中“1”的个数为奇数。  
采用偶校验时，使整个代码组中“1”的个数为偶数。

## 二. 奇偶校验电路

奇偶校验的基本运算是异或运算。

设有 $n$  个输入变量  $X_1, X_2, \dots, X_n$

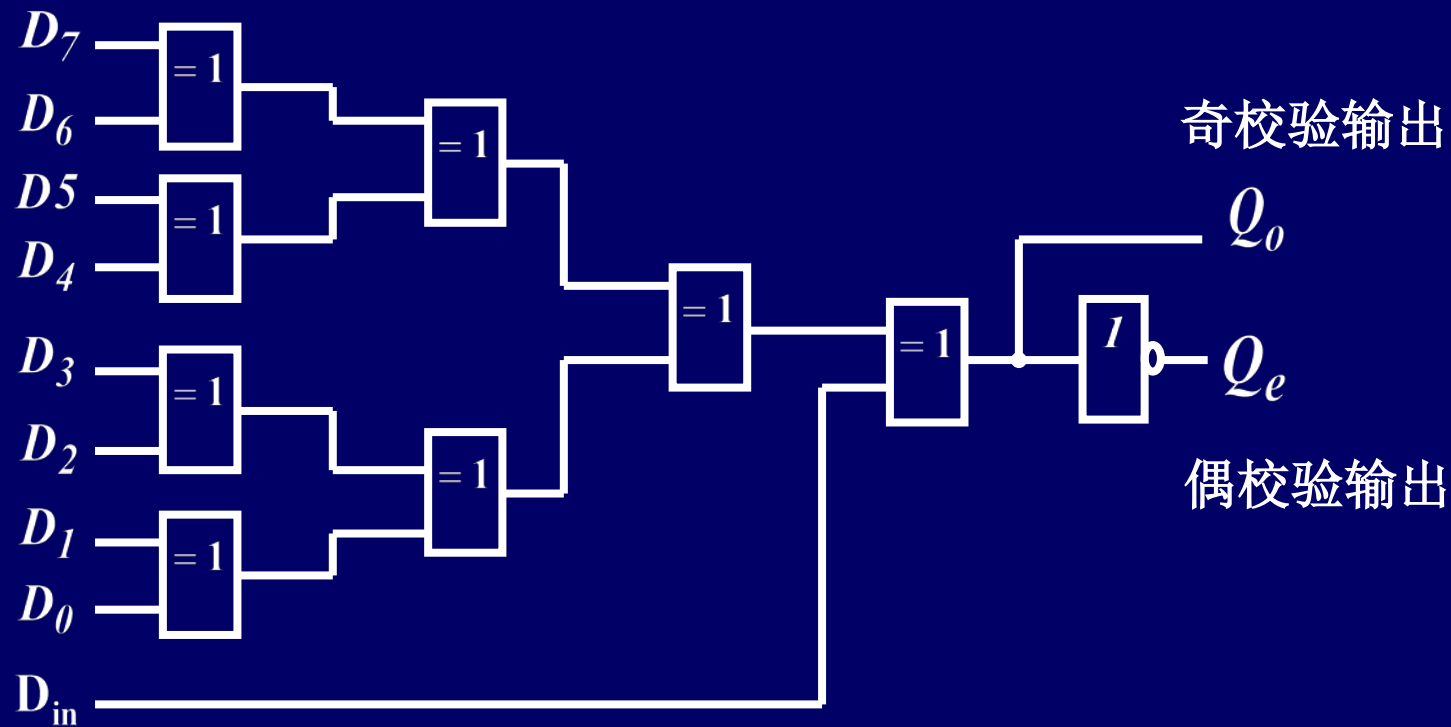
则函数  $F = X_1 \oplus X_2 \oplus \dots \oplus X_n$  的逻辑功能为：

当输入变量为1 的个数是奇数时， $F$  为1 ；

当输入变量为1 的个数是偶数时， $F$  为0 。

实现这一功能的电路称为奇校验电路；输出端加一个非门，则可得偶校验电路。通常合二为一，称为奇偶校验电路。

分析下列奇偶校验电路如何用作 8位数据的奇偶校验位发生器

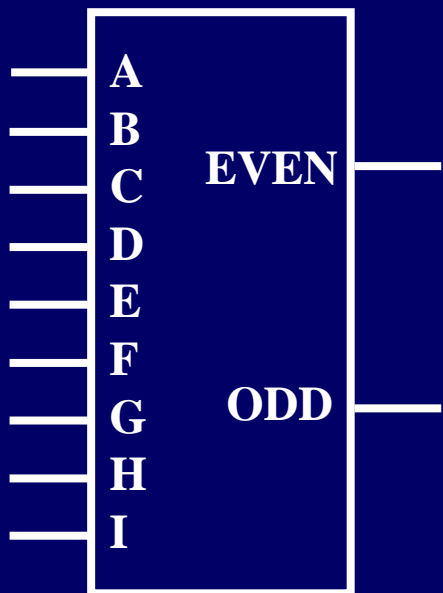


$D_{in}=0$ ,  $Q_e$ 用作奇校验位发生器;  $Q_0$ 用作偶校验位发生器。

$D_{in}=1$ ,  $Q_e$ 用作偶校验位发生器;  $Q_0$ 用作奇校验位发生器。



### 三. 奇偶校验电路的Verilog HDL模型



逻辑符号

功能表

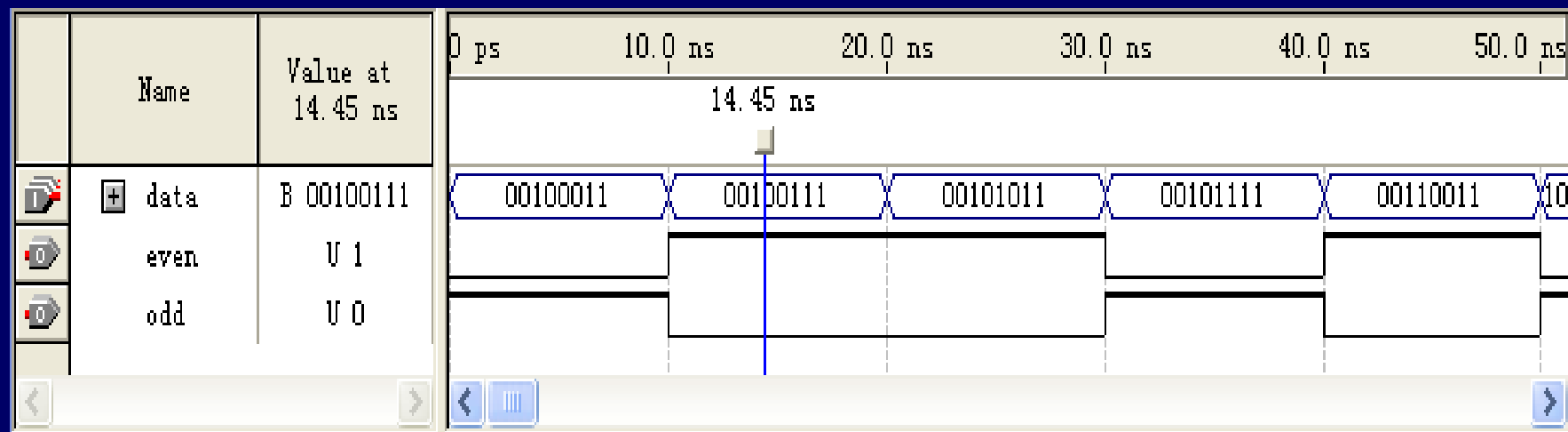
9 位数据中 “1”的个数	EVEN	ODD
偶数	1	0
奇数	0	1

对应74LS280逻辑器件

abc odd\_even.v

```
1 module odd_even(data,odd,even);  
2     input [8:1] data;  
3     output odd,even;  
4     assign odd ^= data;  
5     assign even = ~odd;  
6 endmodule  
7
```

