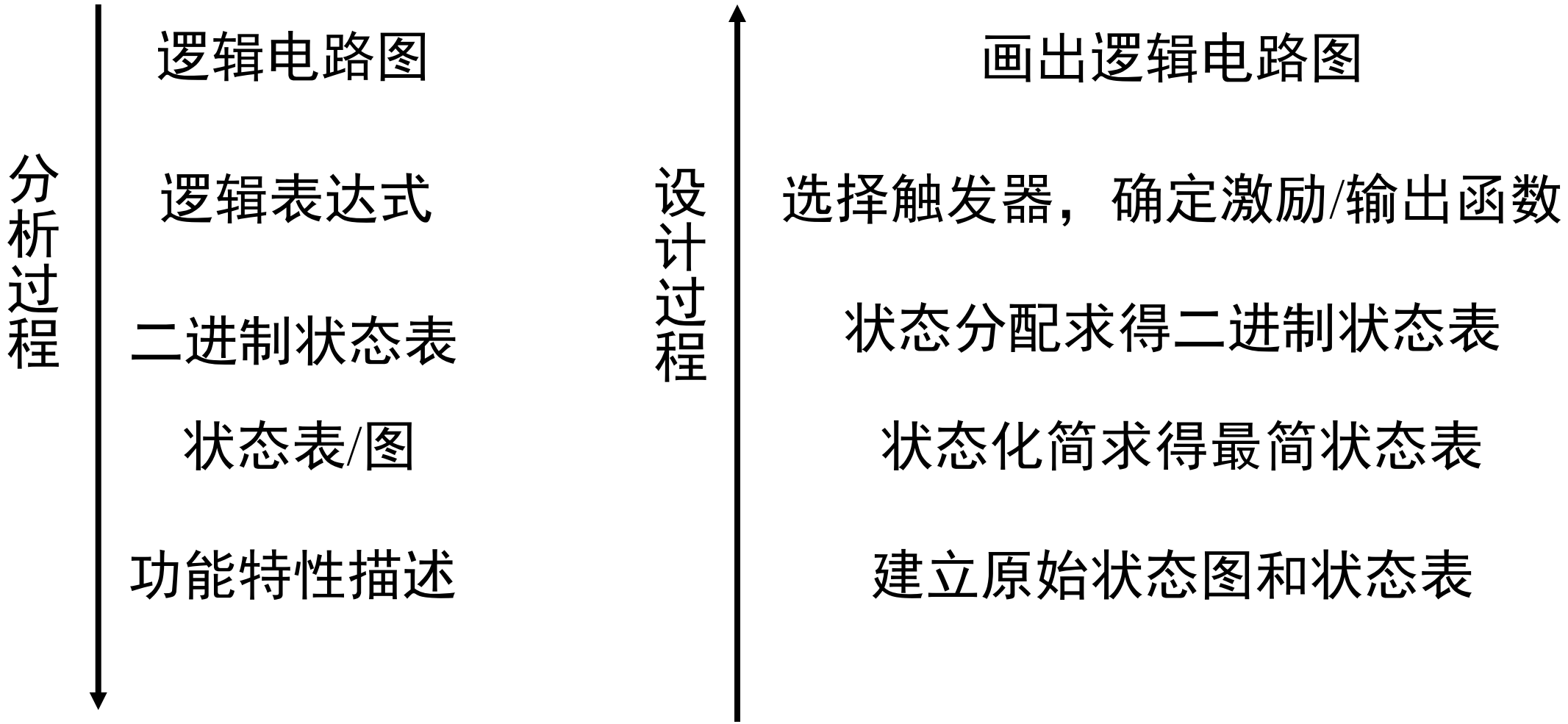
第五章 同步时序电路的设计

5.1 建立原始状态图(表)

5.2 状态化简

5.3 有限状态机（FSM）

同步时序电路分析与设计的比较：



5.1 建立原始状态图(表)

建立原始状态表的关键是确定以下三个问题：

1、所描述电路应包括多少状态？

2、状态之间的转换关系如何？

3、输出情况如何？

设计要求：只求正确，不求最简。确保逻辑功能的正确性（完整性和无二义性）。

设计方法：直接构图（表）法

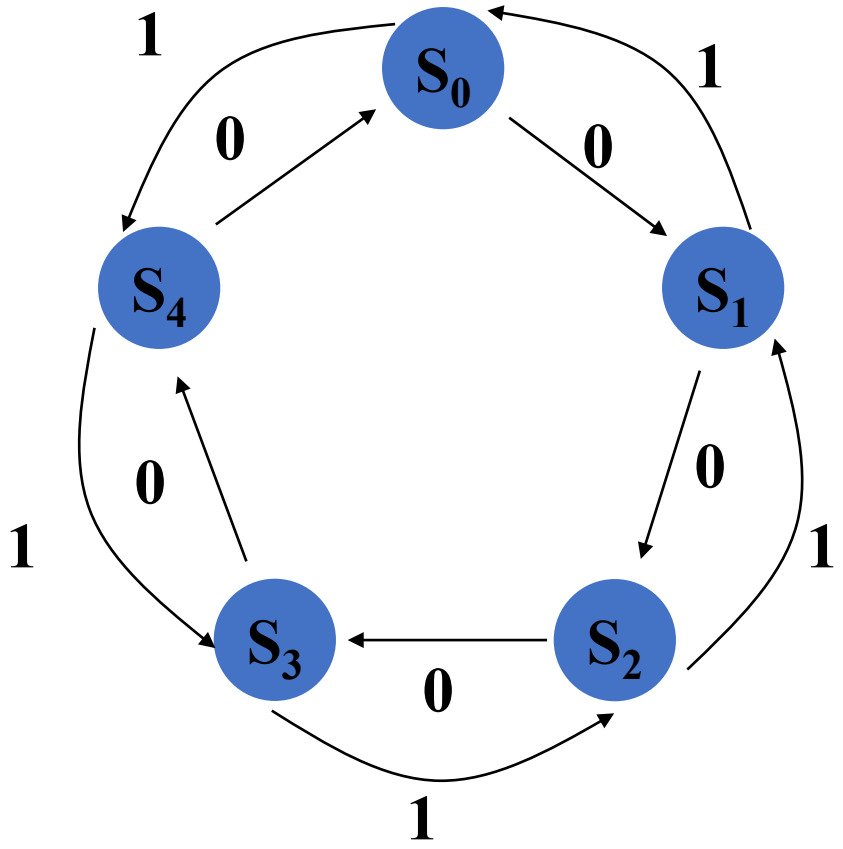
1、 起点—假设一个初态；

2、 输入信号为 n，则每个状态发出 2^n 条带箭头线；

3、 直到不再有新的状态出现。

例1、设计一个五进制可逆计数器。输入 x 为 0 时，加 1 计数；x 为 1 时，减 1 计数。

原始状态图

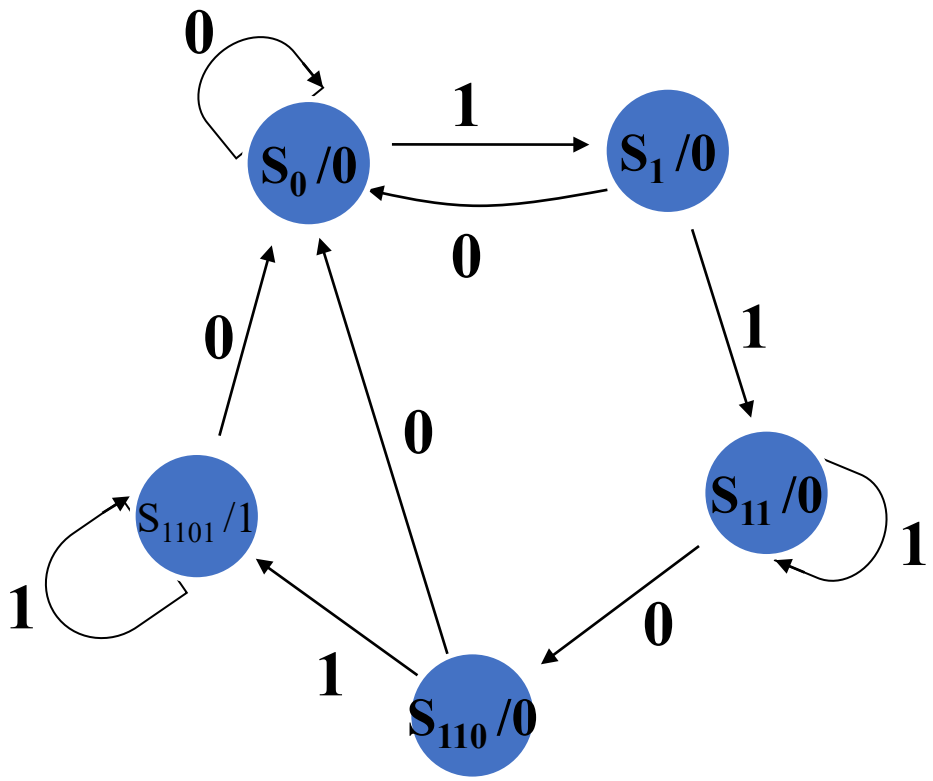


原始状态表

y \ x	0	1
S_0	S_1	S_4
S_1	S_2	S_0
S_2	S_3	S_1
S_3	S_4	S_2
S_4	S_0	S_3

例 2、设计一个“1101”序列检测器。当输入 x 连续出现“1101” (或在出现“1101”后，x 一直保持为1)时，输出 Z = 1；否则 Z = 0 。

原始状态图



原始状态表

y \ x	0	1	z
S_0	S_0	S_1	0
S_1	S_0	S_{11}	0
S_{11}	S_{110}	S_{11}	0
S_{110}	S_0	S_{1101}	0
S_{1101}	S_0	S_{1101}	1

原始状态表

y \ x	0	1
S_0	S_0 /0	S_1 /0
S_1	S_0 /0	S_{11} /0
S_{11}	S_{110} /0	S_{11} /0
S_{110}	S_0 /0	S_{1101} /1
S_{1101}	S_0 /0	S_{1101} /1

由于S_{110}和S_{1101}的次态和输出完全一样，可以合并。

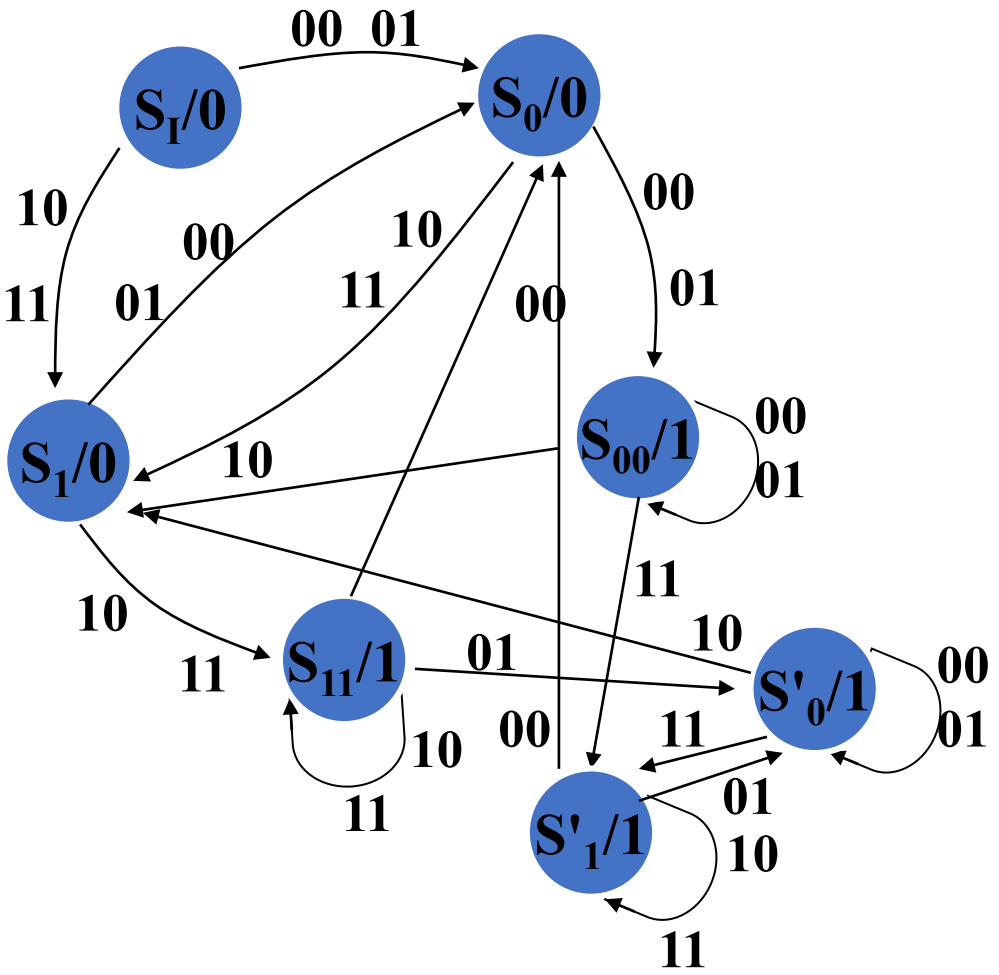
原始状态表

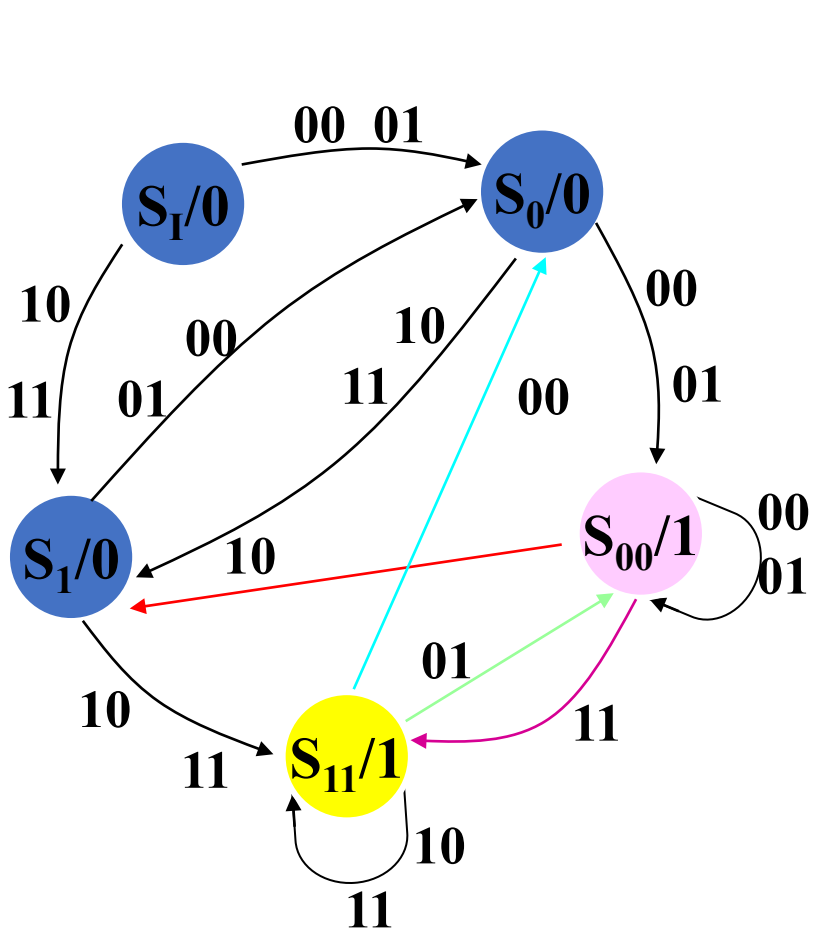
y \ x	0	1
S_0	S_0 /0	S_1 /0
S_1	S_0 /0	S_{11} /0
S_{11}	S_{110} /0	S_{11} /0
S_{110}	S_0 /0	S_{1101} /1
S_{1101}	S_0 /0	S_{1101} /1

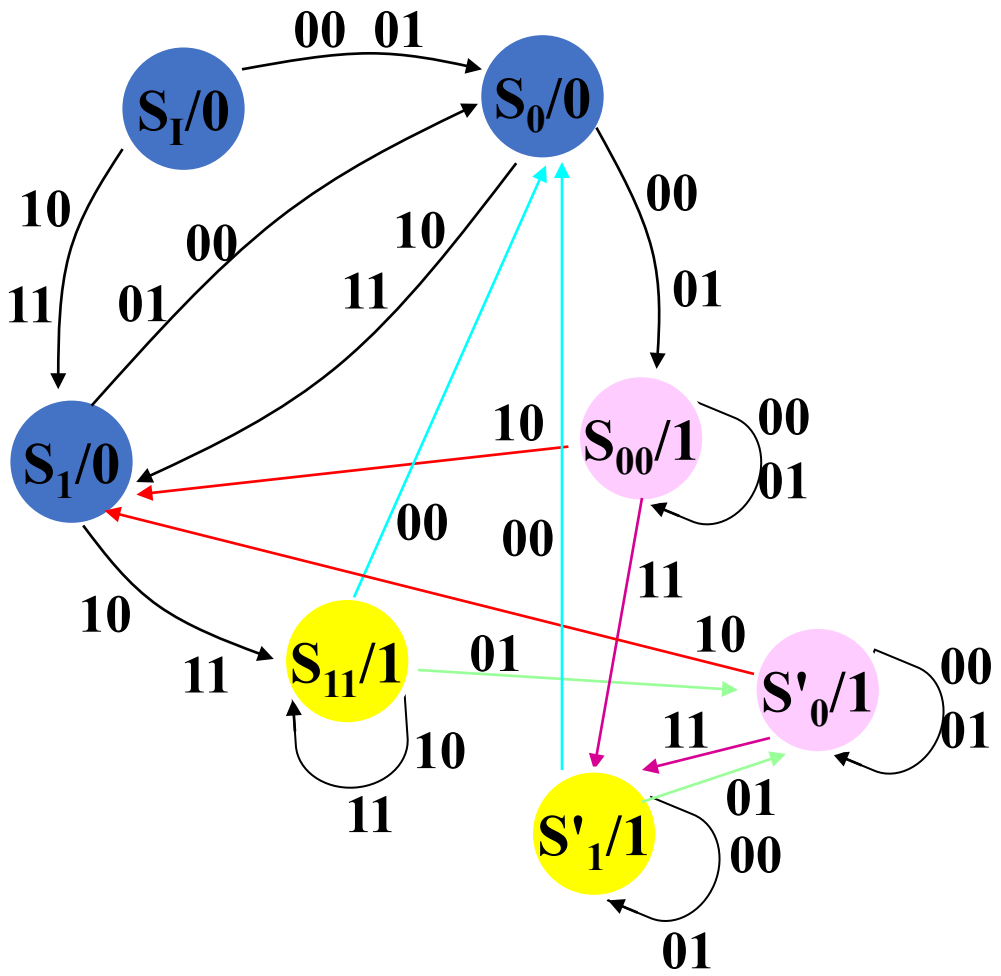
y \ x	0	1
S_0	S_0 /0	S_1 /0
S_1	S_0 /0	S_{11} /0
S_{11}	S_{110} /0	S_{11} /0
S_{110}	S_0 /0	S_{110} /1

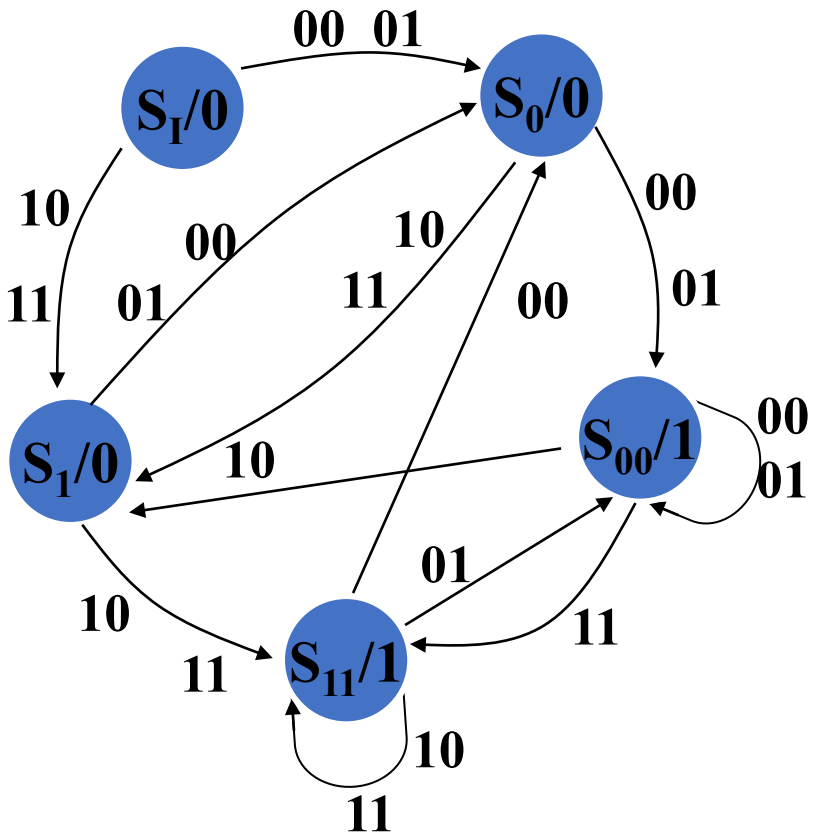
y \ x	0	1	z
S_0	S_0	S_1	0
S_1	S_0	S_{11}	0
S_{11}	S_{110}	S_{11}	0
S_{110}	S_0	S_{1101}	0
S_{1101}	S_0	S_{1101}	1

例3、电路输入x和y，输出z。如果x连续两次输入同样的值，z=1，在此之后如果y输入一直为1，则z为1；否则，z=0。




$S_I/0$
$S_0/0$
$S_1/0$
$S_{00}/1$
$S_{11}/1$
00 01
10
11
00
01
10
11
00
01
10
11
01
10
11
00
01
10
11


$S_I/0$
$S_0/0$
$S_1/0$
$S_{00}/1$
$S_{11}/1$
$S'_0/1$
$S'_1/1$
00 01
10
11
00
01
10
11
00
01
10
11
00
00
11
01
10
11
00
01
10
11
00
01



S \ xy	00	01	10	11	z
S_I	S_0	S_0	S_1	S_1	0
S_0	S_{00}	S_{00}	S_1	S_1	0
S_1	S_0	S_0	S_{11}	S_{11}	0
S_{00}	S_{00}	S_{00}	S_1	S_{11}	1
S_{11}	S_0	S_{00}	S_{11}	S_{11}	1

5.2 状态化简

5.2.1 完全给定同步时序电路状态表的化简

1、等效：S_1和S_2是完全给定时序电路M_1和M_2(M_1和M_2可以是同一个电路)的两个状态，作为初态同时加入任意输入序列，产生的输出完全一致，则S_1和S_2是等效对。等效状态可以合并为一个状态。

即：$(S_1, S_2) \rightarrow S$

2、等效的传递性：$(S_1, S_2), (S_2, S_3) \rightarrow (S_1, S_3)$

3、等效类：所含状态都可以相互构成等效对的等效状态的集合。即：

$(S_1, S_2, S_3) \rightarrow (S_1, S_2)(S_2, S_3)(S_1, S_3)$

$(S_1, S_2)(S_2, S_3)(S_1, S_3) \rightarrow (S_1, S_2, S_3)$

4、最大等效类：在一个原始状态表中，不能被其他等效类所包含的等效类称为最大等效类。

等效对的判断标准

条件1：它们的输出完全相同。

条件2：它们的次态满足下列条件之一：

① 次态相同

② 次态交错

③ 次态维持

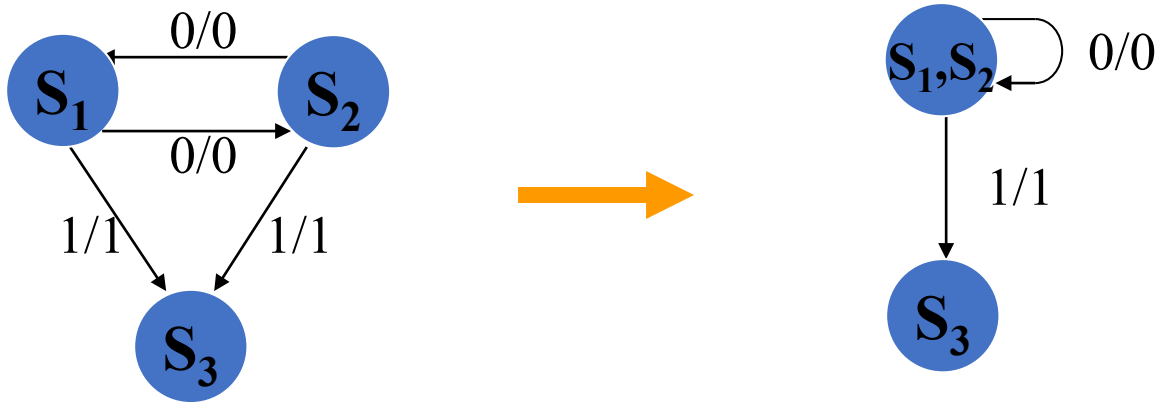
④ 后续状态等效

⑤ 次态循环

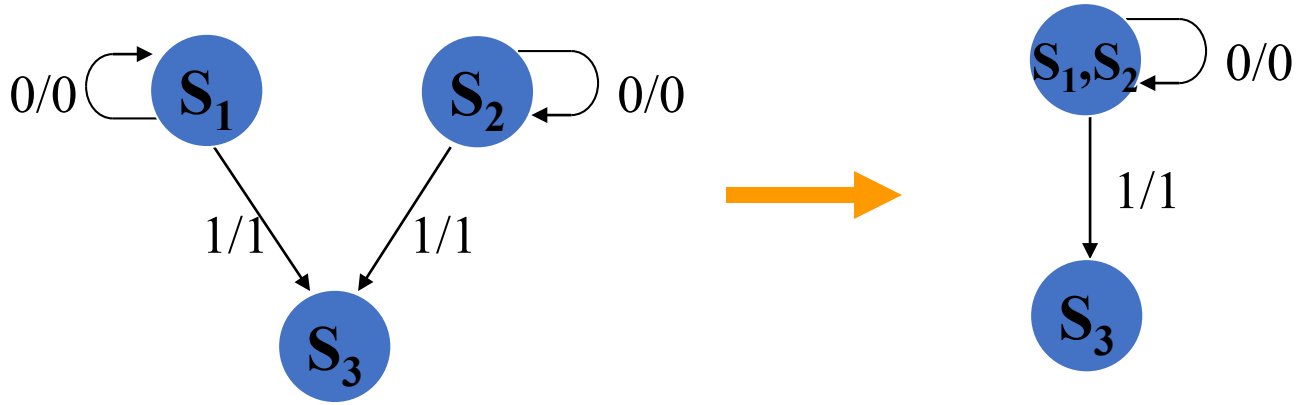
① 次态相同


S1
S2
S3
S4
0/0
1/0
0/0
1/0
S1,S2
0/0
1/0
S3
S4

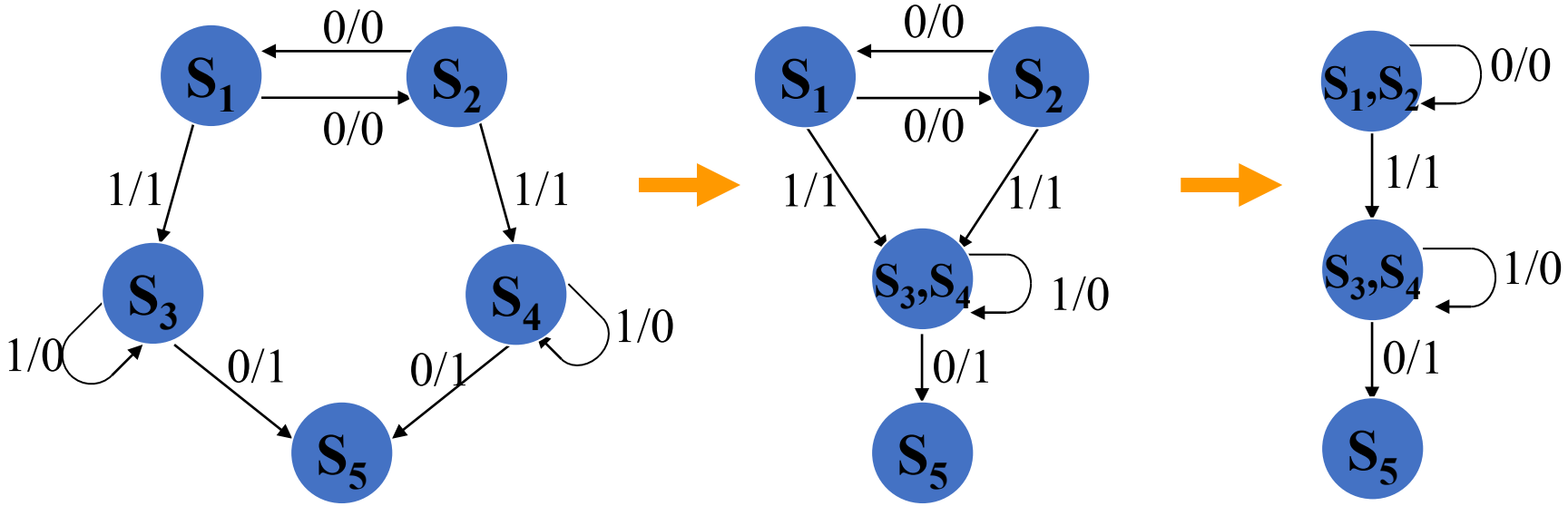
② 次态交错


S1
S2
S3
0/0
0/0
1/1
1/1
S1,S2
0/0
1/1
S3

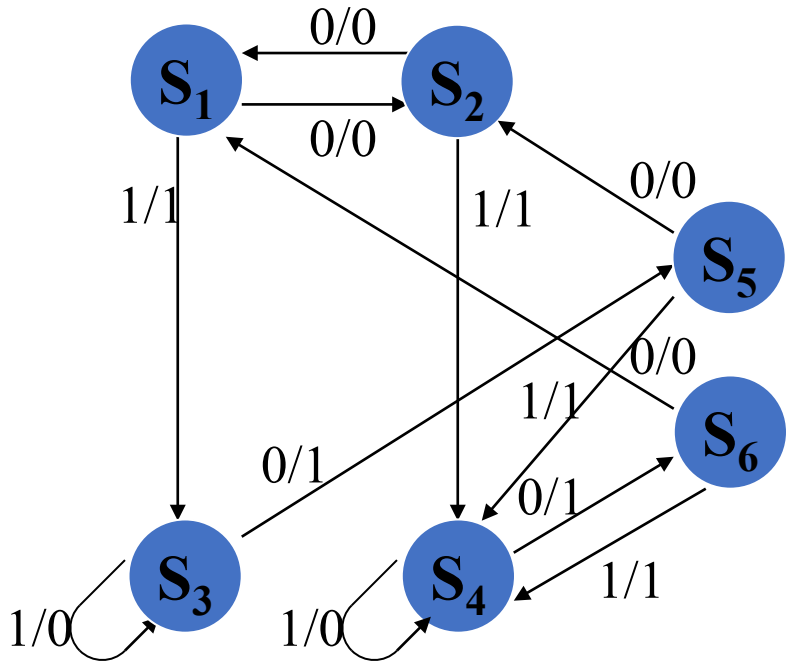
③ 次态维持

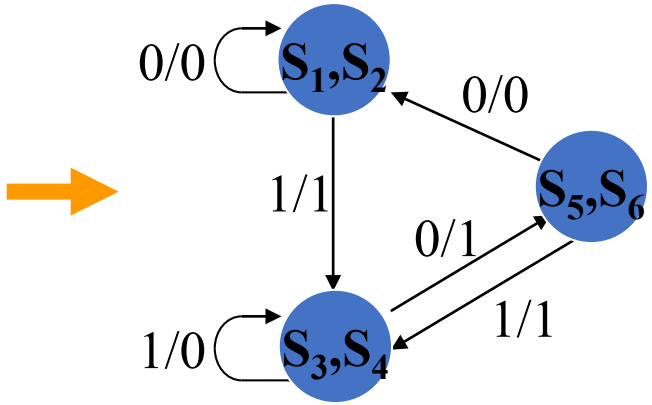


④ 后继状态等效

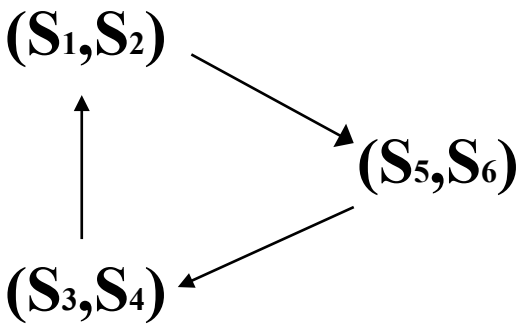


⑤ 次态循环





图中次态的等效依赖关系



利用隐含表进行状态化简

例、化简下图所示的原始状态表

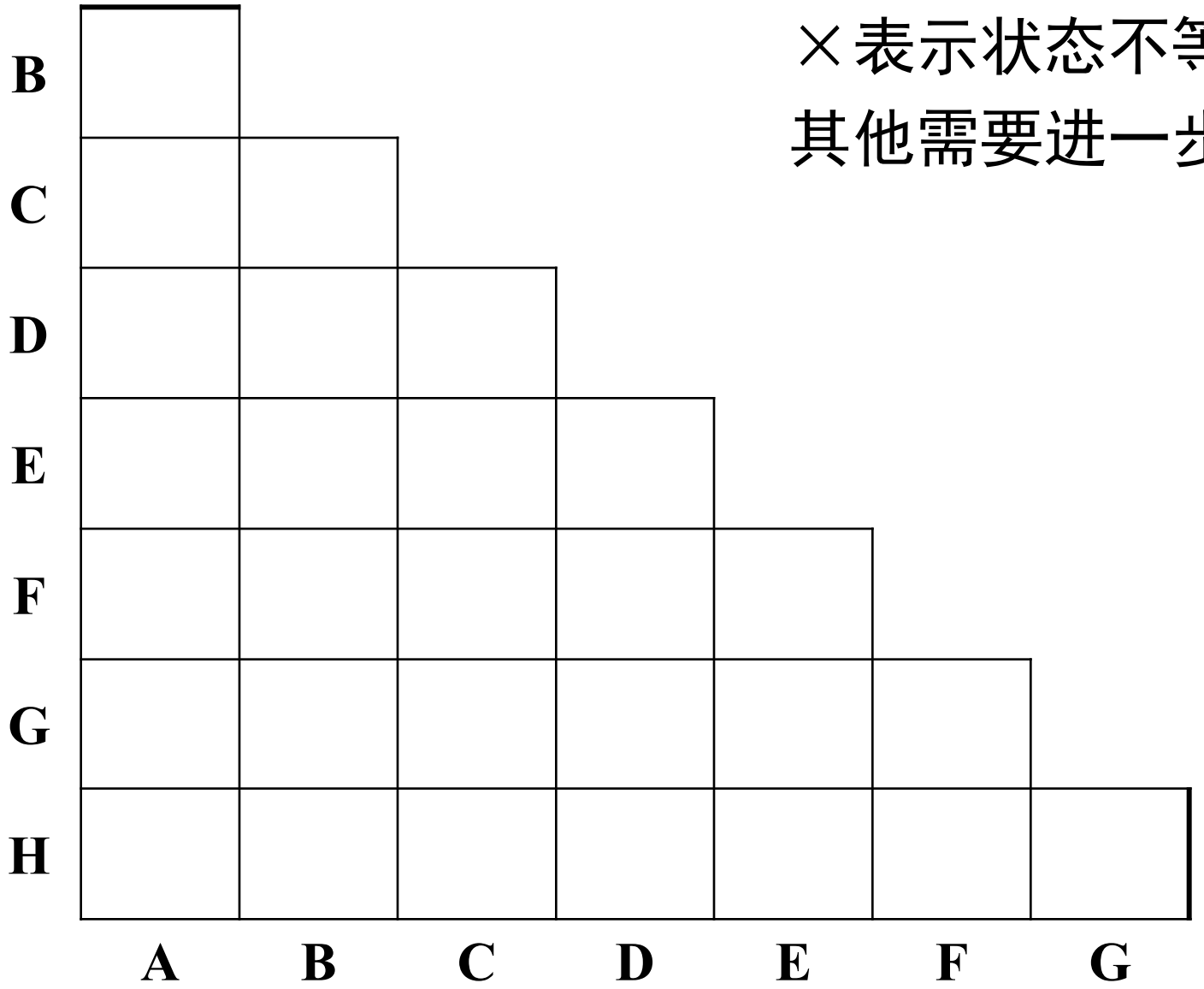
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

（1）画隐含表(缺头少尾表)

