数字电路与逻辑设计B

第十讲

南京邮电大学

电子与光学工程学院

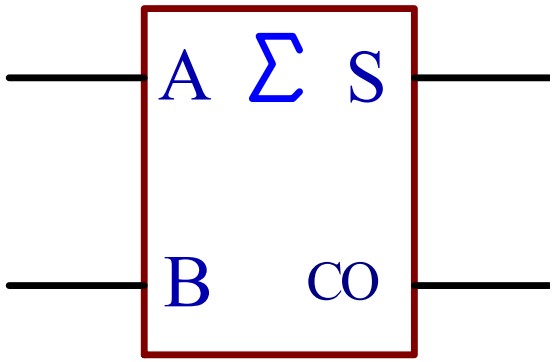
臧裕斌

五、全加器

1.四位串行进位全加器

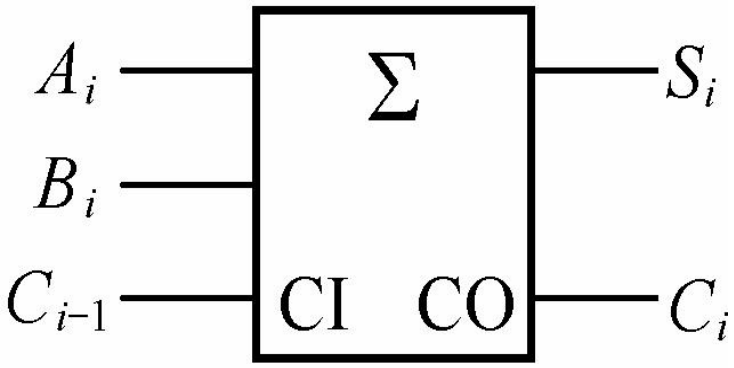
2.四位超前进位全加器

0. 回顾：半加器与全加器



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

$$CO = AB$$



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

$$C_i = A_iB_i + C_{i-1}(A_i \oplus B_i)$$

1. 四位串行进位全加器

$$
\begin{array}{r}
A_3A_2A_1A_0 \\
+\ B_3B_2B_1B_0 \\
\hline
CO\ C_2C_1C_0CI \\
\hline
S_3\ S_2\ S_1\ S_0
\end{array}
$$

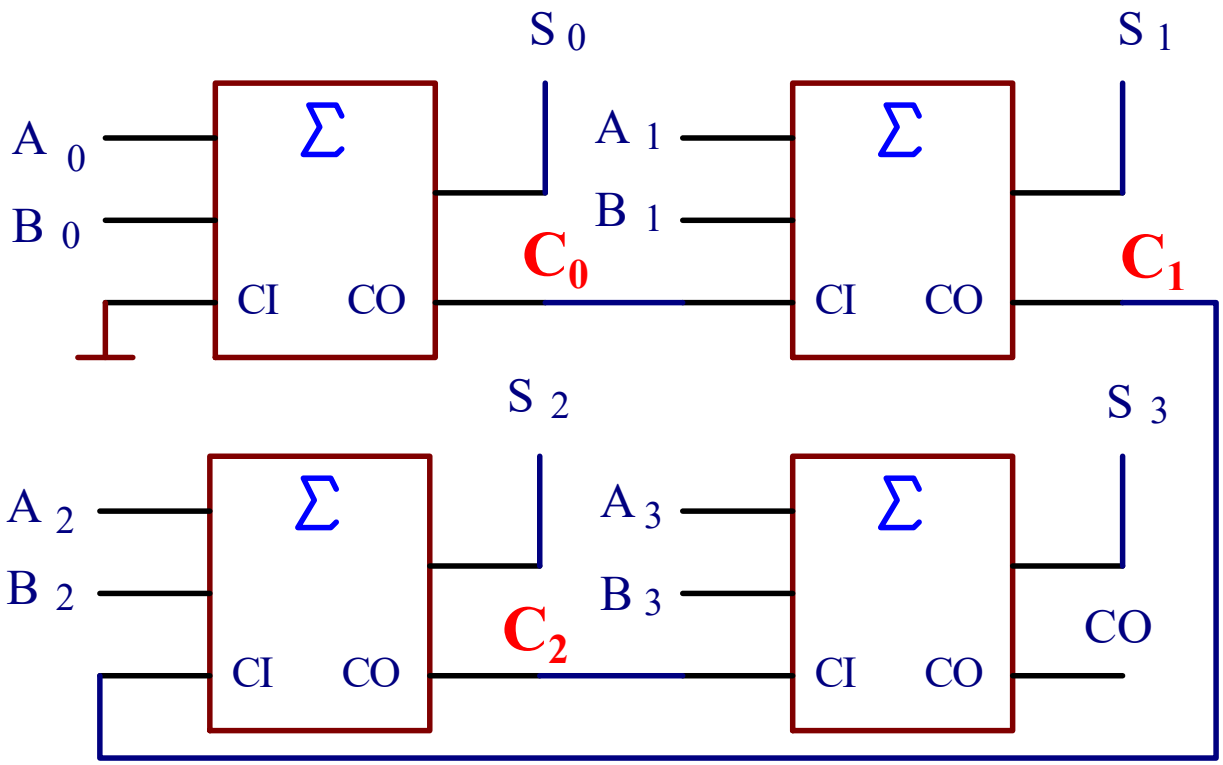


图 3.2.36

$$
\begin{array}{r}
A_3A_2A_1A_0 \\
+\ B_3B_2B_1B_0 \\
CO\ C_2C_1C_0CI \\
\hline
S_3\ S_2\ S_1\ S_0
\end{array}
$$

$S_0 = f_0\ (A_0, B_0, CI)$

$C_0 = g_0\ (A_0, B_0, CI)$

3变量

$S_1 = f_1\ (A_1, B_1, A_0, B_0, CI)$

$C_1 = g_1\ (A_1, B_1, A_0, B_0, CI)$

5变量

$S_2 = f_2\ (A_2, A_1, A_0, B_2, B_1, B_0, CI)$

$C_2 = g_2\ (A_2, A_1, A_0, B_2, B_1, B_0, CI)$

7变量

$S_3 = f_3\ (A_3, A_2, A_1, A_0, B_3, B_2, B_1, B_0, CI)$

$CO = g_3\ (A_3, A_2, A_1, A_0, B_3, B_2, B_1, B_0, CI)$

9变量

变量数增加，函数复杂

2. 四位超前进位全加器74283

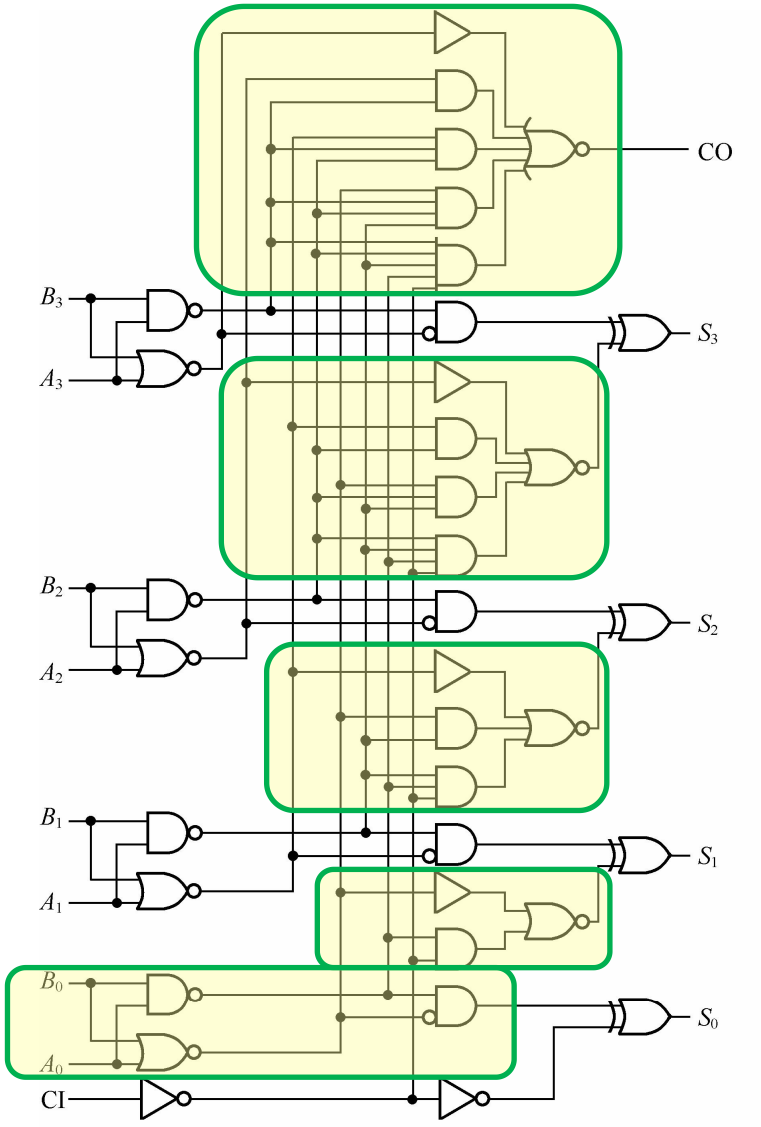
$$S_3 = A_3 \oplus B_3 \oplus C_2$$

$$S_2 = A_2 \oplus B_2 \oplus C_1$$

$$S_1 = A_1 \oplus B_1 \oplus C_0$$

$$S_0 = A_0 \oplus B_0 \oplus CI$$

$$A_0 \oplus B_0$$



（a）逻辑图

CO

C_2

C_1

C_0

$$
\begin{array}{rl}
 & A_3A_2A_1A_0 \\
+ & B_3B_2B_1B_0 \\
CO & C_2C_1C_0CI \\
\hline
 & S_3\ S_2\ S_1\ S_0
\end{array}
$$