按位异或归约运算



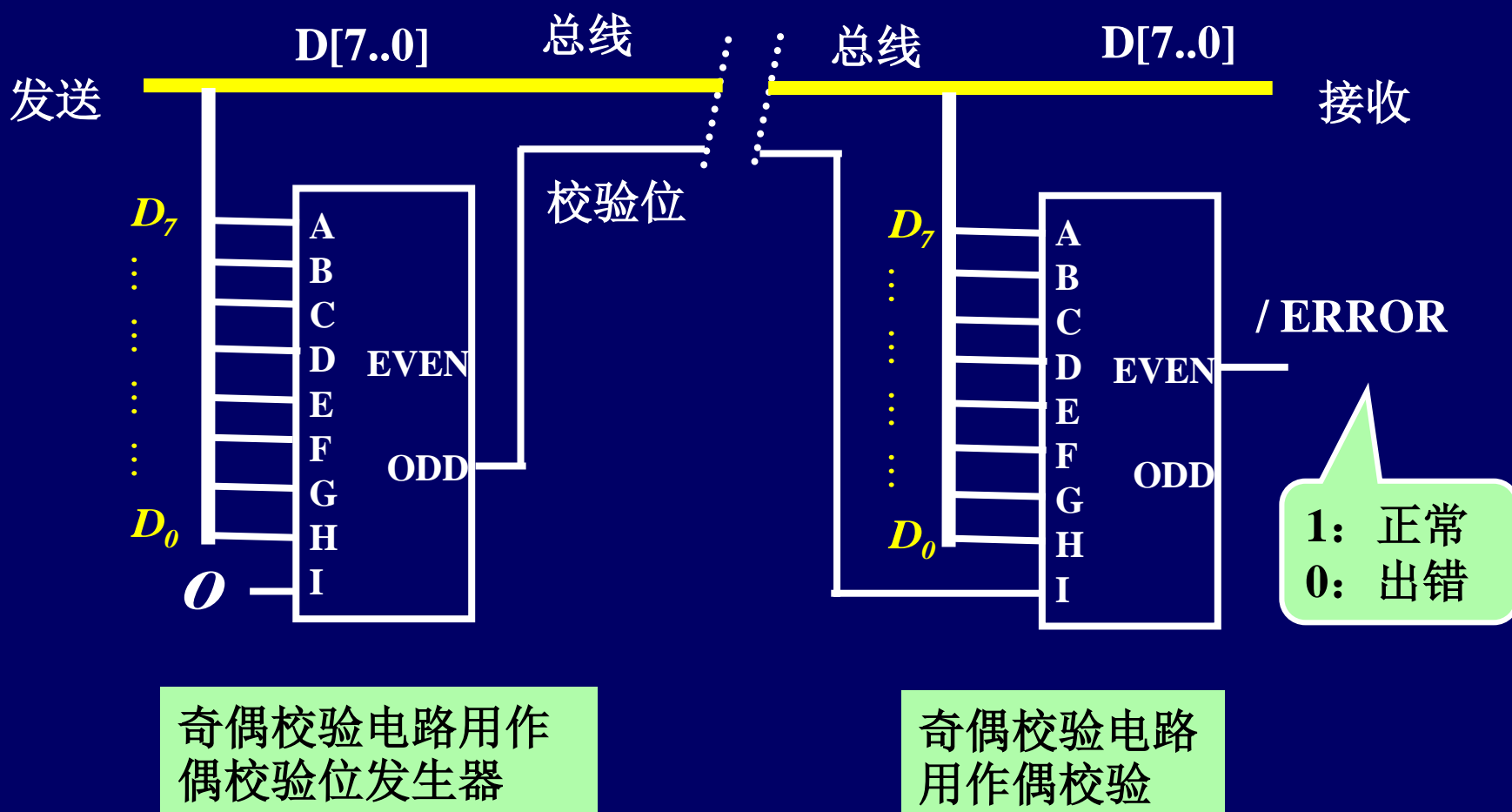
## 四. 奇偶校验电路的应用

奇偶校验电路有奇、偶校验两个输出标志，可用作发送端的奇偶校验位发生器；也可用作接收端的奇偶检验器，产生奇偶校验和。

若约定为奇校验，一般采用偶校验电路产生发送端的校验位，而用奇校验电路在接收端检验数据的正确性；

若约定为偶校验，一般采用奇校验电路产生发送端的校验位，而用偶校验电路在接收端检验数据的正确性。

## 偶校验应用示意

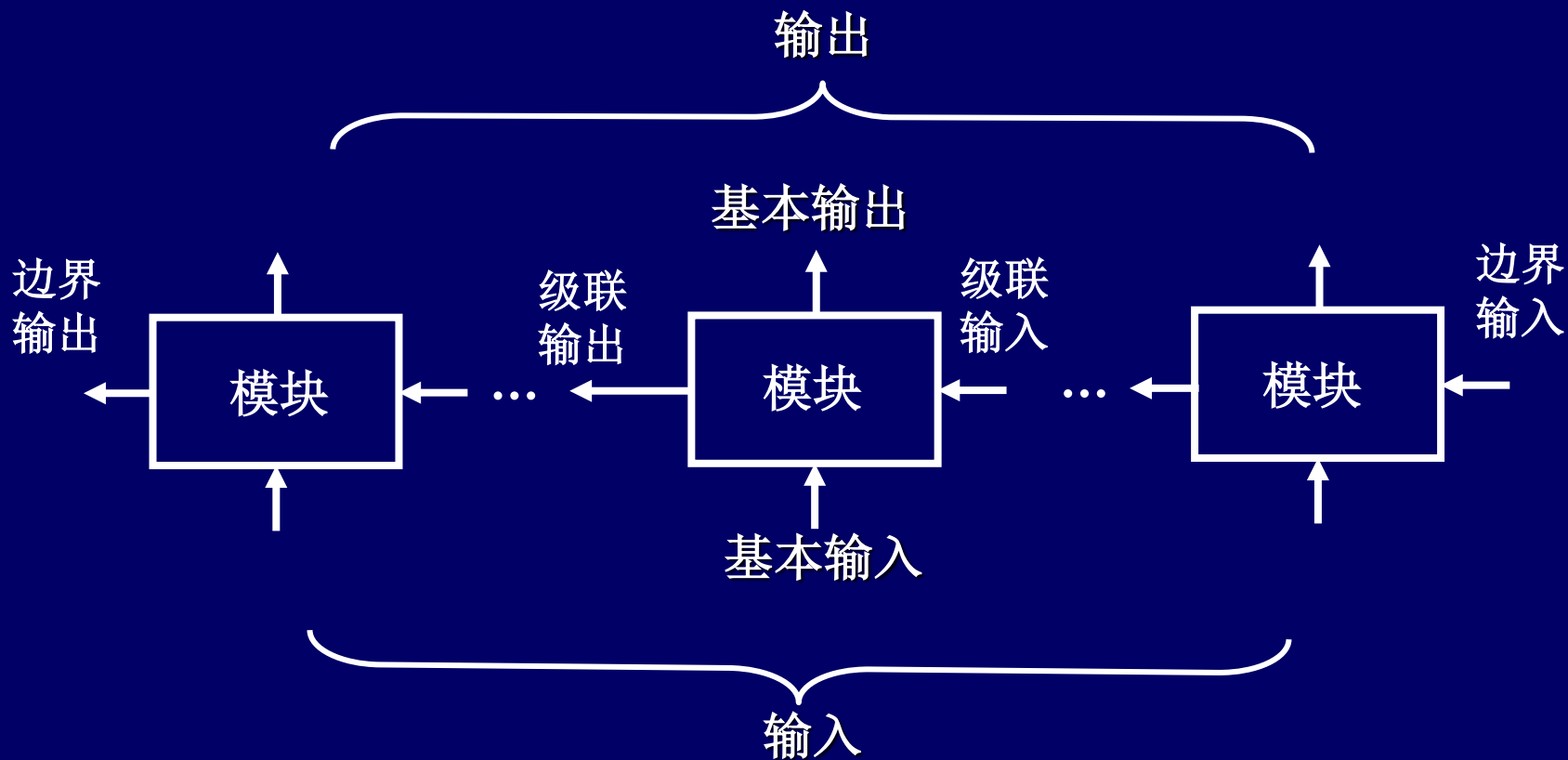


## 7 比较器 (Comparators)

比较器是对两个位数相同的二进制整数进行数值比较，并判断其大小关系的逻辑电路。

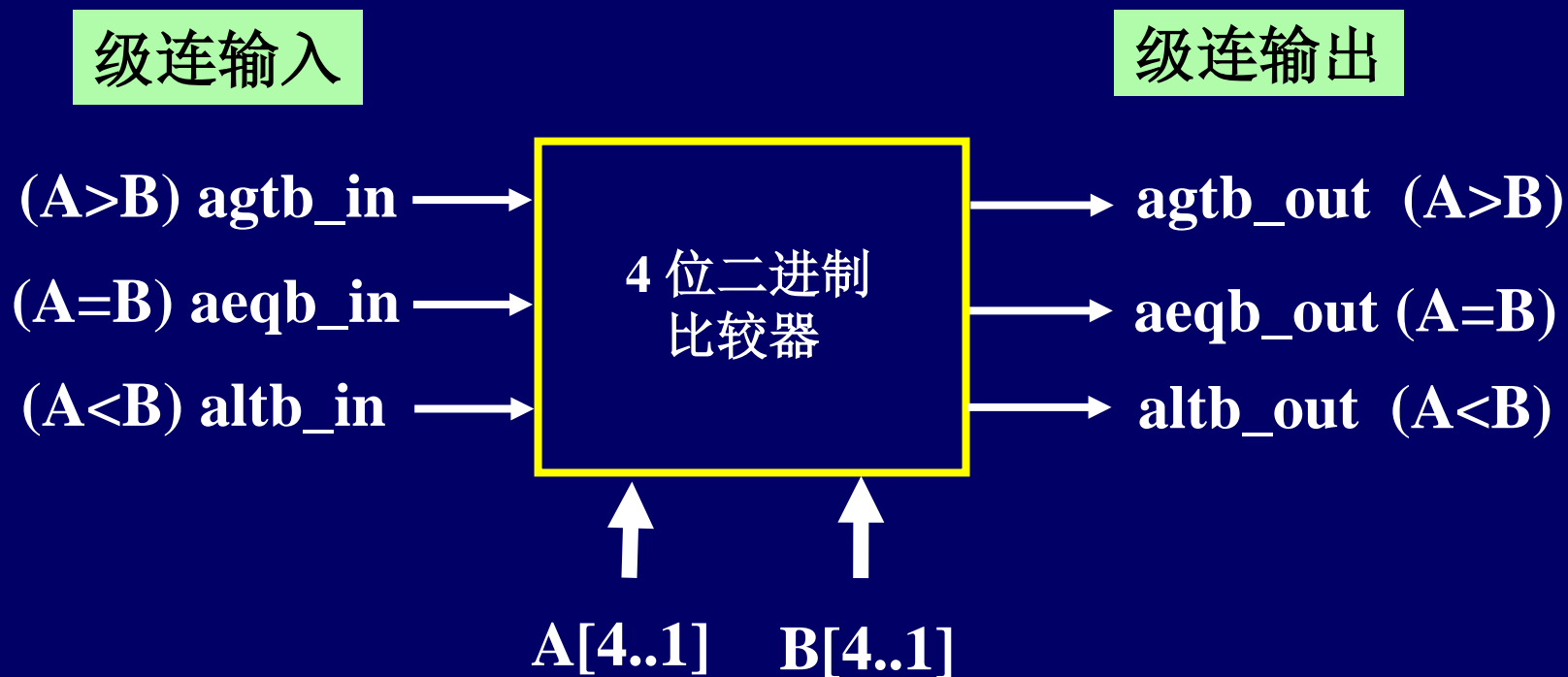
- 大小关系分为：大于 ( $>$ )  
等于 ( $=$ )  
小于 ( $<$ )
- 从高位开始比较，只有在高位相等情况下，才进行低位比较。
- 通常采用便于级连扩展的迭代设计方法（模块设计、重复电路）。

## 一. 迭代设计概述



迭代设计的基本模型

## 二. 4 位二进制比较器的Verilog HDL模型



满足迭代设计要求的功能框图

# 逻辑功能表

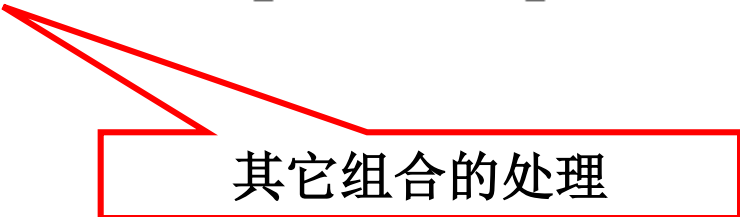
A[4..1]与 B[4..1]比 较结果	级连输入（低位比较结果）			级连输出（比较结果）		
	agtb_in altb_in	aeqb_in		agtb_out altb_out	aeqb_out	
A>B	×	×	×	1	0	0
A<B	×	×	×	0	0	1
A=B	1	0	0	1	0	0
	0	1	0	0	1	0
	0	0	1	0	0	1



## Verilog HDL 描述

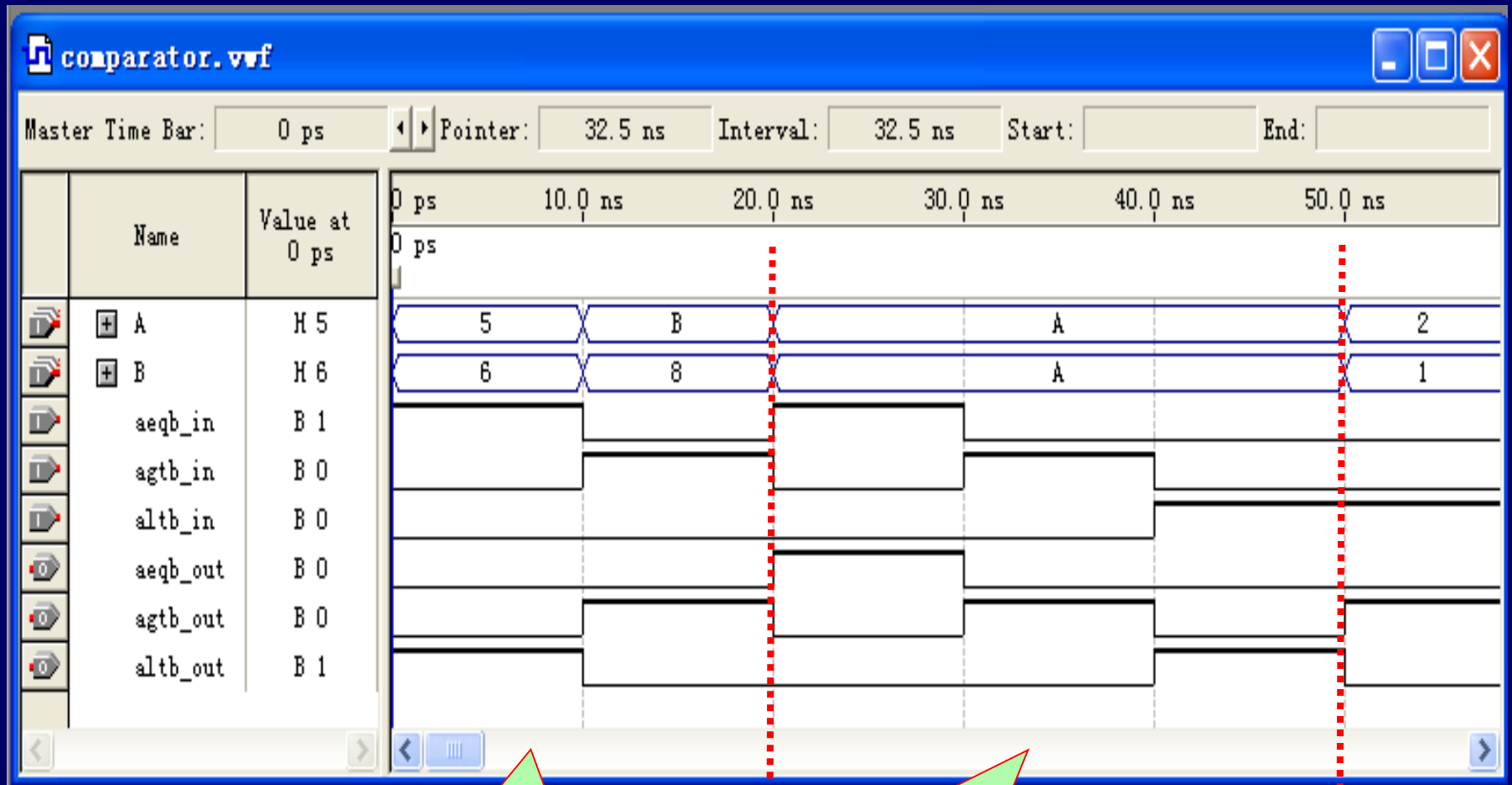
comparator.v\*

```
1 module comparator(A,B,
2                     agtb_in,altb_in,aeqb_in,
3                     agtb_out,altb_out,aeqb_out);
4     input [4:1] A,B;
5     input      agtb_in,altb_in,aeqb_in;
6     output     agtb_out,altb_out,aeqb_out;
7     reg        agtb_out,altb_out,aeqb_out;
8     always @ (A or B or agtb_in or altb_in or aeqb_in)
9         begin
10             if (A>B) begin agtb_out=1'b1;altb_out=1'b0;aeqb_out=1'b0;end
11             else if (A<B) begin agtb_out=1'b0;altb_out=1'b1;aeqb_out=1'b0;end
12             else if (A==B)
13                 case ({agtb_in,altb_in,aeqb_in})
14                     3'b100: begin agtb_out=1'b1;altb_out=1'b0;aeqb_out=1'b0;end
15                     3'b010: begin agtb_out=1'b0;altb_out=1'b1;aeqb_out=1'b0;end
16                     3'b001: begin agtb_out=1'b0;altb_out=1'b0;aeqb_out=1'b1;end
17                     default : begin agtb_out=1'b0;altb_out=1'b0;aeqb_out=1'b1;end
18                 endcase
19         end
20 endmodule
21
```