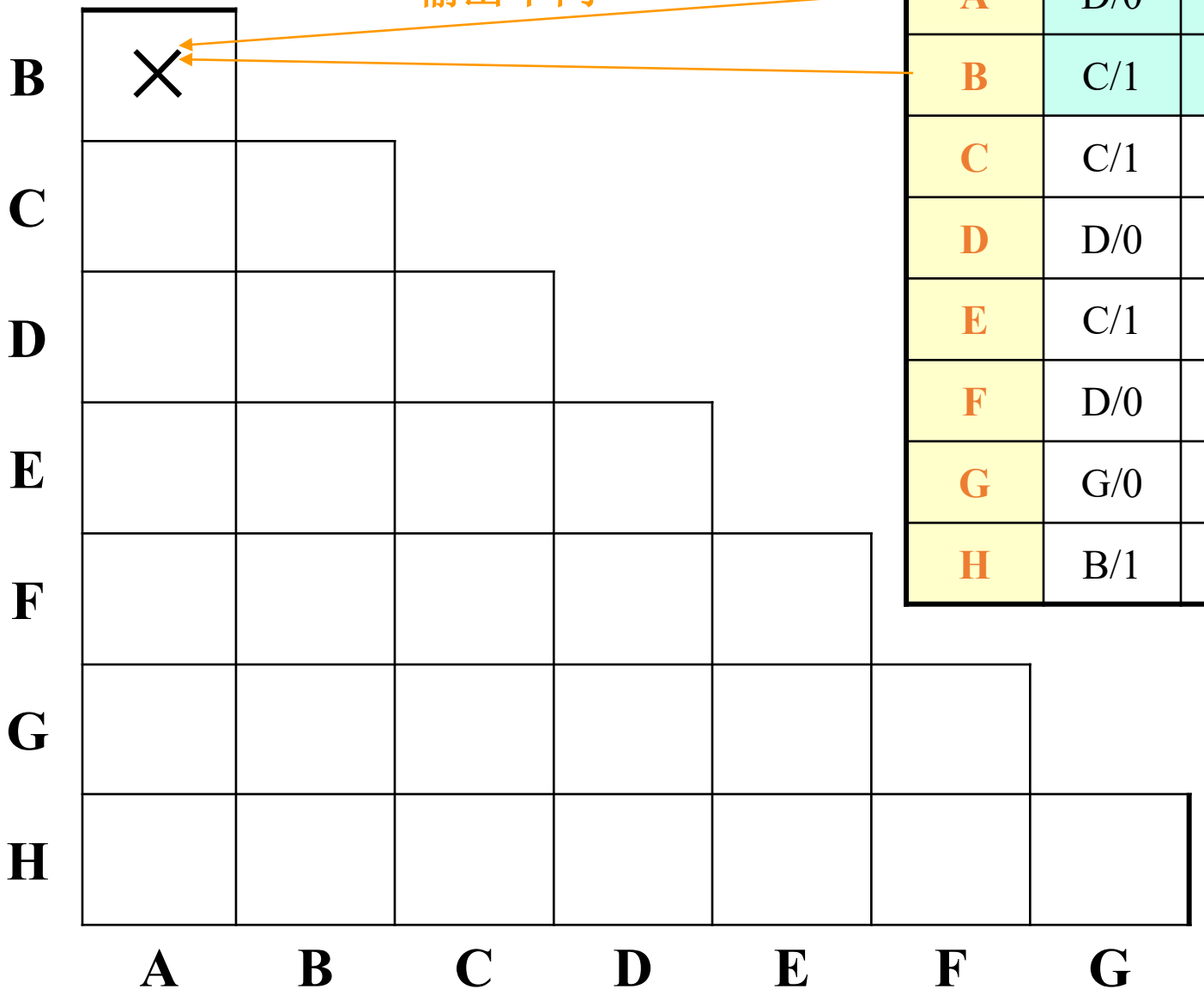
隐含表中有三种状态结果：

×表示状态不等效；√表示状态等效；

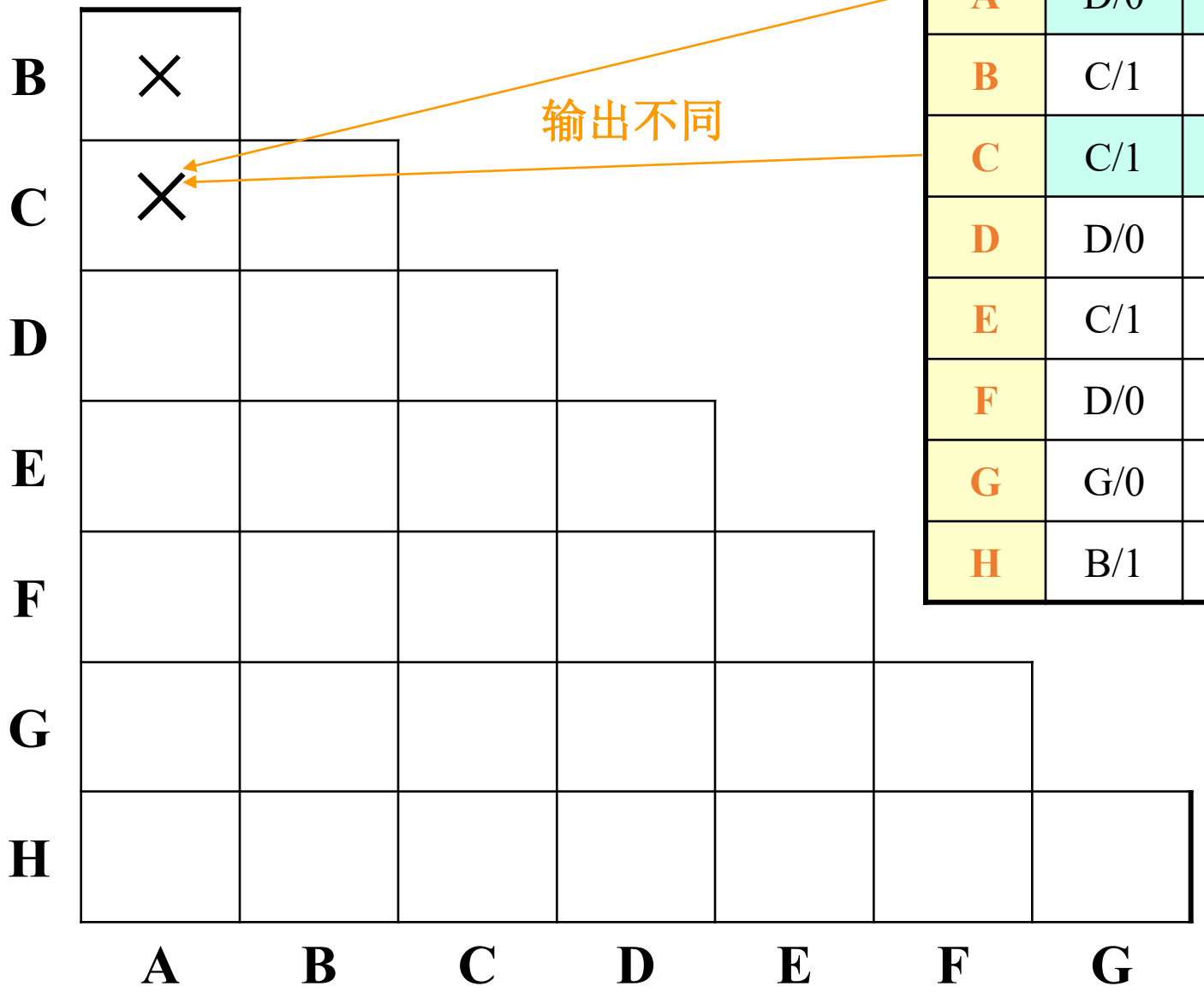
其他需要进一步确定状态对是否等效。



（2）进行顺序比较 AB

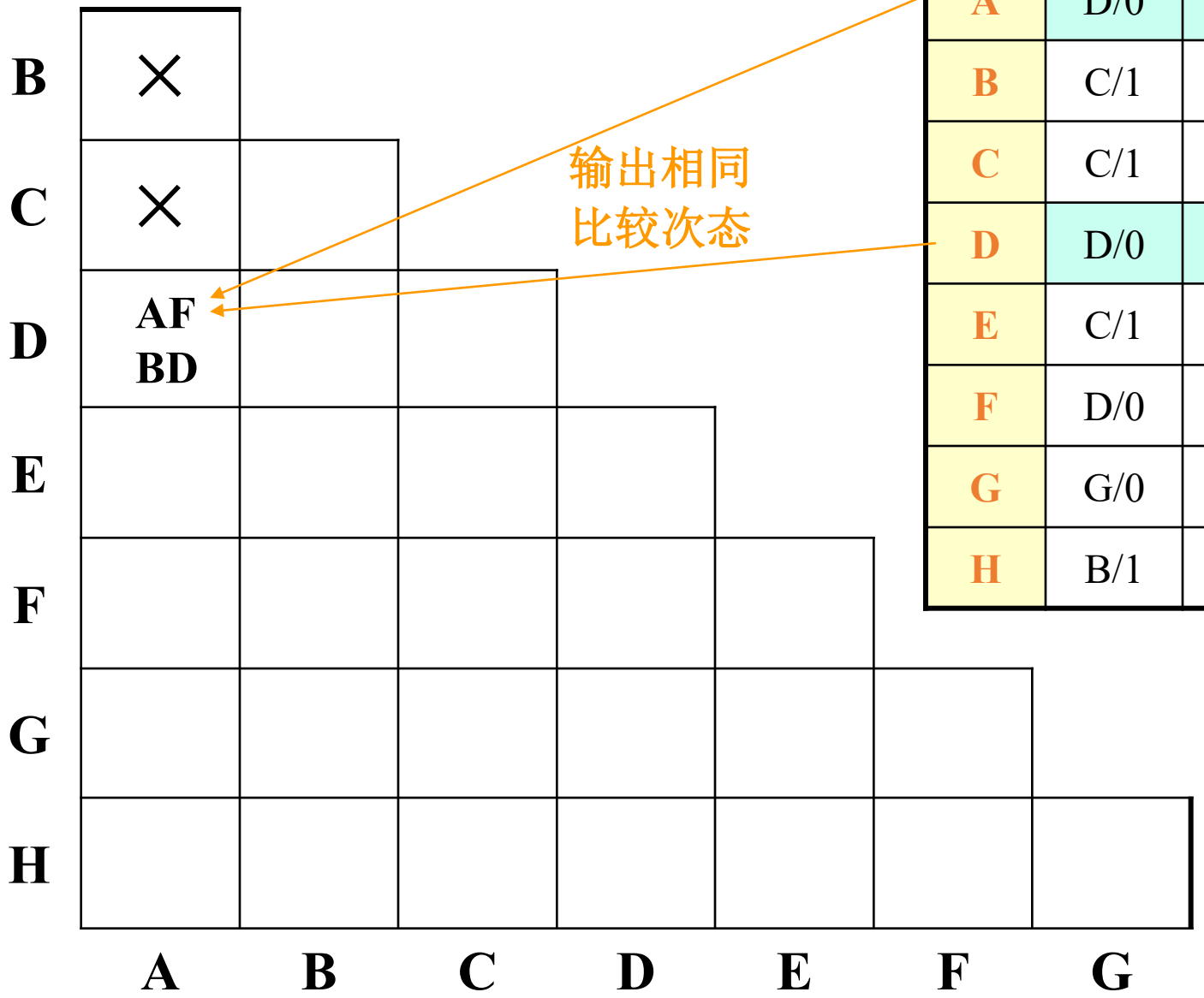


y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



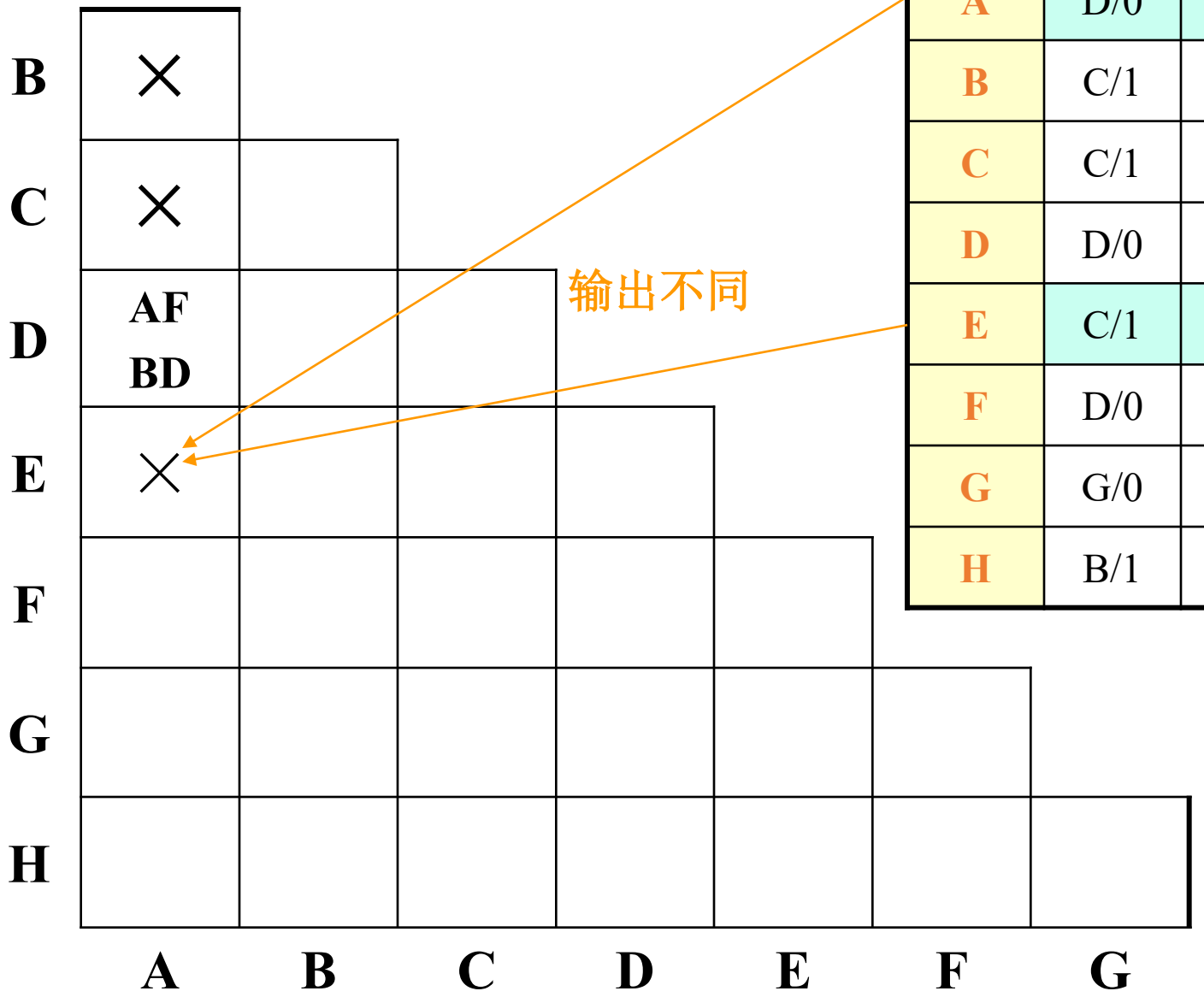
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



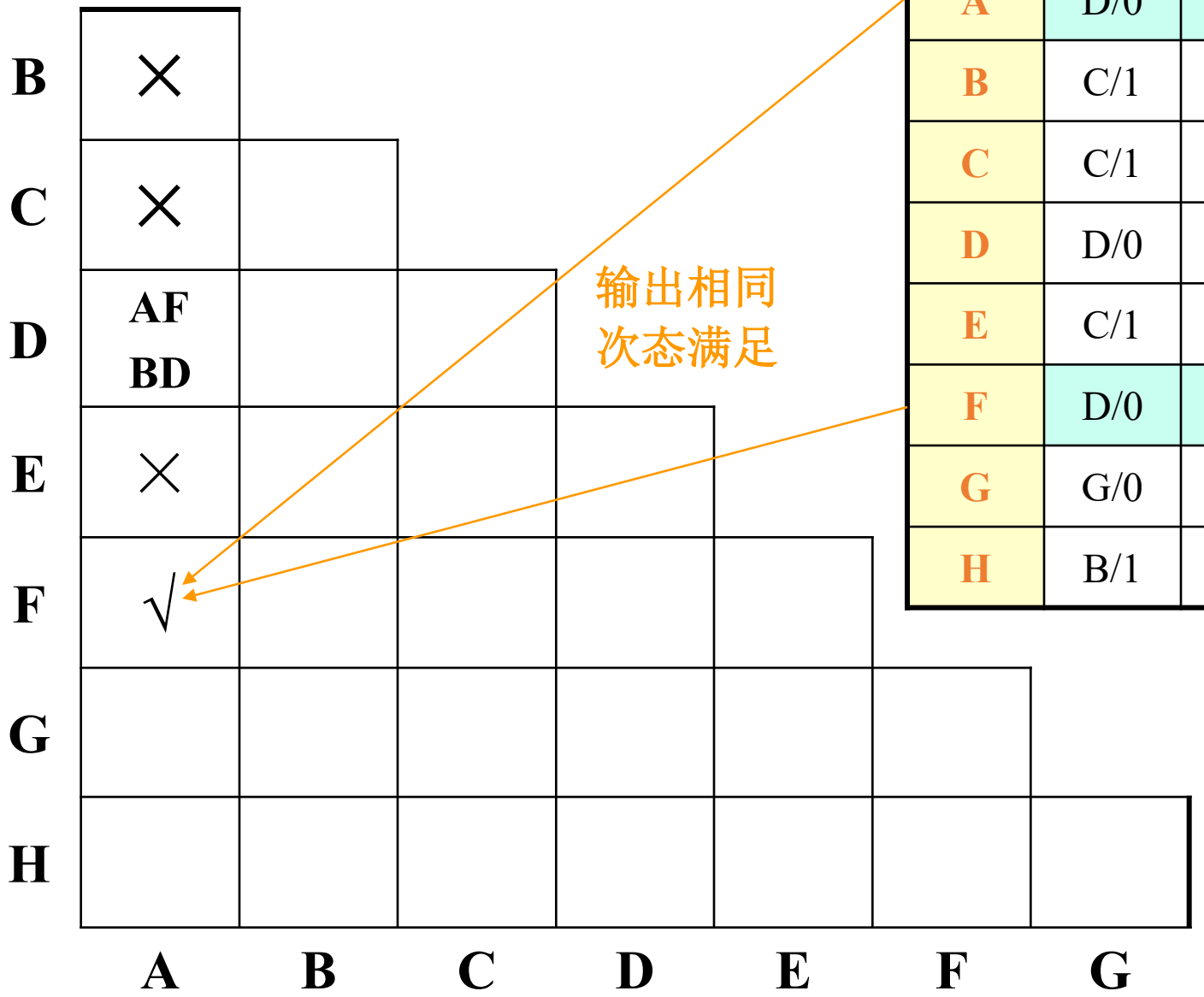
输出相同
比较次态

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



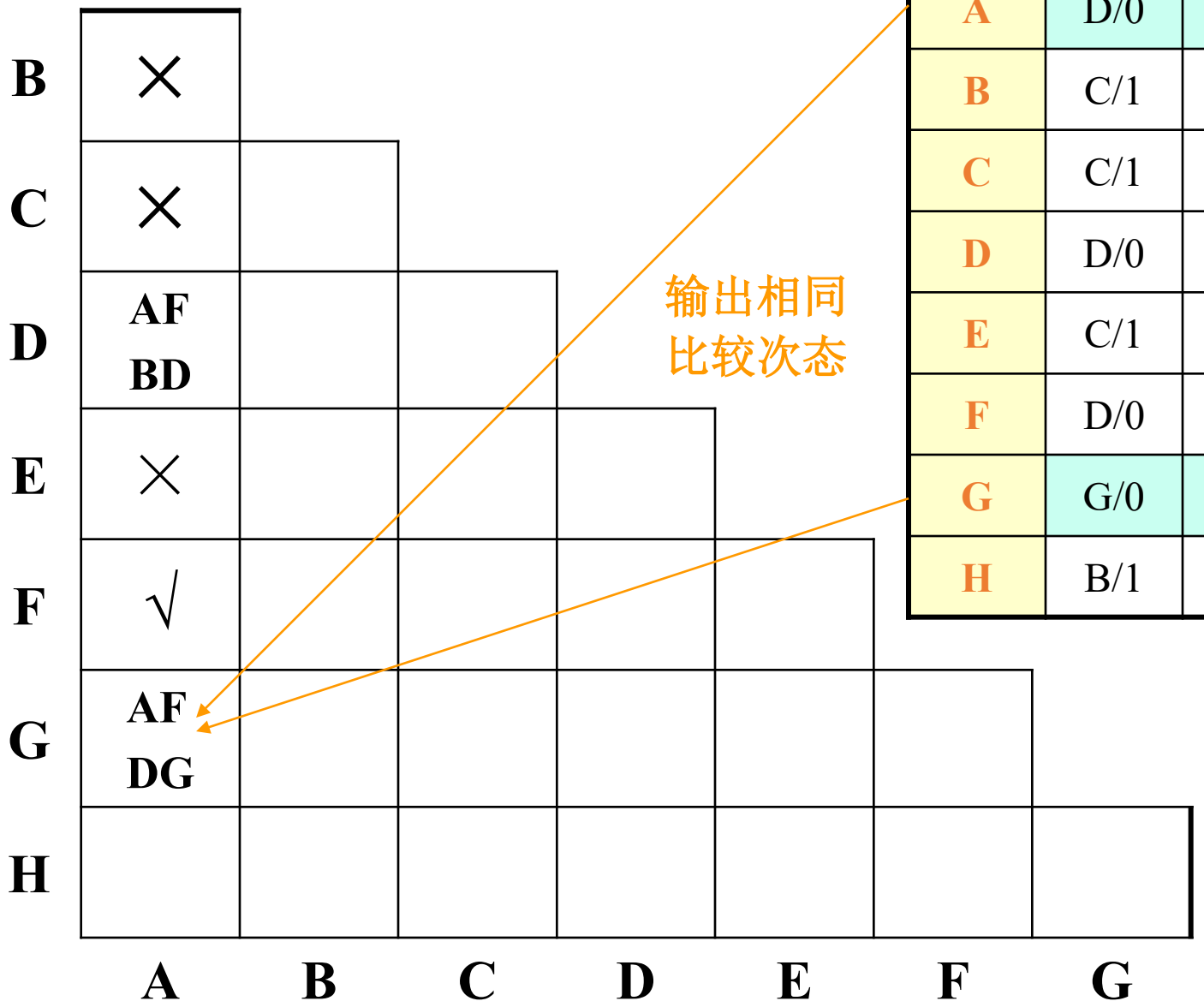
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



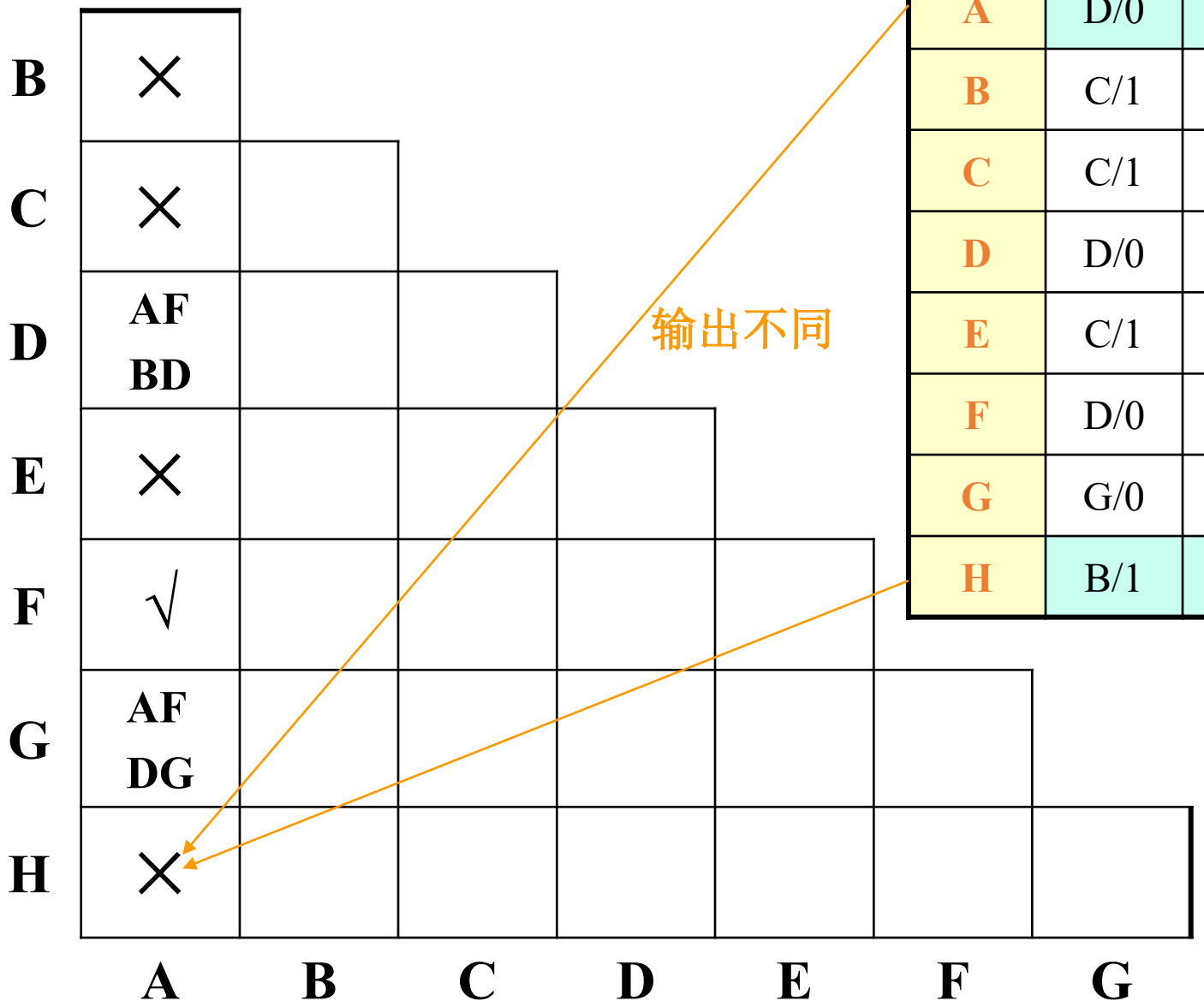
输出相同
次态满足

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



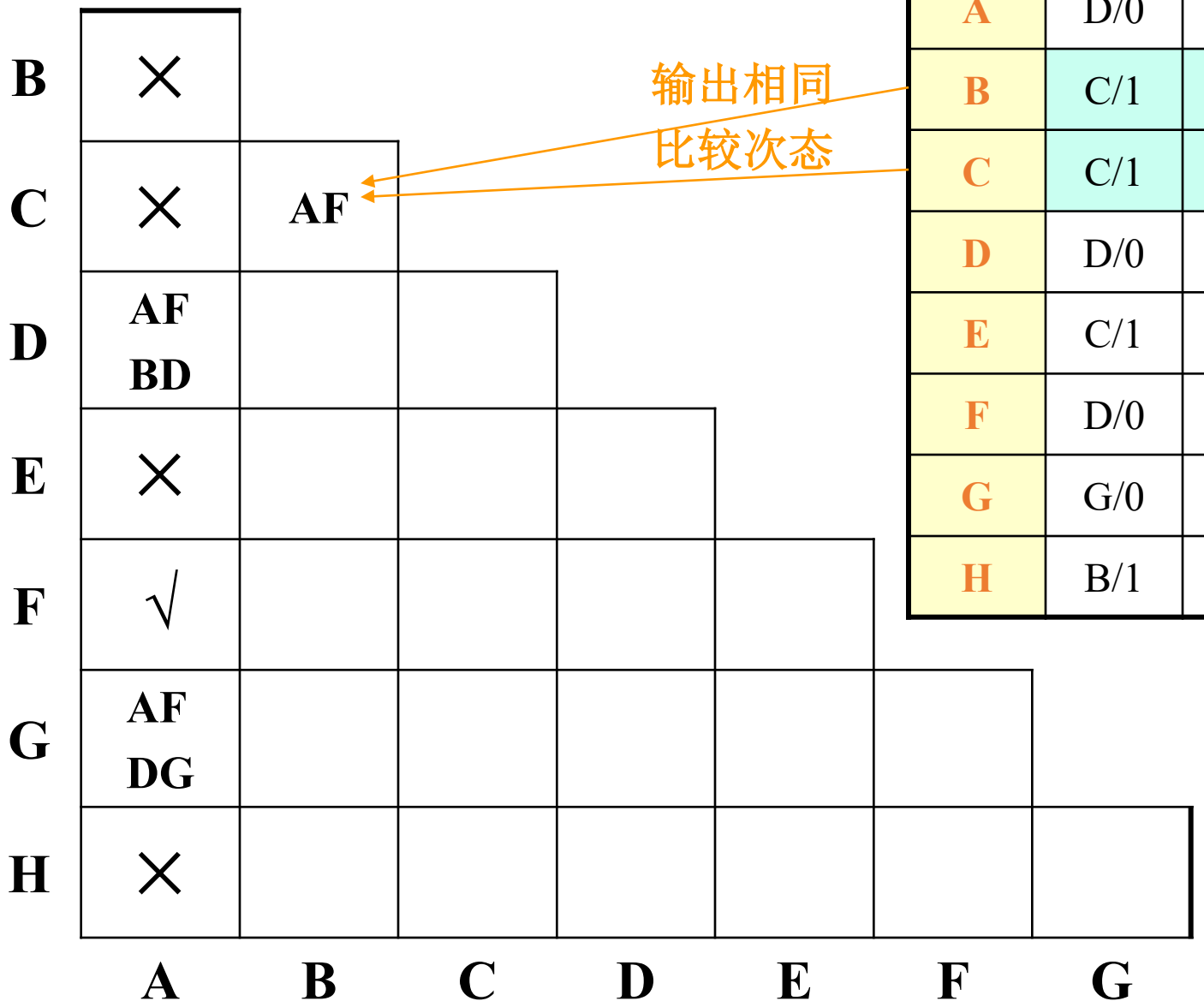
输出相同
比较次态

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



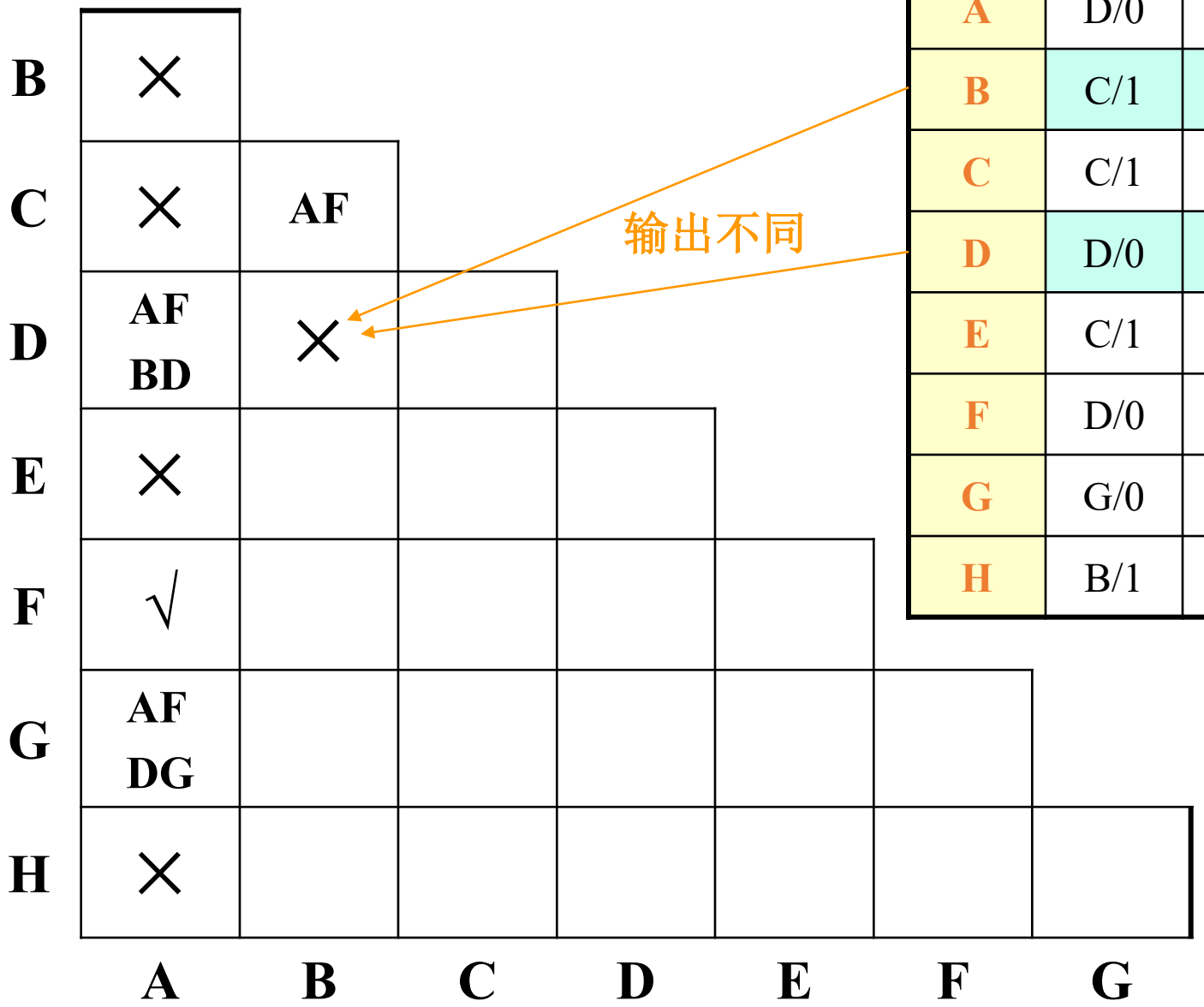
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