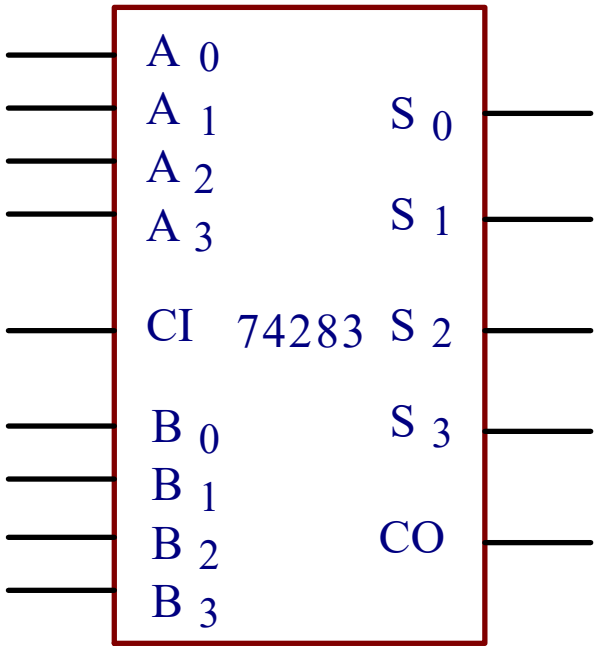


图 3.2.38 (b) 简化符号

六、基于MSI的组合逻辑电路设计

例 已知BCD码 $(A_3A_2A_1A_0.a_3a_2a_1a_0)_{8421BCD}$，试设计一个电路将该数四舍五入。

S1.判断BCD码小数部分 $(a_3a_2a_1a_0)_{8421BCD}$ 是否大于等于5

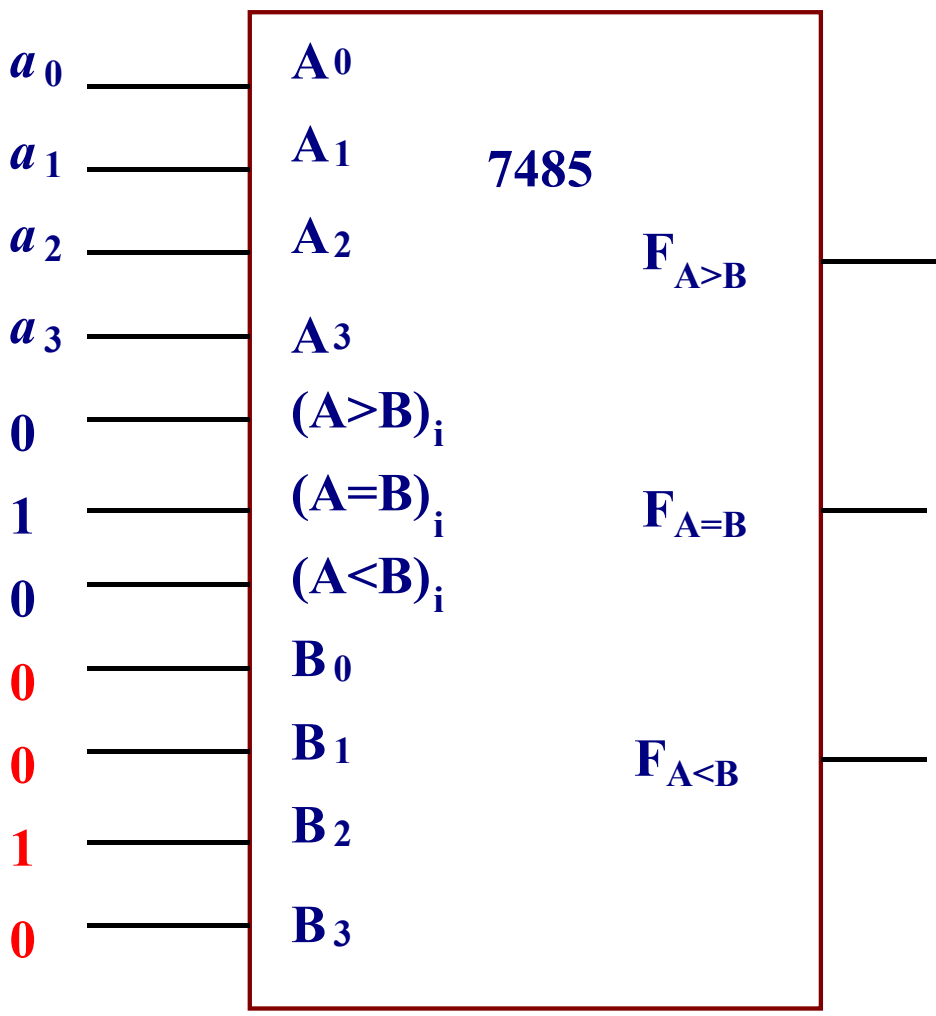
S2.对BCD码整数部分 $(A_3A_2A_1A_0)_{8421BCD}$ 执行加法（0或1）操作

S3.判断中间结果是否合法

S4.对中间结果进行修正（加0000或0110）

$$\begin{array}{rr} & A_3A_2A_1A_0 \\ + & 0/1 \\ \hline & C_3C_2C_1C_0 \\ + & 0000/0110 \\ \hline & F_4F_3F_2F_1F_0 \end{array}$$

S1. 判断BCD码小数部分 $(a_3a_2a_1a_0)_{8421BCD}$是否大于等于5



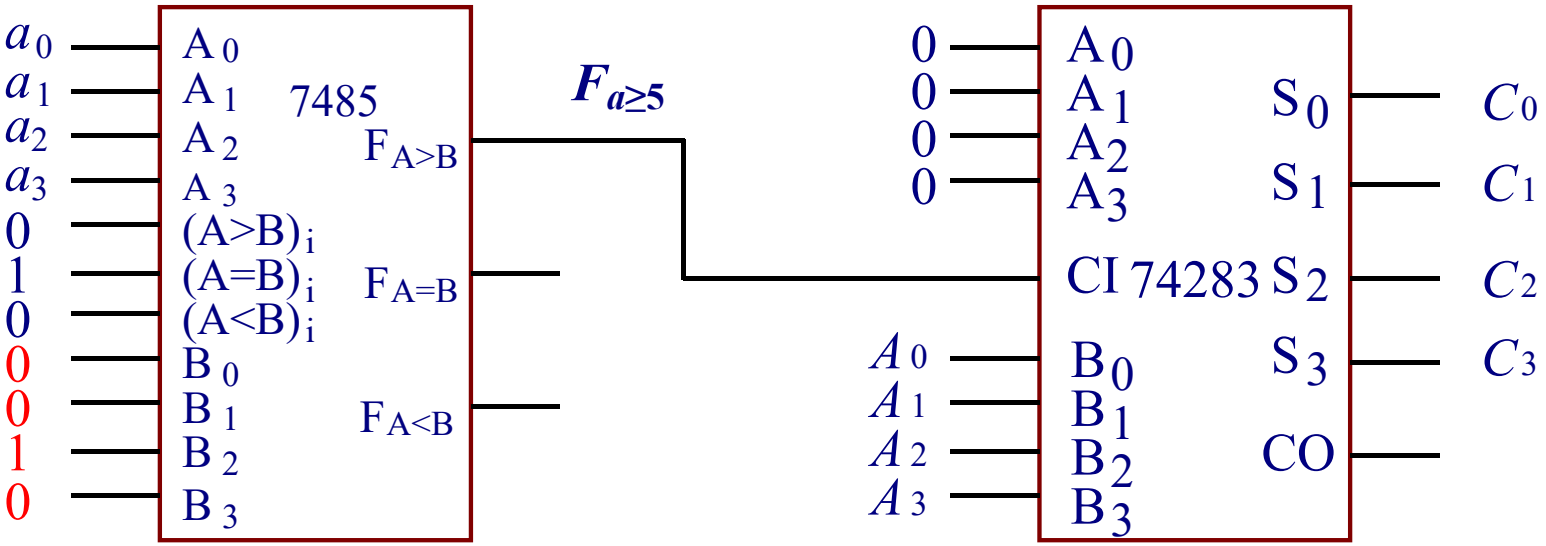
$F_{A>B}$ \ $F_{A<B}F_{A=B}$	00	01	11	10
0	φ	0	φ	0
1	1	φ	φ	φ

$F_{a\geq5}=F_{A>B}$

$F_{A>B}$ \ $F_{A<B}F_{A=B}$	00	01	11	10
0	φ	1	φ	0
1	1	φ	φ	φ

若与5比较，函数关系复杂！

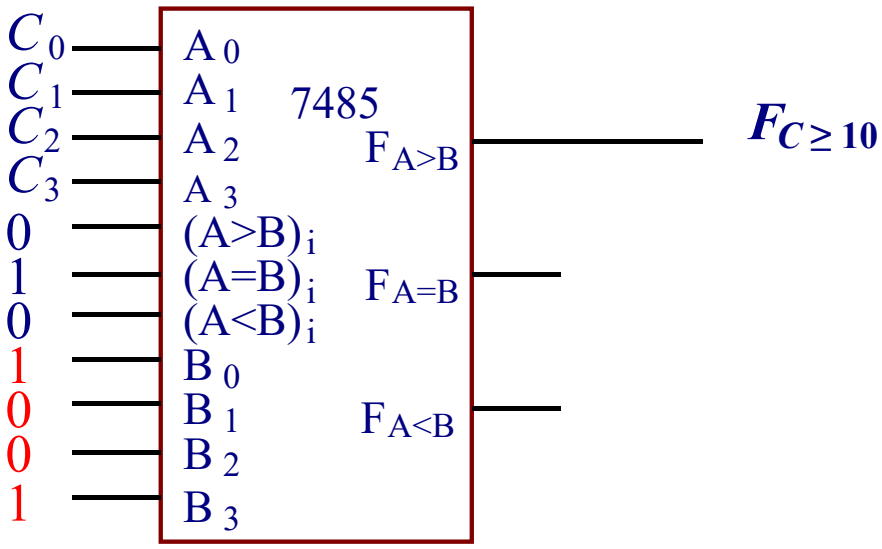
S2.对BCD码整数部分 $(A_3A_2A_1A_0)_{8421BCD}$执行加法（0或1）操作