其它组合的处理

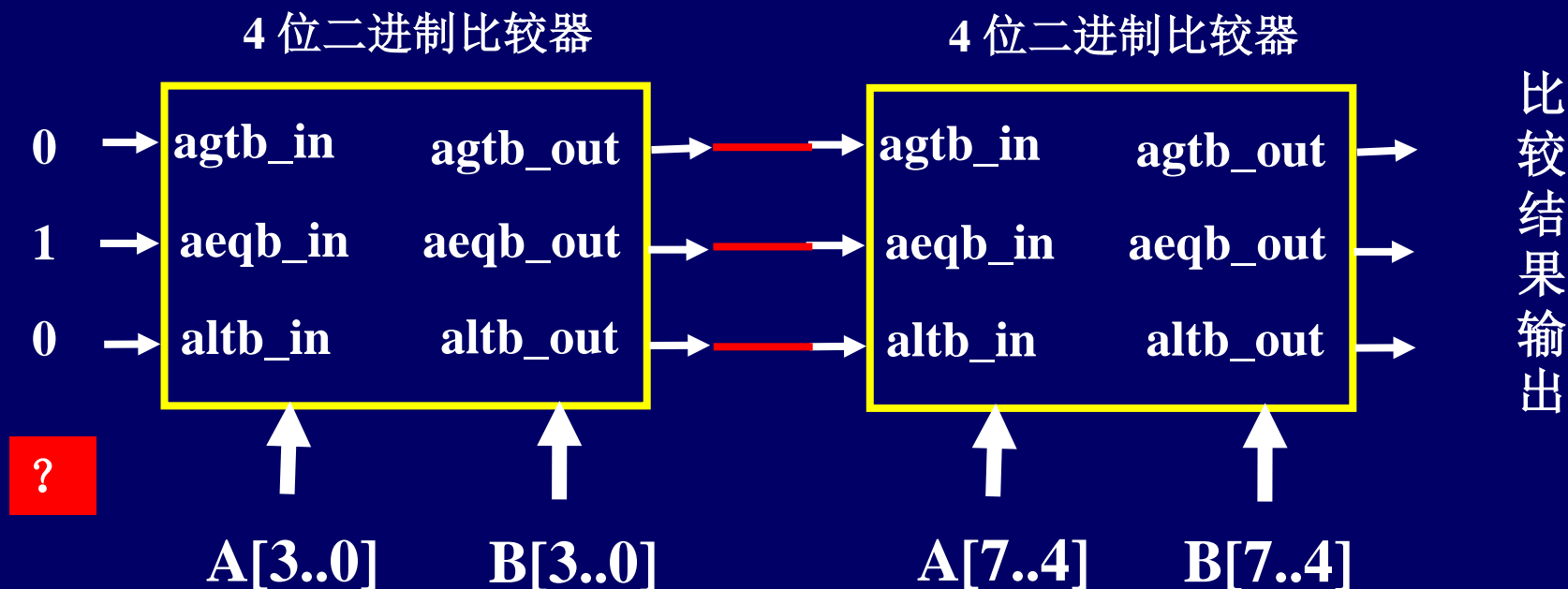
# 仿真波形



本地已比较出结果

本地相等，传递低位比较结果

### 三. 用4位二进制比较器构成8位二进制比较器

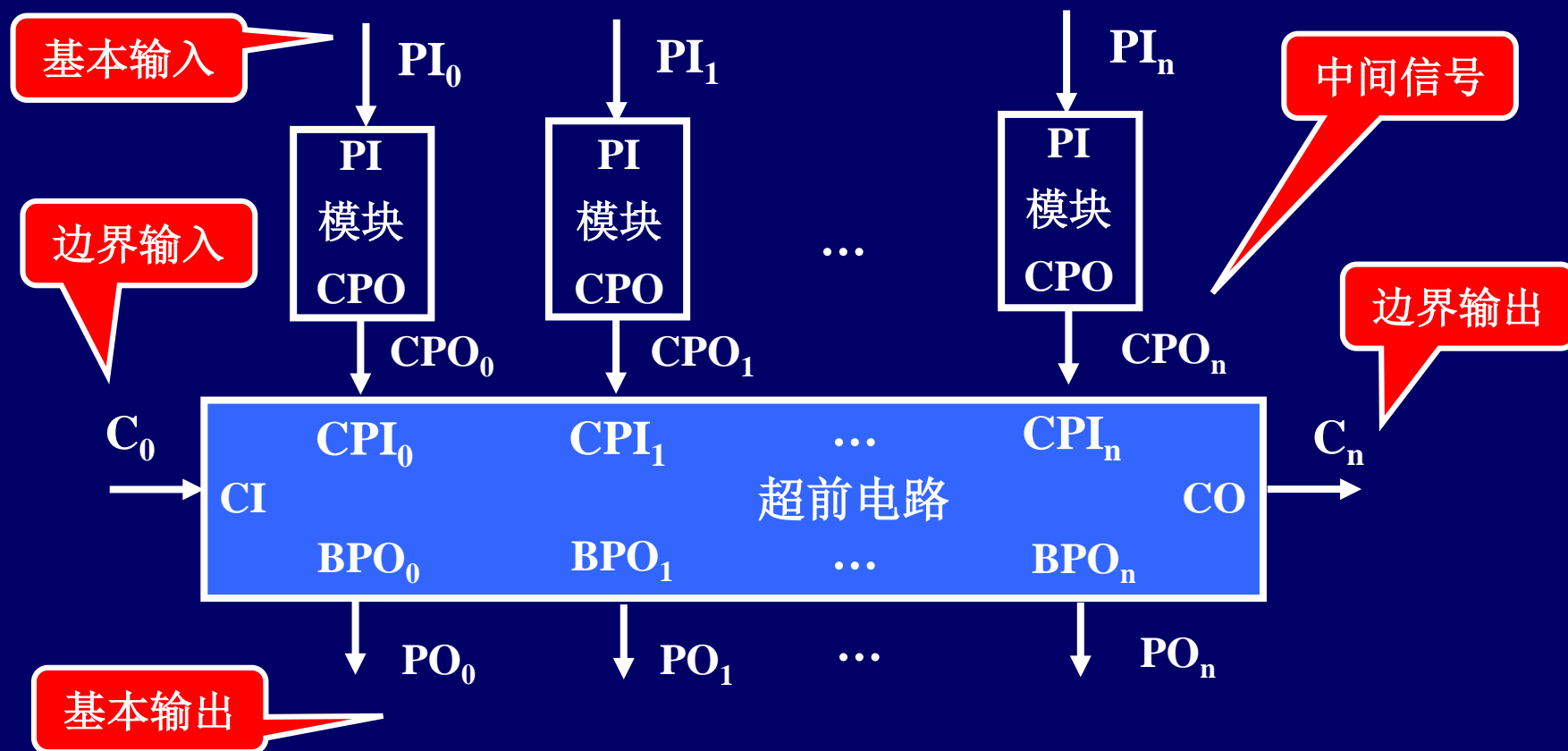


也可通过修改Verilog HDL描述模型，完成设计。

当迭代级数增加时，引发电路响应速度问题。

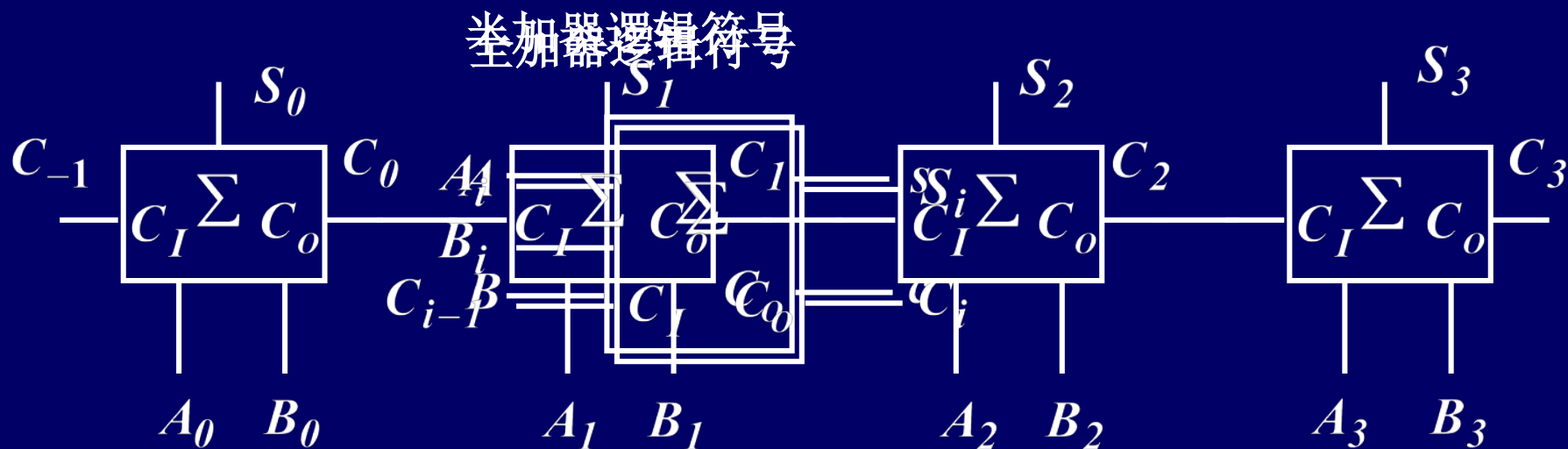
## 四. 超前电路设计模型

在串行迭代比较电路中，从高到低逐位比较，其速度随位数增加而降低。为了提高速度，减少级联信号通过的门的数量，可采用超前电路，即每个模块不产生级联信号，只产生超前电路需要的中间信号，超前电路对这些中间信号同时处理，产生输出结果。