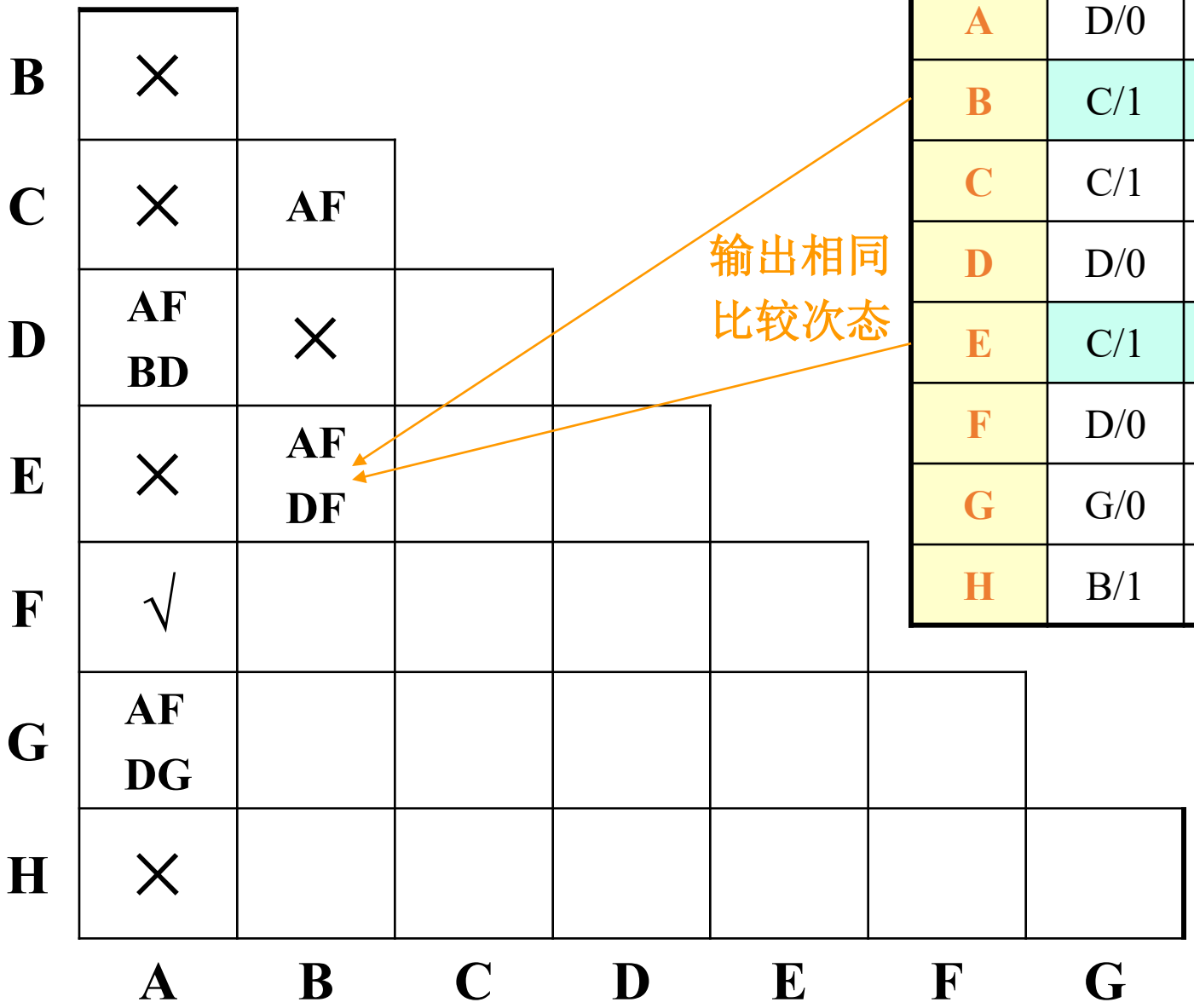
输出相同
比较次态

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



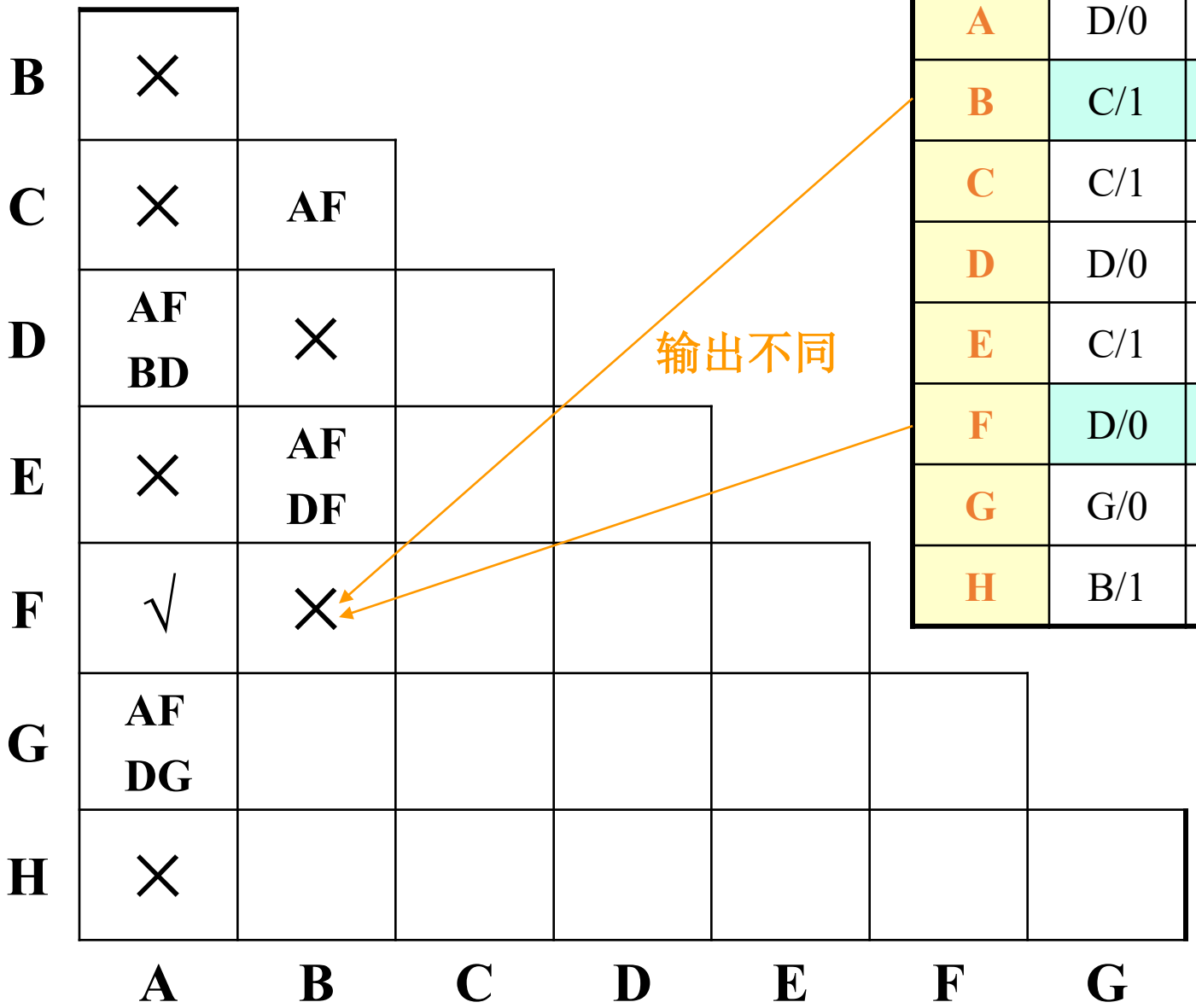
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



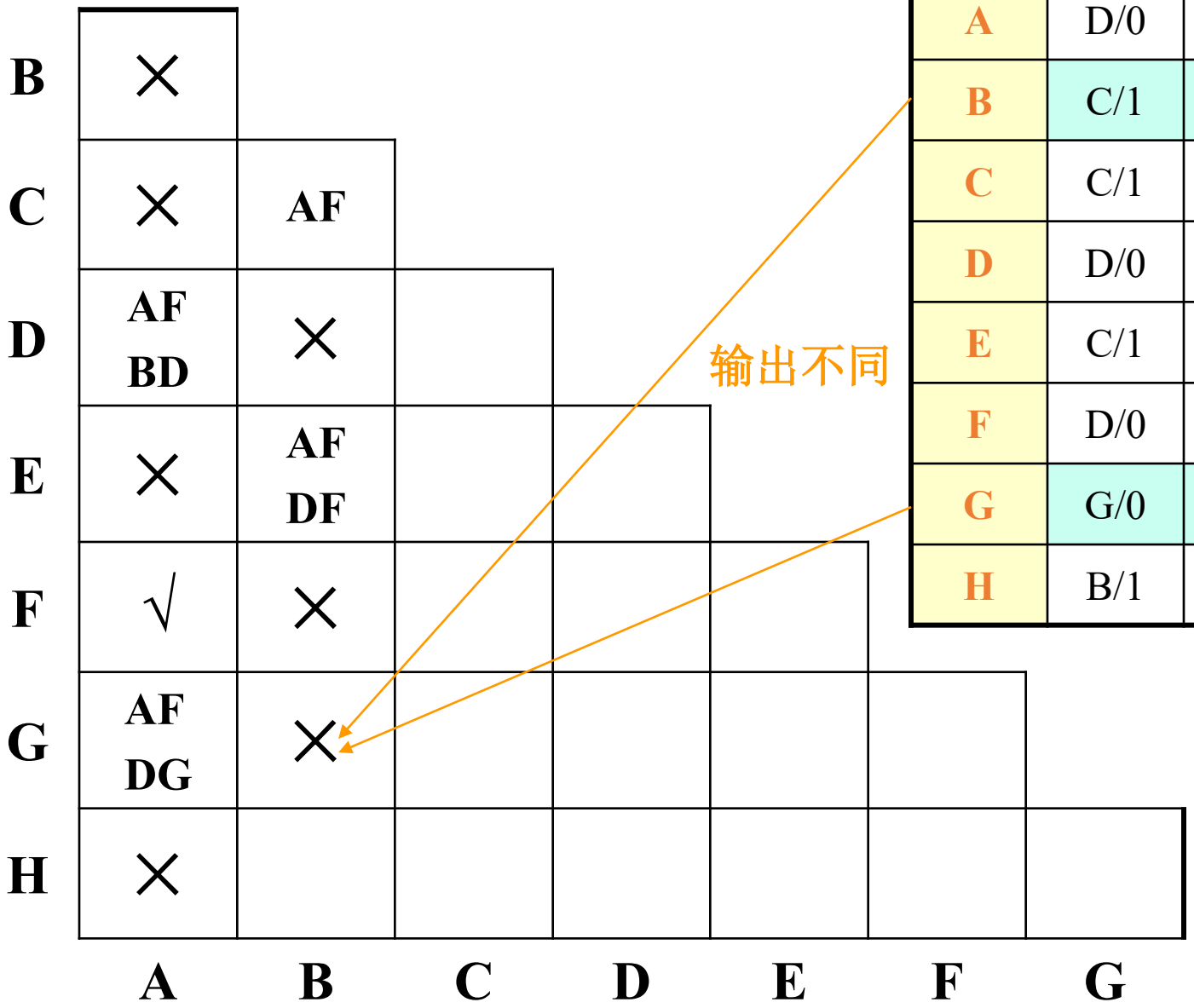
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出相同
比较次态



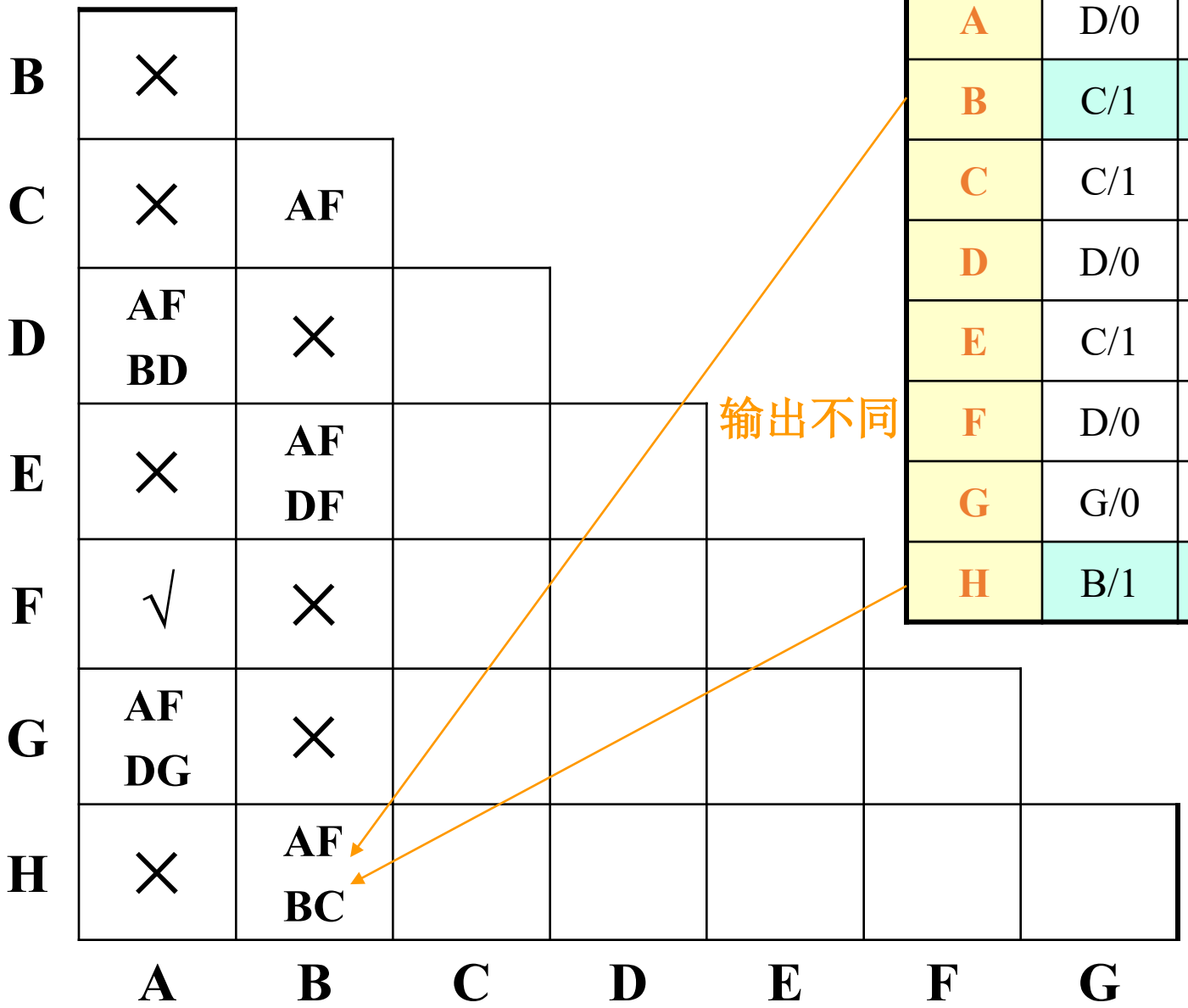
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



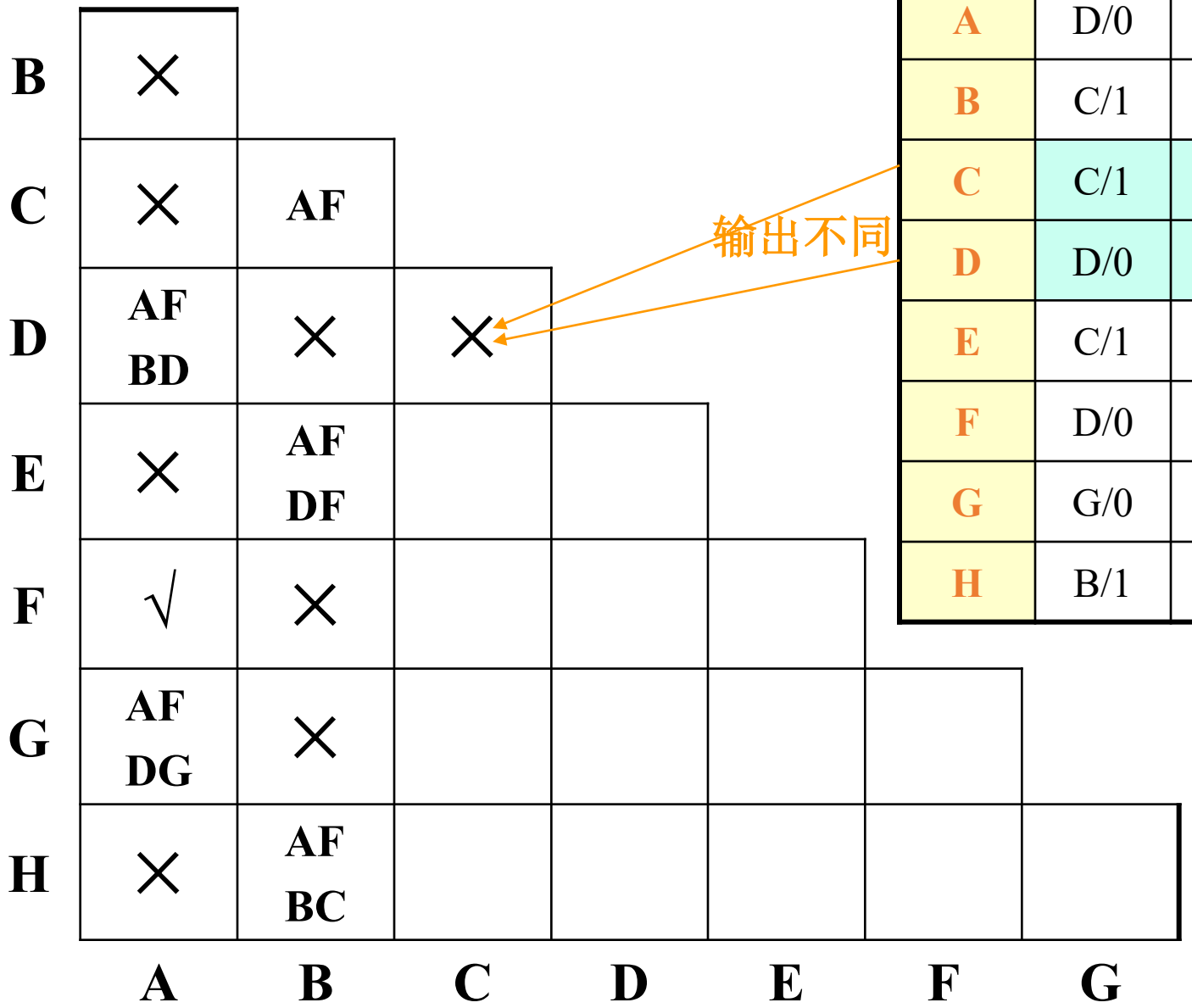
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



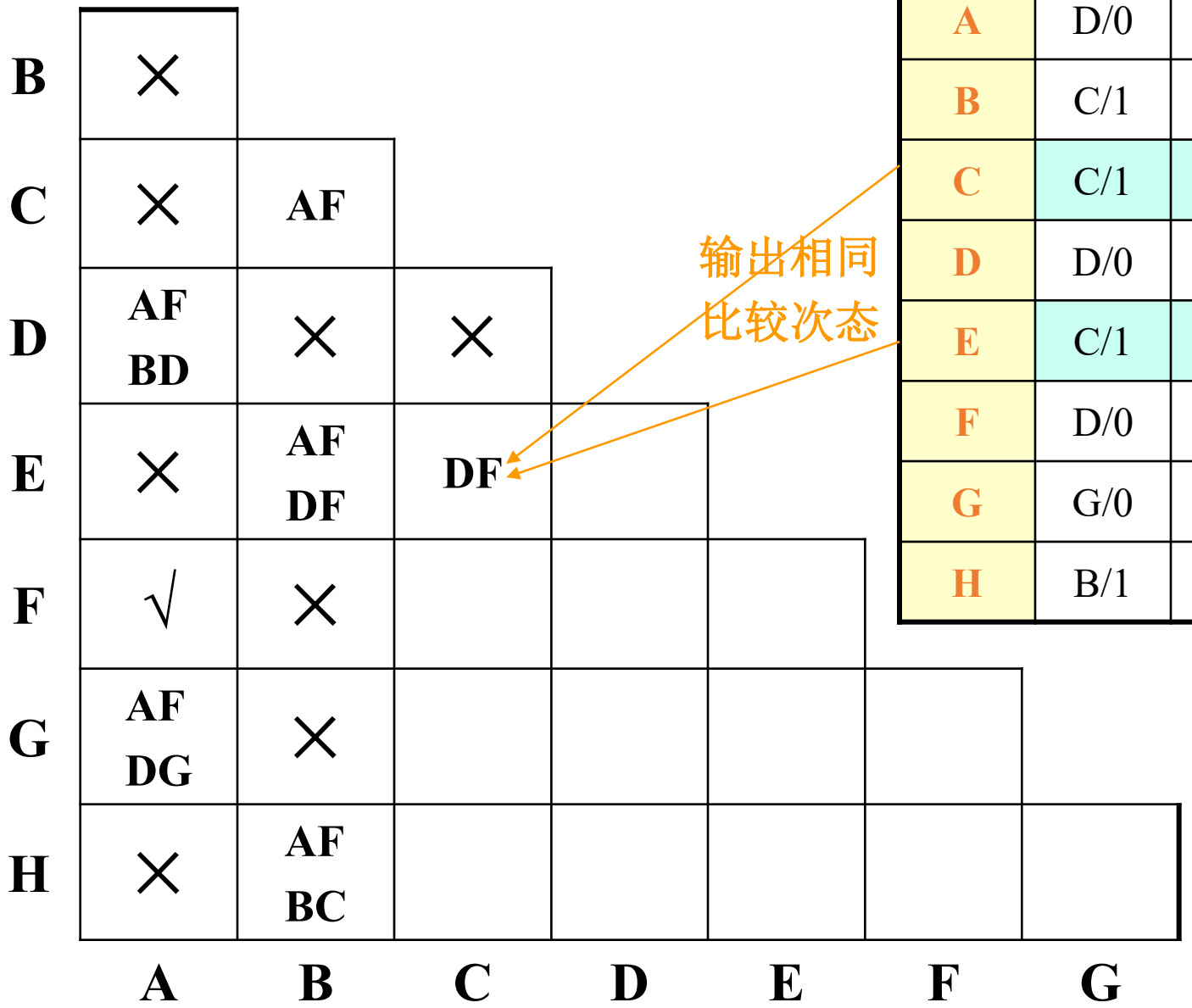
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



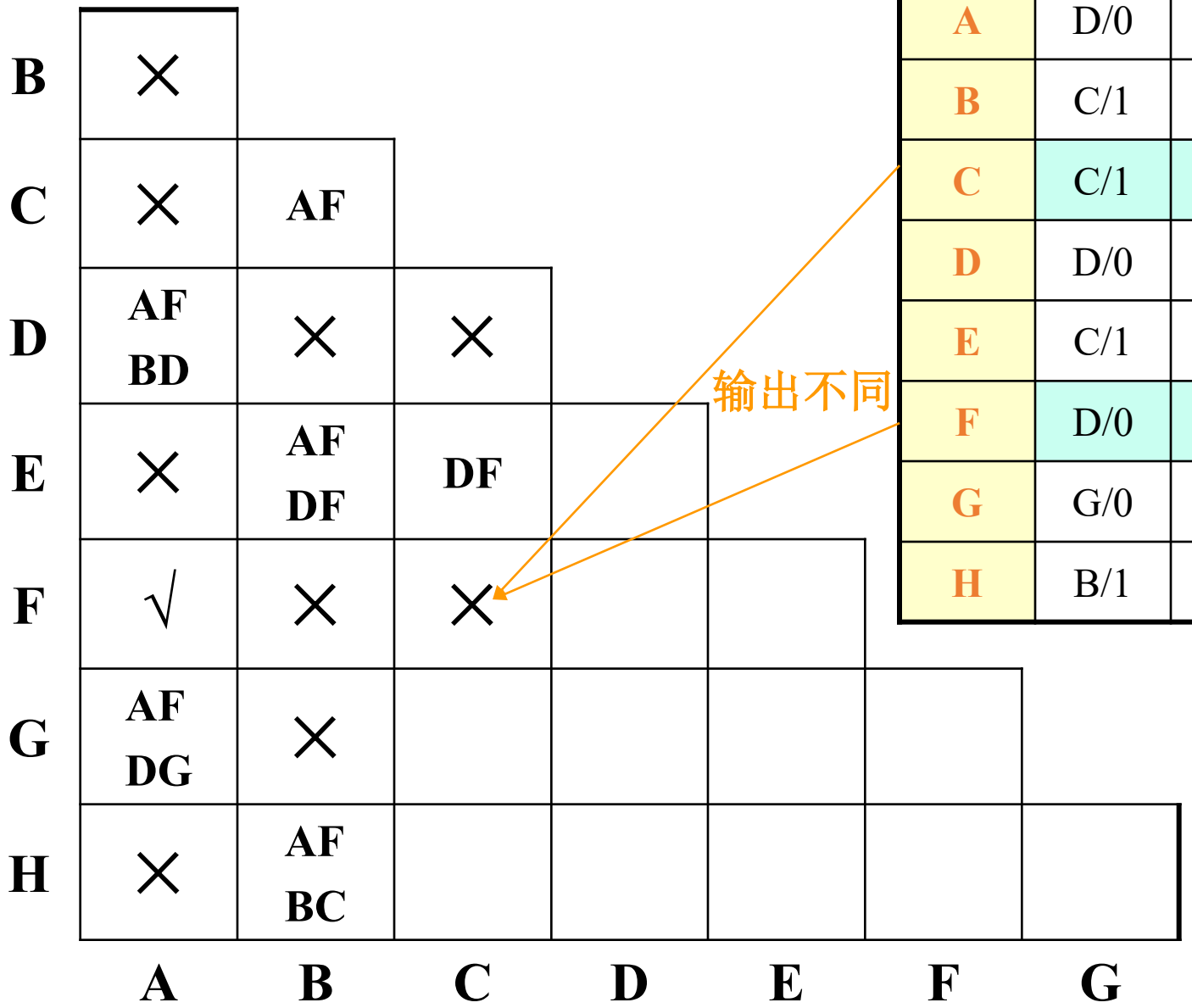
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



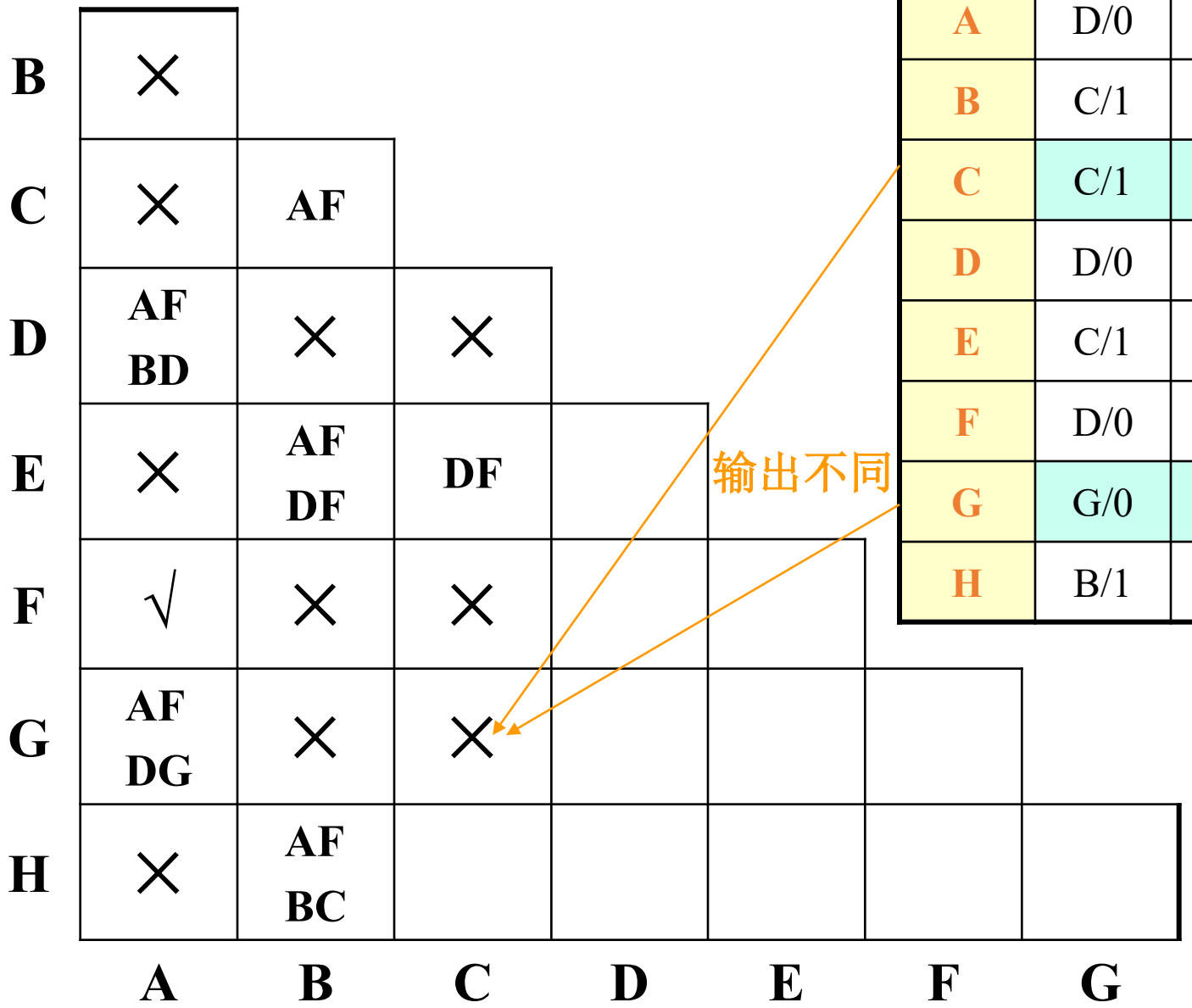
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出相同
比较次态



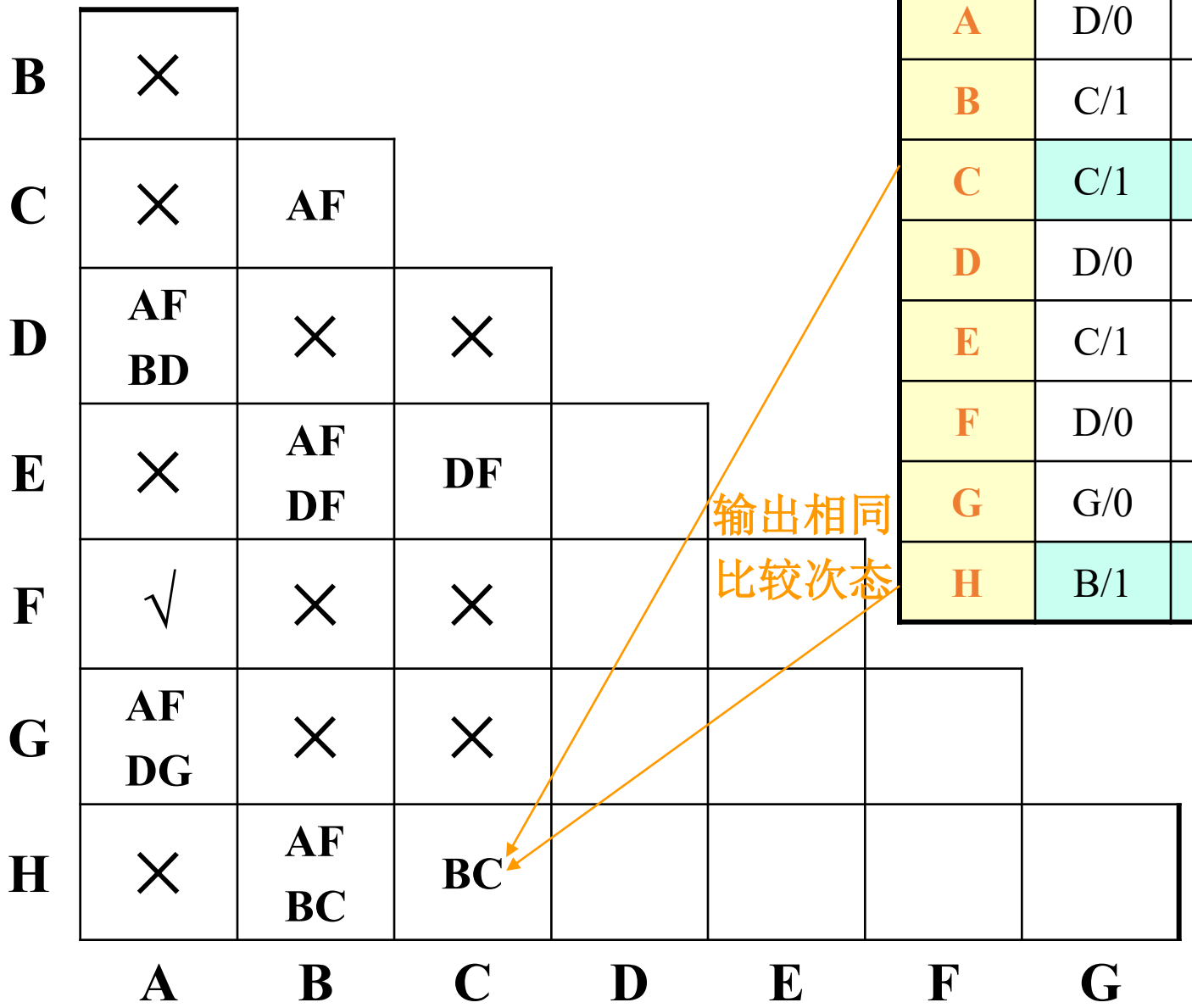
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出不同



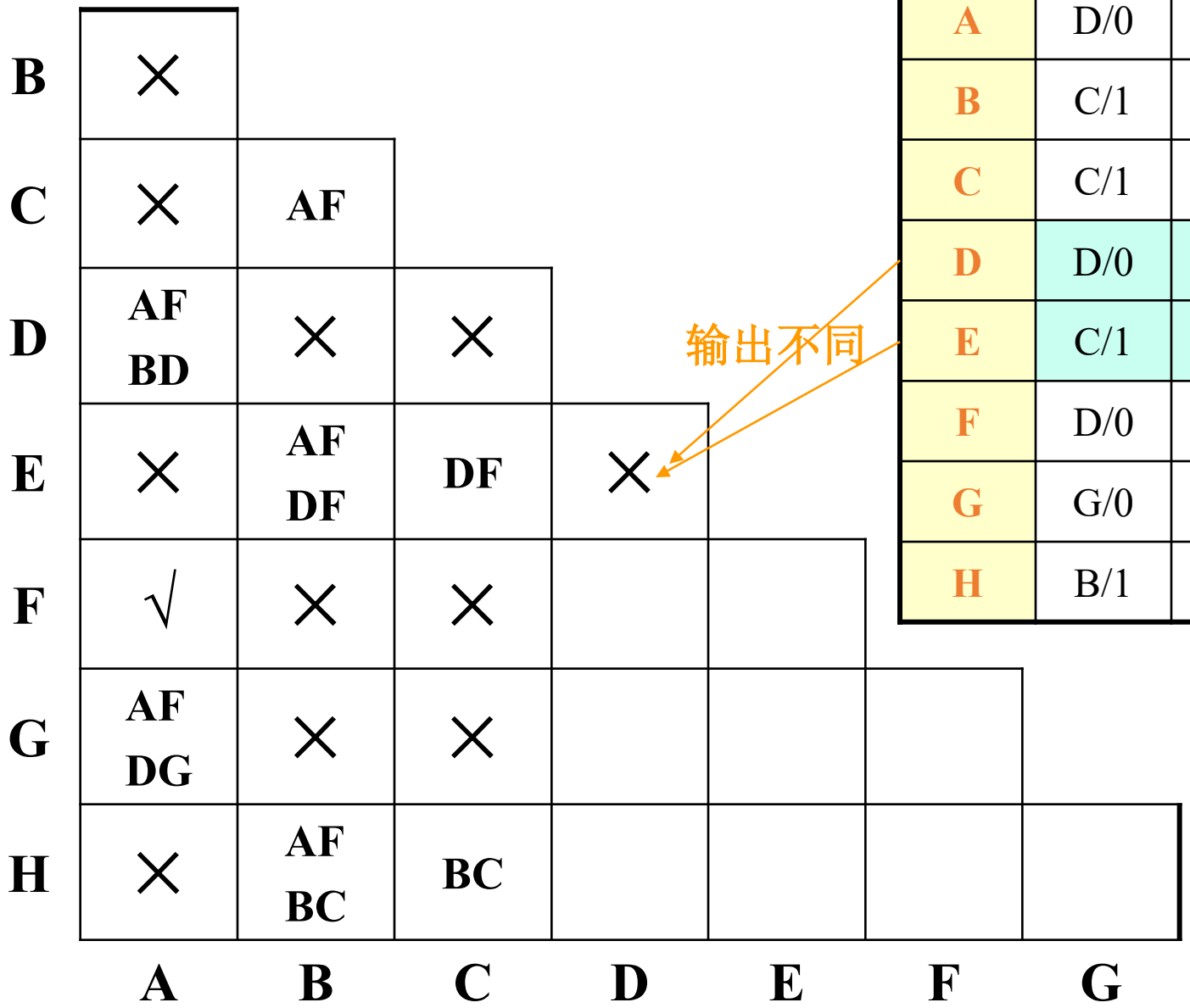
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出不同



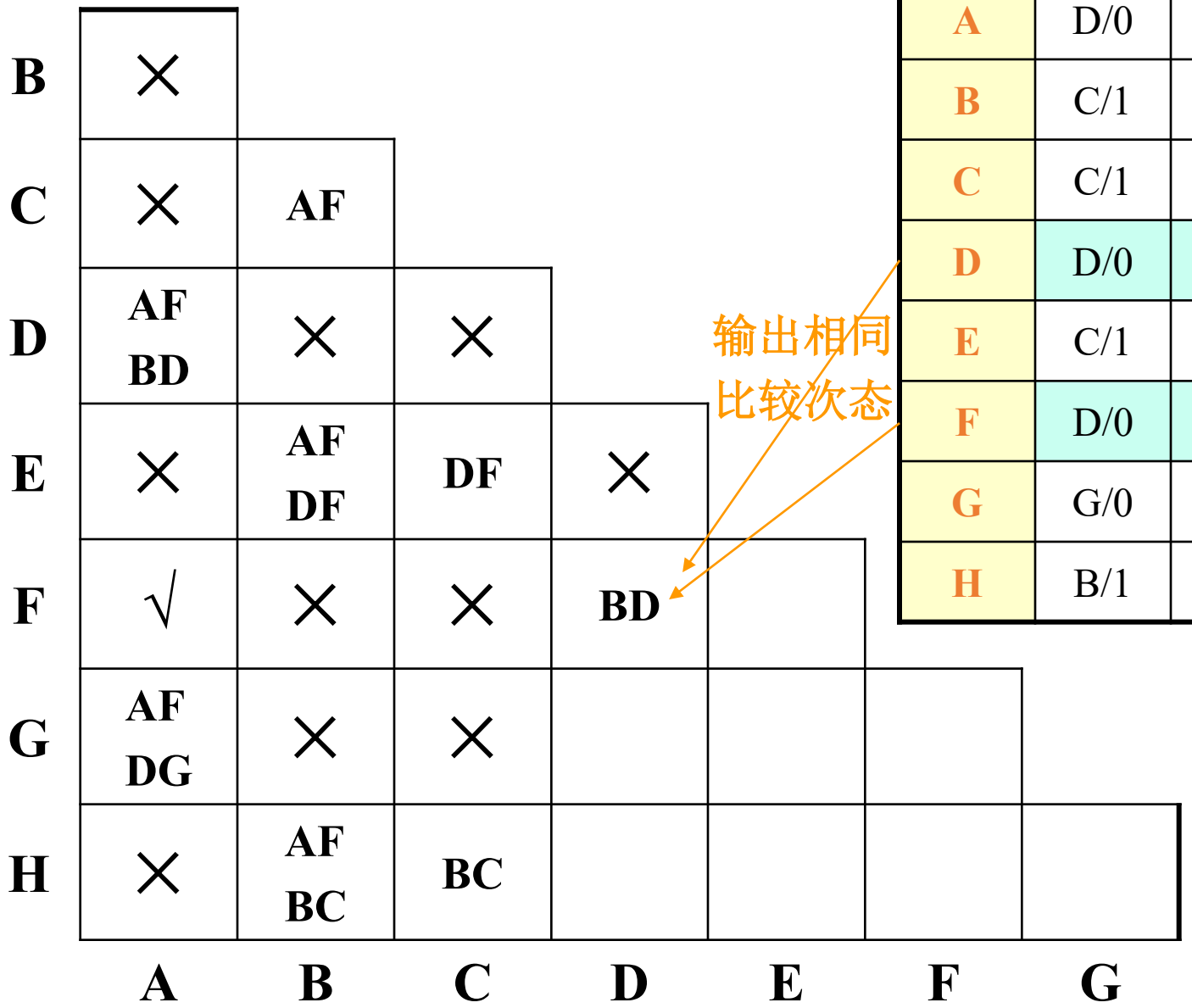
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出相同
比较次态