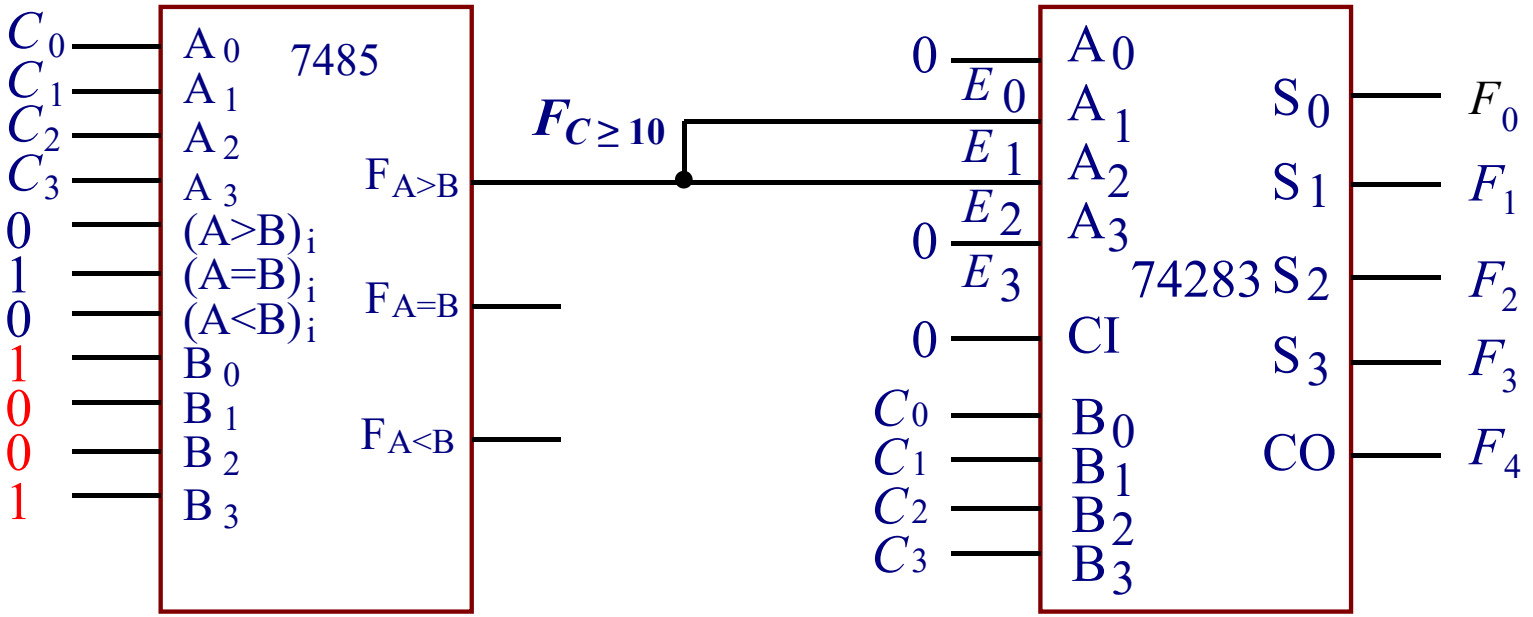
$F_{a\geq5}$	CI
0	0
1	1

$$CI=F_{a\geq5}$$

S3.判断中间结果是否合法



S4.对中间结果进行修正（加0000或0110）

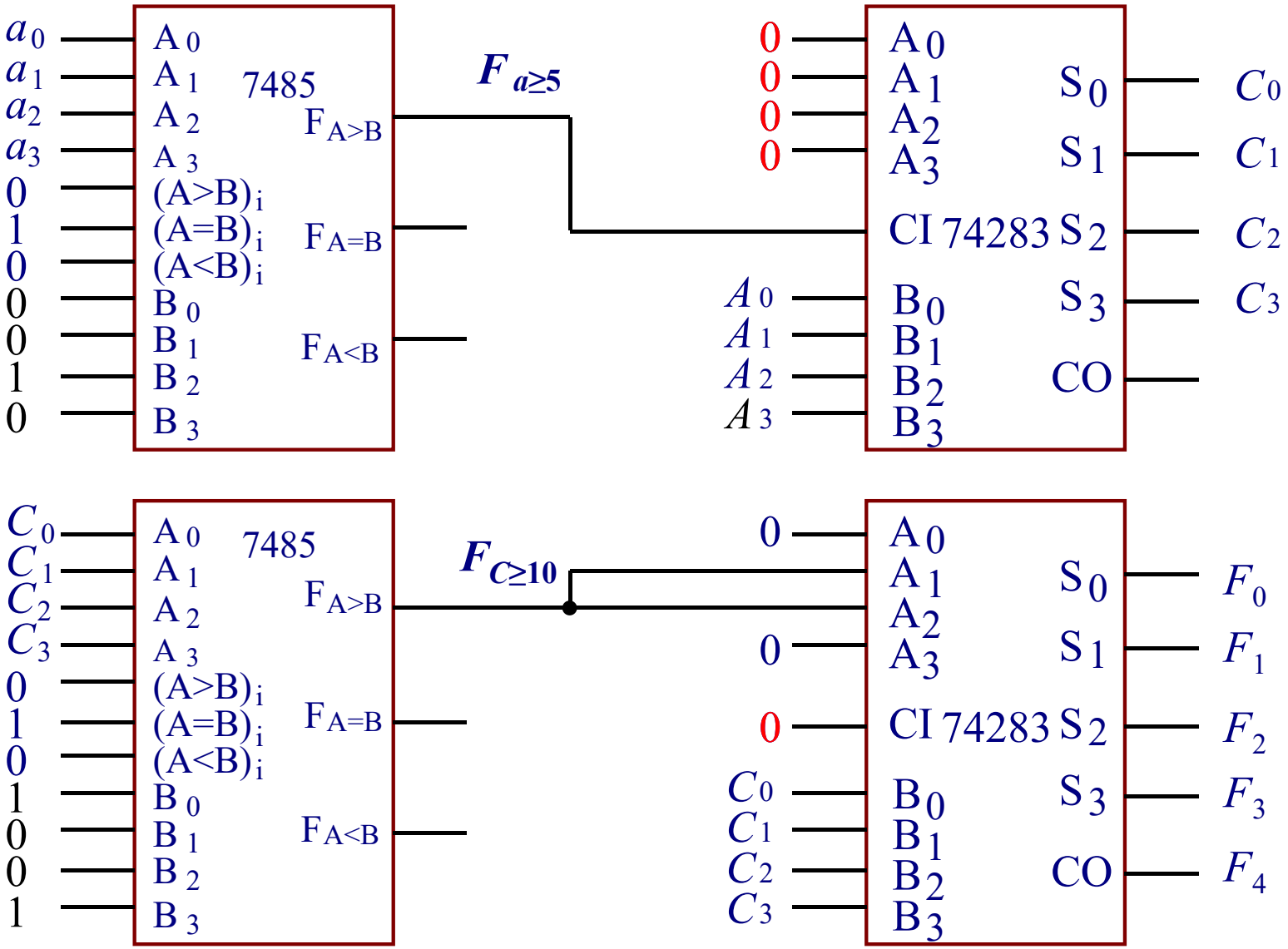


$F_{C\geq10}$	$E_3E_2E_1E_0$
0	0 0 0 0
1	0 1 1 0

$E_2=E_1=F_{C\geq10}$

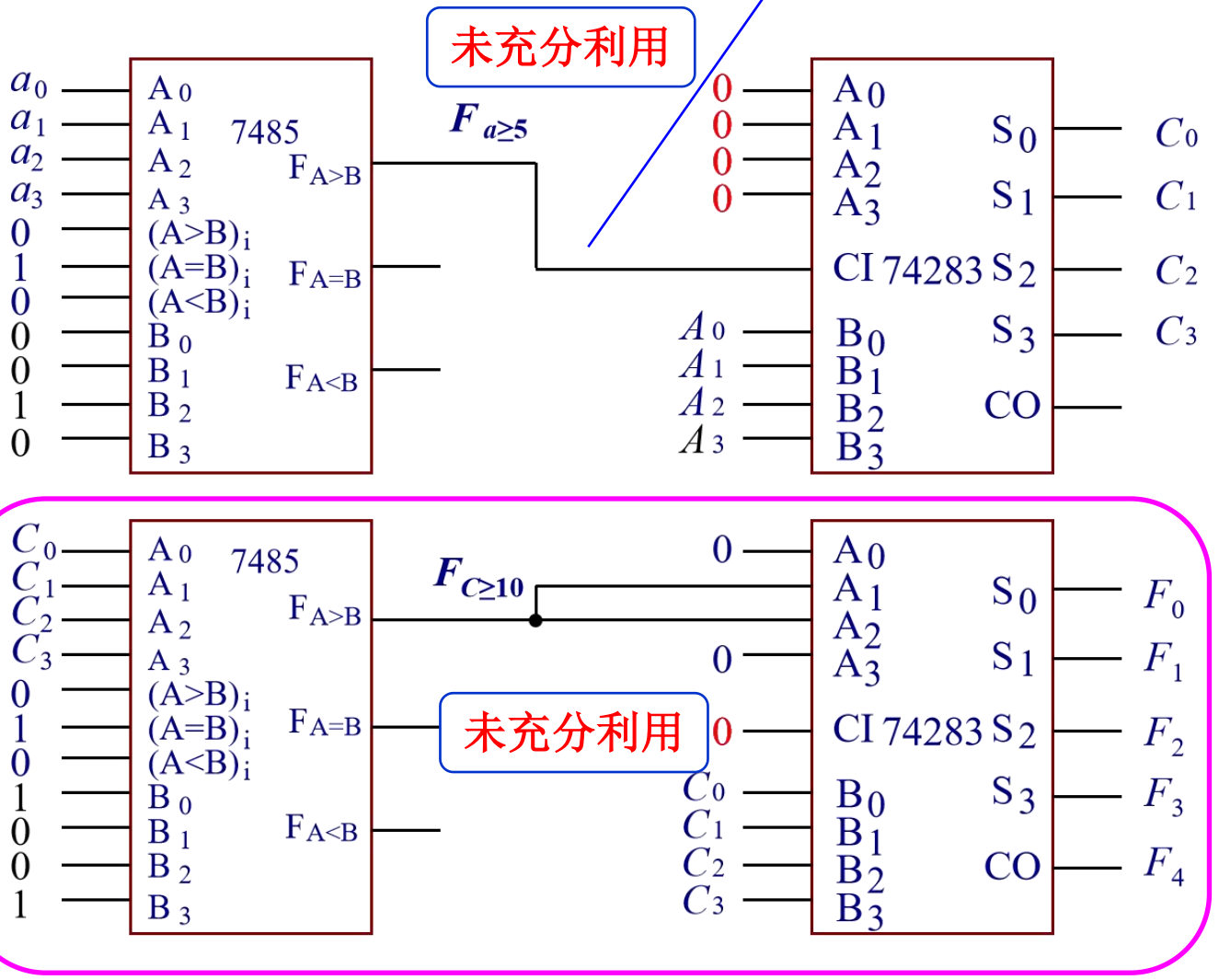
$E_3=E_0=0$

电路整体：


a_0
a_1
a_2
a_3
0
1
0
0
0
1
0
A 0
A 1
A 2
A 3
(A>B)i
(A=B)i
(A<B)i
B 0
B 1
B 2
B 3
7485
FA>B
FA=B
FA<B
$F_{a\geq 5}$
0
0
0
0
A 0
A 1
A 2
A 3
CI 74283
A_0
A_1
A_2
A_3
B 0
B 1
B 2
B 3
S 0
S 1
S 2
S 3
CO
C_0
C_1
C_2
C_3
C_0
C_1
C_2
C_3
0
1
0
0
1
0
$F_{C\geq 10}$
0
0
0
C_0
C_1
C_2
C_3
F_0
F_1
F_2
F_3
F_4

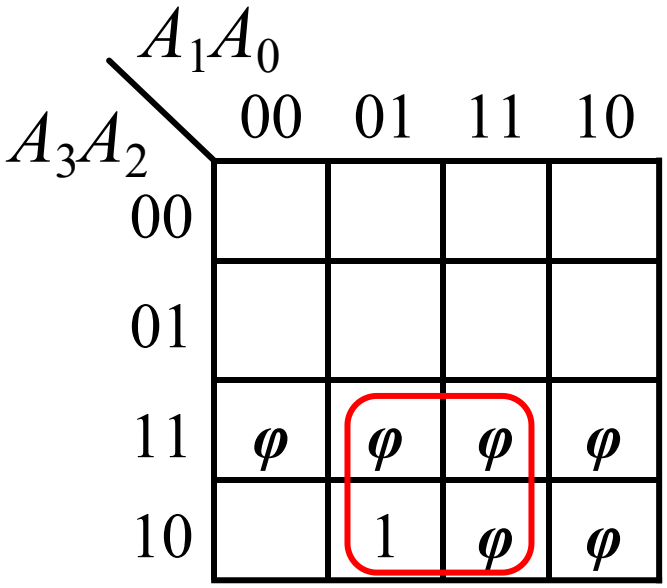
再思考：电路优化