## 8 加法器及其应用

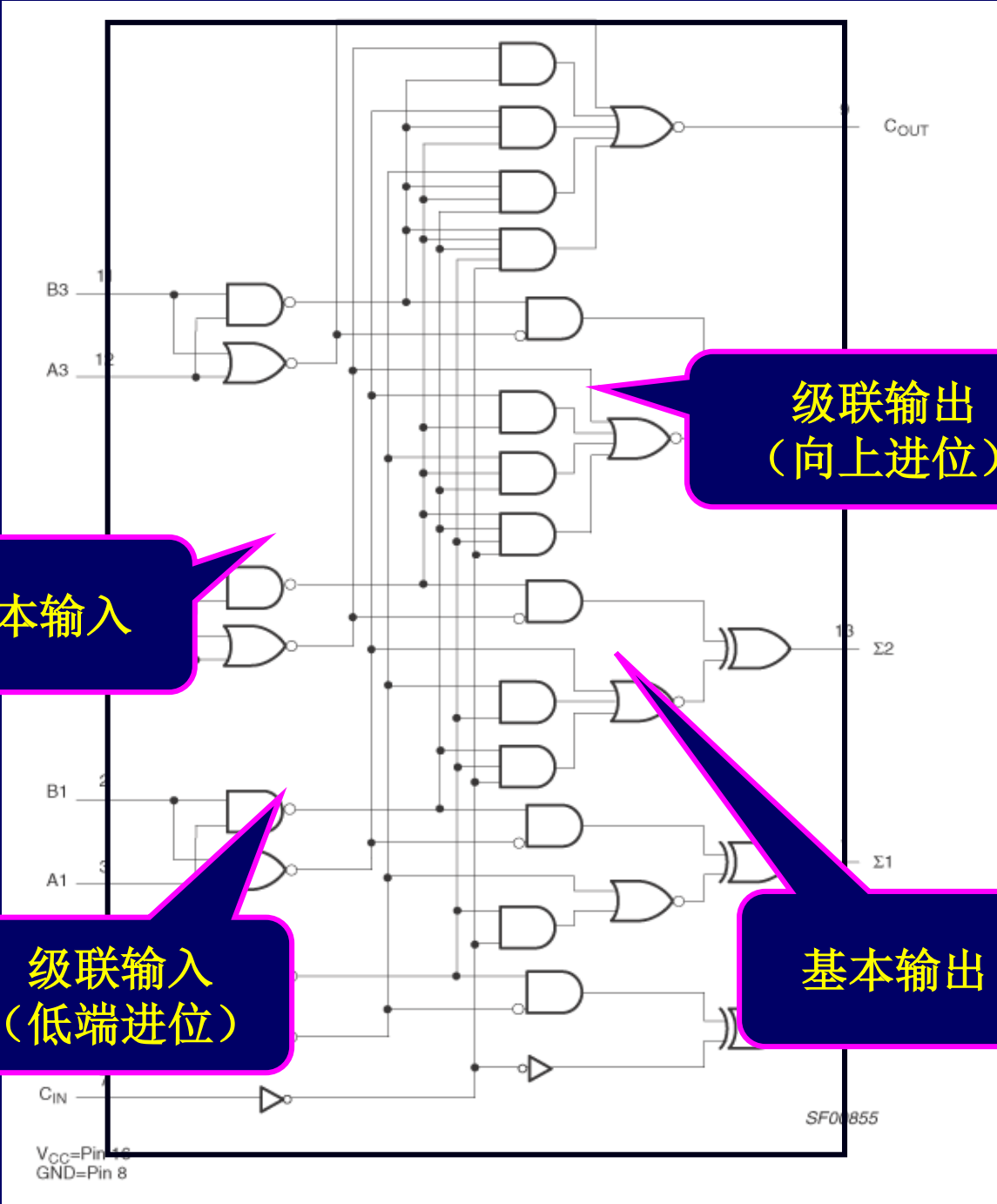
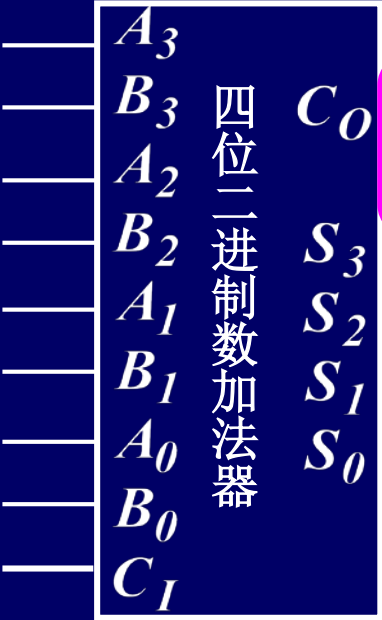
- 半加器 (half-adder):
- 全加器 (full-adder):
- 串行进位 (行波 travelling wave) 的并行加法器。
- 先行进位加法器