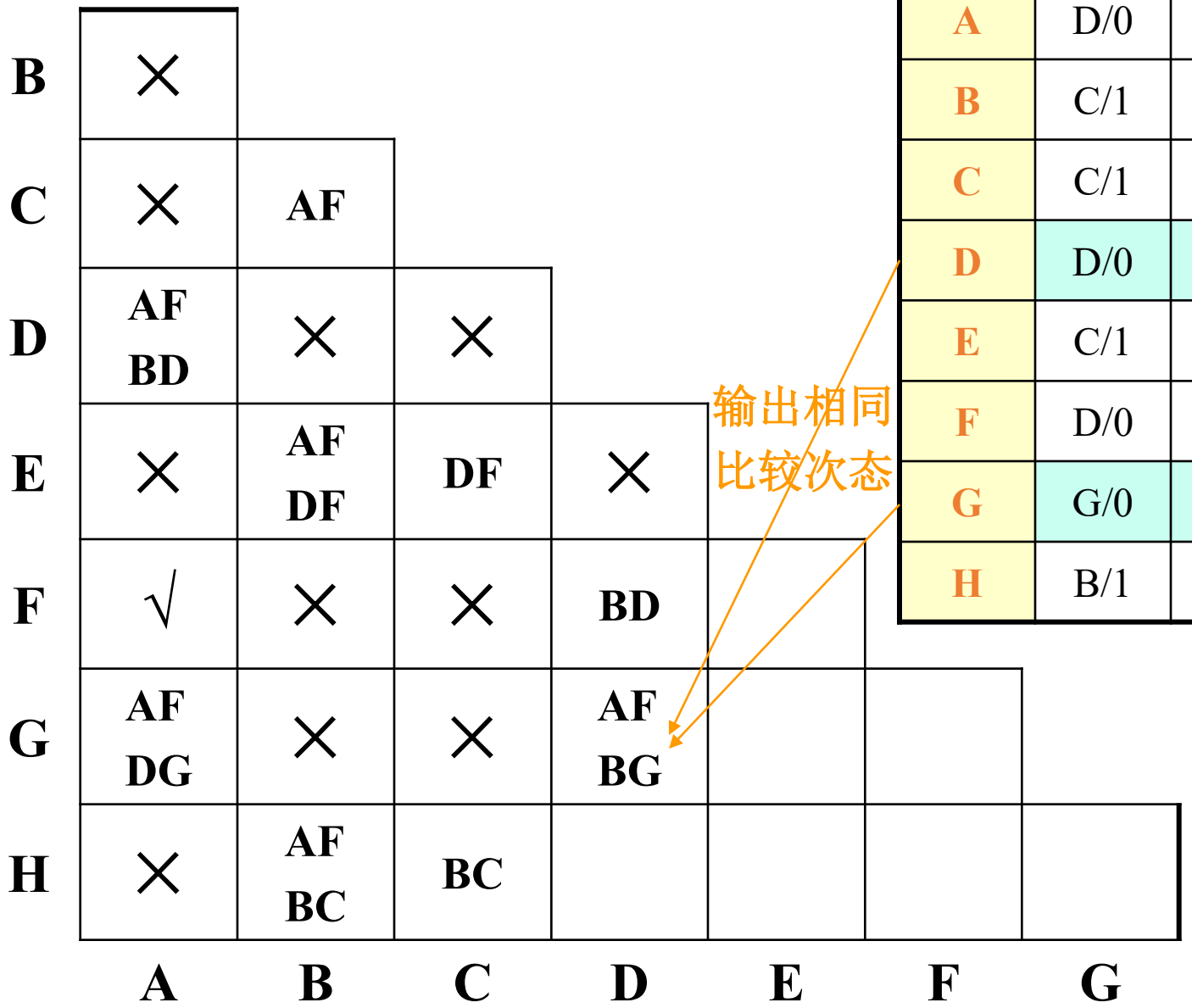
输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



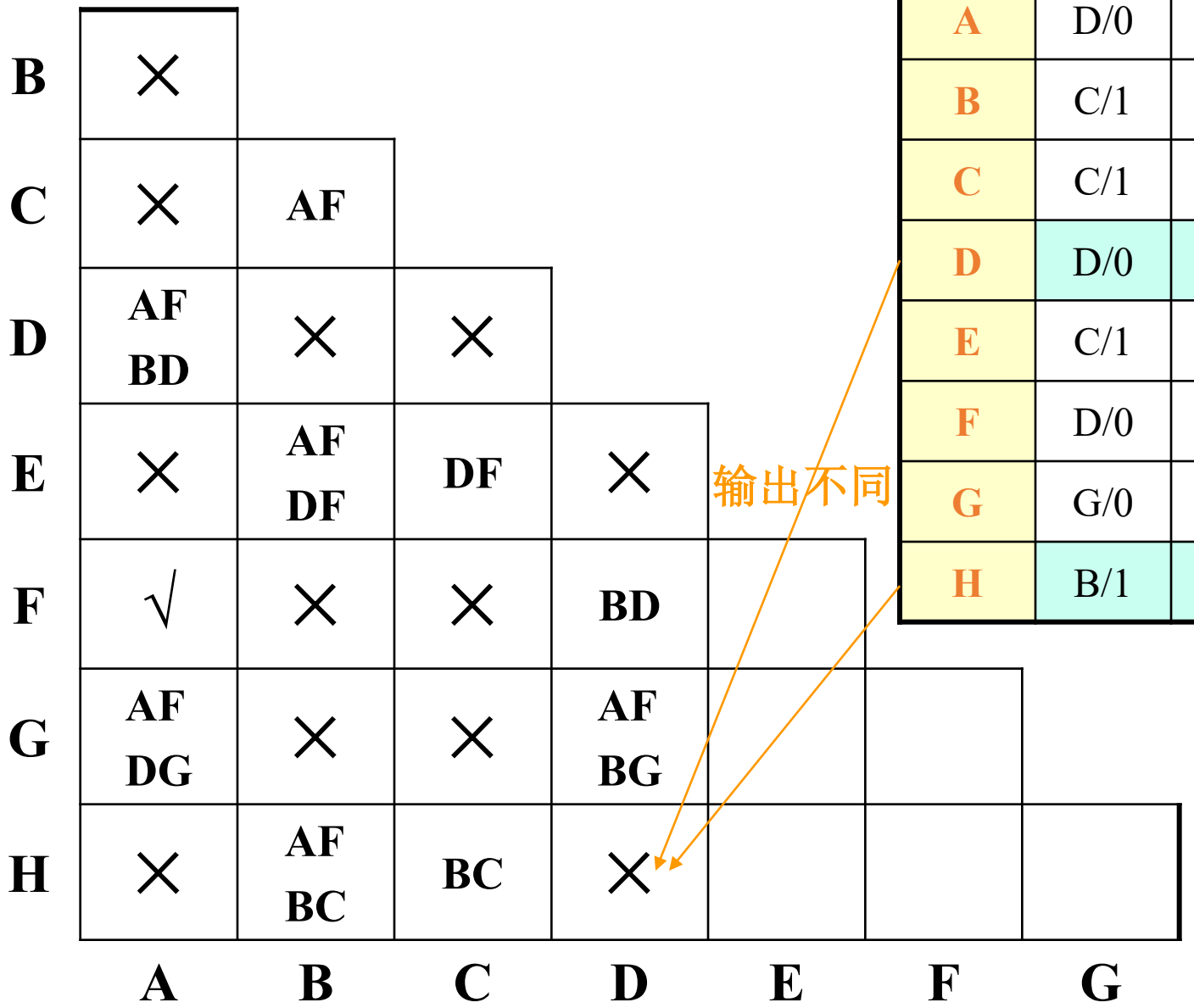
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出相同
比较次态



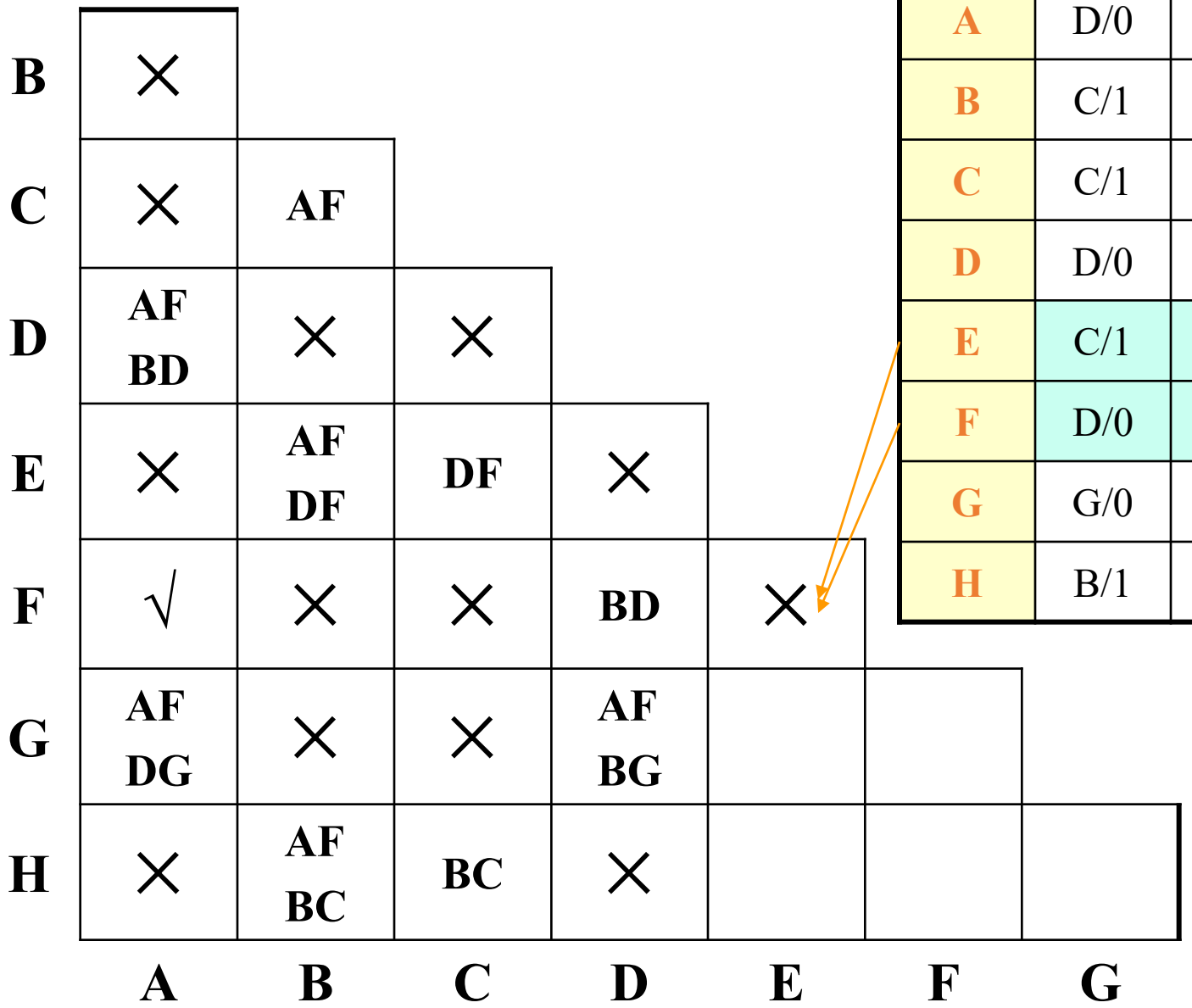
y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

输出相同
比较次态



输出不同

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0



y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×		
H	×	AF BC	BC	×			

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×		
H	×	AF BC	BC	×	BC DF		

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF		

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

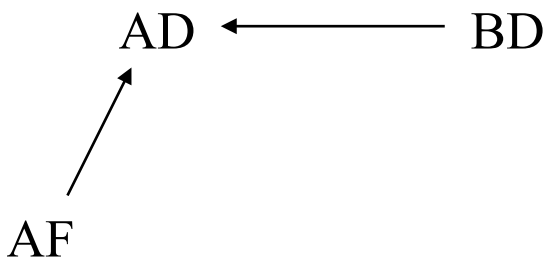
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

(3) 关联比较

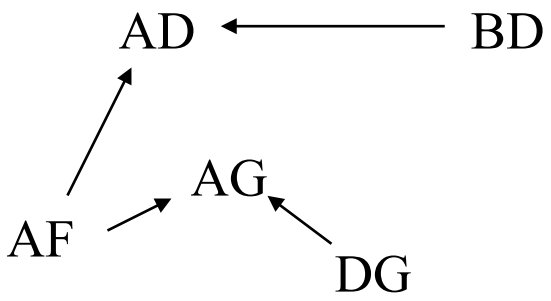
① 找出待定的等效对

AD 等效取决于AF和BD



	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×

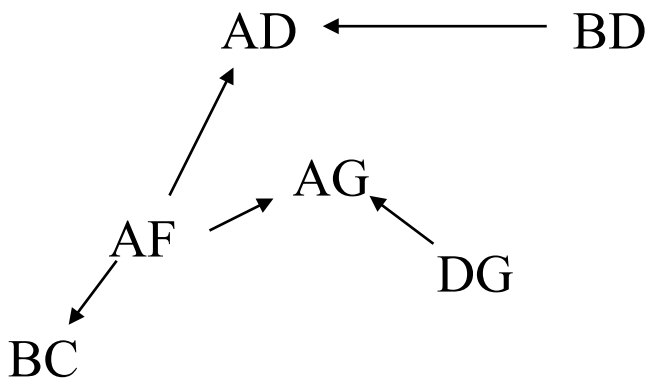
AG 等效取决于AF和DG



	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×

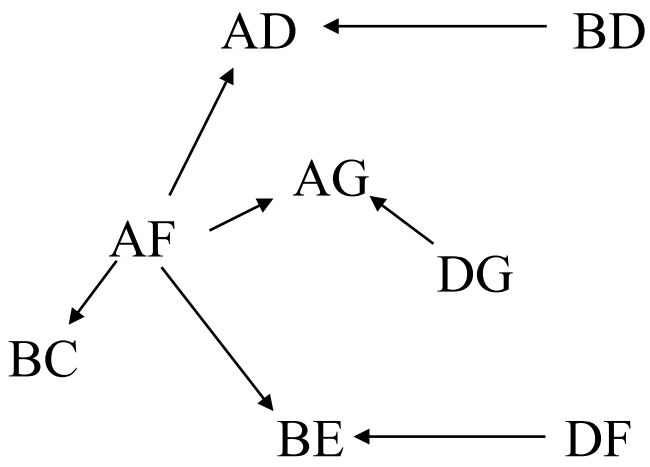
BC 等效取决于AF

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×

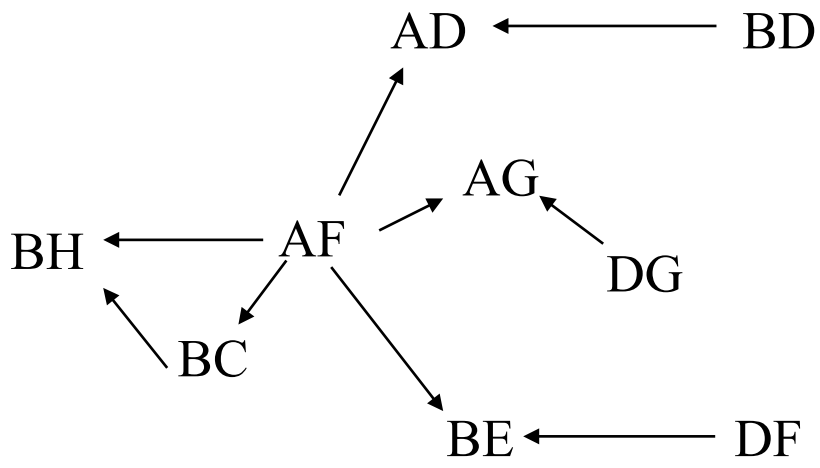


BE 等效取决于AF和DF

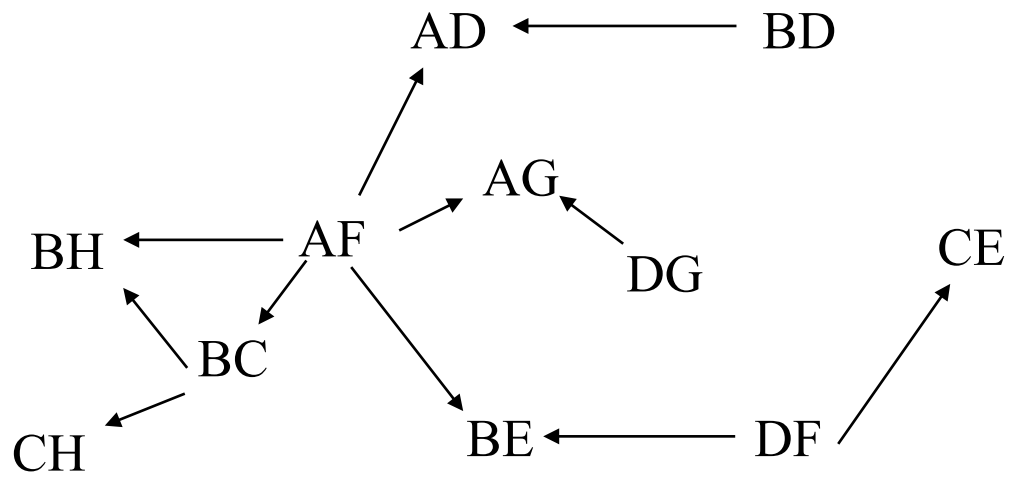
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



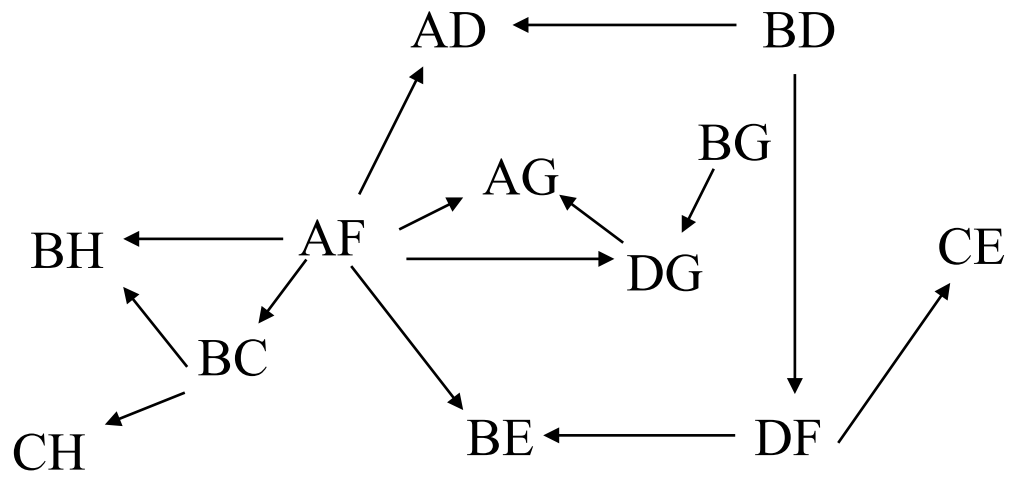
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



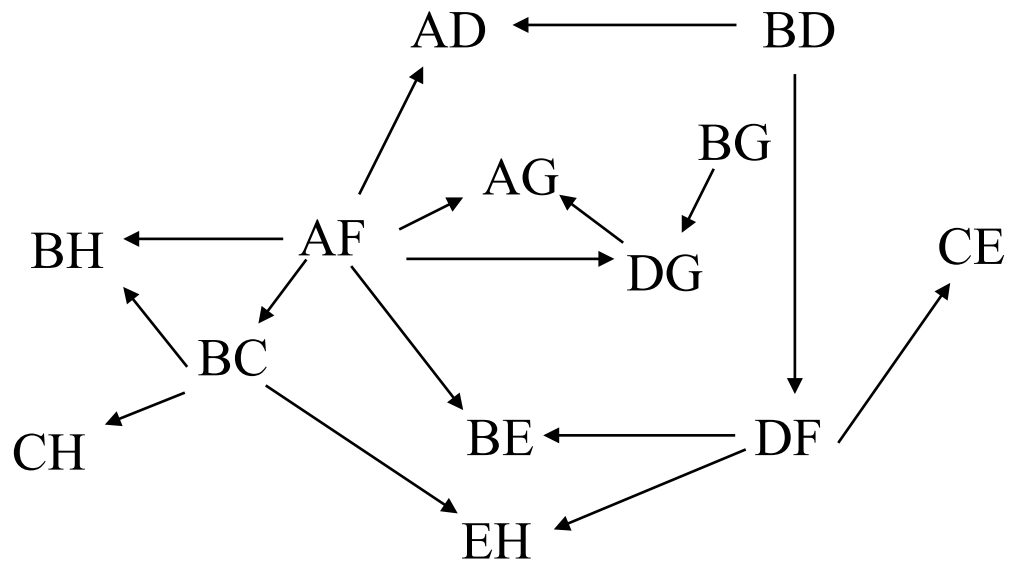
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



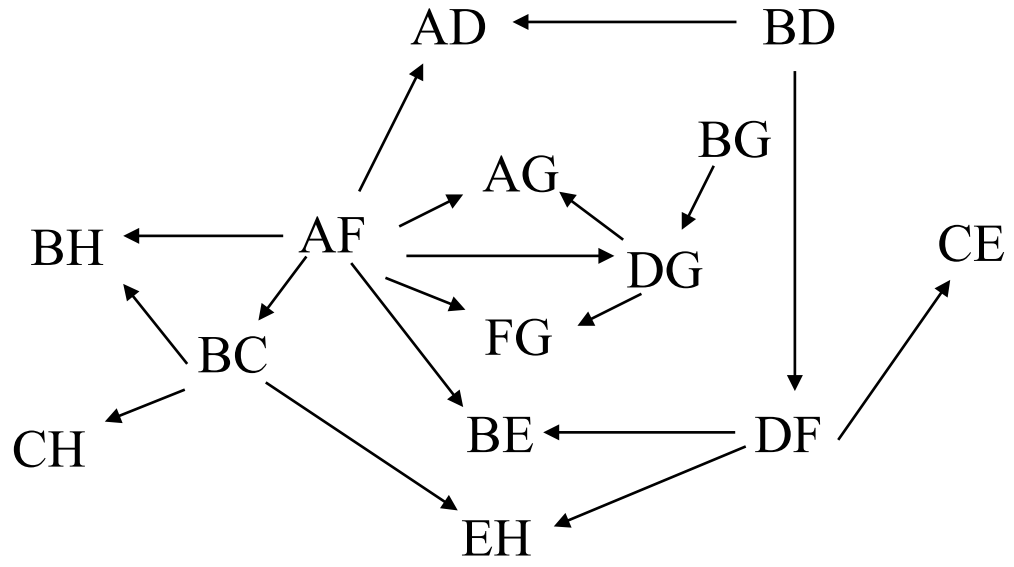
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×

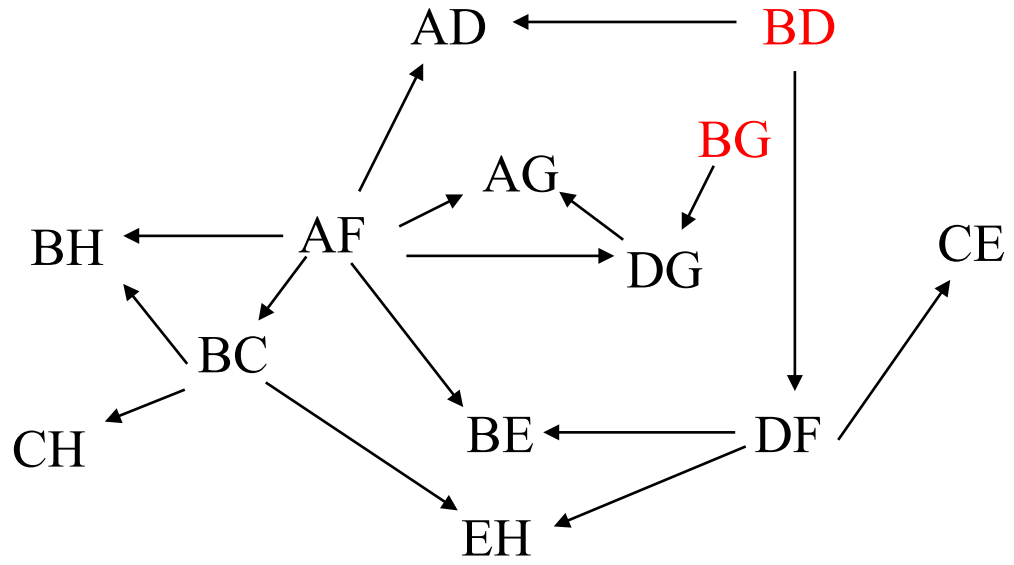


	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



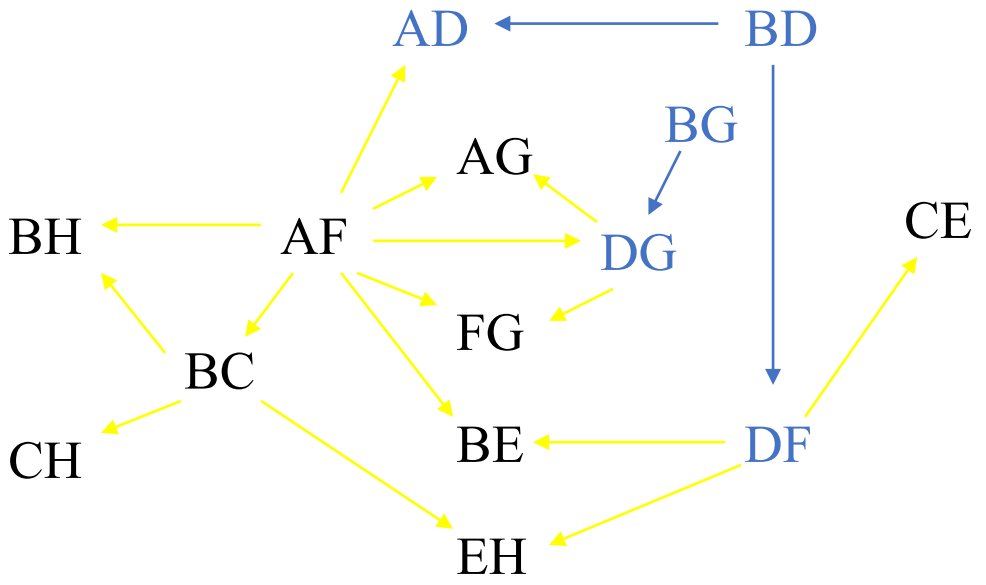
②确定不等效状态对

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



BD , BG不等效

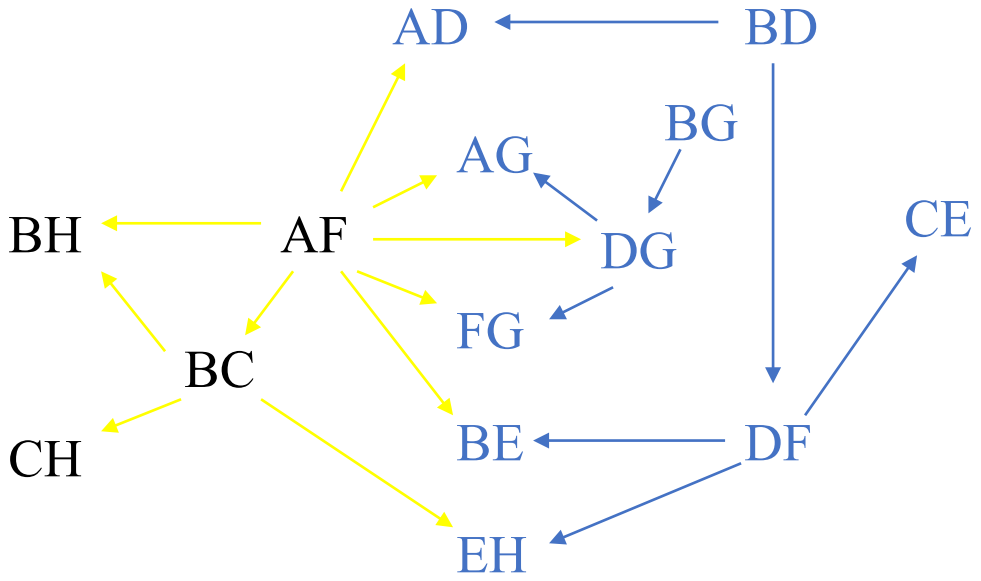
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



BD 不等效→AD,DF 不等效

BG 不等效→DG不等效

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×

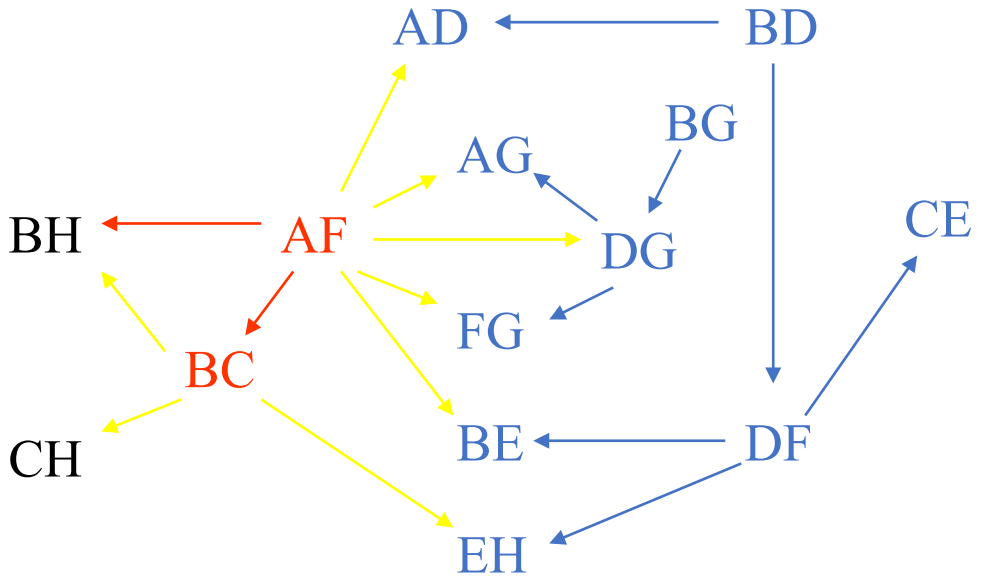


DG 不等效→AG,FG 不等效

DF 不等效→BE,EH,CE不等效

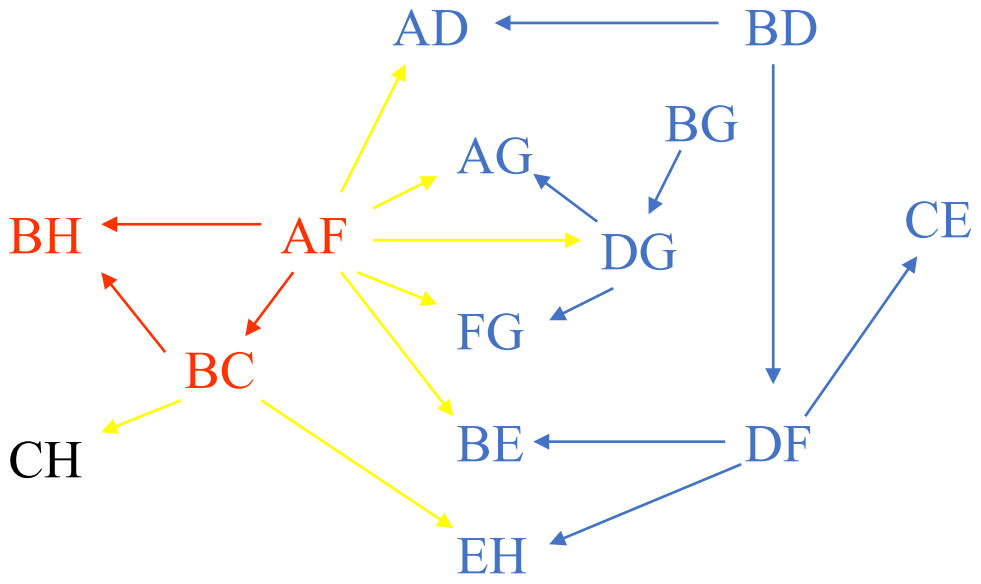
③确定等效状态对

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



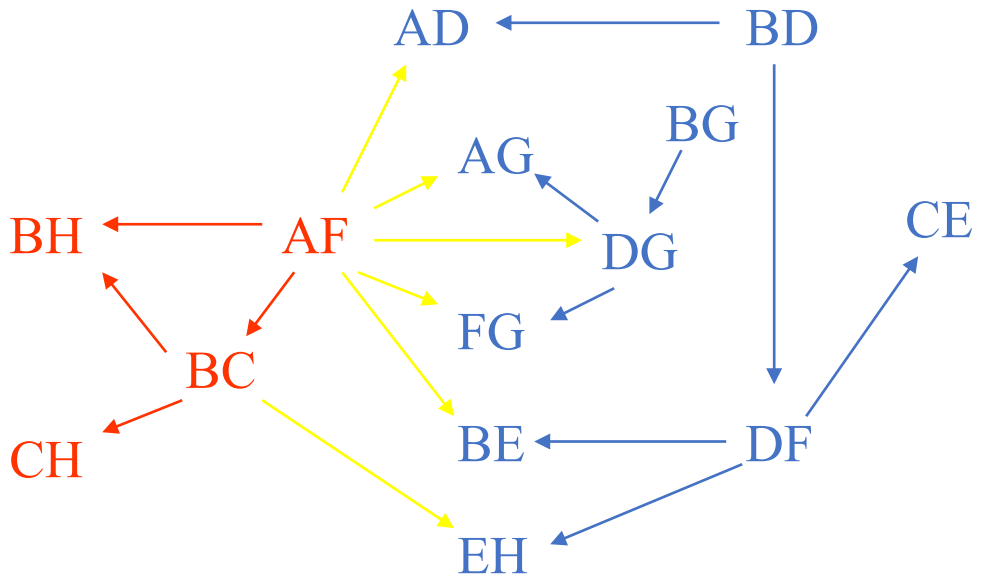
AF 等效→BC 等效

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



AF,BC 等效→BH 等效

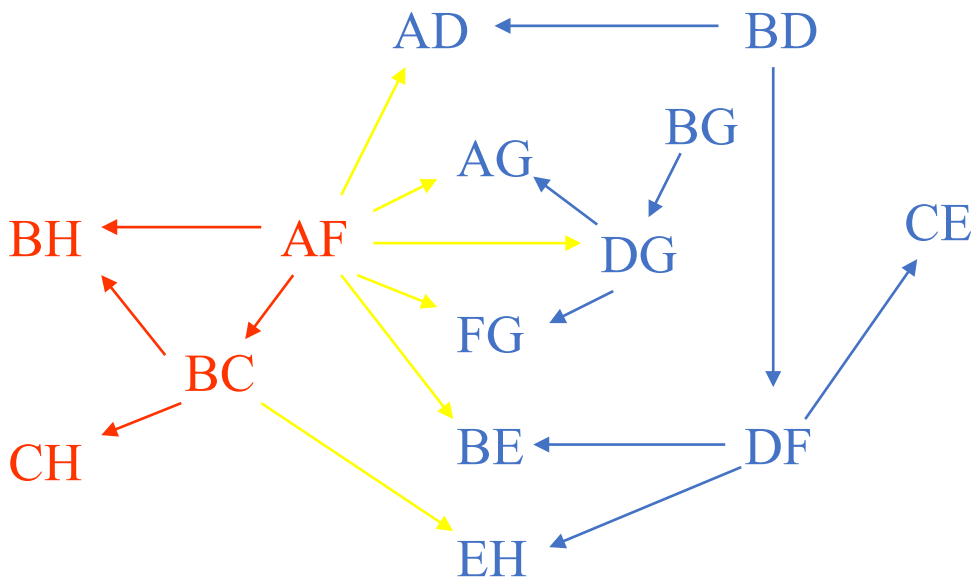
	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	AF BD	×	×				
E	×	AF DF	DF	×			
F	√	×	×	BD	×		
G	AF DG	×	×	AF BG	×	AF DG	
H	×	AF BC	BC	×	BC DF	×	×