能否实现四舍五入的同时进行修正？



$$\begin{array}{r} A_3A_2A_1A_0 \\ + \quad 0/1 \\ \hline C_3C_2C_1C_0 \\ + \quad 0000/0110 \\ \hline F_4F_3F_2F_1F_0 \end{array}$$

$A_3A_2A_1A_0$	$F_{A=9}$
0 0 0 0	0
⋮	
1 0 0 0	0
1 0 0 1	1
1 0 1 0	φ
1 0 1 1	φ
1 1 0 0	φ
1 1 0 1	φ
1 1 1 0	φ
1 1 1 1	φ

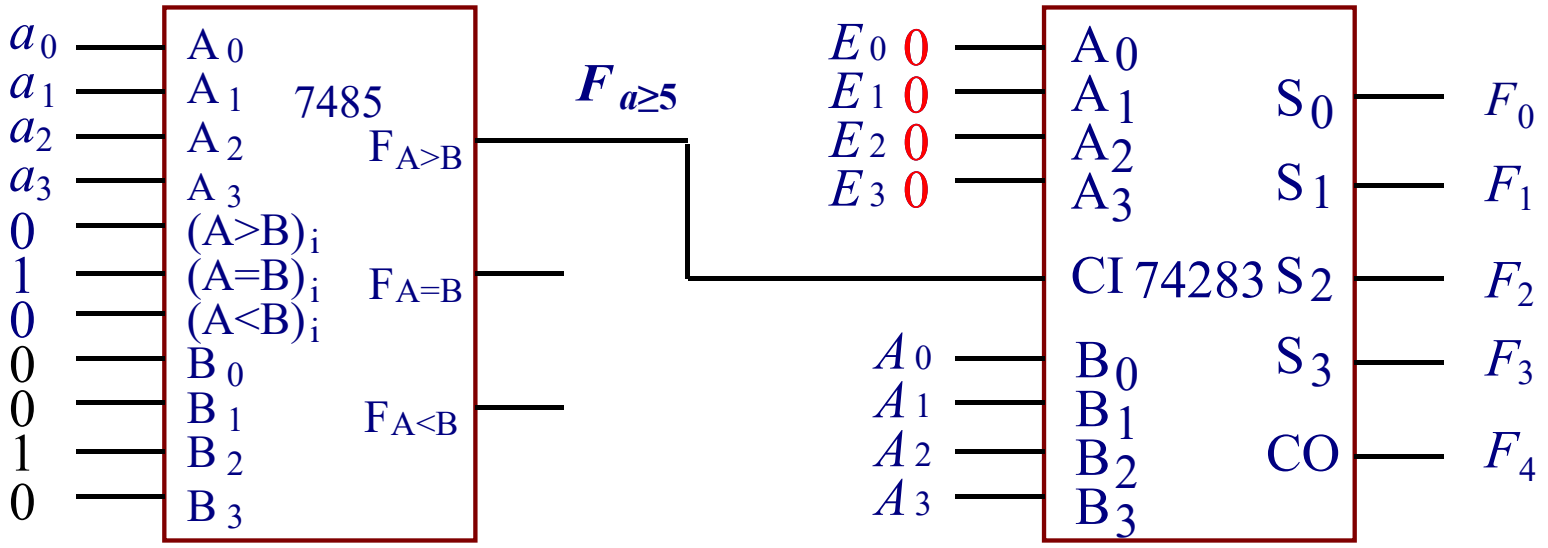


$F_{A=9} = A_3A_0$

$F_{A=9}$	$F_{a\geq5}$	$F_{C\geq10}$
0	0	0
0	1	0
1	0	0
1	1	1

$F_{C\geq10}=F_{A=9}F_{a\geq5}$

$=A_3A_0F_{a\geq5}$



$F_{C\geq10}$	$E_3E_2E_1E_0$
0	0 0 0 0
1	0 1 1 0

$$E_2=E_1=F_{C\geq10}=A_3A_0F_{a\geq5}$$

$$E_3=E_0=0$$

优化后的电路：

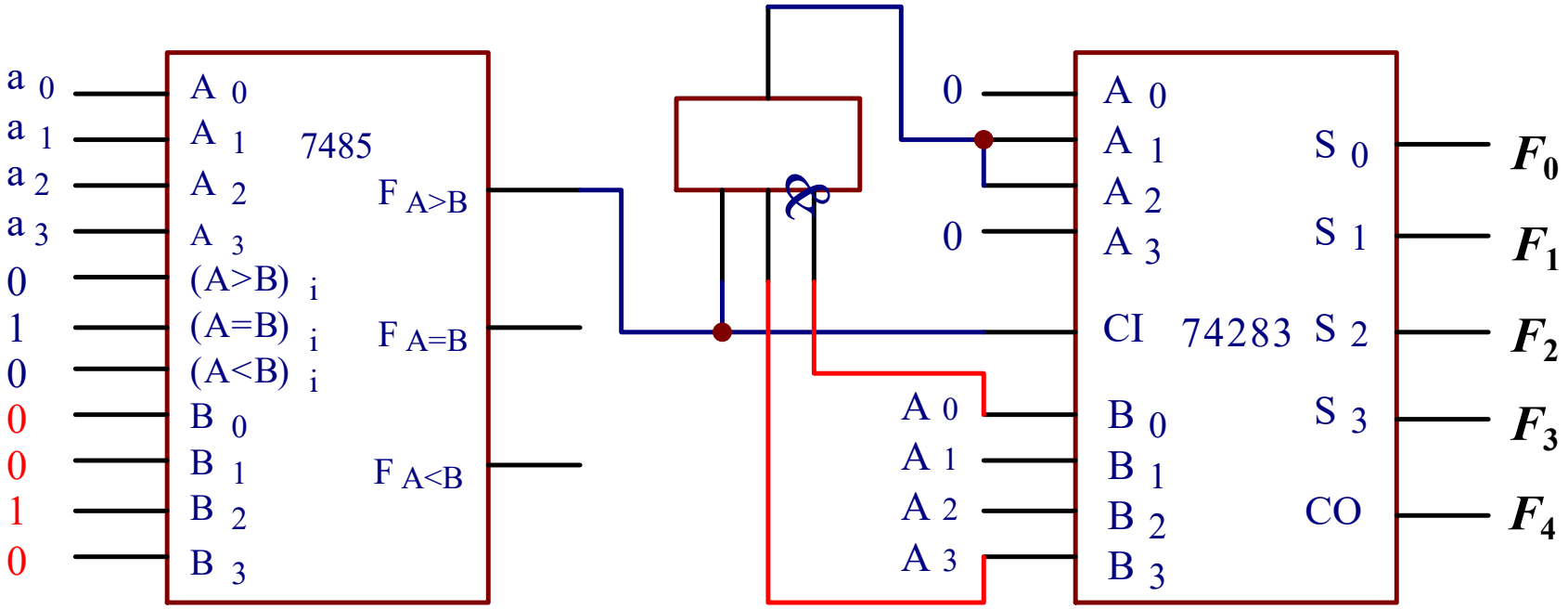
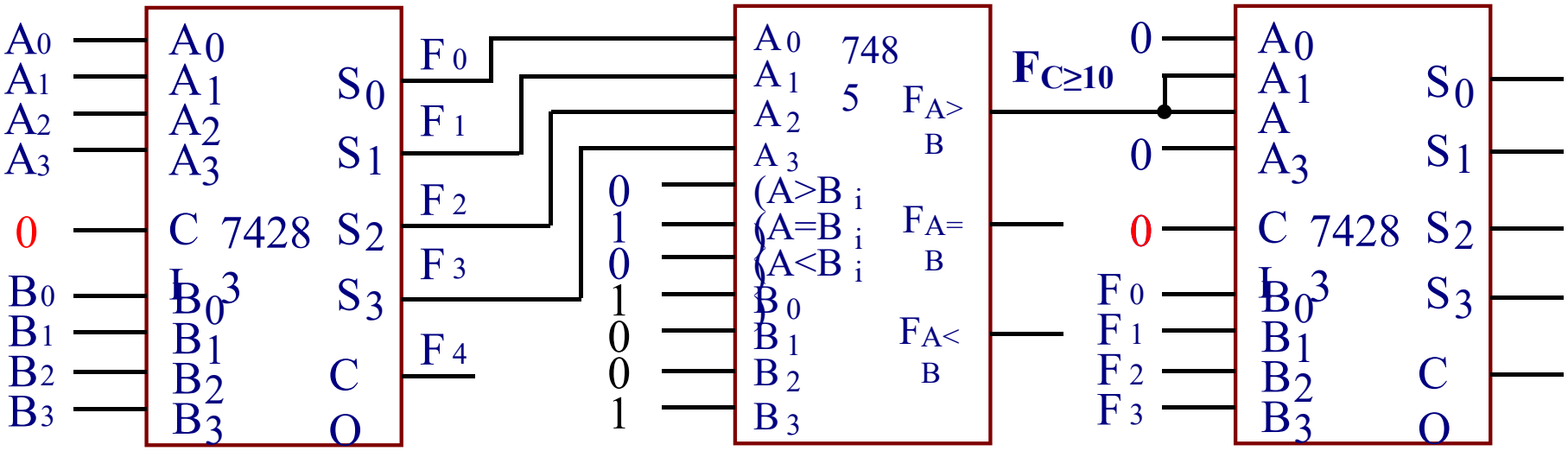