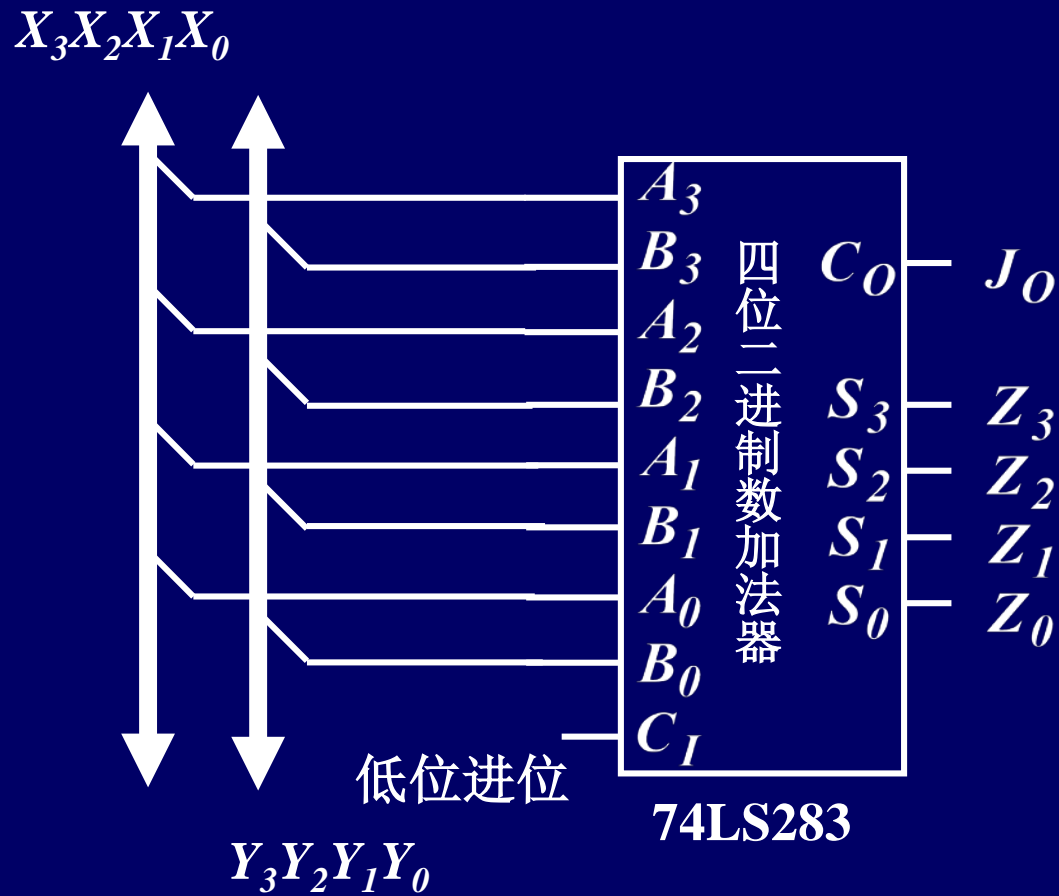
## 超前进位加法器

## 74LS283逻辑图

## 逻辑符号



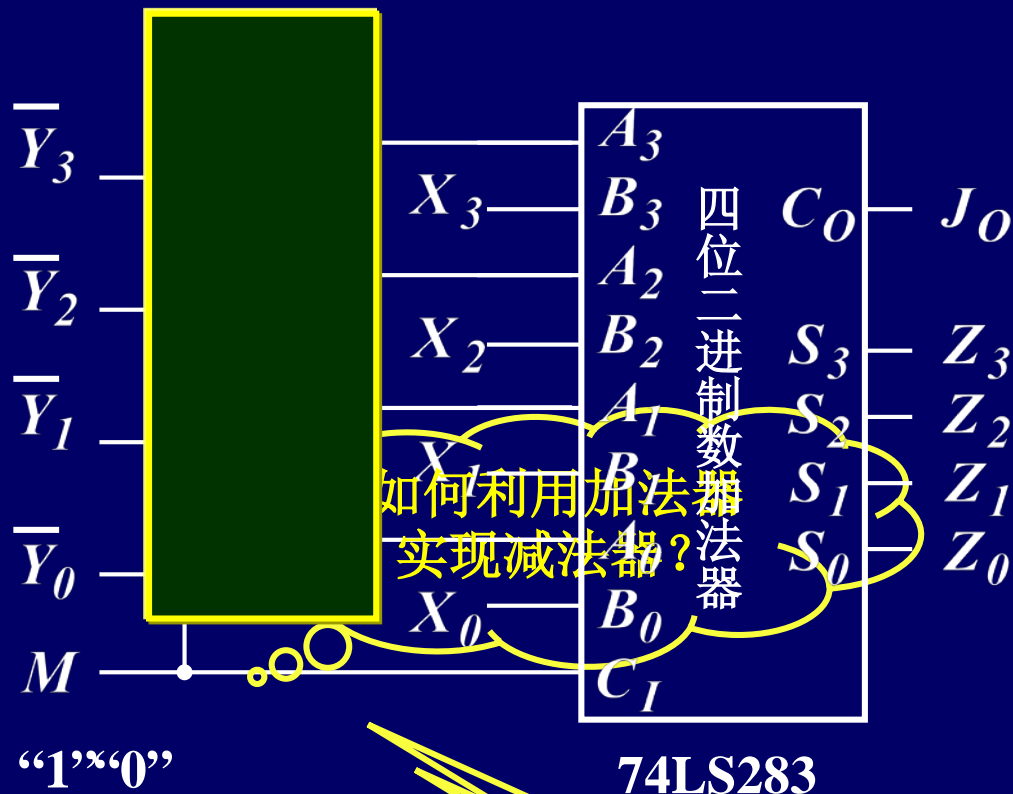
## 四位二进制数加法运算



# 四位二进制数补码加法器

$$[A + B]_{\text{补}} = [A]_{\text{补}} + [B]_{\text{补}}$$

$$[A - B]_{\text{补}} = [A]_{\text{补}} + [\bar{B}]_{\text{补}}$$



M=0: 加法运算

M=1: 减法运算

加法运算  
减法运算



1) 余3码转换成8421码。

解：真值表：

$Y_3Y_2Y_1Y_0$	$F_8F_4F_2F_1$
0011	0000
0100	0001
0101	0010
0110	0011
0111	0100
1000	0101
1001	0110
1010	0111
1011	1000
1100	1001

$$\text{已知: } F_8F_4F_2F_1 = Y_3Y_2Y_1Y_0 - 0011$$

$$= Y_3Y_2Y_1Y_0 + 1101 \pmod{10000}$$

$$[-0011]_{\text{补}} = 1101$$

## 四位加法器设计BCD代码转换电路



解毕。

# 四位加法器设计BCD代码转换电路

2) 2421码转换成余3码。

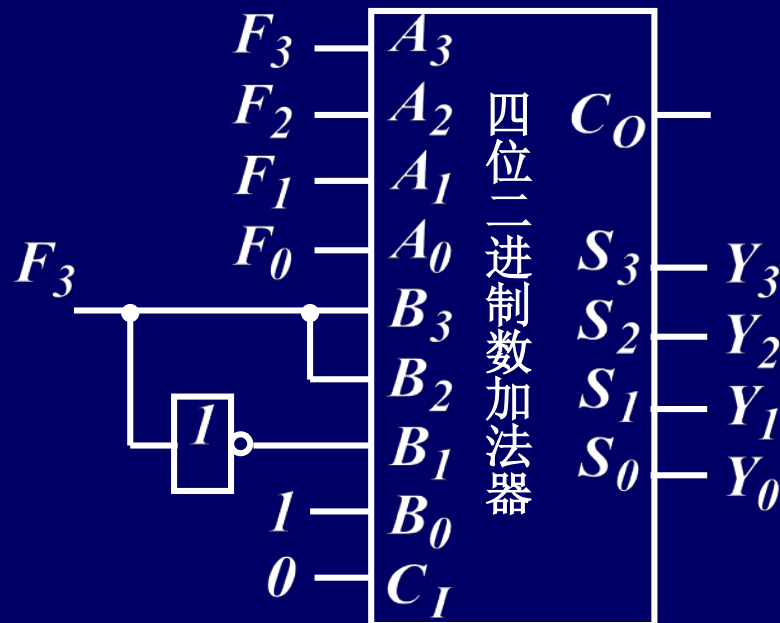
解：真值表如下：

	$F_3F_2F_1F_0$	$Y_3Y_2Y_1Y_0$
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	1011	1000
6	1100	1001
7	1101	1010
8	1110	1011
9	1111	1100

逻辑条件：

$$\begin{array}{rcl}
 F_3 = 0 : & Y = F & + \begin{array}{ccc} F_3 & F_3 & \overline{F_3} \end{array} \begin{array}{ccc} 1 & 1 & 1 \end{array} \\
 F_3 = 1 : & Y = F & + \begin{array}{ccc} 1 & 1 & 0 \end{array} \begin{array}{ccc} 1 & 1 & 1 \end{array}
 \end{array}$$

逻辑电路图：



解毕。

## 两个8421码的加法

解：这个题目分两部分组成，一是四位二进制加法器的构成；二是8421码在加法运算中的加六修正判断。

加六修正命题 $F$ 成立的条件：当和大于9（1001）产生修正判断信号（ $Z=1$ ），  
或者有进位（ $C_o=1$ ）时对8421码的算术和  
进行加六修正。

和数	修正前		修正后	
	$C_o$	$S_3S_2S_1S_0$	$C_o$	$S_3S_2S_1S_0$
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	0000
11	0	1011	1	0001
12	0	1100	1	0010
13	0	1101	1	0011
14	0	1110	1	0100
15	0	1111	1	0101
16	1	0000	1	0110
17	1	0001	1	0111
18	1	0010	1	1000
19	1	0011	1	1001

修正判断信号Z：在8421码进行运算时，需要对运算结果超出1001的数码（1010...1111）进行修正工作，需要给出修正的判定信号。8421码的运算结果为 $S_3, S_2, S_1, S_0$ ，是修正判断信号Z的输入变量。

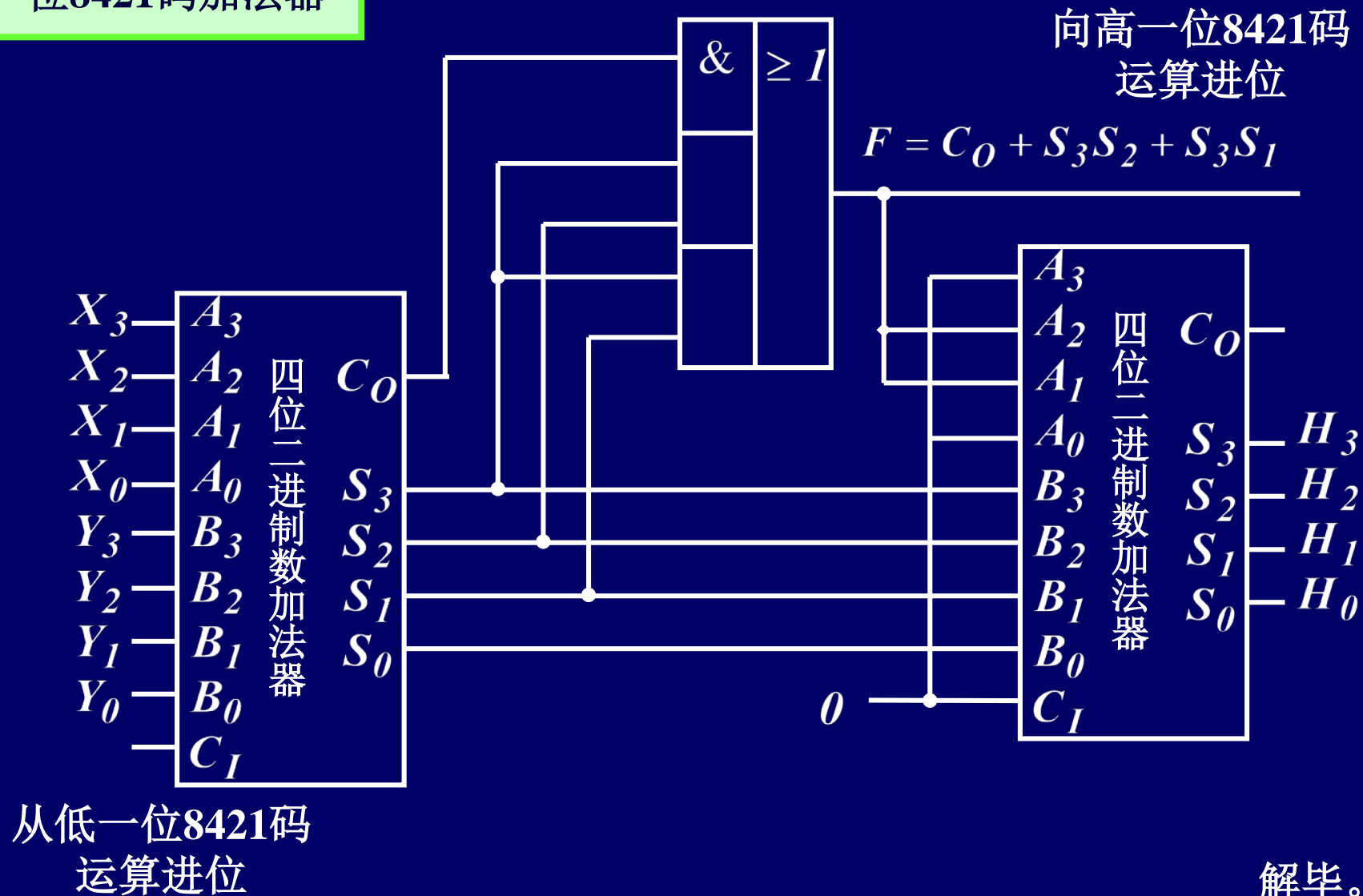
$$\begin{aligned}
 & Z(S_3, S_2, S_1, S_0) \\
 &= m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15} \\
 &= \sum m^4(10, 11, 12, 13, 14, 15)
 \end{aligned}$$