BC 等效→CH 等效

⑷ 列出最大等效类

等效对：(A,F)，(B,C)，
(B,H)， (C,H)



其中，(B,C)，(B,H)，(C,H) → (B,C,H)

因而得到两个最大等效类：(A,F) 和 (B,C,H)

重新命名状态名

(A,F)	(B,C,H)	(D)	(E)	(G)
↓	↓	↓	↓	↓
A'	B'	C'	D'	E'

(5) 最小化状态表

y \ x	00	01	10	11
A'	C'/0	C'/0	A'/0	A'/0

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

(A,F)　(B,C,H)　(D)　(E)　(G)

↓　↓　↓　↓　↓

A'　B'　C'　D'　E'

y \ x	00	01	10	11
A'	C'/0	C'/0	A'/0	A'/0
B'	B'/1	C'/0	D'/1	A'/0

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

y^{n+1}/z

(A,F) (B,C,H) (D) (E) (G)

↓ ↓ ↓ ↓ ↓

A' B' C' D' E'

y \ x	00	01	10	11
A'	C'/0	C'/0	A'/0	A'/0
B'	B'/1	C'/0	D'/1	A'/0
C'	C'/0	B'/0	A'/0	A'/0

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

y^{n+1}/z

(A,F) (B,C,H) (D) (E) (G)

↓ ↓ ↓ ↓ ↓

A' B' C' D' E'

y \ x	00	01	10	11
A'	C'/0	C'/0	A'/0	A'/0
B'	B'/1	C'/0	D'/1	A'/0
C'	C'/0	B'/0	A'/0	A'/0
D'	B'/1	A'/0	D'/1	A'/0

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

y^{n+1}/z

(A,F) (B,C,H) (D) (E) (G)

↓ ↓ ↓ ↓ ↓

A' B' C' D' E'

y \ x	00	01	10	11
A'	C'/0	C'/0	A'/0	A'/0
B'	B'/1	C'/0	D'/1	A'/0
C'	C'/0	B'/0	A'/0	A'/0
D'	B'/1	A'/0	D'/1	A'/0
E'	E'/0	E'/0	A'/0	A'/0

y \ x	00	01	10	11
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

y^{n+1}/z

(A,F) (B,C,H) (D) (E) (G)

↓ ↓ ↓ ↓ ↓

A' B' C' D' E'

5.2.2 不完全给定同步时序电路状态表的化简

1、不完全给定：次态或输出中包含有无关项(d)。

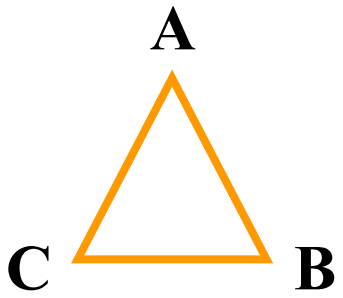
y \ x	0	1
A	A/0	D/d
B	A/0	D/0
C	A/0	D/1
D	A/0	C/1

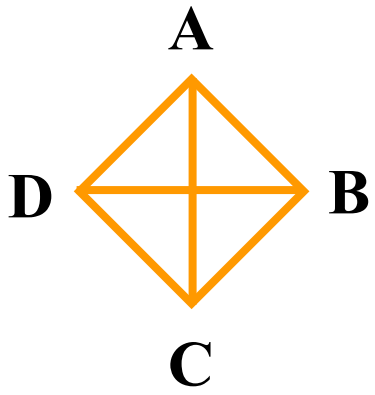
2、相容

(1) 状态相容：S1和S2作为初态，同时加入输入序列（除最后一个次态外，其他次态都是确定的），所产生的输出序列一致（认为确定的输出与对应的不确定输出相同），则状态S1和S2是相容对。记为：(S1，S2)。

(2) 状态相容无传递性，(A，B) 、(A，C)相容，但(B ,C)不相容

(3) 相容类：两两相容的状态的集合。





x / y	0	1
A	A/0	D/d
B	A/0	D/0
C	A/0	D/1
D	A/0	C/1

(4) 最大相容类：不能被其他相容类所包含的相容类。

相容对的判别标准:

条件一：它们的输出相同；
条件二：它们的次态必须满足下列情况之一：次态相同、次态交错、次态维持、后继状态相容、次态循环。
注意：一方给定，一方不给定的次态均当作相同。

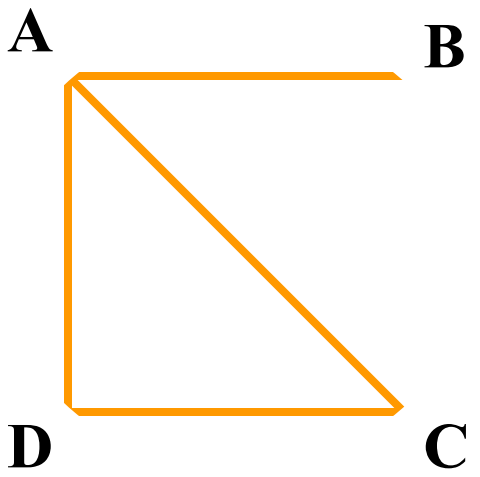
例、化简不完全给定状态表。

右表中的相容对为：(A,B)，(A,C)，(A,D)，(C, D)

状态合并图：将所有相容对填入合并图，可以得到两个最大相容类为：(A,B)，(A,C,D)

y \ x	0	1
A	A/0	D/d
B	A/0	D/0
C	A/0	D/1
D	A/0	C/1

y^{n+1}/z



3、最小化状态表

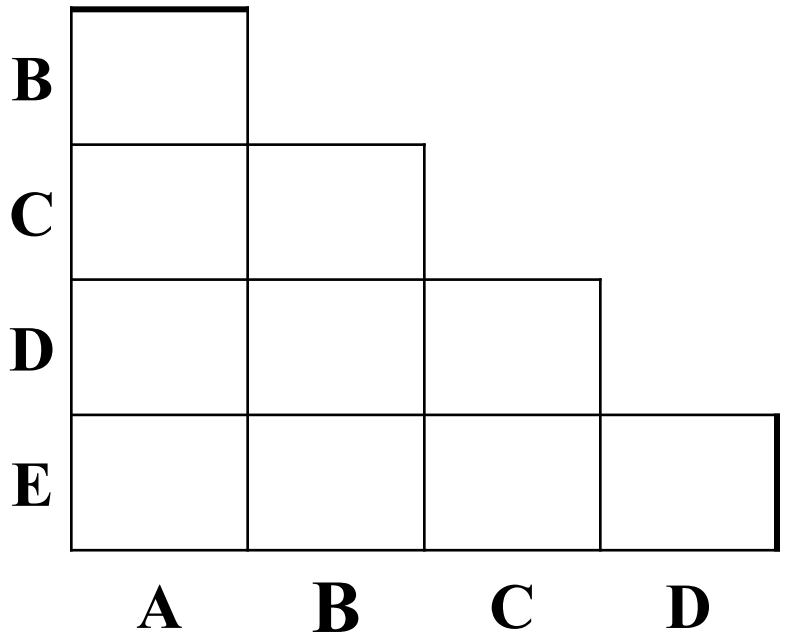
(1)覆盖性：能包含全部的原始状态。
(2)闭合性：任一个相容类的次态应属于该集内的一个相容类。
(3)最小化：选择满足“覆盖”和“闭合”的相容类且数目最少。

4. 不完全给定状态表的化简过程
(1)利用隐含表寻找相容对
(2)用合并图确定最大相容类
(3)采用覆盖闭合表进行相容类集的选择，建立最小化状态表

例1、化简如图所示的原始状态表。

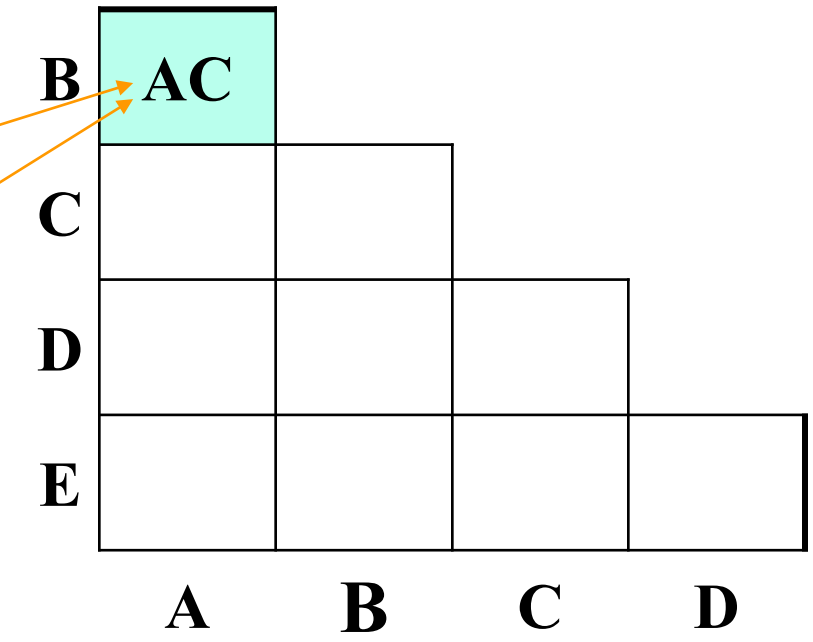
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



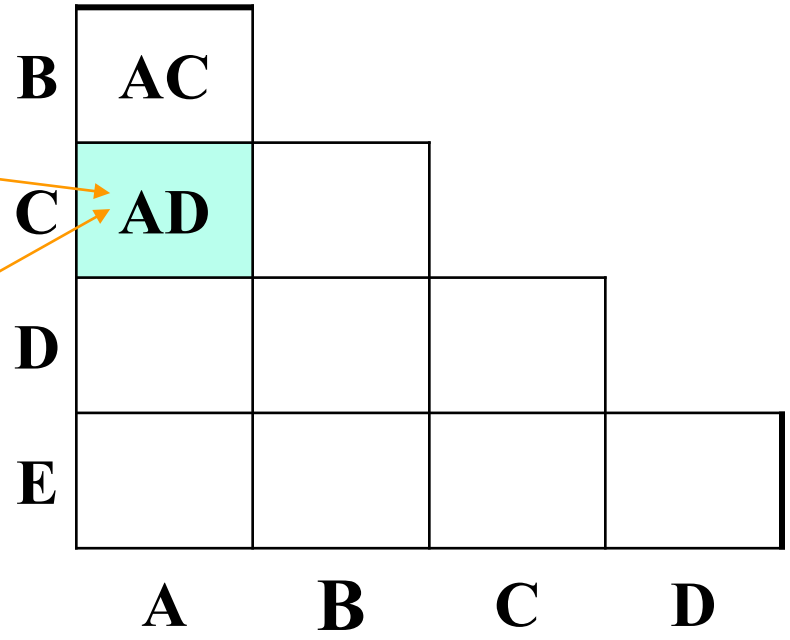
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



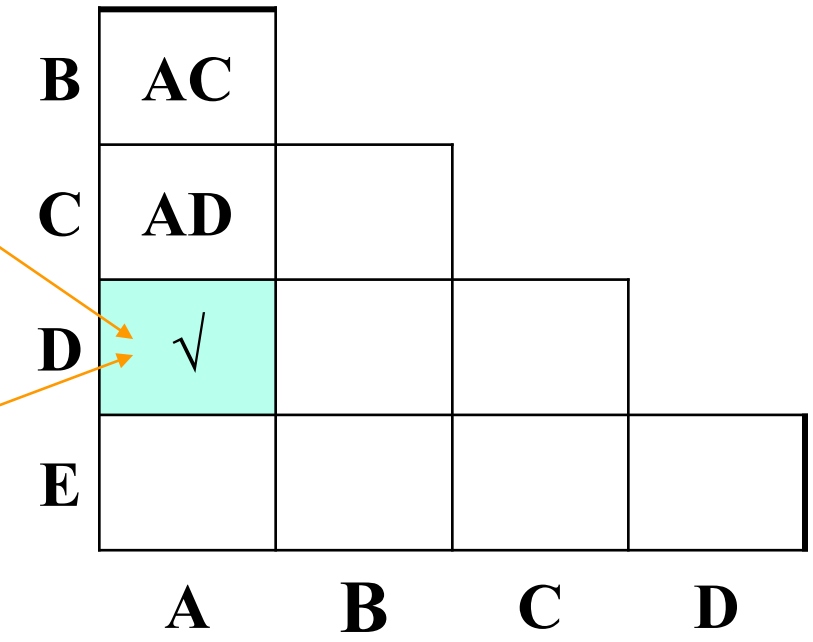
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



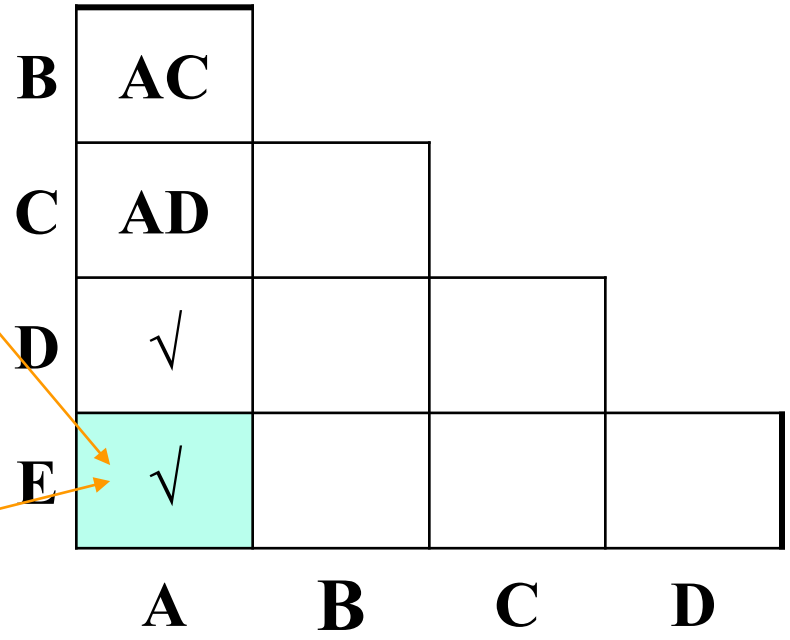
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



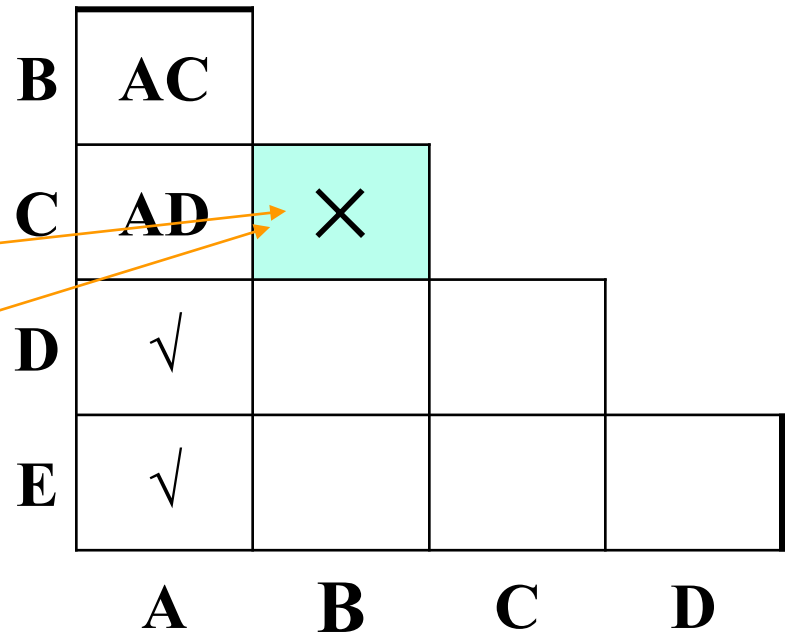
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



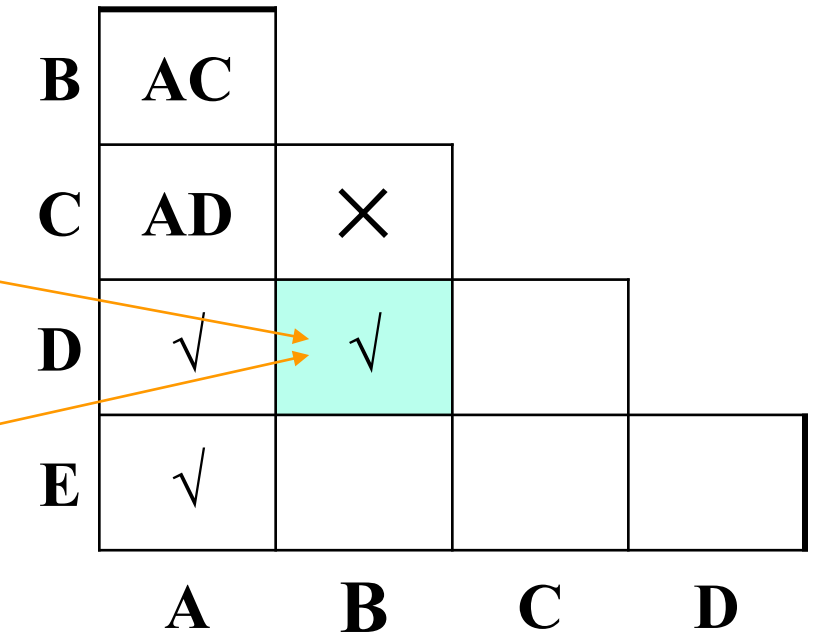
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



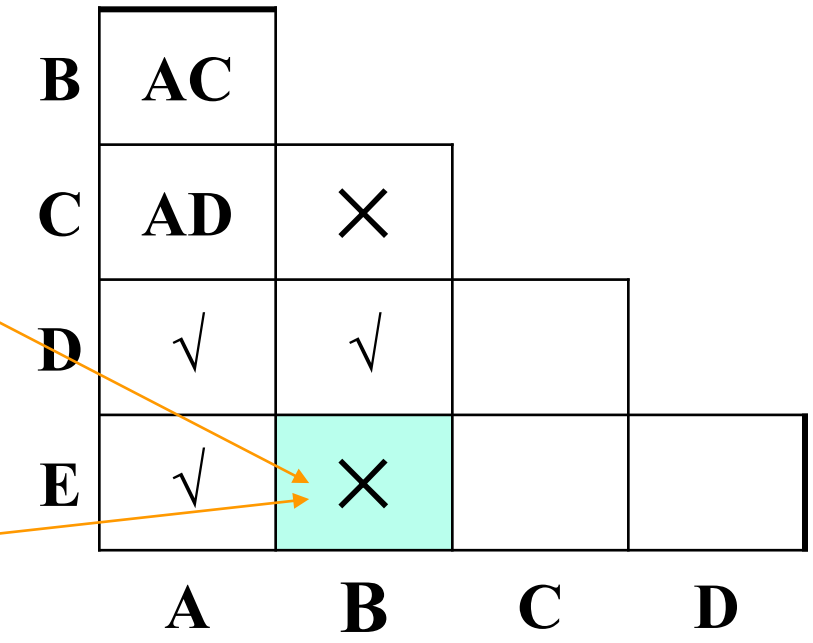
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



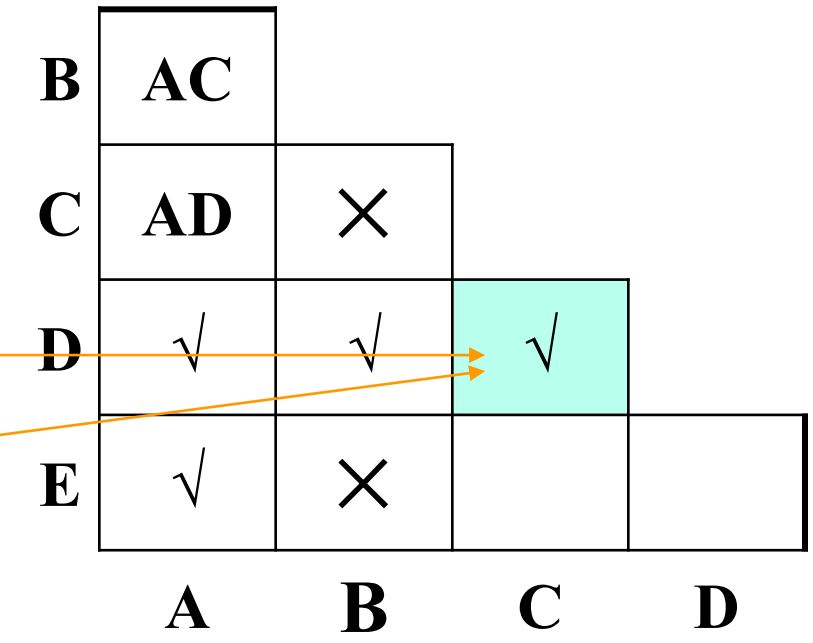
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



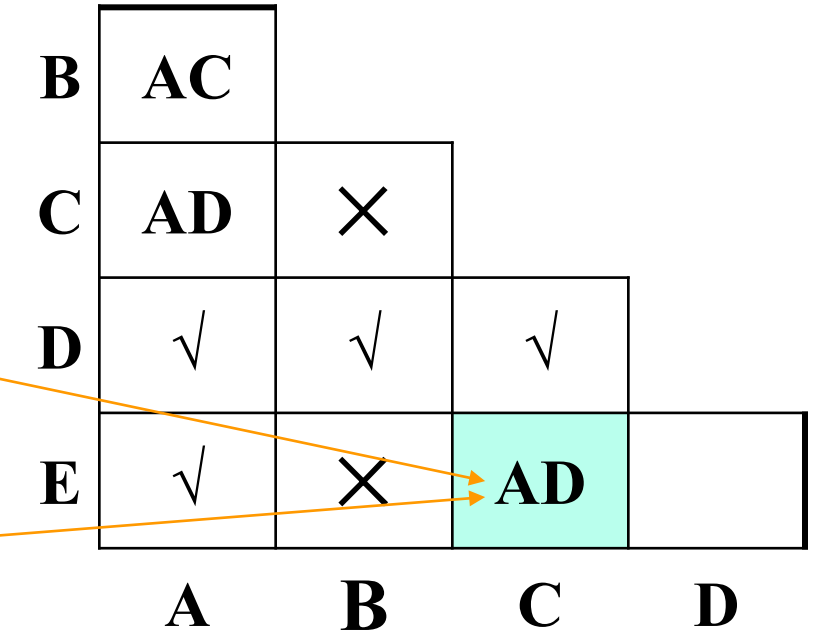
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



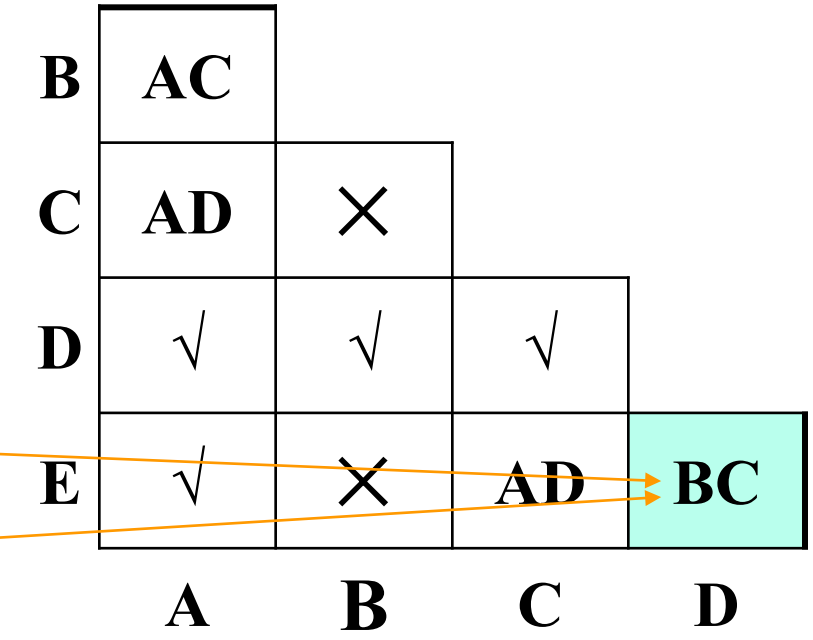
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z



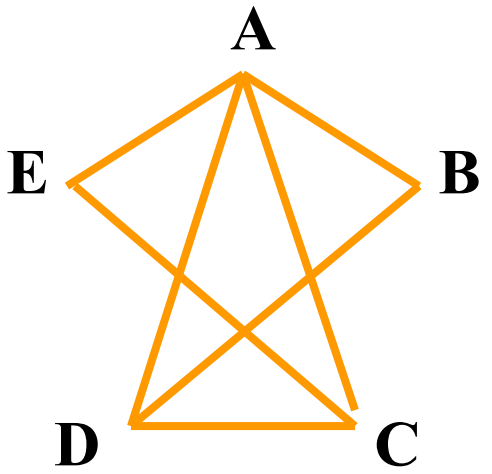
y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z

B	A/C			
C	A/D	×		
D	√	√	√	
E	√	×	A/D	B/C
	A	B	C	D

（1）利用隐含表寻找相容对：(A,B), (A,C), (A,D), (A,E), (B,D), (C,D), (C,E)

（2）用合并图确定最大相容类：(A,B,D), (A,C,D), (A,C,E)



3、作出最小化状态表

y \ x	0	1
A	A/d	d/d
B	C/1	B/0
C	D/0	d/1
D	d/d	B/d
E	A/0	C/1

y^{n+1}/z

覆盖闭合表

相容类	覆盖					闭合	
	A	B	C	D	E	X = 0	X = 1
ABD	A	B		D		AC	B
ACD	A		C	D		AD	B
ACE	A		C		E	AD	C

选择最小化：

(ABD)　　(ACE)

↓　　↓

A'　　B'

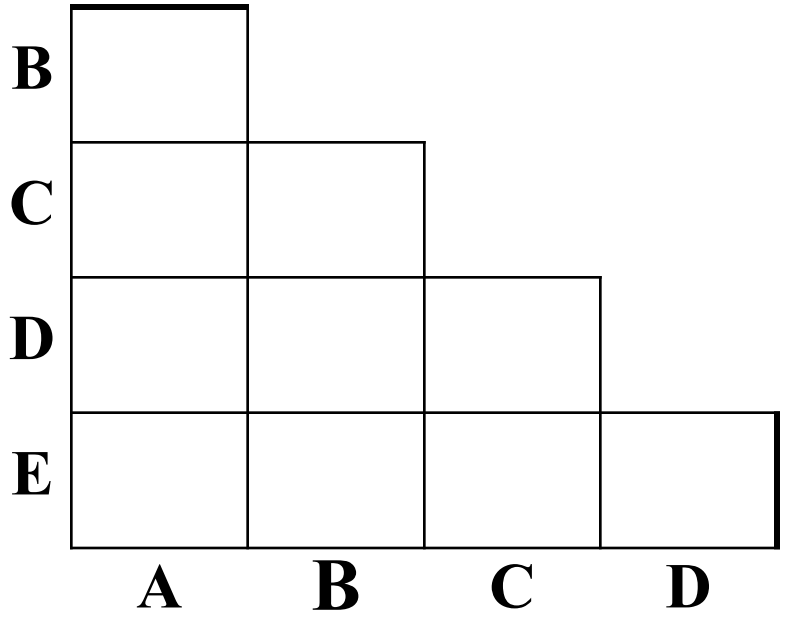
最小化状态表

y \ x	0	1
A'	B'/1	A'/0
B'	A'/0	B'/1

例2、化简如图所示的原始状态表。

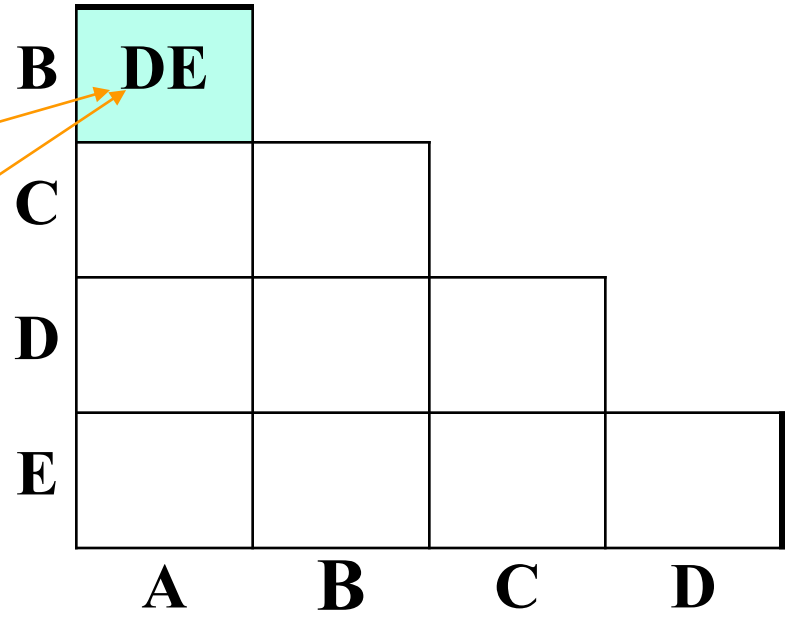
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



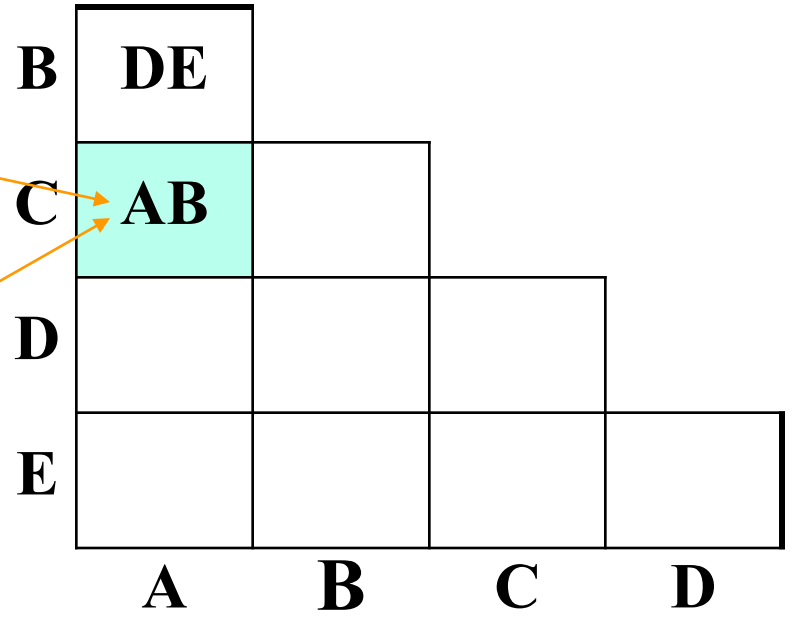
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



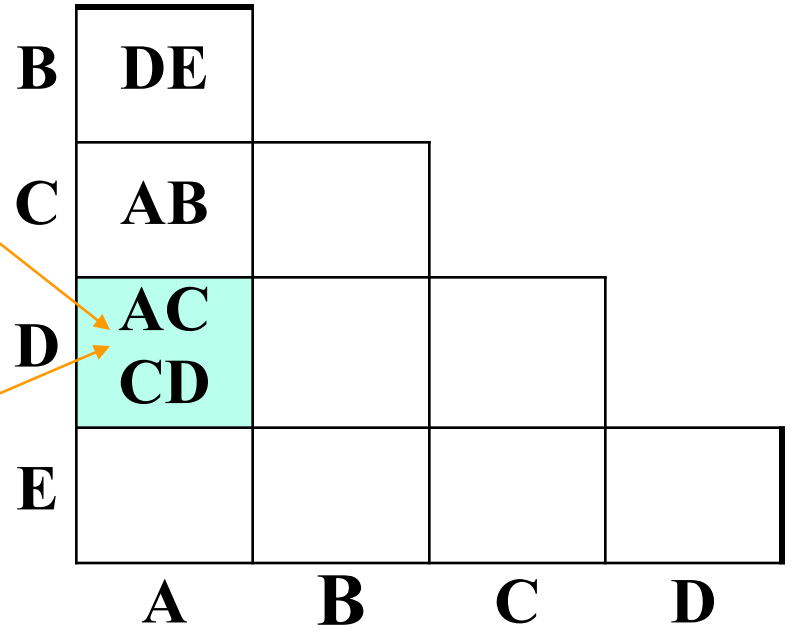
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



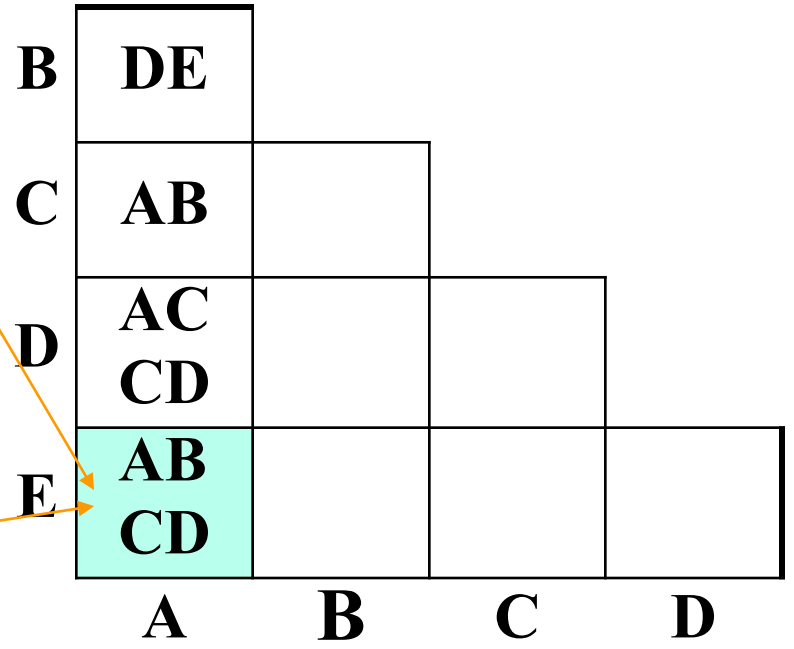
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



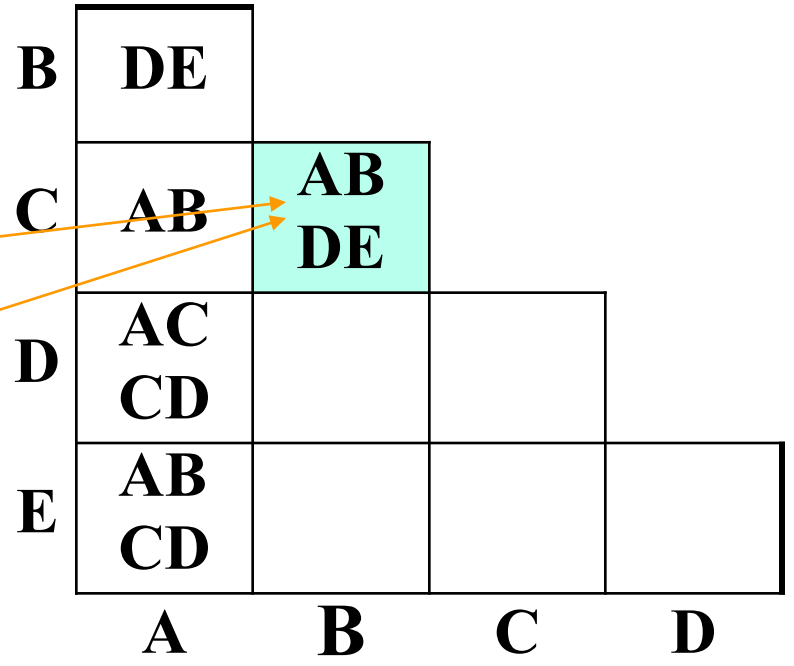
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