图 3.2.43

主观题 10分

设置

思考题1：$(A_3A_2A_1A_0)_{8421BCD}+(B_3B_2B_1B_0)_{8421BCD}=(000C_4C_3C_2C_1C_0)_{8421BCD}$，若产生非法码，要进行加0110修正，在以下图中标注输出端$C_4C_3C_2C_1C_0$。

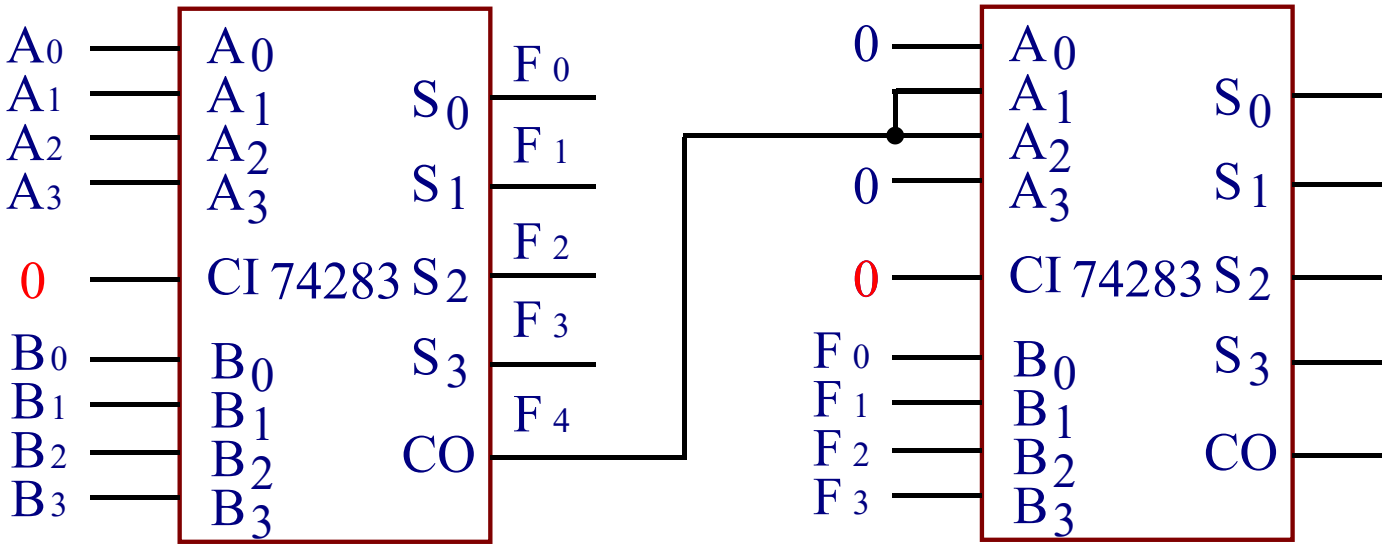


作答

主观题 10分

设置

思考题2：$(A_3A_2A_1A_0)_{8421BCD}+(B_3B_2B_1B_0)_{8421BCD}=$（$000C_4C_3C_2C_1C_0$）$_{8421BCD}$，若产生进位，要进行加0110修正，在以下图中标注输出端$C_4C_3C_2C_1C_0$。