$S_3 S_2 \backslash S_1 S_0$		$S_3 S_2$			
		00	01	11	10
00	00	0	0	1	0
01	01	0	0	1	0
11	11	0	0	1	1
10	10	0	0	1	1

$$Z = S_3 S_2 + S_3 S_1$$

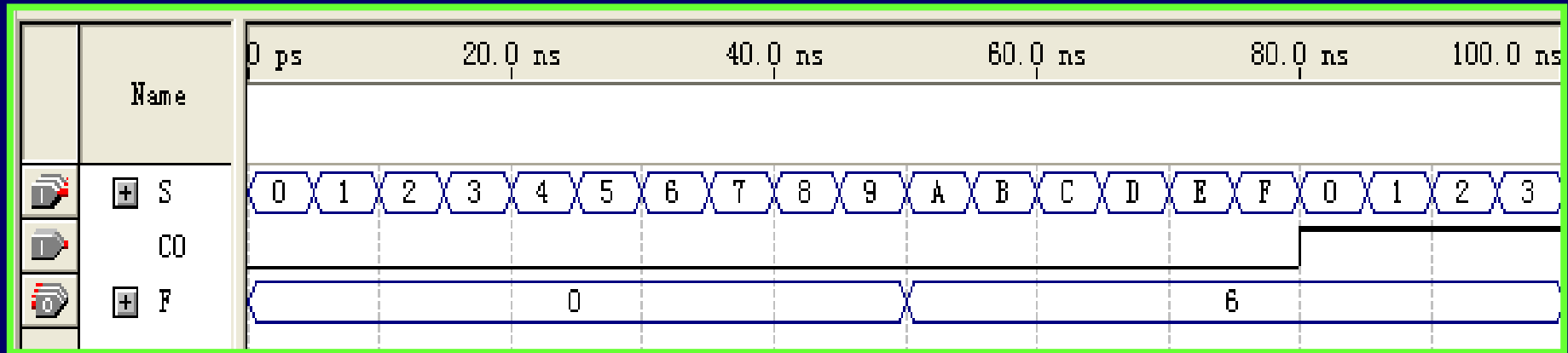
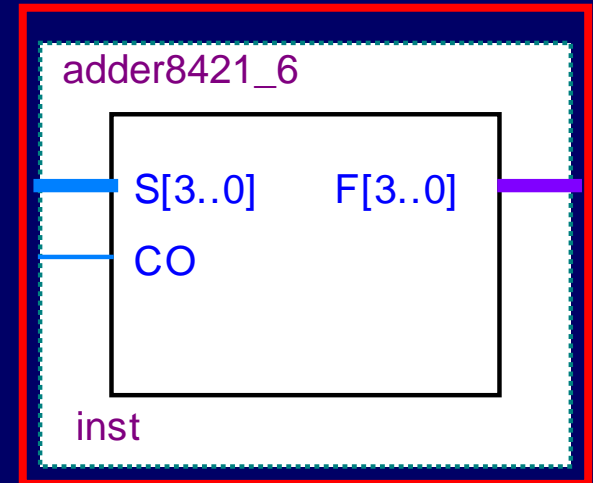
加六修正的条件是  $F = C_O + Z = C_O + S_3 S_2 + S_3 S_1$

# 一位8421码加法器



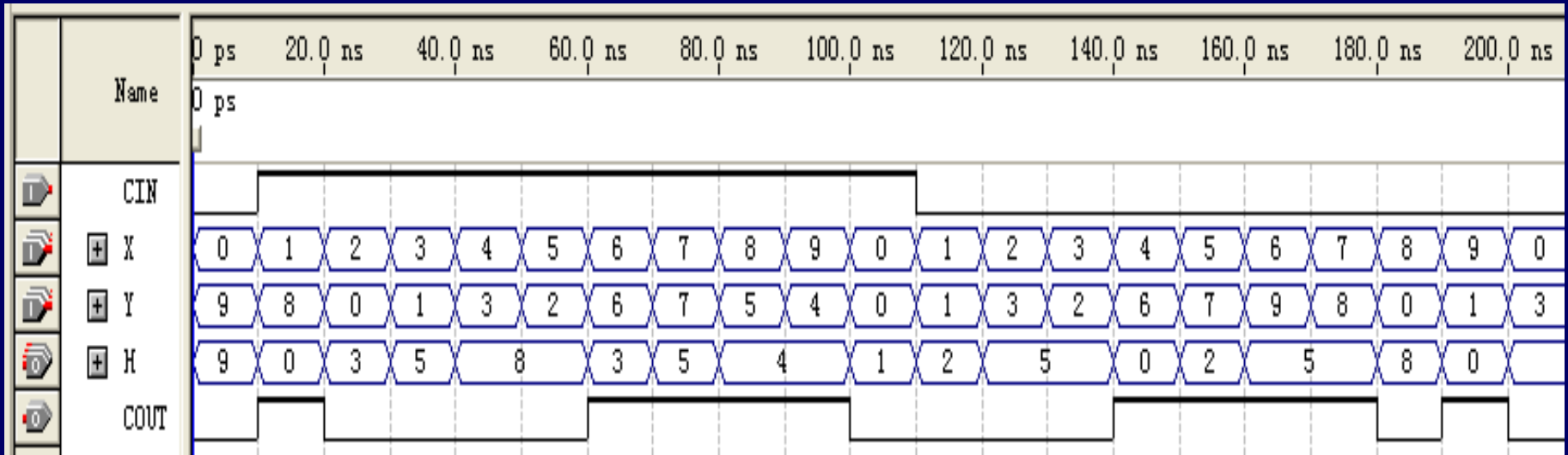
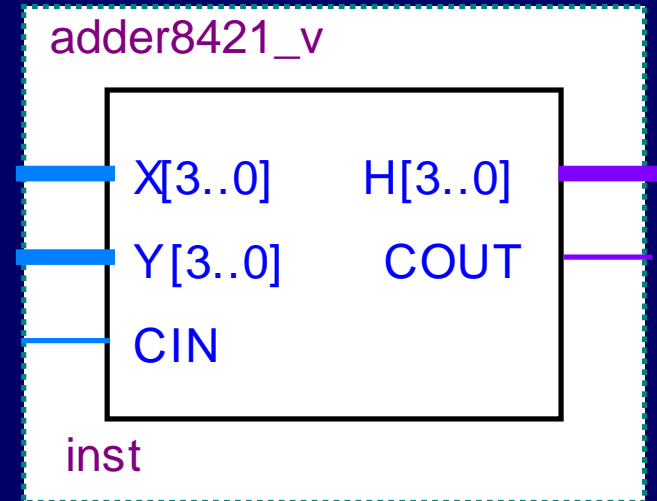
应用Verilog HDL编写加六修正的逻辑模块:

```
module adder8421_6(S,CO,F);  
    input[3:0]S;  
    input CO;  
    output [3:0]F;  
    assign F=CO/S[3]&S[2]/S[3]&S[1]?'b0110:0000;  
endmodule
```