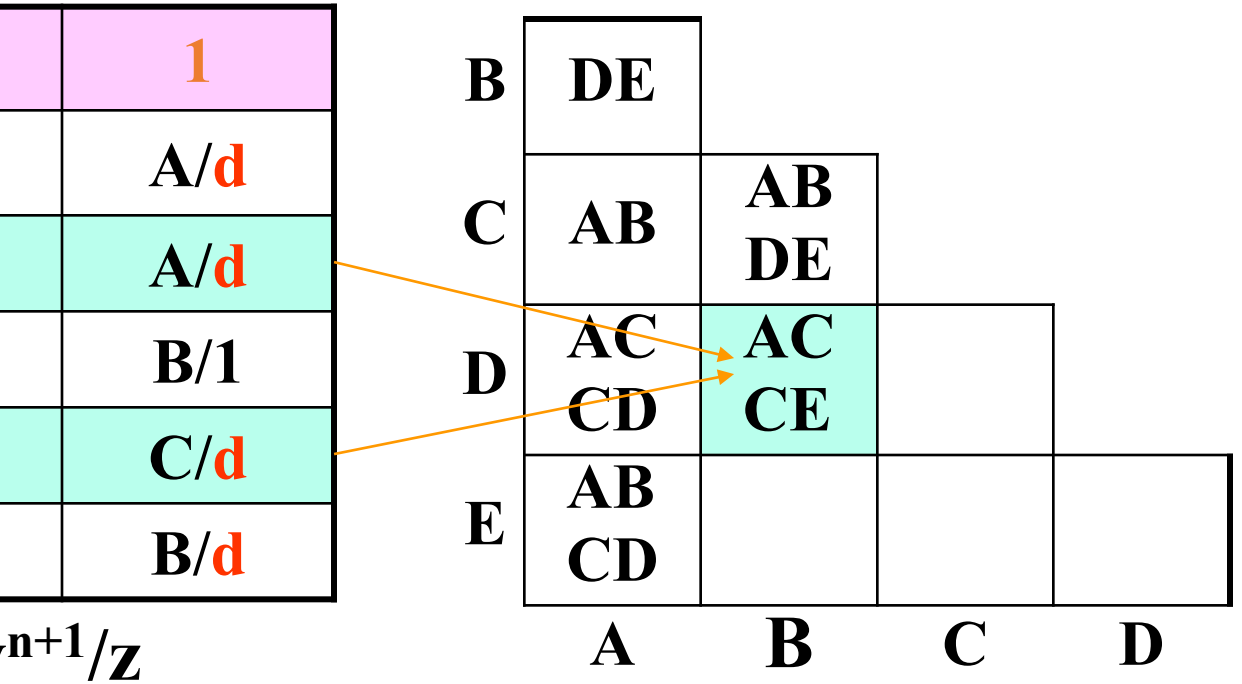
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



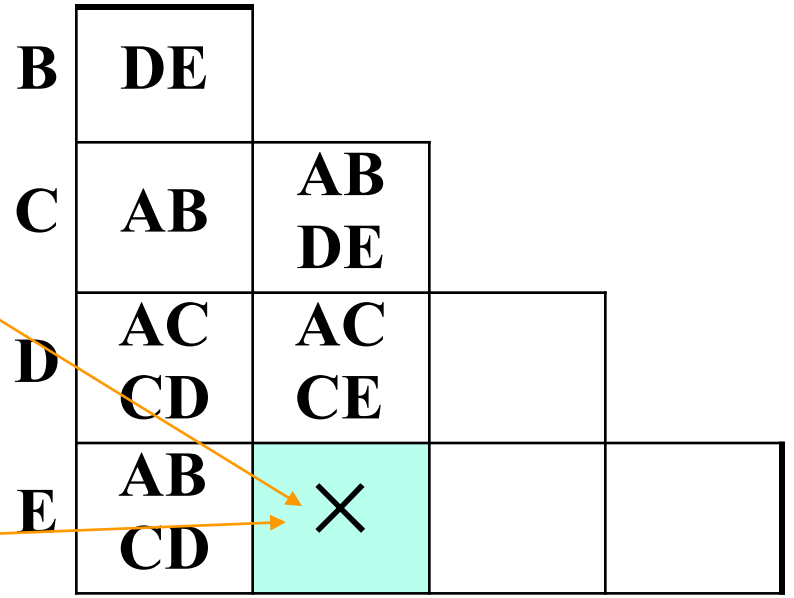
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



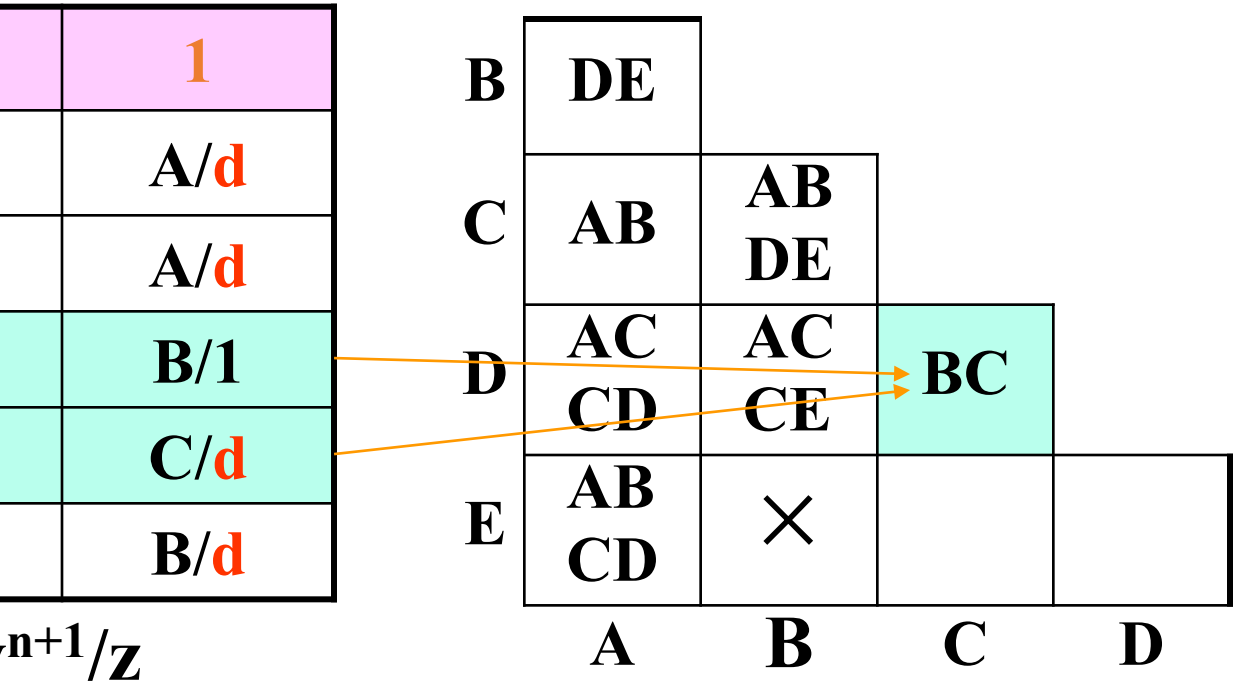
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



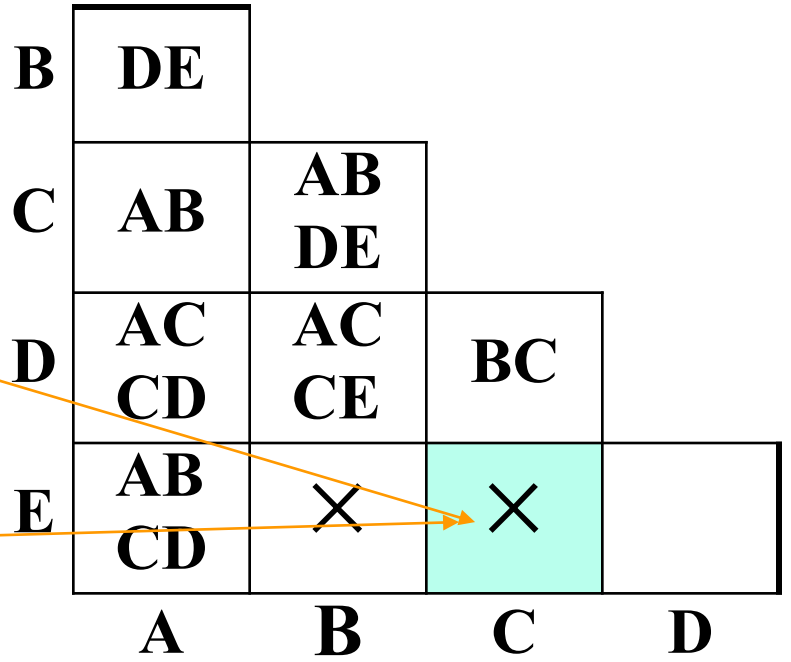
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



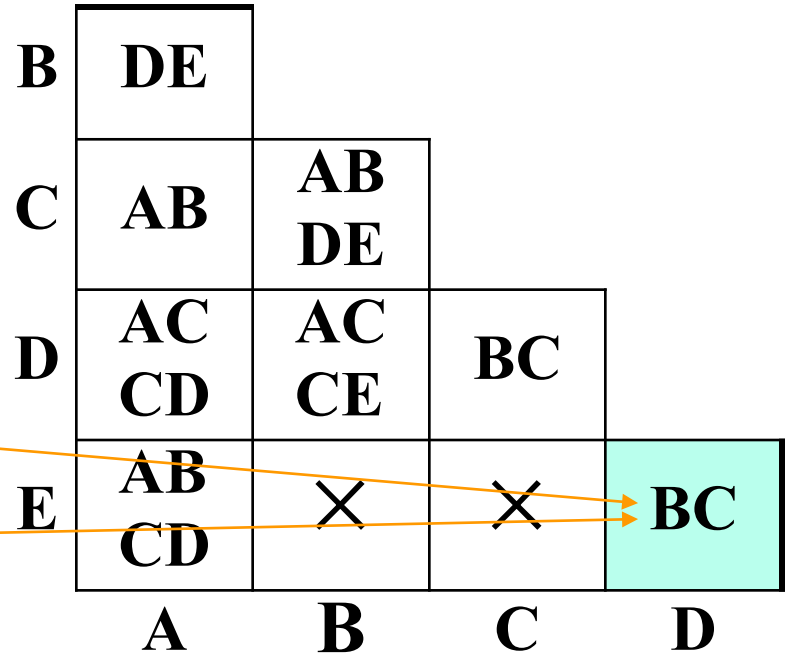
y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z



y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z

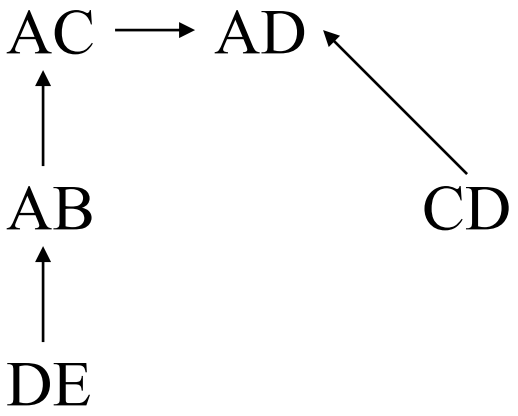


AB

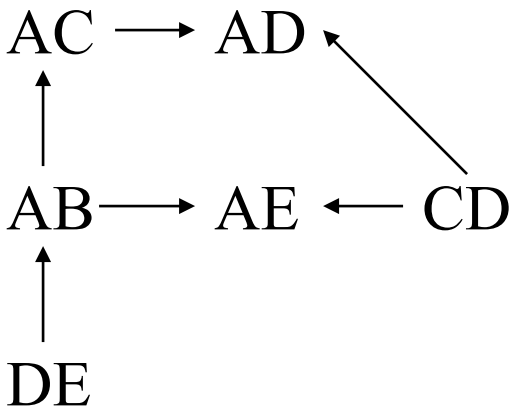
DE

	A	B	C	D
B	DE			
C	AB	AB DE		
D	AC CD	AC CE	BC	
E	AB CD	×	×	BC

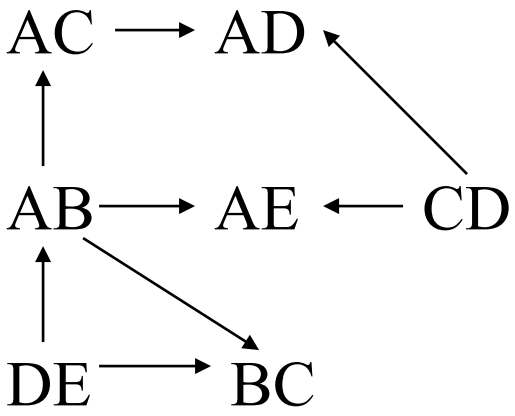
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**



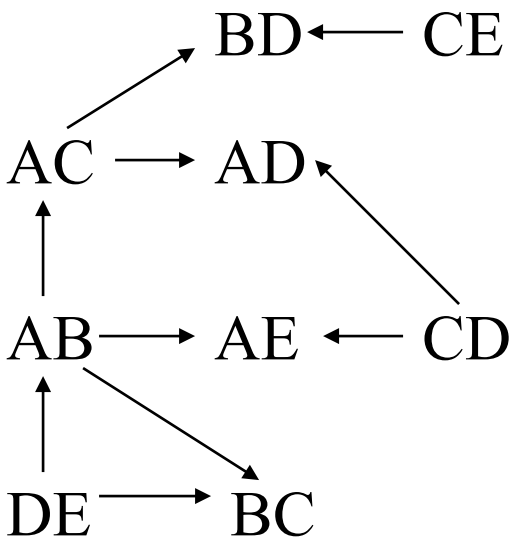
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**



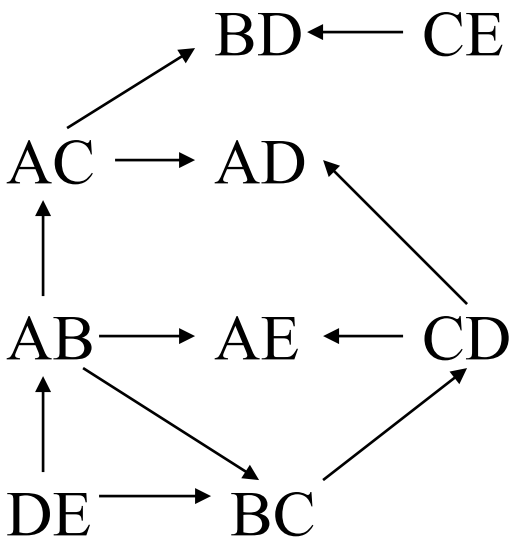
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**



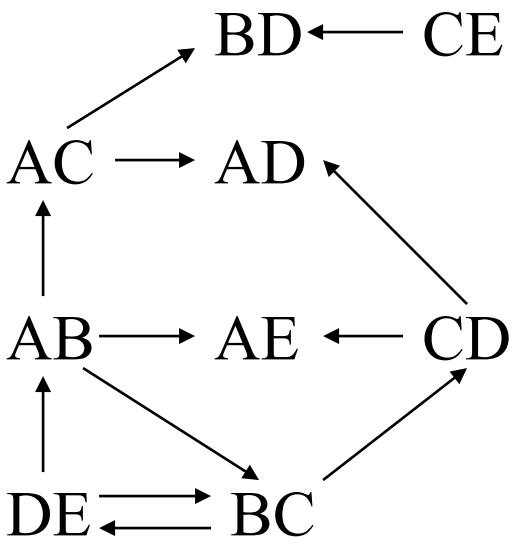
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**



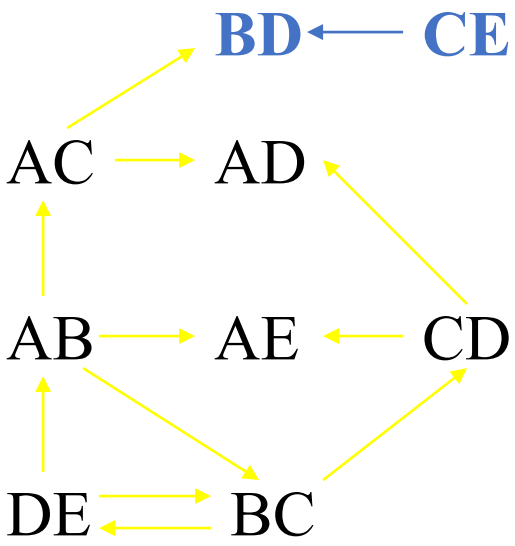
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**



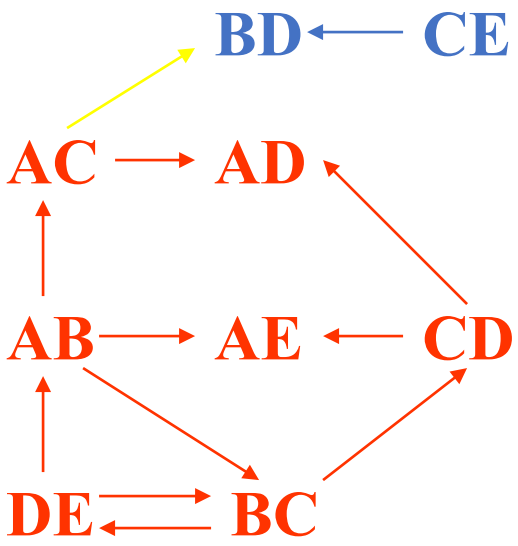
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**



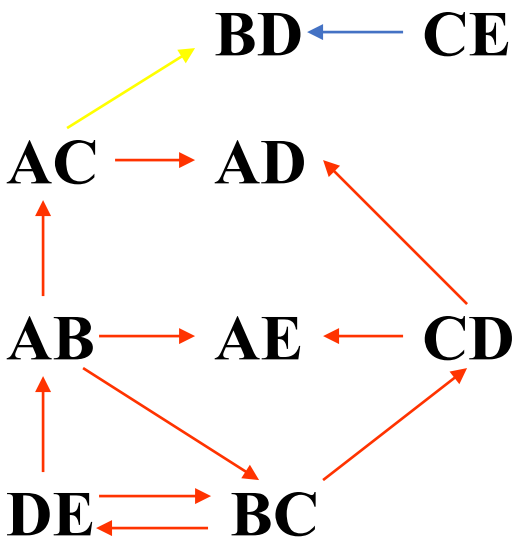
B	**DE**			
C	**AB**	**AB** **DE**		
D	**AC** **CD**	**AC** **CE**	**BC**	
E	**AB** **CD**	×	×	**BC**
	A	**B**	**C**	**D**

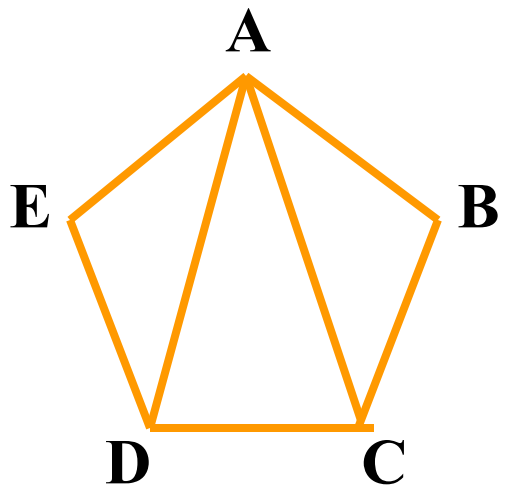


B	DE			
C	AB	AB DE		
D	AC CD	AC CE	BC	
E	AB CD	×	×	BC
	A	B	C	D



B	DE			
C	AB	AB DE		
D	AC CD	AC CE	BC	
E	AB CD	×	×	BC
	A	B	C	D





⑴ 利用隐含表找出相容对：(A,B), (A,C), (A,D), (A,E), (B,C), (C,D), (D,E)

⑵ 用合并图确定最大相容类：(A,B,C),(A,C,D),(A,D,E)

(3) 作出最小化状态表一

y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z

相容类	覆盖					闭合	
	A	B	C	D	E	X = 0	X = 1
ABC	A	B	C			DE	AB
ACD	A		C	D		CD	ABC
ADE	A			D	E	CD	ABC

选择最小化：

(ABC) (ACD) (ADE)

↓ ↓ ↓

A' B' C'

y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z

最小化状态表

y \ x	0	1
A'	C'/0	A'/1
B'	B'/0	A'/1
C'	B'/1	A'/d

选择最小化：

(ABC) (ACD) (ADE)

↓ ↓ ↓

A' B' C'

y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z

最小化状态表

y \ x	0	1
A'	C'/0	A'/1
B'	B'/0	A'/1
C'	B'/1	A'/d

选择最小化：

(ABC) (ACD) (ADE)

↓ ↓ ↓

A' B' C'

y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z

最小化状态表

y \ x	0	1
A'	C'/0	A'/1
B'	B'/0	A'/1
C'	B'/1	A'/d

选择最小化：

(ABC) (ACD) (ADE)

↓ ↓ ↓

A' B' C'

(3) 作出最小化状态表二

y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

相容类	覆盖					闭合	
	A	B	C	D	E	X = 0	X = 1
ABC	A	B	C			DE	AB
ACD	A		C	D		CD	ABC
DE				D	E	C	BC

选择最小化：

(ABC)　(DE)

↓　↓

A'　B'

最小化状态表

y \ x	0	1
A'	B'/0	A'/1
B'	A'/1	A'/d

y \ x	0	1
A	D/d	A/d
B	E/0	A/d
C	D/0	B/1
D	C/d	C/d
E	C/1	B/d

y^{n+1}/z

相容类	覆盖					闭合	
	A	B	C	D	E	X = 0	X = 1
ABC	A	B	C			DE	AB
ACD	A		C	D		CD	ABC
DE				D	E	C	BC

选择最小化：

(ABC)　(DE)

↓　↓

A'　B'

y \ x	0	1
A'	B'/0	A'/1
B'	A'/1	A'/d

5.3 有限状态机（FSM）

有限状态机可以实现外部激励和当前状态的响应机制。有限状态机（Finite State Machine, FSM）简称状态机，是用来表示系统中的有限个状态及这些状态之间的转移和动作的模型。分为：Moore型和Mealy型。

状态转移和动作依赖于当前状态和外部输入

对于有限状态机，通常次态逻辑要遵循状态图的逻辑转移流向
状态寄存器代码独立，然后将次态逻辑和输出逻辑结合

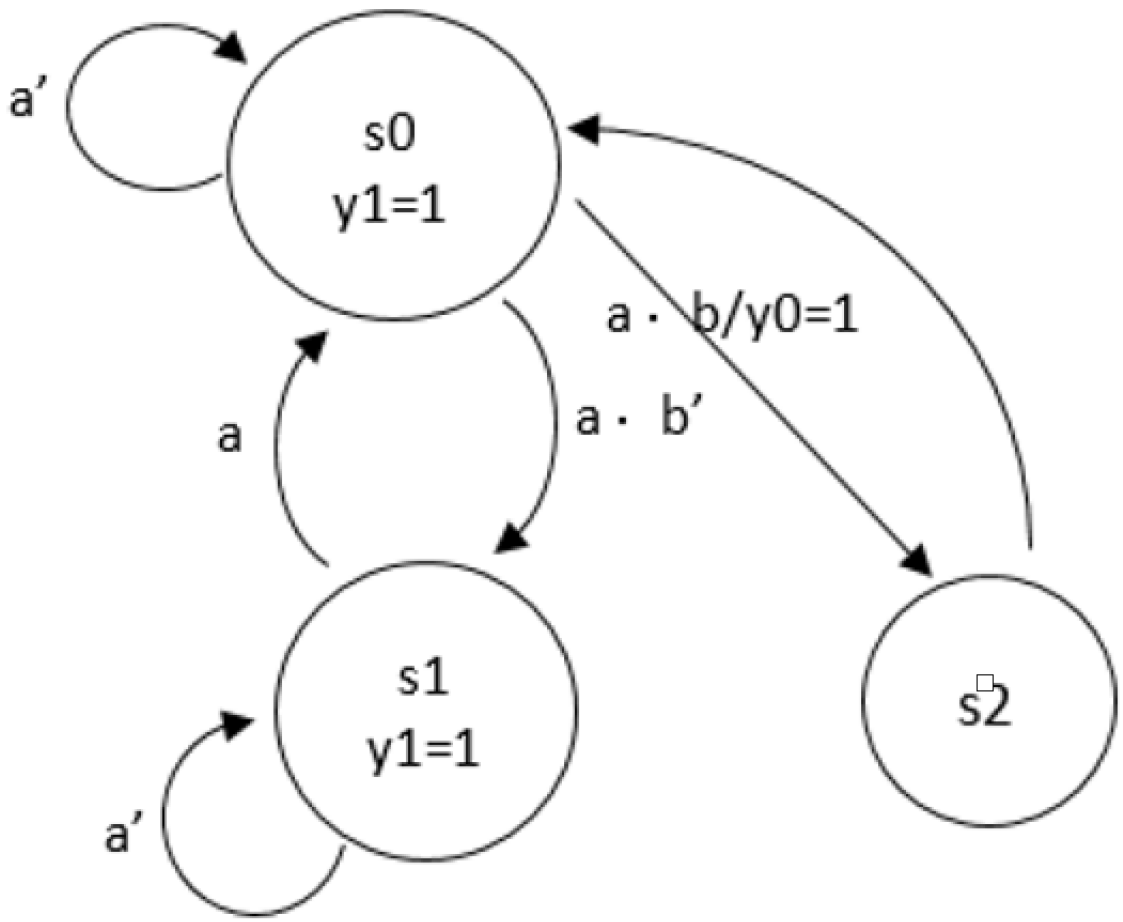
状态机的优点

- 高效的顺序控制模型，是高速高效控制的首选；
- 容易利用EDA工具进行优化设计：状态机构建简单，设计方案相对固定，可以发挥HDL综合器的优化功能；
- 性能稳定：状态机容易构成良好的同步时序逻辑模块，可用于解决大规模逻辑电路设计中的竞争和冒险现象。
- 高速性能：在高速通信和高速控制方面，状态机具有其巨大的优势；
- 高可靠性：状态机是由纯硬件电路构成；能使用各种容错技术；如果状态机进入非法状态，其从中跳出进入正常状态的时间短暂，对系统的危害不大。

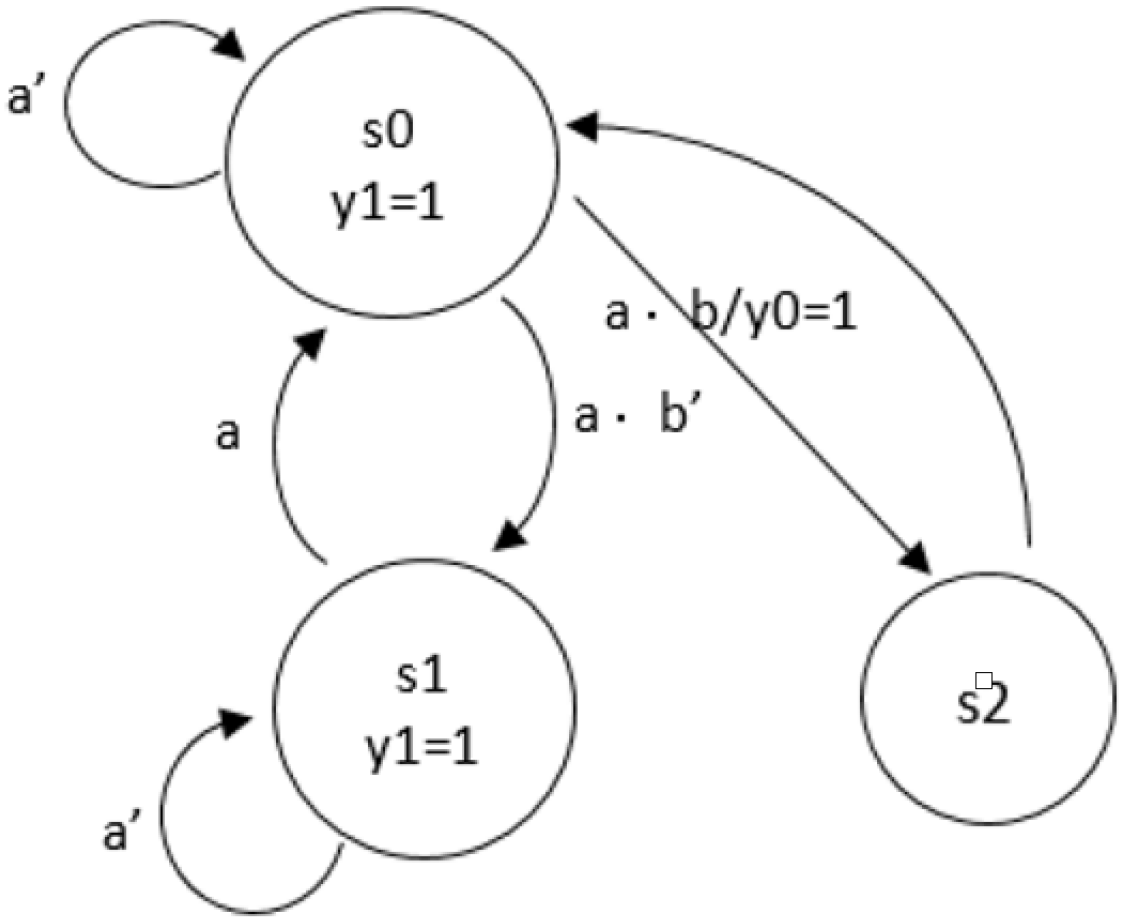
考虑到简便性和灵活性，用一个符号常量来表示有限状态机的状态：

```
localparam[1:0] s0=2’b00,
                s1=2’b01,
                s2=2’b10;
```

综合的时候，软件会根据有限状态机的结构将符号常量映射为对应的二进制字符（如独热码），这就是状态分配。



```
// 次态逻辑
always @*
case ( state_reg )
 s0:  if ( a )
                    if ( b )
          state_next = s2 ;
                    else
          state_next = s1 ;
       else
          state_next = s0;
 s1:   if  ( a )
         state_next = s0;
       else
         state_next = s1;
  s2:   state_next = s0;
  default: state_next = s0;
endcase
```



状态的转移要遵循状态图的流程
次态逻辑单元是关键：次态（state_next）
由当前状态(state_reg)和外部输入信号决定

例、序列检测器：检测一组或多组由二进制码组成的脉冲序列信号，当序列检测器连续收到一组串行二进制码后，如果这组码与检测器中预先设置的码相同，则输出1，否则输出0。

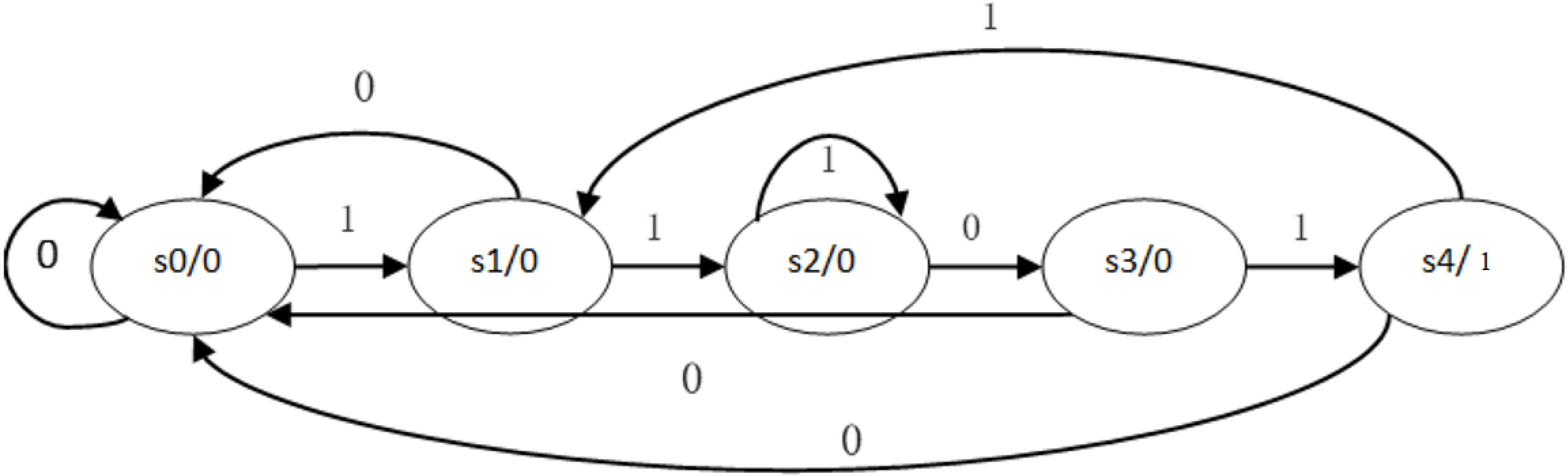
关键步骤：正确码的接收必须是连续的，要求检测器必须记住输入脉冲序列，直到连续检测中收到的每一位码都与预置数的对应码相同。

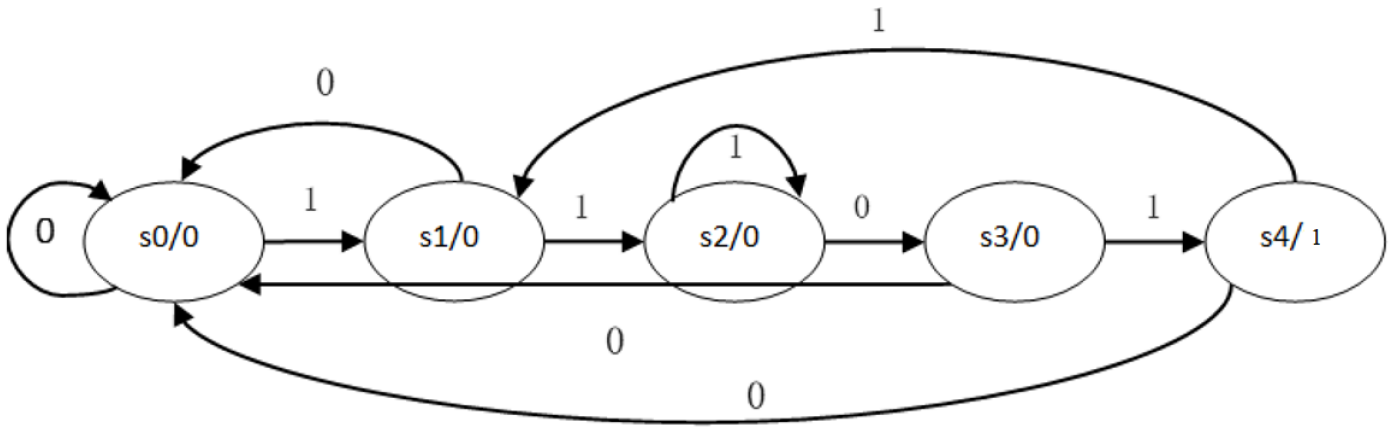
用Moore状态机和Mealy状态机分别实现输入序列"1101"的检测

Moore状态机序列检测器设计

定义以下状态
s0:未检测到“1”
s1:收到“1”
s2:收到“11”
s3:收到“110”
s4:收到“1101”

如果现态是s0，输入为0，那么下一状态还是停留在s0；
如果输入1，则转移到状态s1。
在状态s1，如果输入为0，则回到状态s0；如果输入为1，
那么就转移到s2。
在s2状态，如果输入为1，则停留在状态s2；如果输入为





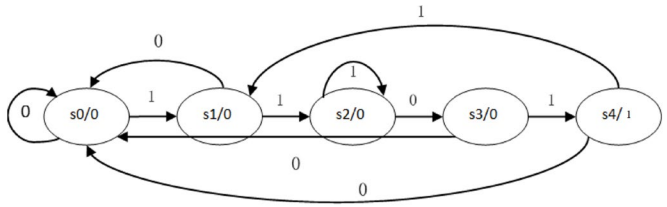
状态声明代码

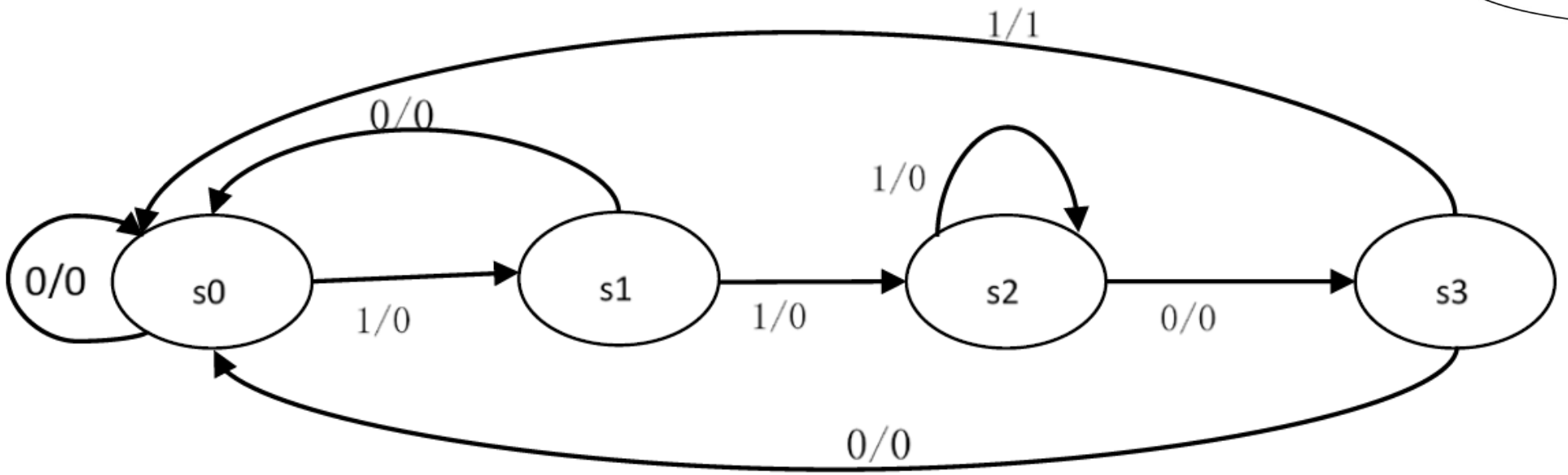
localparam[2:0] s0=3’b000, s1=3’b001,
s2=3’b010, s3=3’b011, s4=3’b100;