作答

作业题

3.13(1)(3)
3.18
3.20

3.3　竞争和冒险

一、竞争和冒险的概念

1. 竞争

2.冒险

二、冒险的消除方法

1. 增加多余项，消除逻辑冒险

3.4 Verilog描述组合电路

一、门级建模

1. 2-4线译码器

2. 4位全加器

二、数据流建模

1. 2选1数据选择器

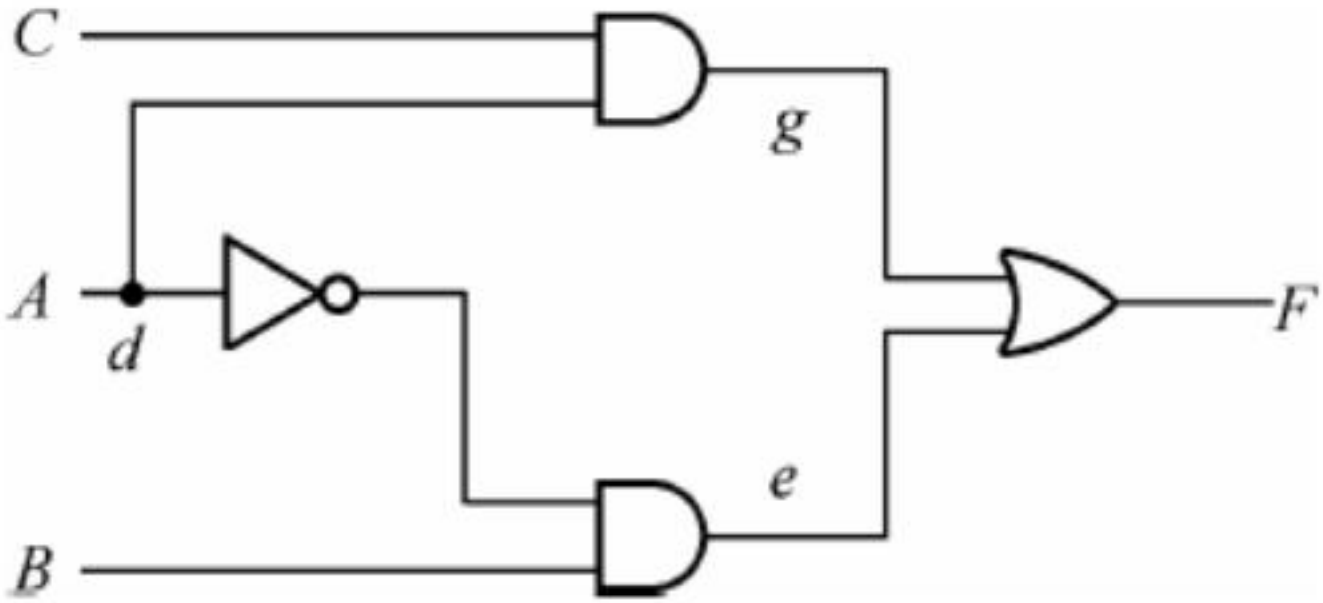
2. 4位全加器

3. 2-4线译码器

3.3 竞争和冒险

观察：

描述逻辑函数 $F = AC + \overline{A}B$ 的电路



分析时，假设各门的延迟时间均为t_{pd}, 且C＝B＝1

- 事实上，在B=C=1时，F=1 ！

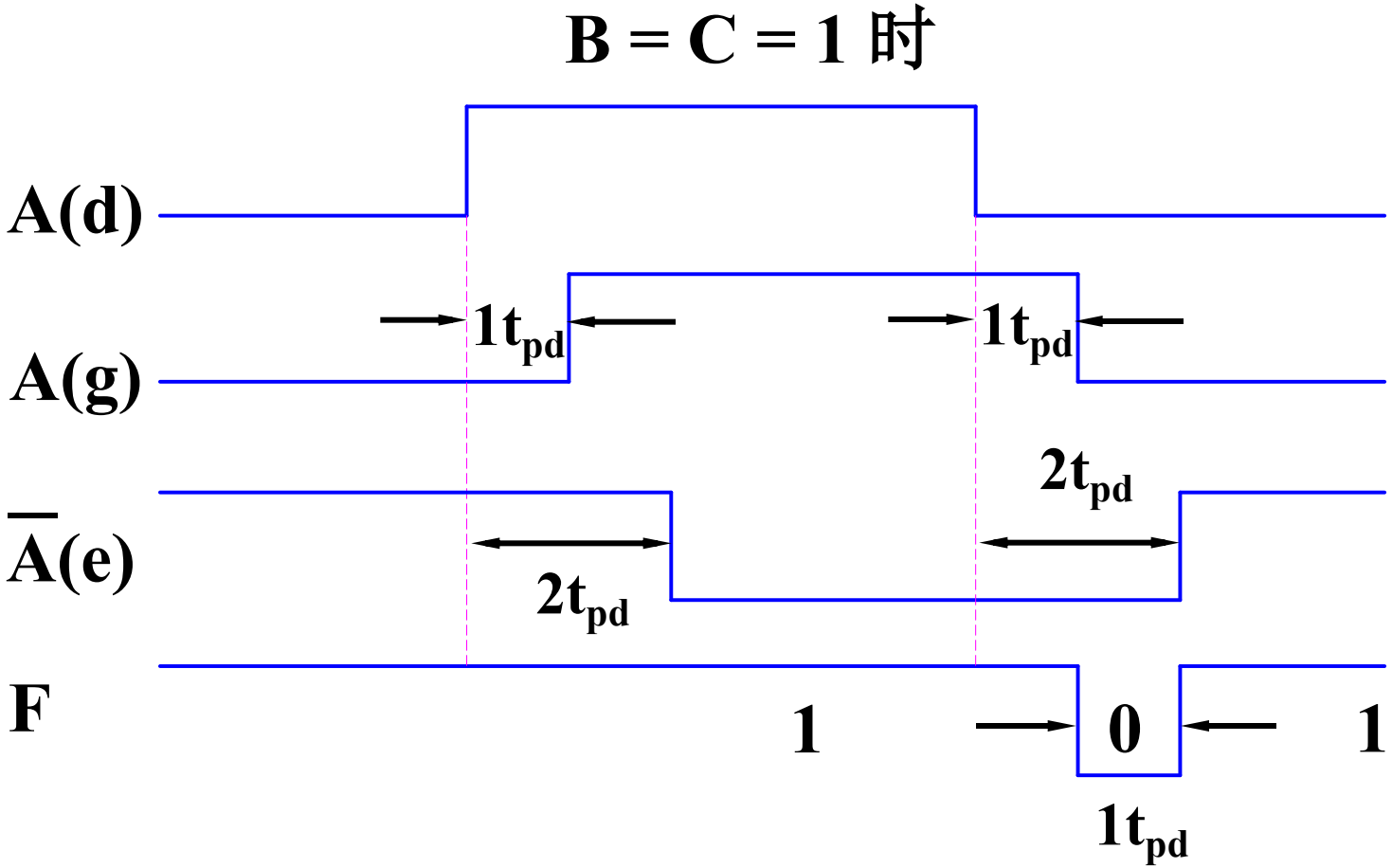


图 3.3.2 冒险的产生

3.3 竞争和冒险

一、竞争和冒险的概念

1. 竞争

(1) 由于连线和集成门有一定的延迟时间，致使同一输入信号经过不同路径到达输出端有先有后（1个或1个以上输入信号变化）；

(2) 多个输入信号同时变化，由于变化的快慢不同，致使多个输入信号到达输出端有先有后（2个或2个以上输入信号变化）。

2.冒险

(1) 冒险的概念

(2) 冒险的分类

① 按短暂尖峰极性

1型冒险和0型冒险

② 按产生短暂尖峰的原因

逻辑冒险:由于输入信号经过的路径不同而引起的冒险

功能冒险: 由于若干个信号同时变化，变化的快慢不同而引起的冒险

二、竞争和冒险的判别

1）逻辑冒险的判别

（1）代数法

$F = A + \overline{A}$（对应0型逻辑冒险的判别式）

$F = A \cdot \overline{A}$（对应1型逻辑冒险的判别式）

以上两式有时亦称为竞争冒险的判别式，只要根据电路写出的逻辑式在一定的条件下，能转化成判别式的形式，就说明在一定条件下，可能存在冒险。

代数法判断的核心与步骤：

(a)判断表达式中是否有变量同时以原、反变量的形式存在;
(b)若有，则将其余变量取某固定值，看能否转化成判别式。

（2）卡诺图法

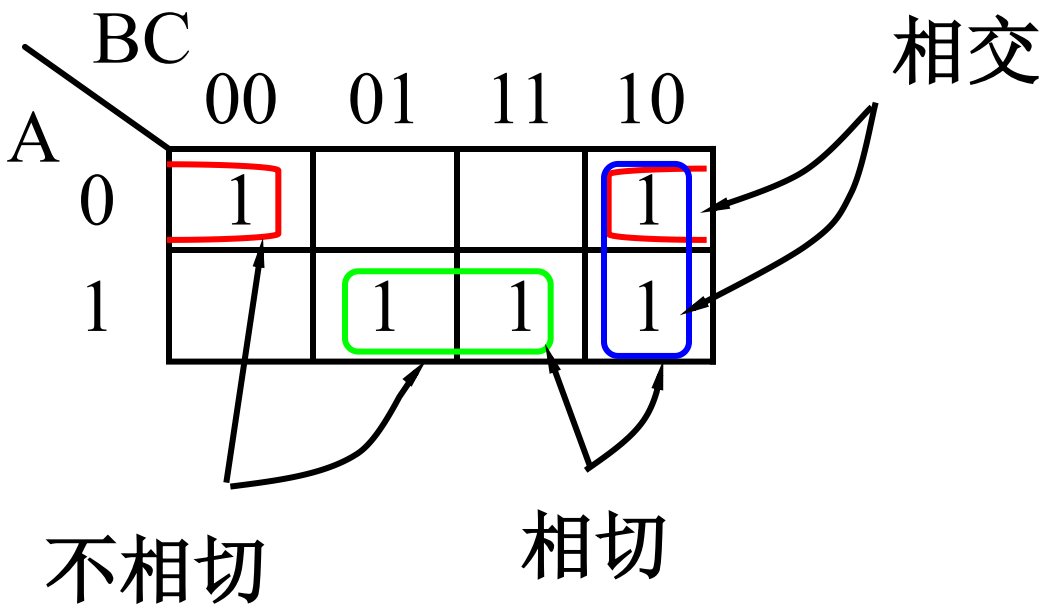
方法：两个卡诺圈部分相切，而这个相切部分又没有被另外的卡诺圈包围，则可能存在逻辑冒险。