## 应用Verilog HDL直接写出一位8421码加法器:

```
module adder8421_v(X,Y,CIN,H,COUT);  
  input[3:0]X,Y;  
  input CIN;  
  output[3:0]H;  
  output COUT;  
  assign H=X+Y+CIN>9?X+Y+CIN+6:X+Y+CIN;  
  assign COUT=X+Y+CIN>9?1:0;  
endmodule
```



# 组合电路的分析与设计小结及教学要求

## 基本概念（了解、理解）

组合电路的特点

逻辑门符号及等效符号

信号名及其有效级

引端有效级和等效变换（了解）

常用MSI逻辑器件的逻辑功能和逻辑符号

熟练掌握：

**74LS138、139、151、153、283**



# 组合电路的分析

## 基于逻辑门的组合电路的分析

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。(注意平时积累)

## 基于MSI逻辑器件的组合电路的分析

MSI逻辑器件 + 逻辑门附加电路

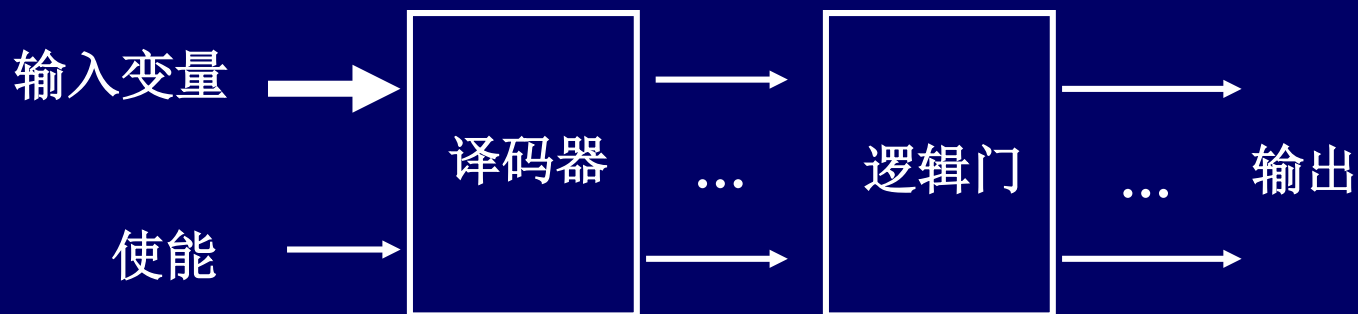
逻辑功能、逻辑符号、使能端、  
必要的输出表达式 .....

上述的分析方法

综合分析、列真值表、功能评述

- 基于加法器（74LS283）的电路分析  
（加减运算、代码转换）

- 基于译码器（74LS138、139）的电路分析



译码器实现逻辑函数的原理

译码器的每一个输出与输入变量构成的最小项存在对应关系。

逻辑函数都可展开为最小项形式。

## 参考分析步骤:

- 1) 分析译码器部分，求出每个译码器输出与输入变量的对应关系，即电路输出包含了哪些输入变量的最小项（最大项）。
- 2) 分析逻辑门电路部分，求得函数的表达式。
- 3) 列真值表或卡诺图。
- 4) 分析逻辑功能

## ●基于数据选择器的电路分析

例如：74LS151、74LS153

### 数据选择器实现逻辑函数的原理

数据选择器的输出表达式具有标准与或式的形式，其与项构成是  $m_i \cdot D_i$  ( $m_i$  为选择控制变量的最小项； $D_i$  为该最小项对应的数据输入端)，其逻辑功能是：在选择控制变量的控制下，将对应的数据输入端的逻辑状态传递到输出。

逻辑函数可以转换为标准与或式的形式。当确定了哪些函数变量作为选择控制变量后，其  $m_i$  也就确定了；其余变量的组合就对应于  $D_i$ 。

分析方法:

列真值表, 根据数据选择器的逻辑功能, 计算输出, 分析功能。

或

根据数据选择器的输出表达式, 直接写出函数表达式, 列真值表, 分析功能。

或

根据数据选择器的卡诺图, 直接得到函数的卡诺图, 分析功能。

# 组合电路的设计

## 基于逻辑门的组合电路的设计

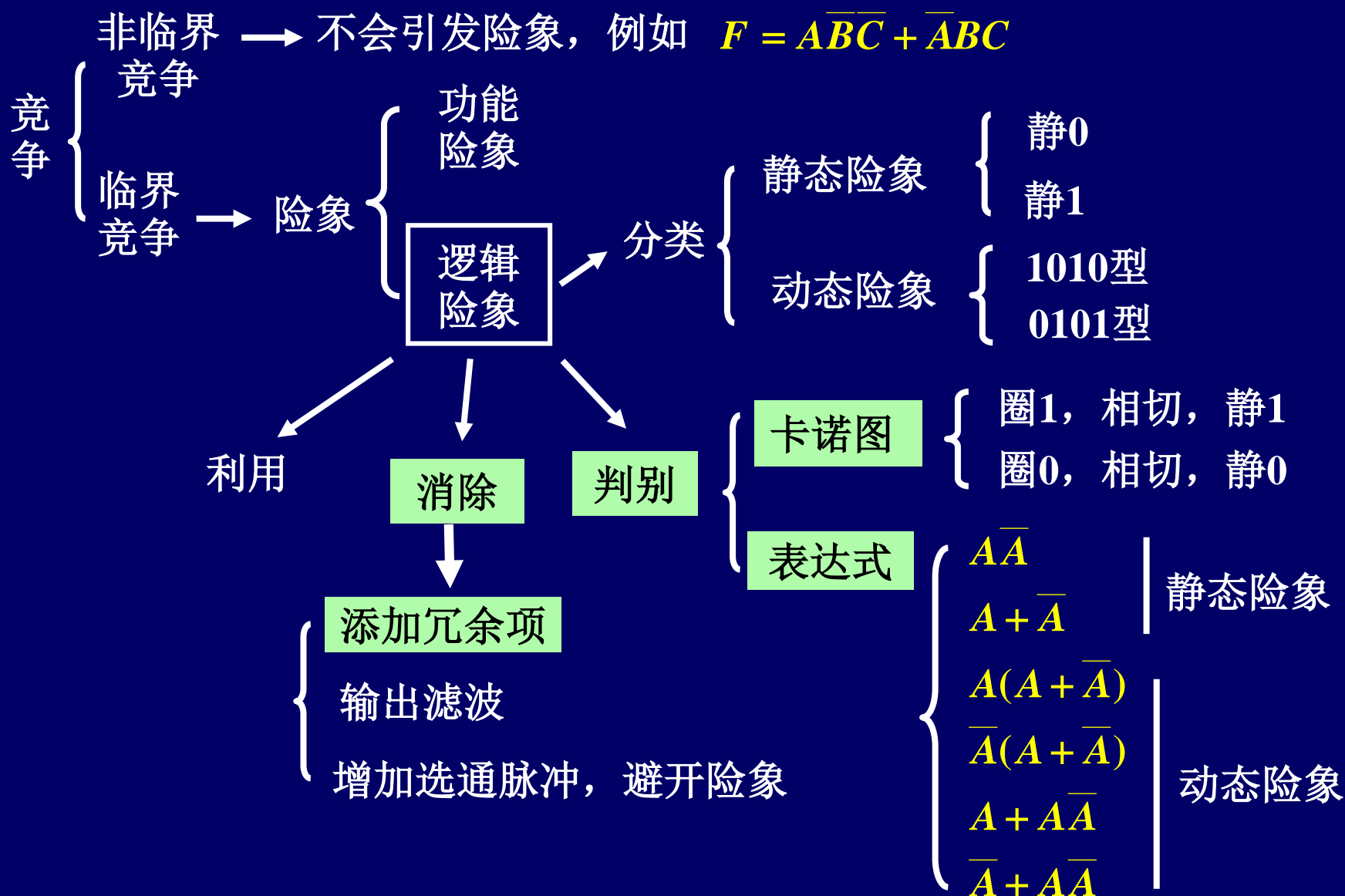
- 1) 将设计要求转化为逻辑关系，列真值表（注意是否包含无关项）
- 2) 用代数法或卡诺图法求得最简逻辑函数表达式
- 3) 根据设计要求变换表达式形式（混合电路、与非门、或非门）
- 4) 画出规范的逻辑电路图

## 基于Verilog HDL的组合电路建模

三种模型：门级模型、数据流模型、行为模型

- 1) 分析逻辑门电路，求最简表达式， Verilog HDL建模；
- 2) 分析以译码器、数据选择器、加法器等逻辑器件为核心的组合电路， Verilog HDL行为建模；
- 3) 分析给定的逻辑命题， Verilog HDL建模；

# 组合逻辑中的竞争与险象





其他：

编码器的原理、优先权编码器的原理、七段显示译码器的原理、比较器的原理、奇偶校验器原理 .....

理解，为采用Verilog HDL进行描述打基础。

## 作业12:

4.20

4.21

4.22

4.26 (3, 5, 7)