次级逻辑代码

```
case(cs)
s0: if(din==1’b1) nst=s1;
      else              nst=s0;
s1: if(din==1’b1) nst=s2;
      else              nst=s0;
s2: if(din==1’b0) nst=s3;
      else              nst=s2;
s3: if(din==1’b1) nst=s4;
      else              nst=s0;
s4: if(din==1’b1) nst=s1;
      else              nst=s0;
default: nst=s0;
endcase
```

Mealy状态机序列检测器：

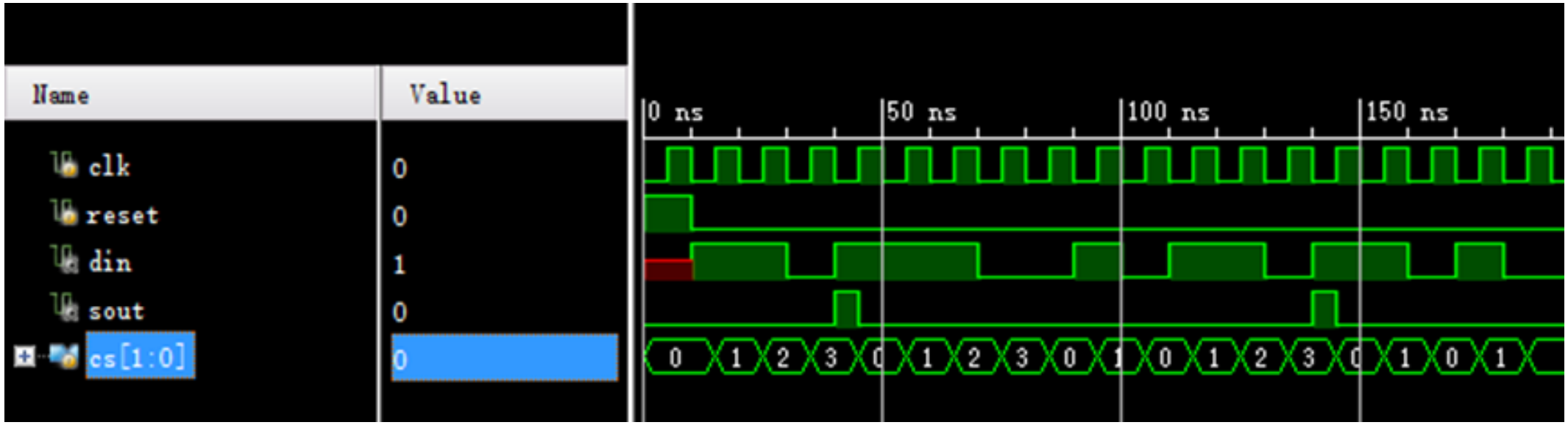




对比Mealy状态机与Moore状态机的状态图可知:
Moore状态机的检测结果输出是与时钟同步的；而Mealy状态机的检测结果输出是异步的，当输入发生变化时，输出就立即变化。因此Mealy状态机的输出比Moore状态机状态的输出提前一个周期。

```
localparam[1:0]
s0=2'b00,
s1=2'b01,
s2=2'b10,
s3=2'b11;
```

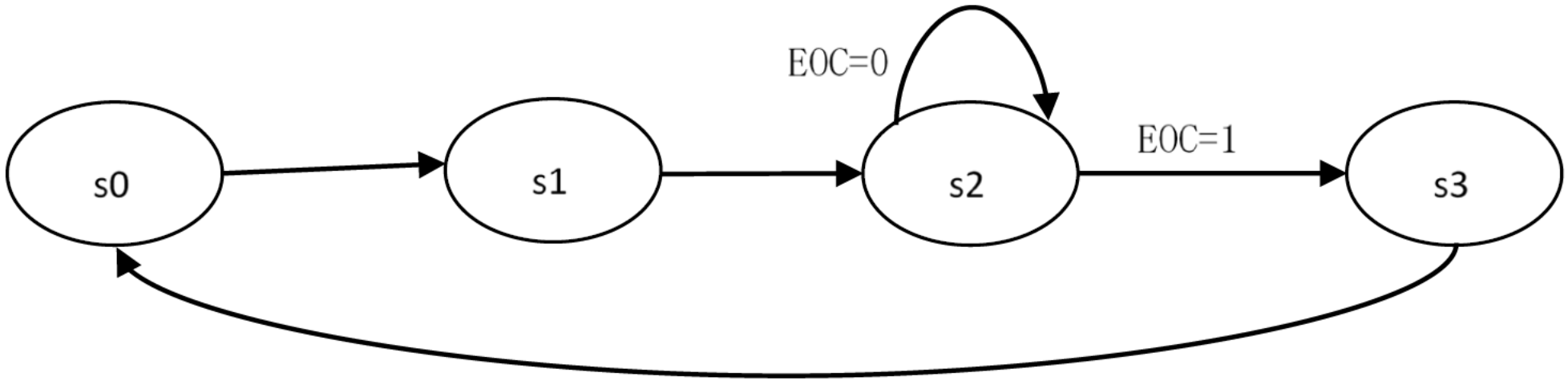
```
case(cs)
s0: if(din==1'b1) nst=s1;
      else                nst=s0;
s1: if(din==1'b1) nst=s2;
      else                nst=s0;
s2: if(din==1'b0) nst=s3;
      else                nst=s2;
s3: nst=s0;
default: nst=s0;
endcase
```



例、ADC采样控制电路设计

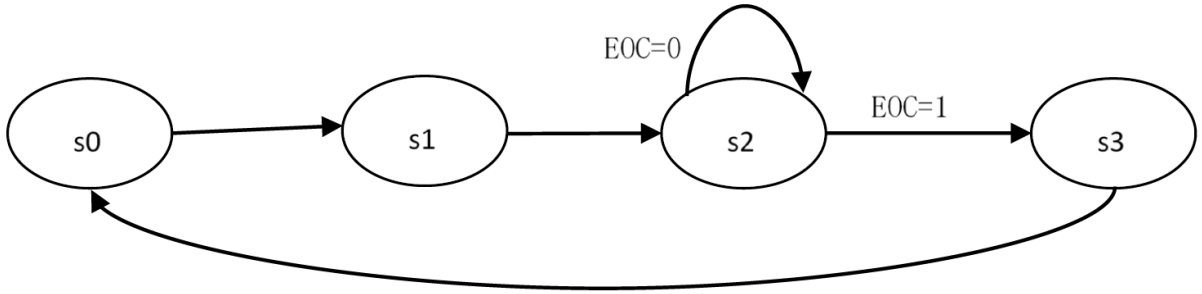
ADC转换控制状态机共有4个状态:
1. 初始化状态s0
2. 启动ADC状态s1
3. 等待ADC转换结束状态s2
4. 转换数据读取状态s3。

ADC0809控制的状态：s0-s1-s2-s3-s0，在时钟上升沿直接变化，只有在s2状态时，根据输入信号EOC来判断状态转移的下一状态。ADC在状态机控制下，依次在这4个状态切换，完成AD转换功能。



状态声明代码

```
localparam[1:0]
s0=2’b00,
s1=2’b01,
s2=2’b10
s3=2’b11;
```



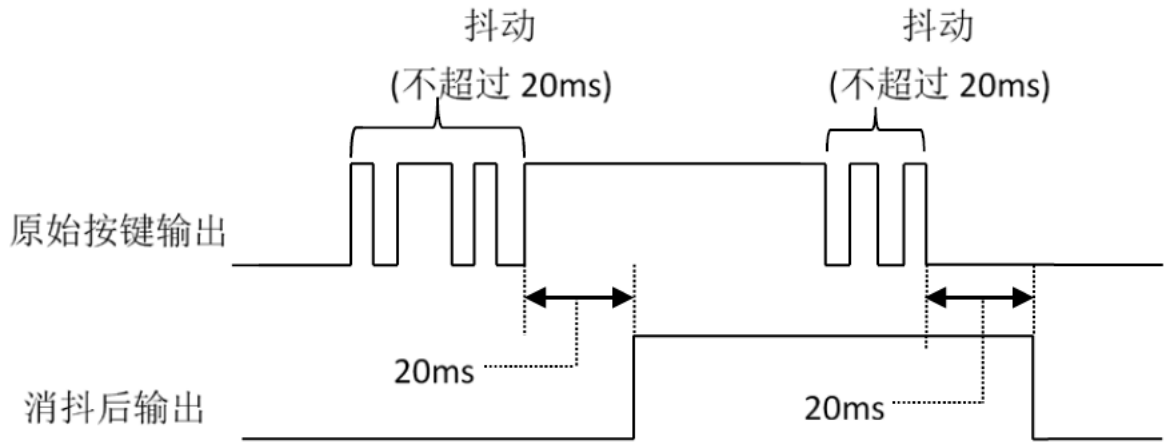
状态转移代码

```
reg[1:0] state_reg, state_next;
always@ (posedgeclk, posdge reset)
if(reset)
    state_reg<=s0;
else
    state_reg<=state_next;
```

次级逻辑代码

```
always@*
begin
    case(state_reg)
        s0: next_state=s1;
        s1: next_state=s2;
        s2: if(eoc==1’b0) next_state =s3;
              else                next_state =s2;
        s3: next_state=s0;
    endcase
end
```

例、按键消抖电路设计

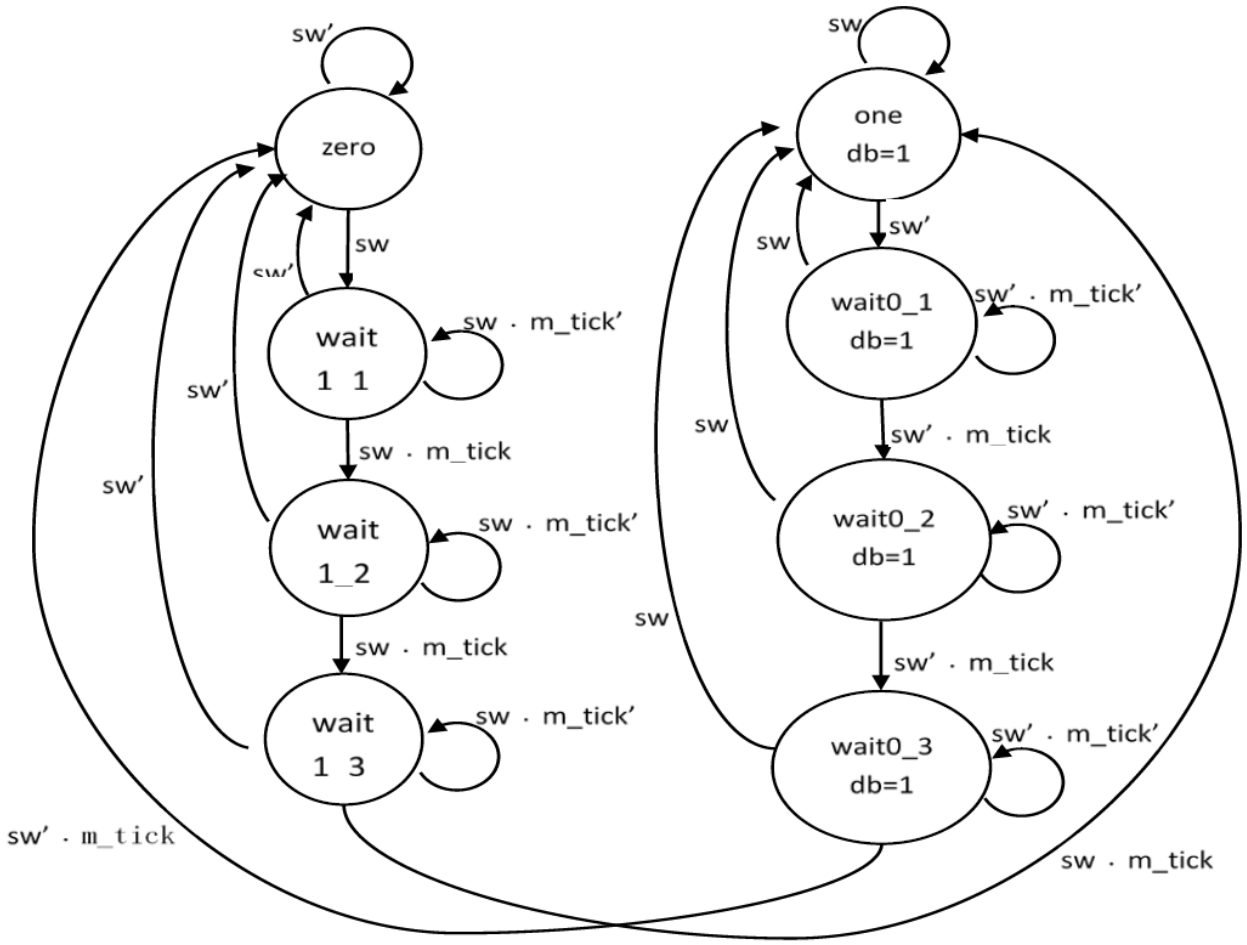


设计基于FSM的消抖电路，利用一个10ms的非同步定时器和有限状态机，计时器每10ms产生一个使能周期信号，有限状态机利用此信号来确定输入信号是否稳定。

有限状态机将消除时间较短的抖动，当输入信号稳定20ms以后才改变去抖动以后的输出值。

假定系统起始态是zero(one)态，当sw变为1(0)时，系统转换为wait1_1态。当处于wait1_1态时，有限状态机处于等待状态并将m_tick置为有效电平态。若sw变为0则表示1值所持续的时间过短有限状态机返回zero态。这个动作在wait1_2态和wait1_3态也将再重复2次。

zero：sw稳定在0值
one：sw稳定在1值。



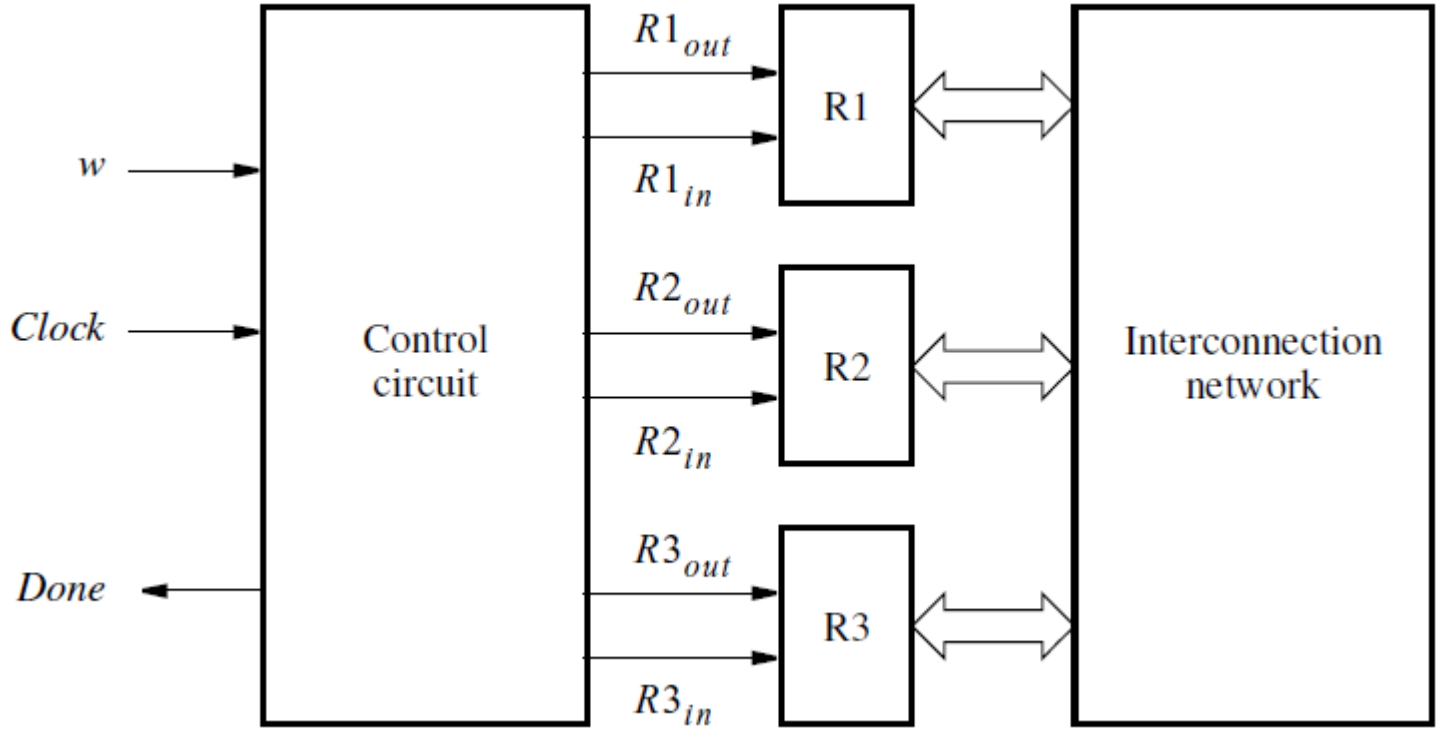
```
localparam[2:0]
zero=3’b000,
wait1_1=3’b001,
wait1_2=3’b010,
wait1_3=3’b011,
one=3’b100,
wait0_1=3’b101,
wait0_2=3’b110,
wait0_3=3’b111；
```

```
case(state_reg)
zero:if(sw) state_next=wait1_1;
wait1_1: if(~sw) state_next=zero;
          else if(m_tick)
            state_next=wait1_2;
wait1_2: if(~sw) state_next=zero;
          else if(m_tick)
            state_next=wait1_3;
wait1_3: if(~sw) state_next=zero;
          else if(m_tick)
            state_next=one;
one: begin db=1’b1;
          if(~sw) state_next=wait0_1;
      end
wait0_1:
    begin db=1'b1;
      if(sw) state_next=one;
else if(m_tick)
state_next=wait0_2;
end
wait0_2: begin db=1'b1;
            if(sw) state_next=one;
            else if(m_tick)
              state_next=wait0_3;
          end
wait0_3: begin db=1'b1;
            if(sw) state_next=one;
            else if(m_tick)
              state_next=zero;
          end
default: state_next=zero;
endcase
```

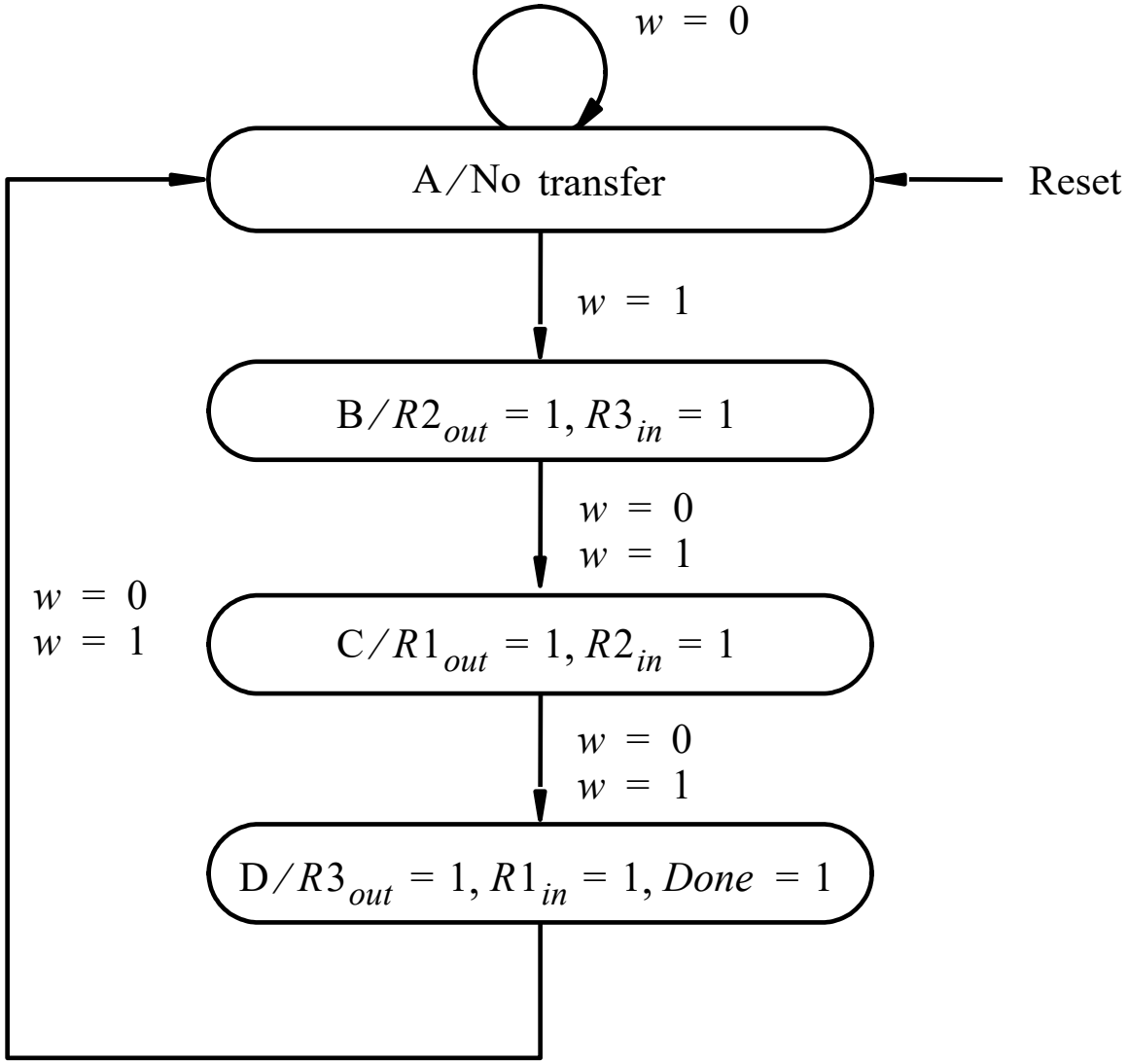
书中寄存器没有初始化

布朗等，数字逻辑基础与Verilog设计（原书第3版），机械工业出版社

例、当输入w=1时，交换R1和R2内容的控制电路。

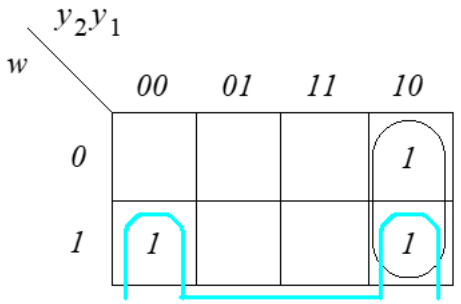


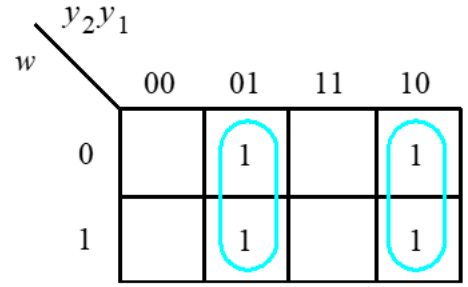
计算机系统中寄存器内容互换。Rk_{out}信号使得寄存器Rk的内容传输至互联网络。Rk_{in}信号使互联网络上的数据传输至寄存器Rk



Present state	Next state		Outputs						
	$w=0$	$w=1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

	Present state	Next state		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	0 1	0	0	0	0	0	0	0
B	01	10	1 0	0	0	1	0	0	1	0
C	10	11	1 1	1	0	0	1	0	0	0
D	11	00	0 0	0	1	0	0	1	0	1





$$Y_1 = w\bar{y}_1 + \bar{y}_1 y_2$$

$$Y_2 = y_1\bar{y}_2 + \bar{y}_1 y_2$$

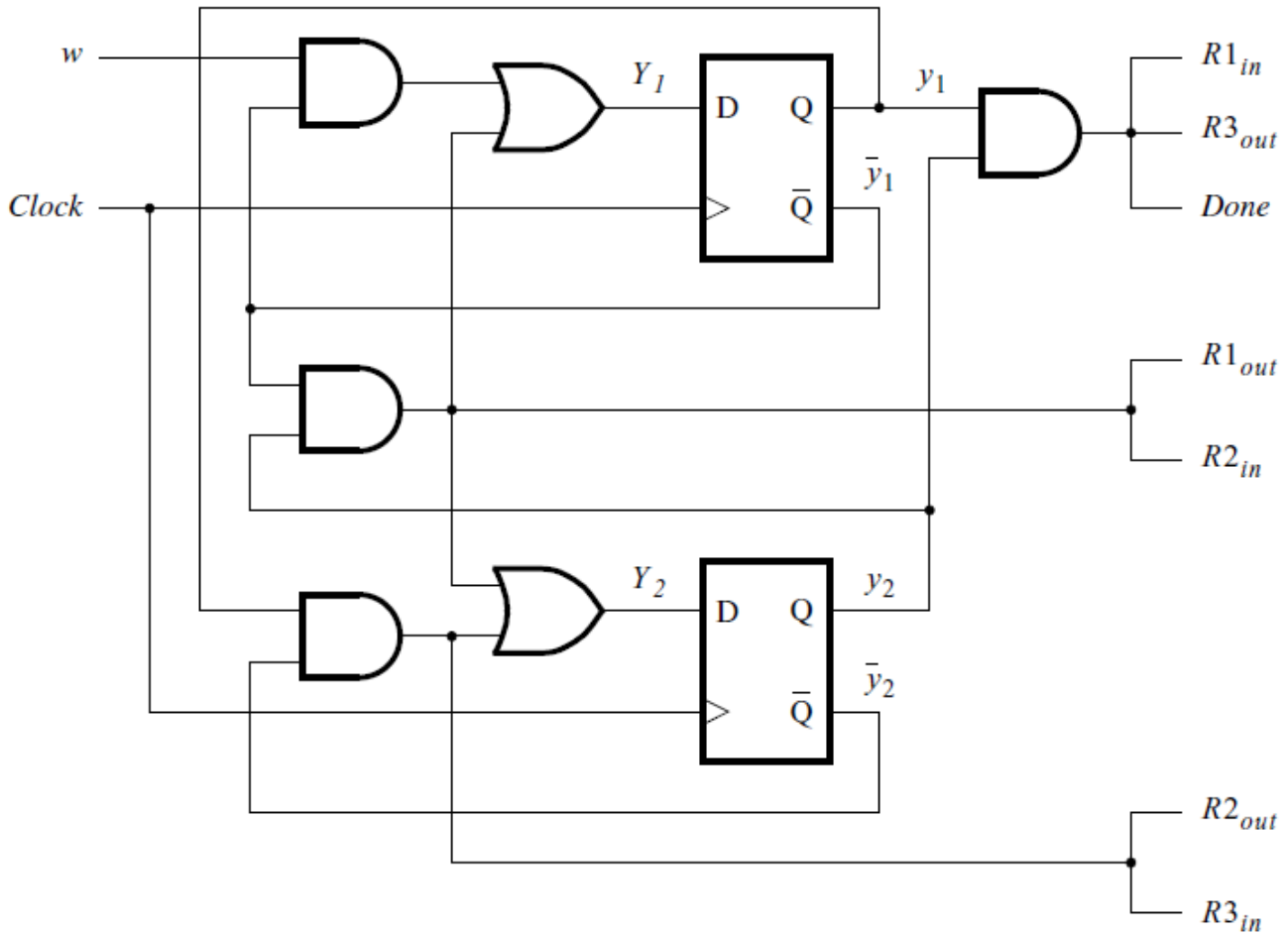
$$Y_1 = w\bar{y}_1 + \bar{y}_1 y_2$$

$$Y_2 = y_1\bar{y}_2 + \bar{y}_1 y_2$$

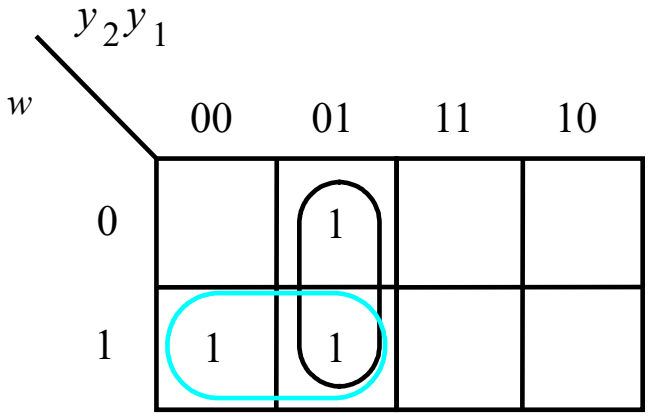
	Present state	Next state		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	01	0	0	0	0	0	0	0
B	01	10	10	0	0	1	0	0	1	0
C	10	11	11	1	0	0	1	0	0	0
D	11	00	00	0	1	0	0	1	0	1

Present state	Next state		Outputs						
	$w=0$	$w=1$							
			$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

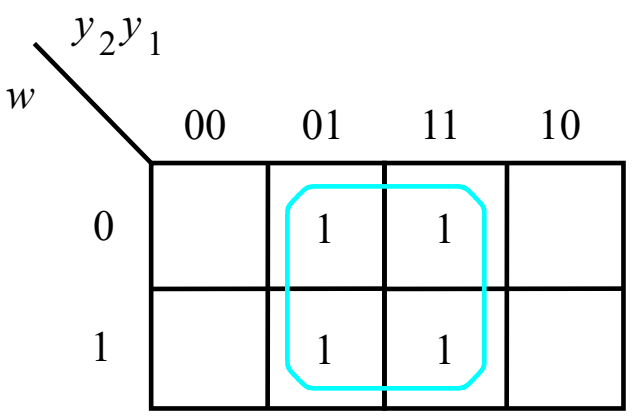
	Present state	Nextstate		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	01	0	0	0	0	0	0	0
B	01	11	11	0	0	1	0	0	1	0
C	11	10	10	1	0	0	1	0	0	0
D	10	00	00	0	1	0	0	1	0	1



	Present state	Nextstate		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	0 0	01	0	0	0	0	0	0	0
B	01	1 1	11	0	0	1	0	0	1	0
C	11	1 0	10	1	0	0	1	0	0	0
D	10	0 0	00	0	1	0	0	1	0	1



$$Y_1 = w\bar{y}_2 + y_1\bar{y}_2$$



$$Y_2 = y_1$$

Present state	Next state		Outputs						
	$w=0$	$w=1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

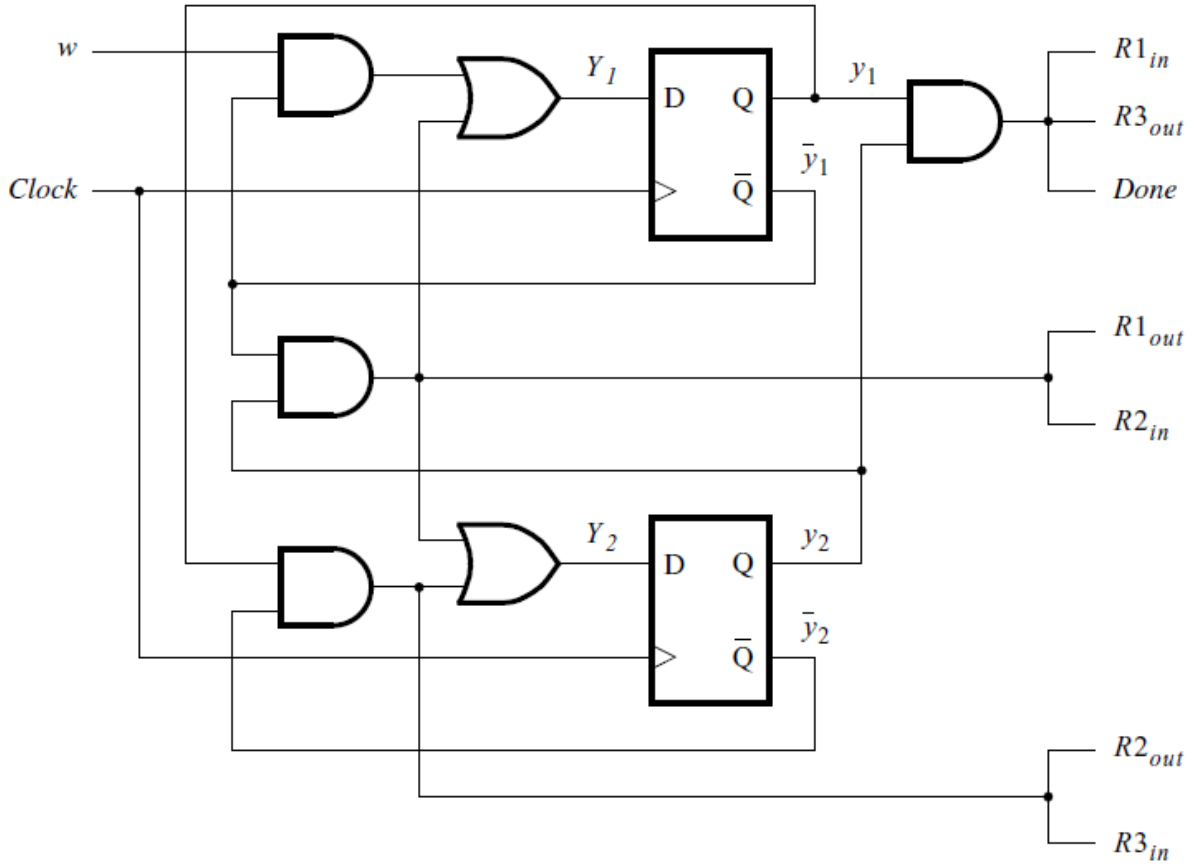
	Present state	Next state		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	0 1	0	0	0	0	0	0	0
B	01	10	1 0	0	0	1	0	0	1	0
C	10	11	1 1	1	0	0	1	0	0	0
D	11	00	0 0	0	1	0	0	1	0	1

$$Y_1 = w\bar{y}_1 + \bar{y}_1y_2$$

$$Y_2 = y_1\bar{y}_2 + \bar{y}_1y_2$$

独热（ One-hot ）编码

	Present state	Nextstate	
		$w = 0$	$w = 1$
	$y_4y_3y_2y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$
A	0 001	0001	0010
B	0 010	0100	0100
C	0 100	1000	1000
D	1 000	0001	0001



	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	0 0	01	0	0	0	0	0	0	0
B	01	1 1	11	0	0	1	0	0	1	0
C	11	1 0	10	1	0	0	1	0	0	0
D	10	0 0	00	0	1	0	0	1	0	1