条件：相切部分取值相同的变量此时的取值组合。

卡诺圈相切示意图

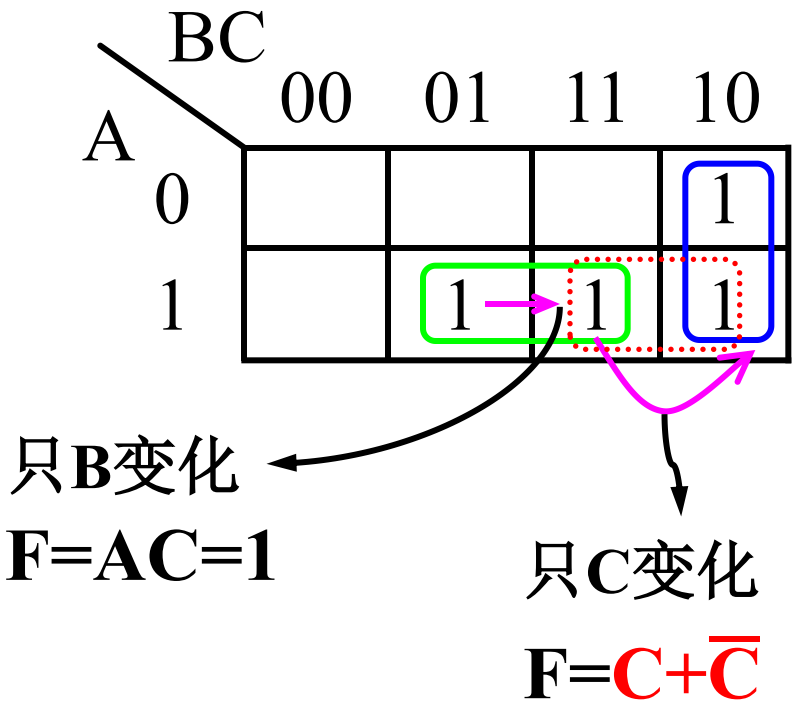
卡诺图法判断逻辑冒险的原理

例1 $F = AC + B\overline{C} + \overline{A}\,\overline{C}$ 。



几何相邻或对称相邻的“1”格，被两个卡诺圈分别独自包围，称卡诺圈相切。

例2： 判别函数$F = A\,C + \overline{B}\,C$是否存在逻辑冒险现象

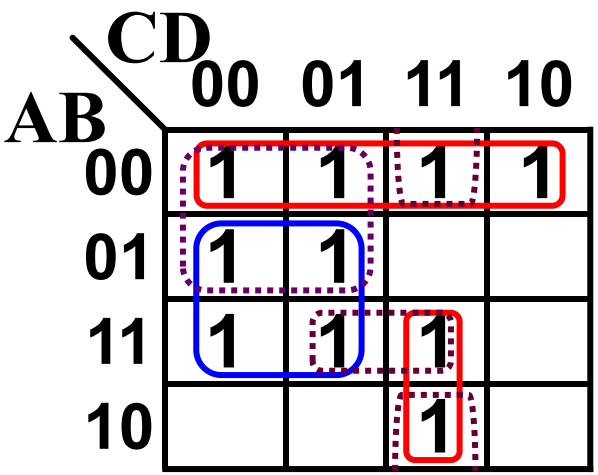


变量取值在卡诺圈内变化，不存在逻辑冒险；

变量取值在相切的卡诺圈跳变时，而这个相切部分又没有被另外的卡诺圈包围，则可能存在逻辑冒险。

$A = B = 1$ 时，存在0型逻辑冒险。

例3：$F = B\overline{C} + \overline{A}\,\overline{B} + ACD$ 。试判断是否存在逻辑冒险。



在 AC = 00 时，或

在 BCD = 011 时，或

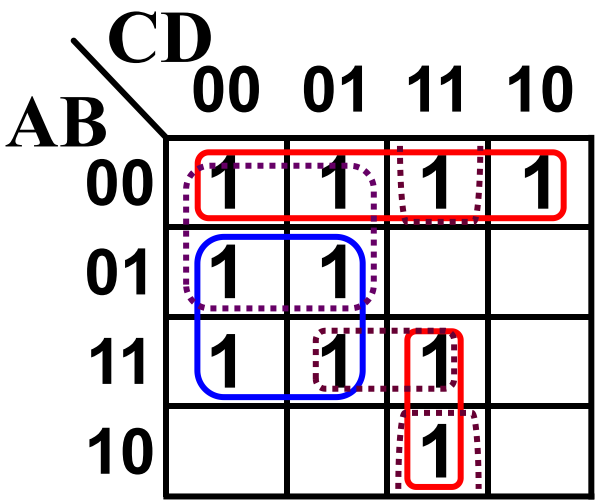
在 ABD = 111 时，

存在0型逻辑冒险。

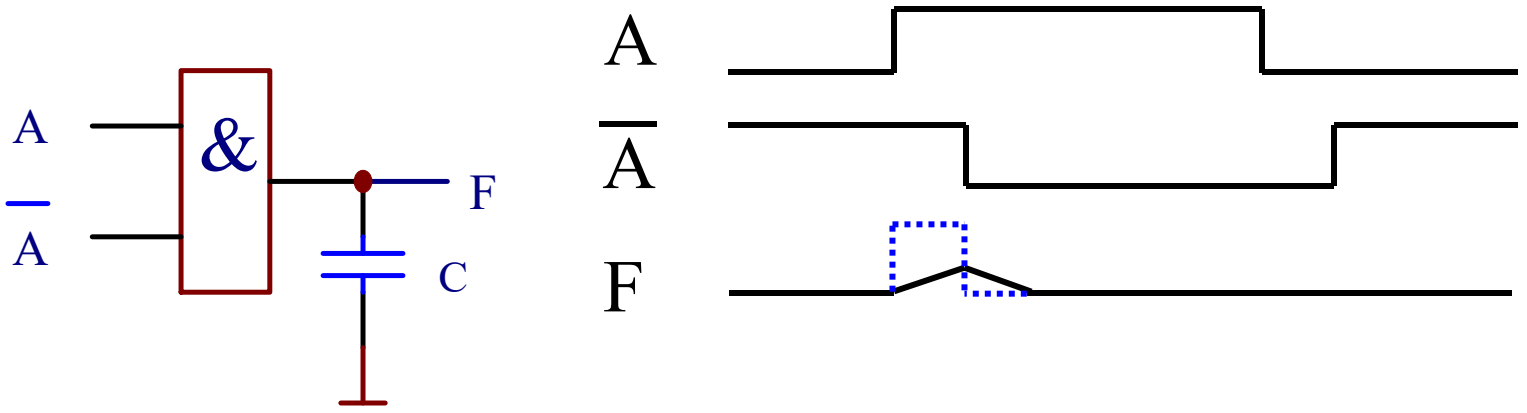
三、竞争和冒险的规避

1. 增加多余项，消除逻辑冒险

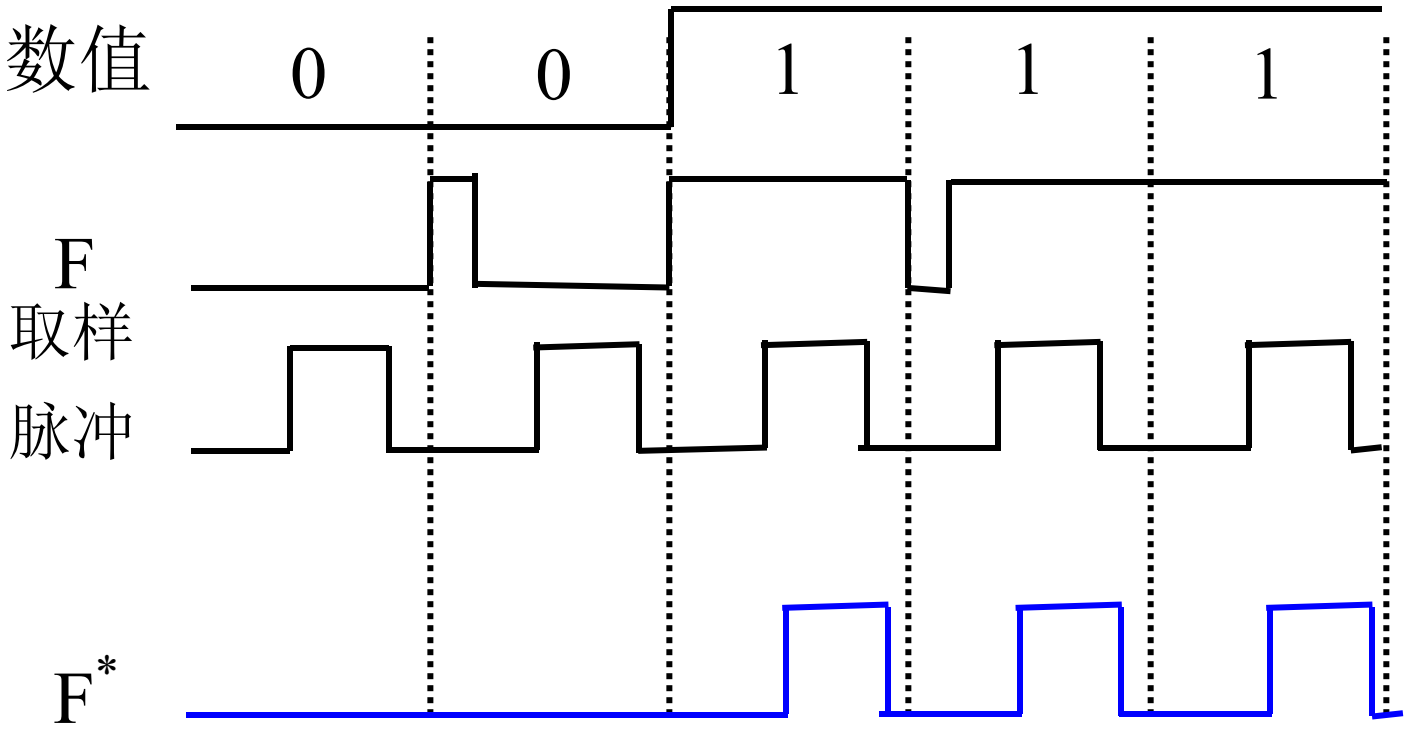
当输入变量取值在卡诺圈内部变化时，不会存在逻辑冒险，因此可以通过增加多余卡诺圈的方法，消除逻辑冒险。



2.加滤波电容 （对输出波形边沿要求不高的情况下运用)



3.加取样脉冲

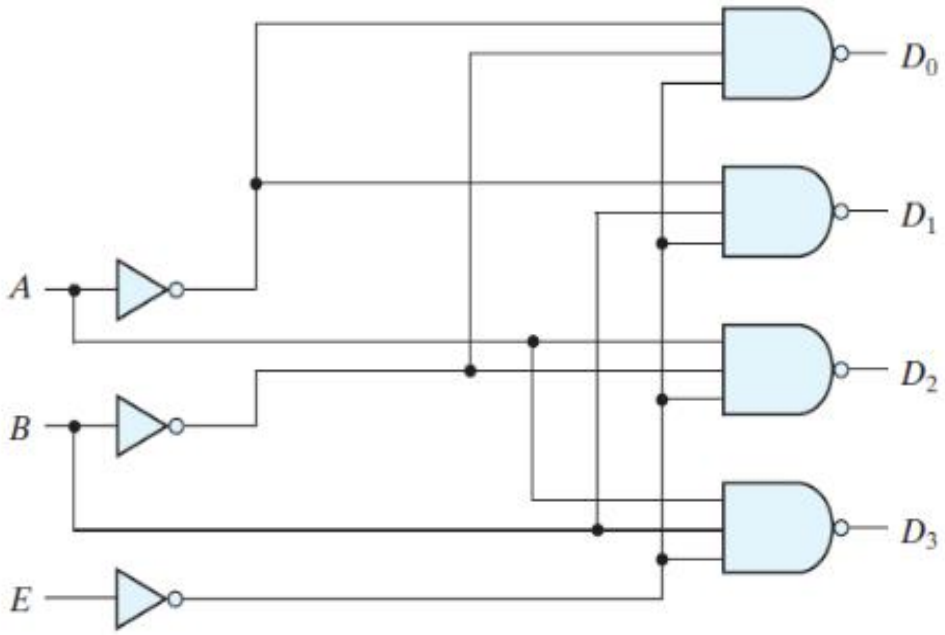


其中，F和F^* 分别表示组合电路加取样脉冲之前、之后的输出。

3.4 Verilog描述组合逻辑电路

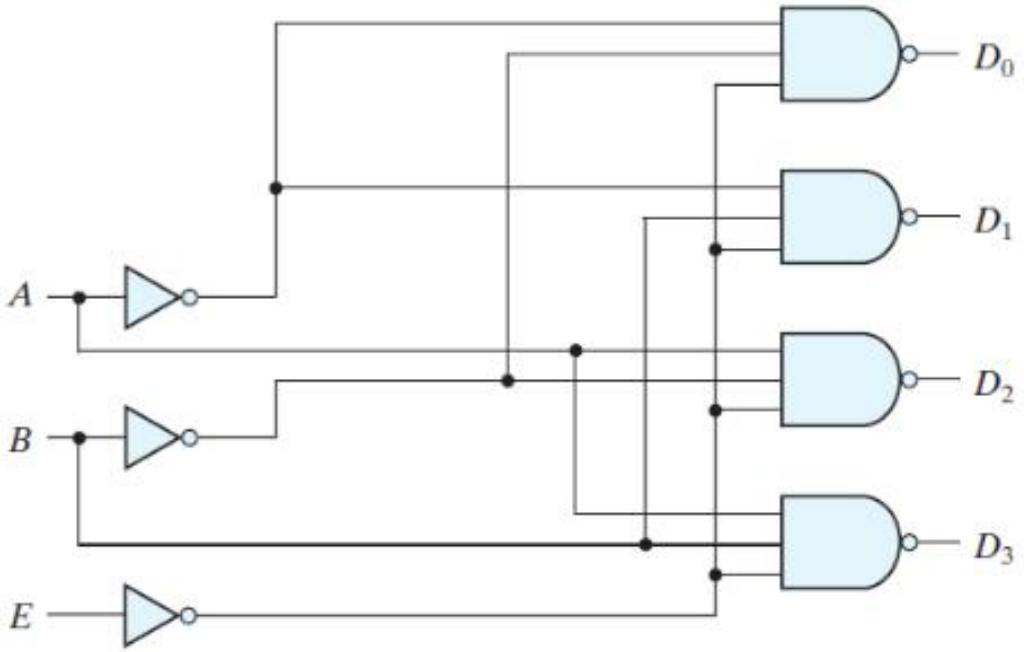
一、门级建模（描述电路结构）

1. 2-4线译码器



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	1

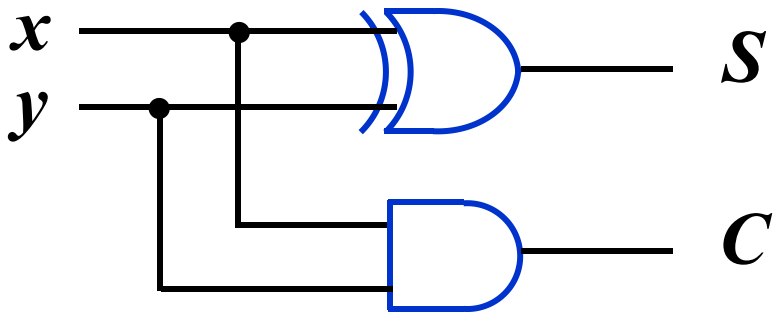
```
module decoder_2x4_gates (D, A, B, E);
output  [3:0]  D;
input  A,B;
input E;
wire A_not, B_not, E_not;
not
  G1 (A_not, A),
  G2 (B_not, B),
  G3 (E_not, E);
nand
  G4 (D[0], A_not, B_not, E_not),
  G5 (D[1], A_not, B, E_not),
  G6 (D[2], A, B_not, E_not),
  G7 (D[3], A, B, E_not);
endmodule
```



2. 加法器

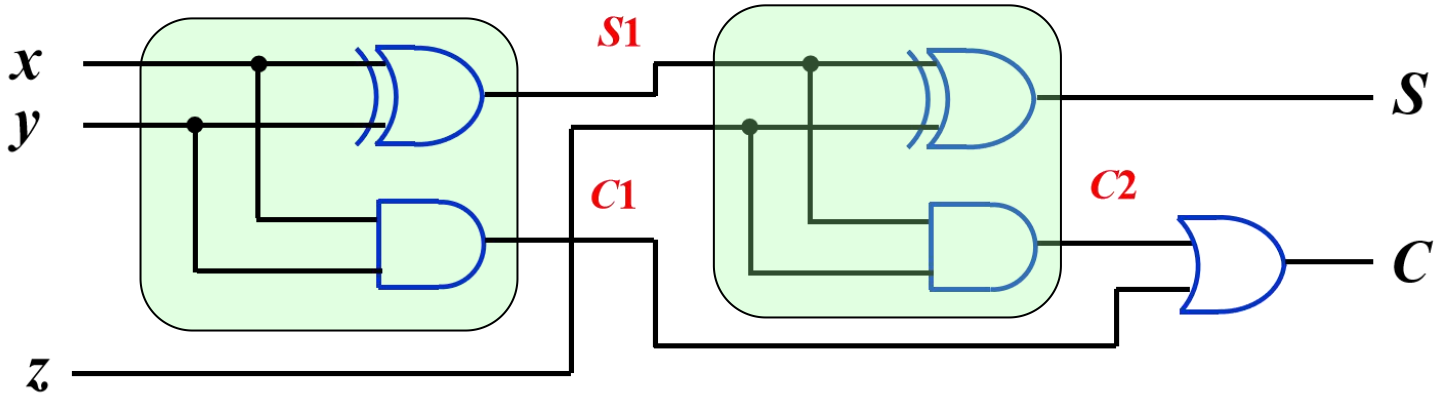
1). 1位半加器

```
//module half_adder (S, C, x, y); 1995版
//output  S, C;
//input  x,y;
module half_adder (output S, C, input x, y); 2001、2005版
xor (S, x, y);
and (C, x, y);
endmodule
```



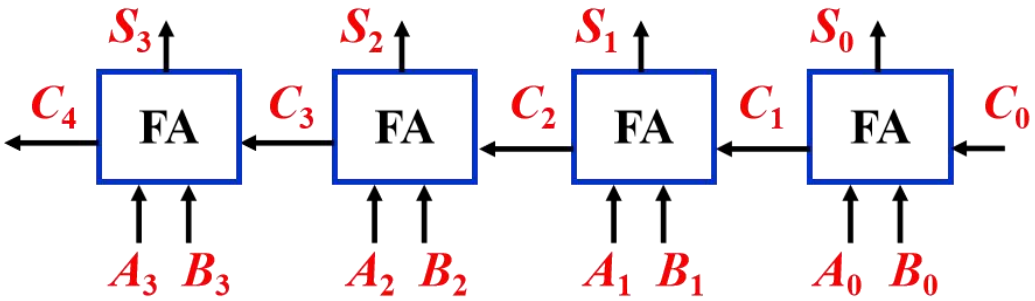
2). 1位全加器

```
//module full_adder (S, C, x, y, z); 1995版
//output  S, C;
//input  x, y, z;
module full_adder (output S, C, input x, y, z); 2001、2005版
wire S1, C1, C2;
//Instantiate half adders
half_adder HA1 (S1, C1, x, y);
half_adder HA2 (S, C2, S1, z);
or G1 (C, C2, C1);
endmodule
```

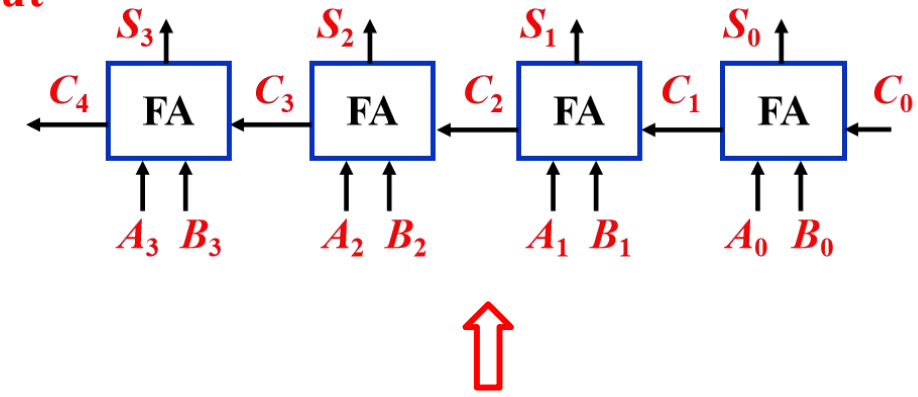


3). 4位全加器

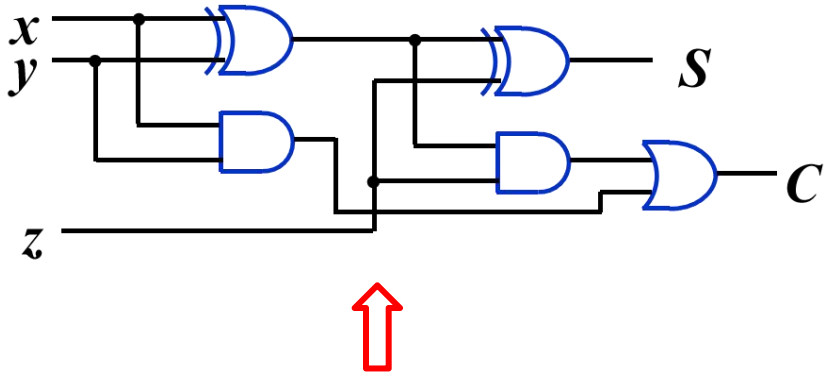
```
//module ripple_carry_4_bit_adder (S, C4, A, B, C0); //1995版
//output  [3:0] S;
//output  C4;
//input  [3:0] A, B;
//input C0;
module ripple_carry_4_bit_adder (output [3:0] S, output C4,
input [3:0] A, B, input C0); //2001、2005版
wire C1, C2, C3;
//Instantiate chain of full adders
full_adder FA0 (S[0], C1, A[0], B[0], C0),
           FA1 (S[1], C2, A[1], B[1], C1),
           FA2 (S[2], C3, A[2], B[2], C2),
           FA3 (S[3], C4, A[3], B[3], C3);
endmodule
```



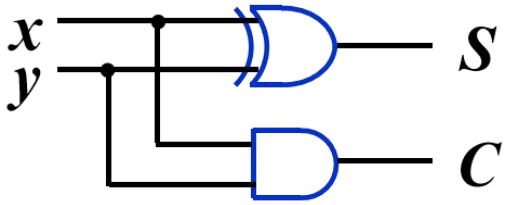
```
module ripple_carry_4_bit_adder (output
[3:0] S, output C4, input [3:0] A, B, input
C0);
wire C1, C2, C3;
full_adder FA0 (S[0], C1, A[0], B[0], C0),
           FA1 (S[1], C2, A[1], B[1], C1),
           FA2 (S[2], C3, A[2], B[2], C2),
           FA3 (S[3], C4, A[3], B[3], C3);
endmodule
```




```
module full_adder (output S, C, input x, y, z);
wire S1, C1, C2;
half_adder HA1 (S1, C1, x, y);
half_adder HA2 (S, C2, S1, z);
or G1 (C, C2, C1);
endmodule
```




```
module half_adder (output S, C, input x, y);
xor (S, x, y);
and (C, x, y);
endmodule
```



二、数据流建模（描述逻辑功能）

1. 2选1数据选择器

```
//dataflow description of two-to-one-line multiplexer
module mux_2x1_df  (m_out, A, B, select);
output  m_out;
input A, B;
input select;
assign m_out = (select)? A : B;
endmodule
```

2. 4位全加器

```
//dataflow description of four-bit adder
module binary_adder  (
    output [3:0] S,
    output  C_out,
    input [3:0] A, B,
    input C_in
);
assign {C_out, S} =  A + B + C_in;
endmodule
```

3. 2-4线译码器

```
module decoder_2x4_df  (
    output [3:0] D,
    input A, B, enable
);
assign D[0] =  !((!A) && (!B) && (!enable))
           D[1] =  !((!A) && B && (!enable)),
           D[2] =  !(A && (!B) && (!enable)),
           D[3] =  !(A && B && (!enable));
endmodule
```