$Y_1=w'y_1+y_4$

$Y_2=wy_1$

$Y_3=y_2$

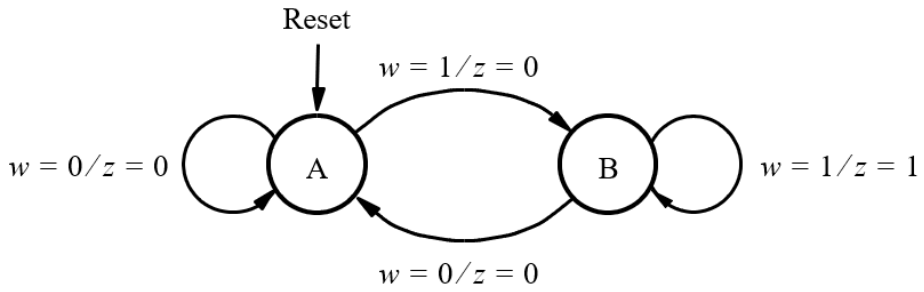
$Y_4=y_3$

$R1_{out}=R2_{in}=y_3$

$R1_{in}=R3_{out}=Done=y_4$

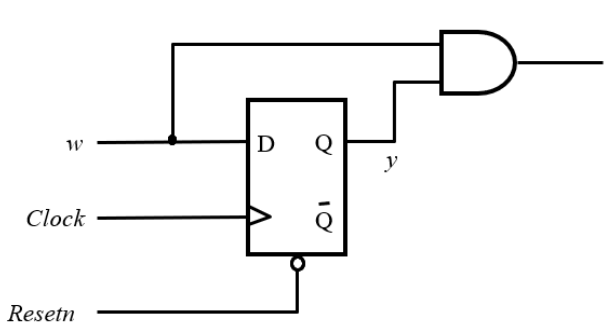
$R2_{out}=R3_{in}=y_2$

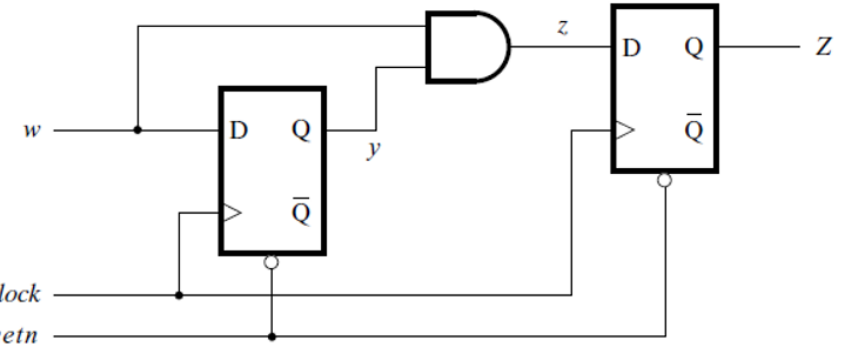
Mealy状态模型

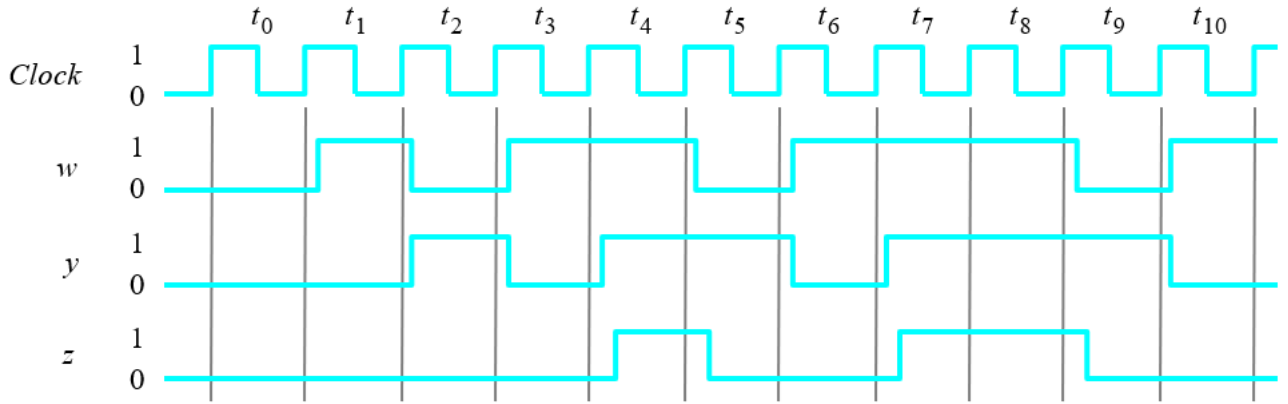


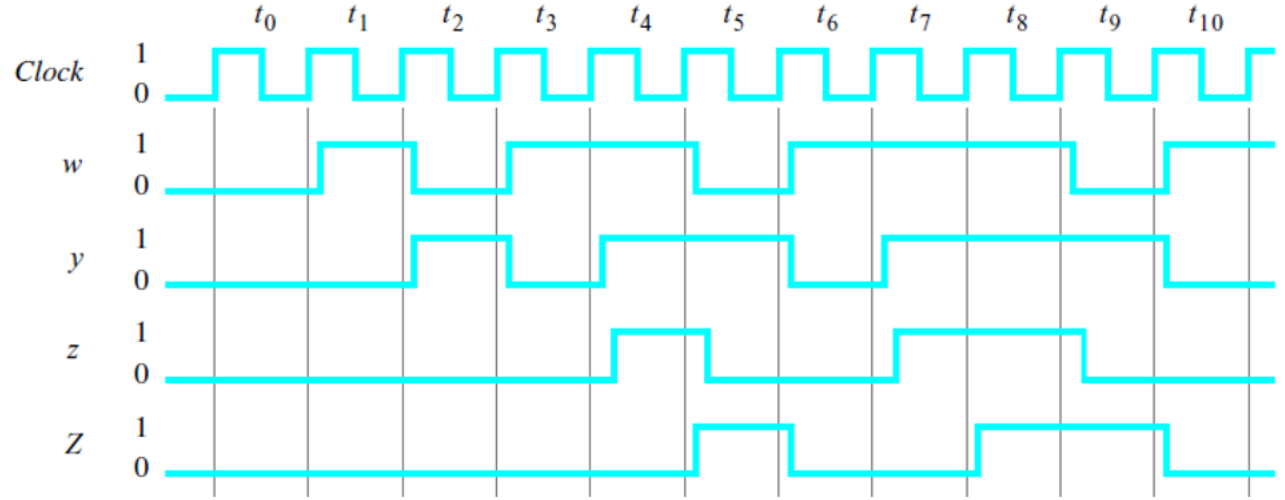
Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	A	B	0	1

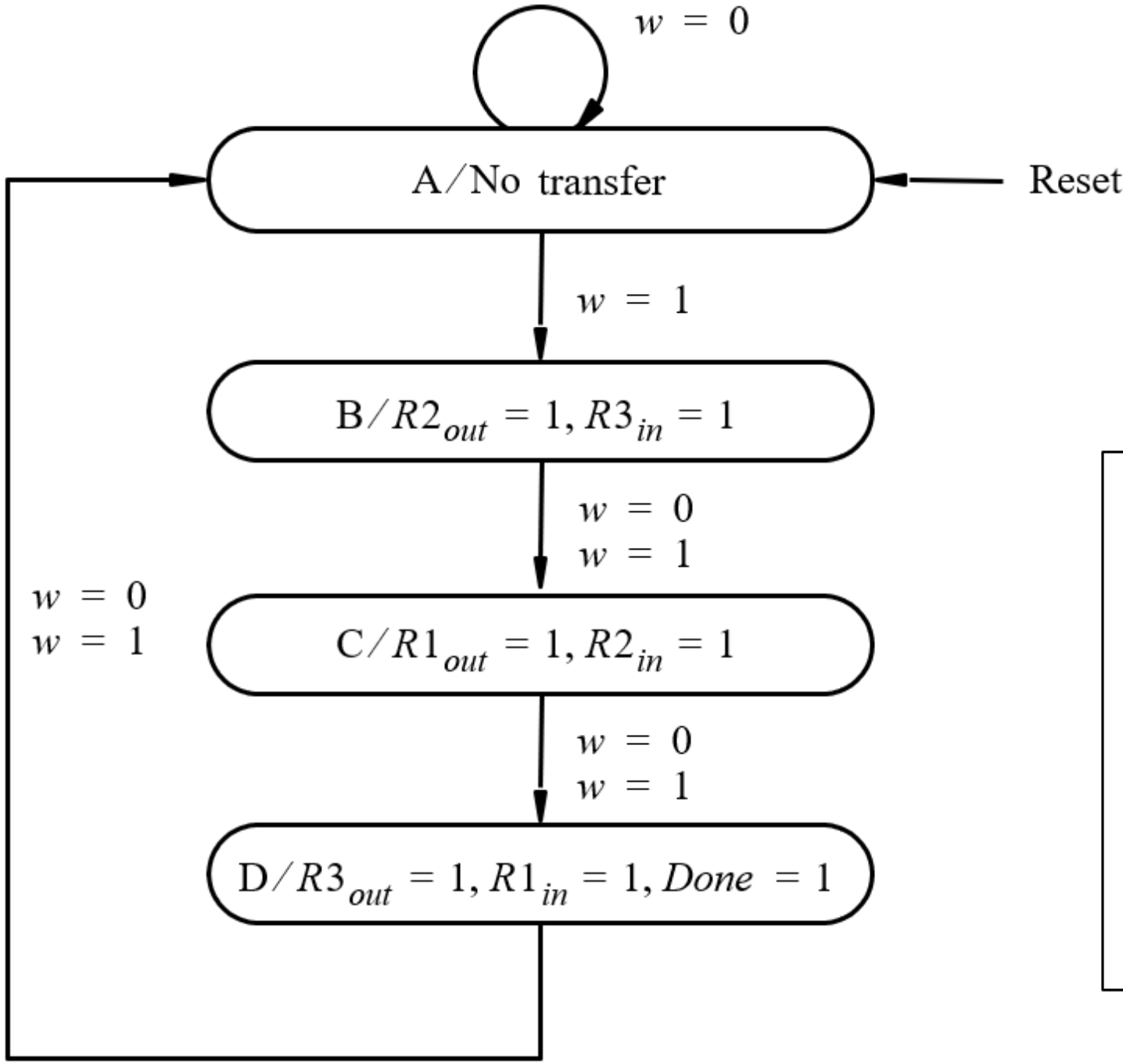
Present state	Next state		Output	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
y	Y	Y	z	z
A 0	0	1	0	0
B 1	0	1	0	1

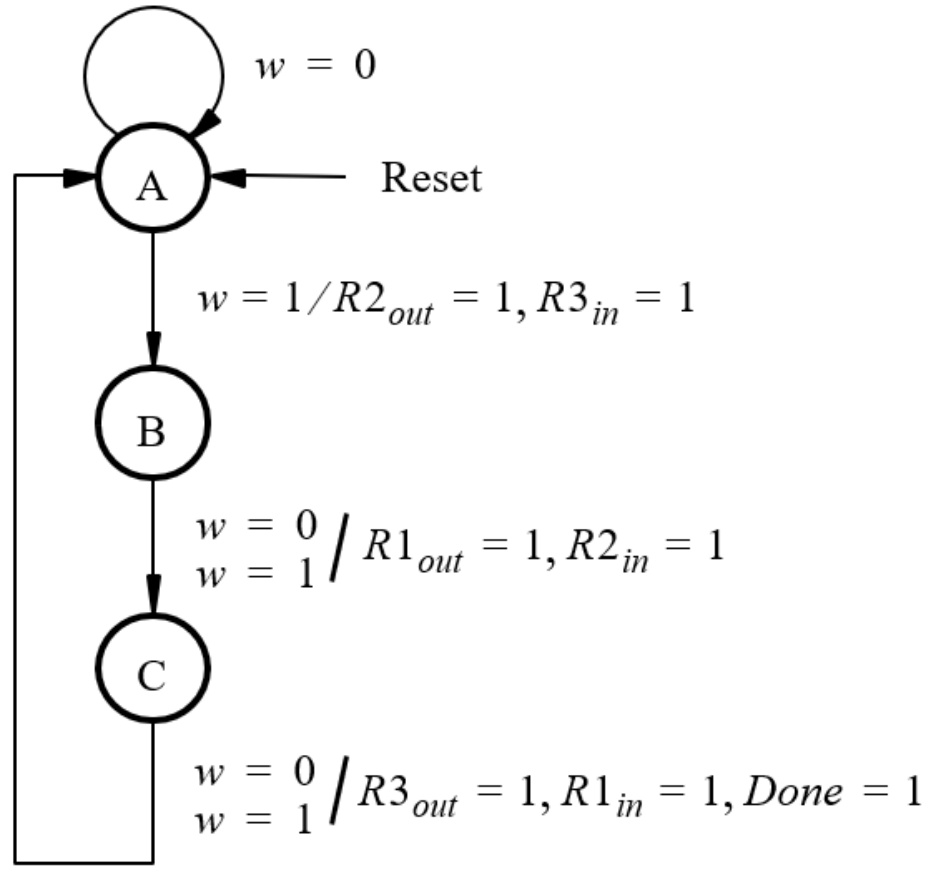


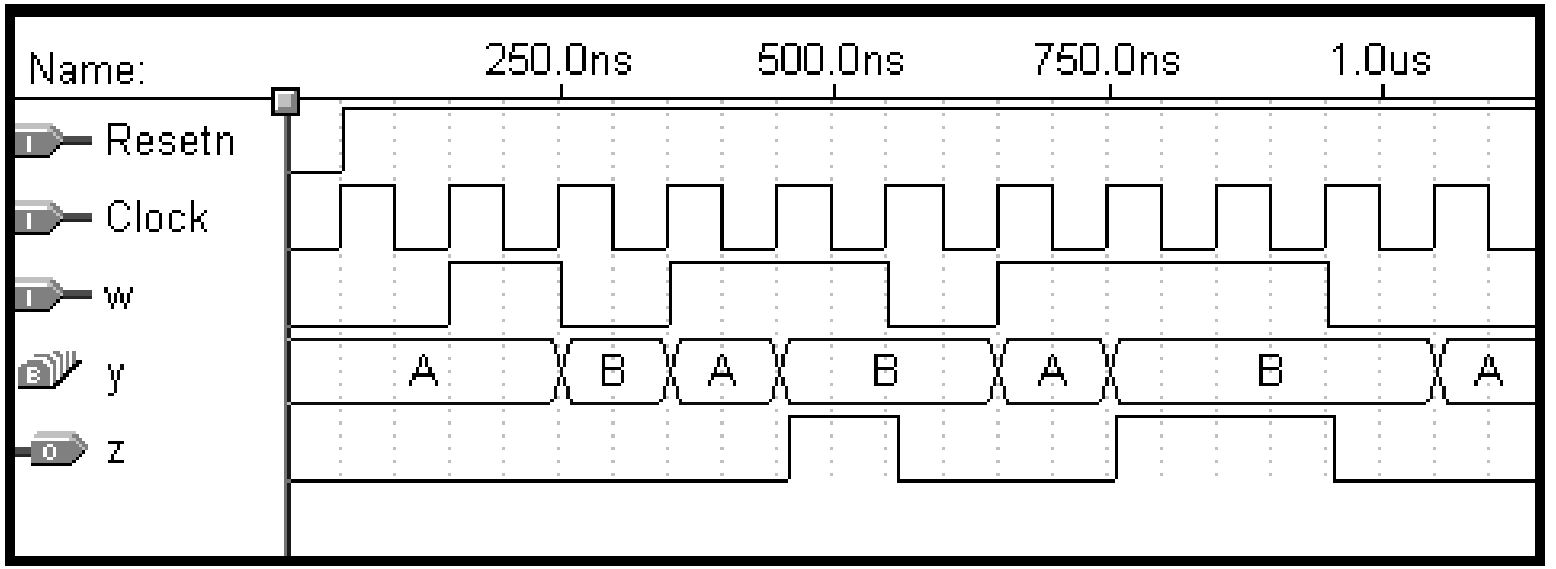


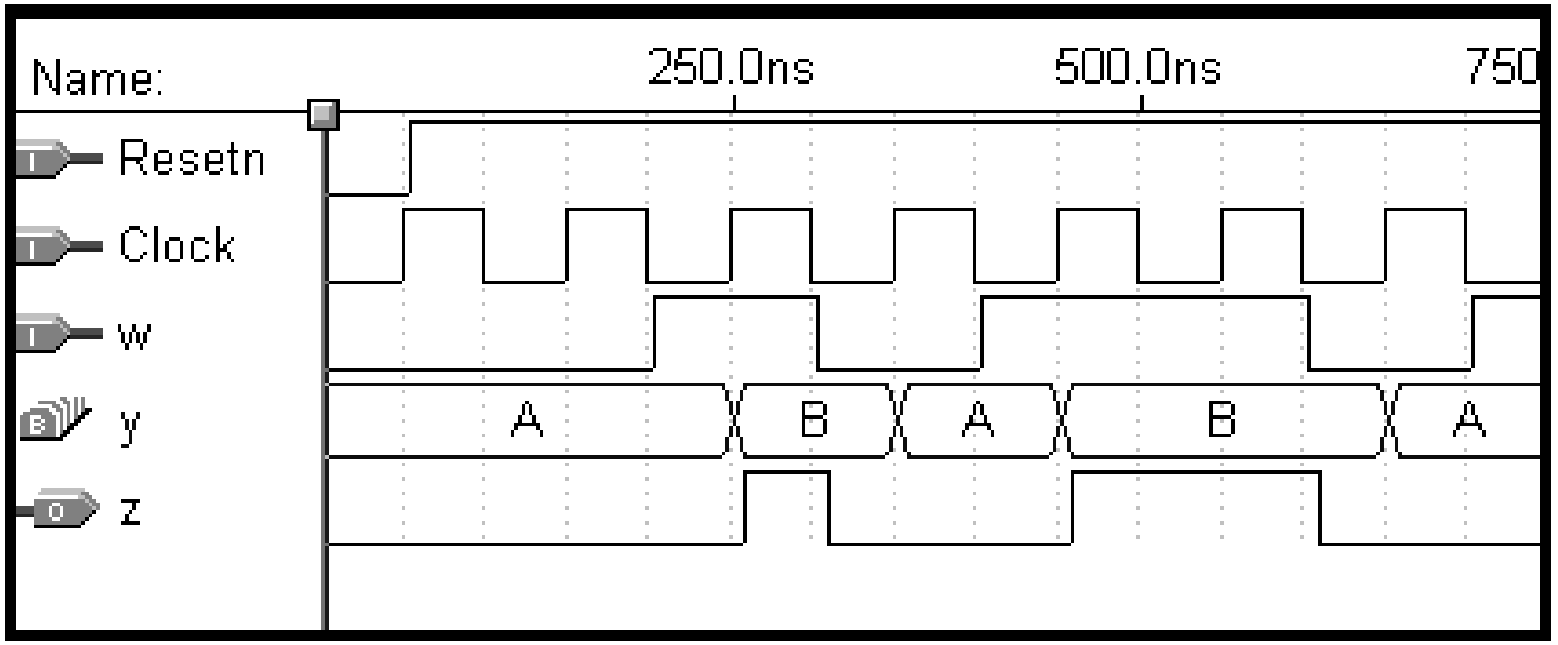





w = 0
A/No transfer
Reset
w = 1
B/R2out = 1, R3in = 1
w = 0
w = 1
w = 0
w = 1
C/R1out = 1, R2in = 1
w = 0
w = 1
D/R3out = 1, R1in = 1, Done = 1

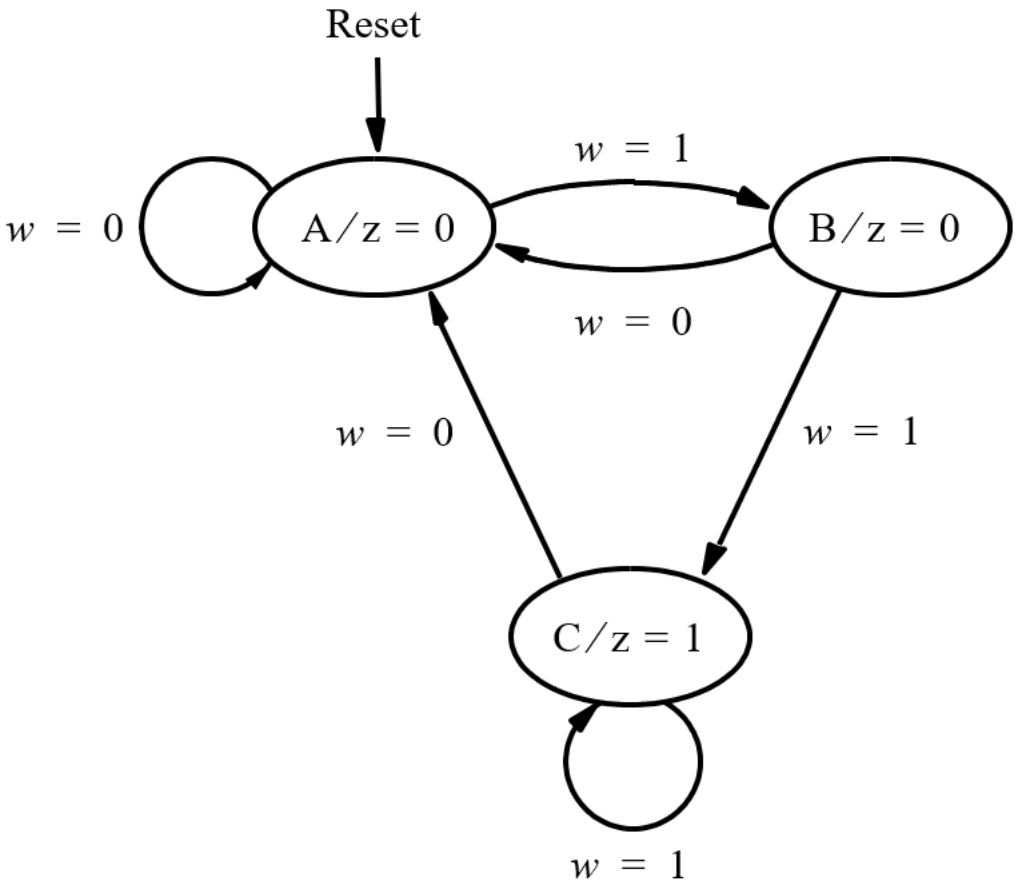

w = 0
A
Reset
w = 1/R2out = 1, R3in = 1
B
w = 0
w = 1
/ R1out = 1, R2in = 1
C
w = 0
w = 1
/ R3out = 1, R1in = 1, Done = 1





Mealy型状态机异步输入时的潜在问题

状态机的Verilog实现风格




```
module simple (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output z;
    reg [2:1] y, Y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;

    // Define the next state combinational circuit
    always @(w, y)
        case (y)
            A: if (w)   Y = B;
               else     Y = A;
            B: if (w)   Y = C;
               else     Y = A;
            C: if (w)   Y = C;
               else     Y = A;
            default:    Y = 2'bxx;
        endcase

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0)                   y <= A;
        elsey <= Y;

    // Define output
    assign z = (y = = C);

endmodule
```

```
module simple (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output z;
    reg [2:1] y, Y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;

    // Define the next state combinational circuit
    always @(w, y)
        case (y)
            A: if (w)   Y = B;
               else     Y = A;
            B: if (w)   Y = C;
               else     Y = A;
            C: if (w)   Y = C;
               else     Y = A;
            default:    Y = 2'bxx;
        endcase

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0)                y <= A;
        elsey <= Y;

    // Define output
    assign z = (y = = C);

endmodule
```

```
module simple (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output reg z;
    reg [2:1] y, Y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;

    // Define the next state combinational circuit
    always @(w, y)
    begin
        case (y)
            A: if (w)   Y = B;
               else     Y = A;
            B: if (w)   Y = C;
               else     Y = A;
            C: if (w)   Y = C;
               else     Y = A;
            default:    Y = 2'bxx;
        endcase
        z = (y = = C); //Define output
    end

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0)  y <= A;
        elsey <= Y;

endmodule
```

```
module simple (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output z;
    reg [2:1] y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0)   y <= A;
        else
            case (y)
                A: if (w)   y <= B;
                   else     y <= A;
                B: if (w)   y <= C;
                   else     y <= A;
                C: if (w)   y <= C;
                   else     y <= A;
                default:    y <= 2'bxx;
            endcase

    // Define output
    assign z = (y = = C);

endmodule
```

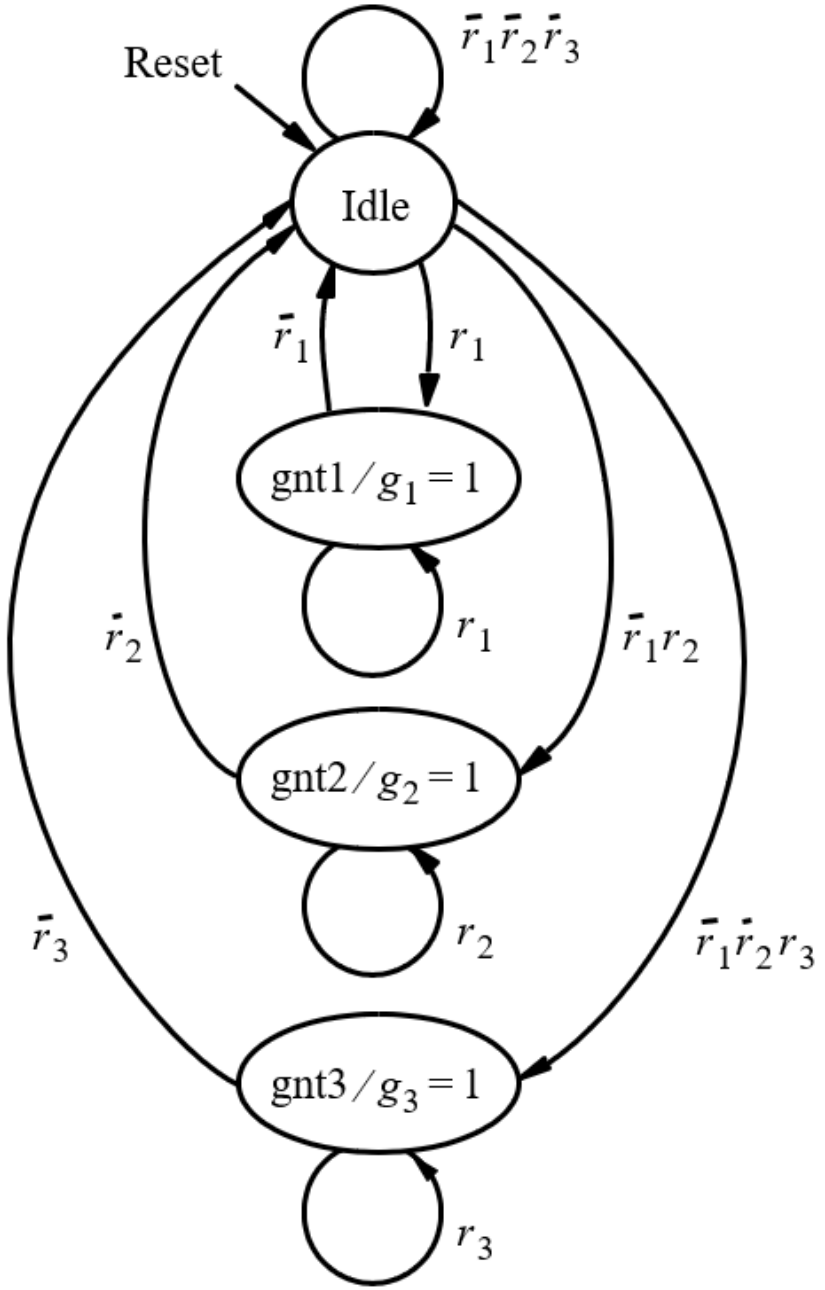
状态机用三个不同的过程实现。一般一个always中只对一个信号赋值。如果多个信号的判断条件总是完全相同，也可以在同一个always中对多个信号同时赋值。

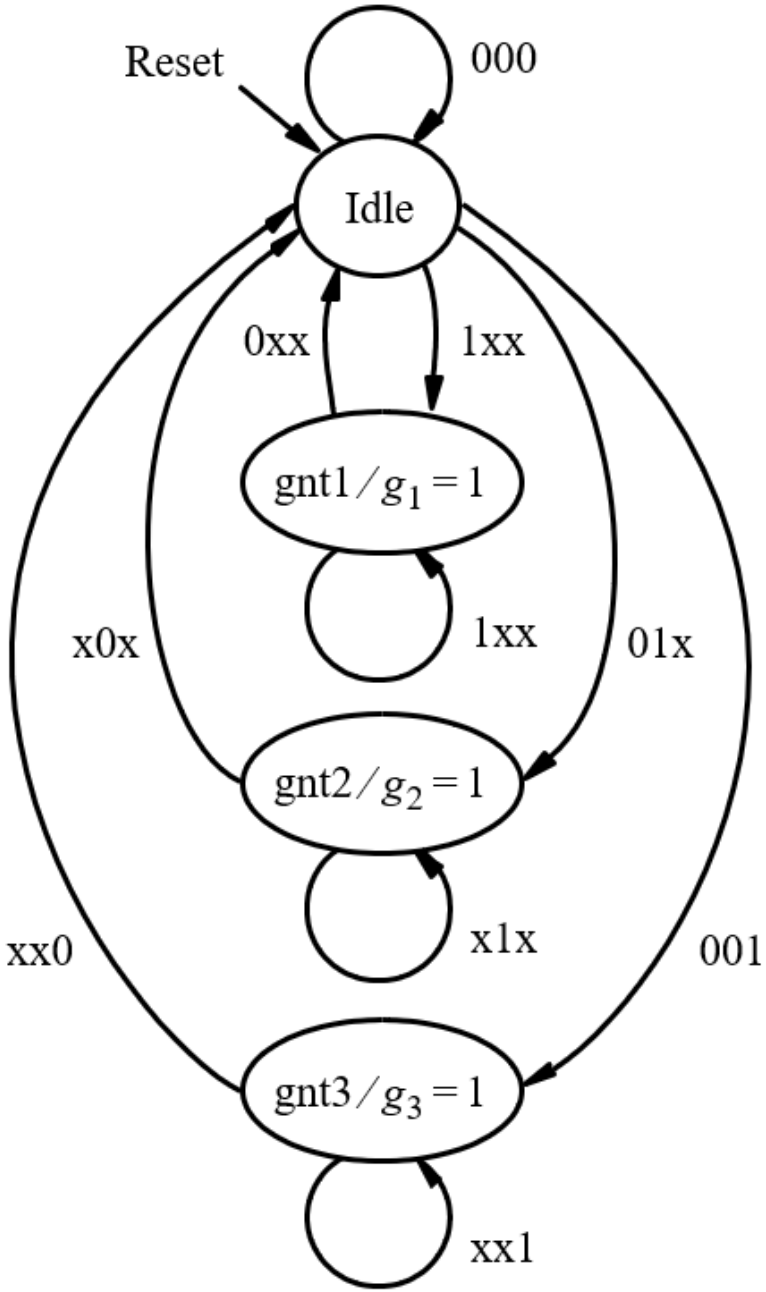
```
//第一段，状态变化赋值，时序逻辑
reg [3:0]cs,ns;
always @ (posedge clk or negedge rstn)
  if(!rstn)
     cs<=sidle;
  else
     cs<=ns;

//第二段，状态变化逻辑，组合逻辑
always @ (cs or …)
case (cs)
  sidle :if(…) ns= s1;          else ns=sidle;     //不允许出现  ns=ns;
  s1      :if(…) ns= sover;    else ns= s1;
  sover :if(…) ns= sidle;     else ns= sover;
default:ns=sidle;
endcase
```

```
//第三段
//对输出赋值：组合电路
//一般来说，一个 always 模块，只对一个信号赋值。如果某几个信号的变化条件
//总是一样，也可以在同一个 always 中对这几个信号同时赋值。
always @(敏感信号列表)
begin
if(……)
   output_reg=……;
else
   output_reg=……;
end
//对输出赋值：时序电路
//一般来说，一个 always 模块，只对一个信号赋值。如果某几个信号的变化条件
//总是一样，也可以在同一个 always 中对这几个信号同时赋值。
always @ (posedge clk or negedge rstn)
begin
if(!rstn)
   signal_a_reg <= 1'b0;
else
if (cs == sover && signal_x_reg == 1'b1 && ….)
   signal_a_reg <= 1'b1;
end
```

例、仲裁电路：给
源的接入控制。在
用该资源，假设系
的上升沿之后才发
一个输入（请求信
应地给每个设备产
grant）。一个设备
需要先发送请求信
FSM会考虑所有有
先级列表，选择一
信号。当设备完成

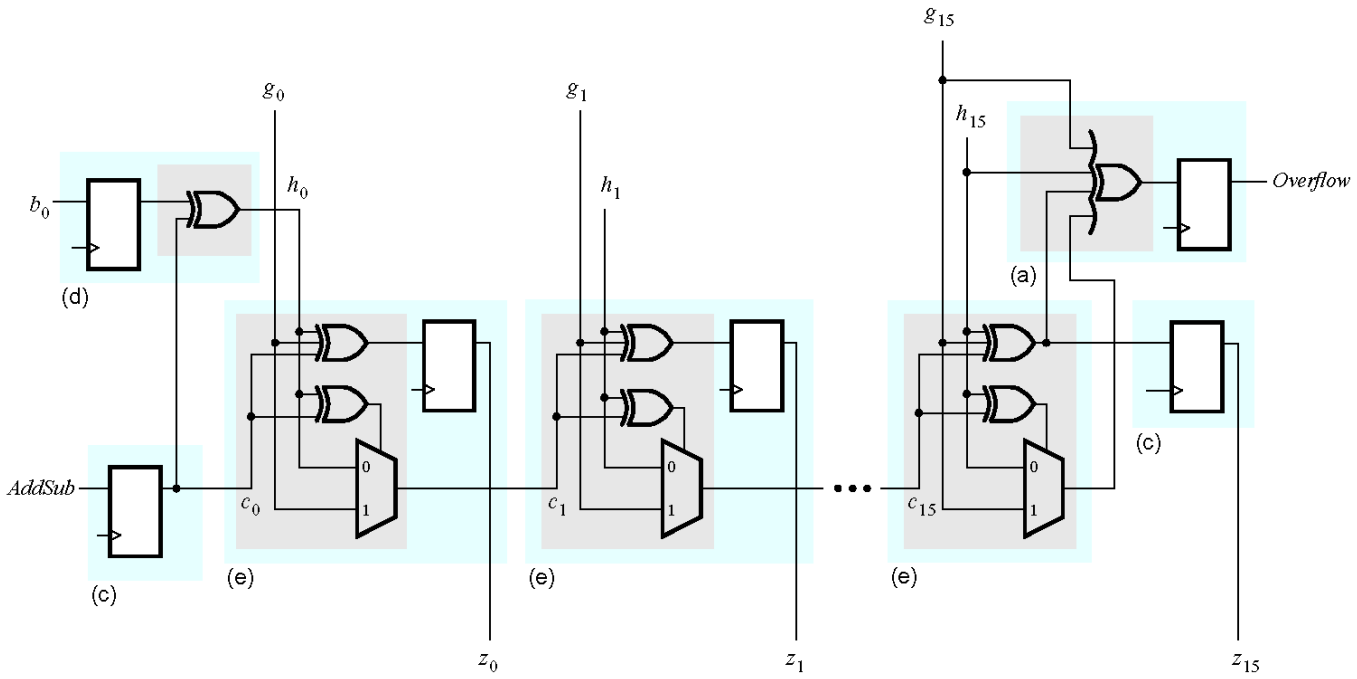
假设系统有3个设备
r1、r2、r3，授权信
先级依次为设备1>

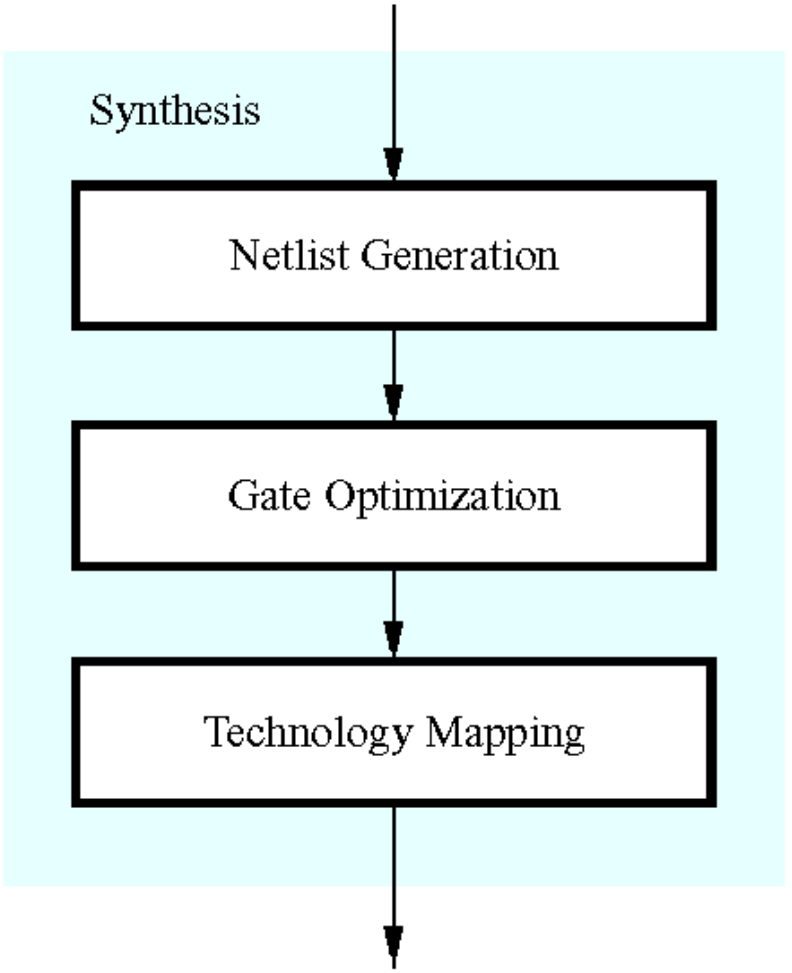




计算机辅助设计工具（综合工具）

网表生成：检查语法、生成网表；
门级优化：逻辑优化（成本、速度）；
工艺映射：网表中的元件如何在目标芯片的可用资源中实现。





综合工具中的3个阶段

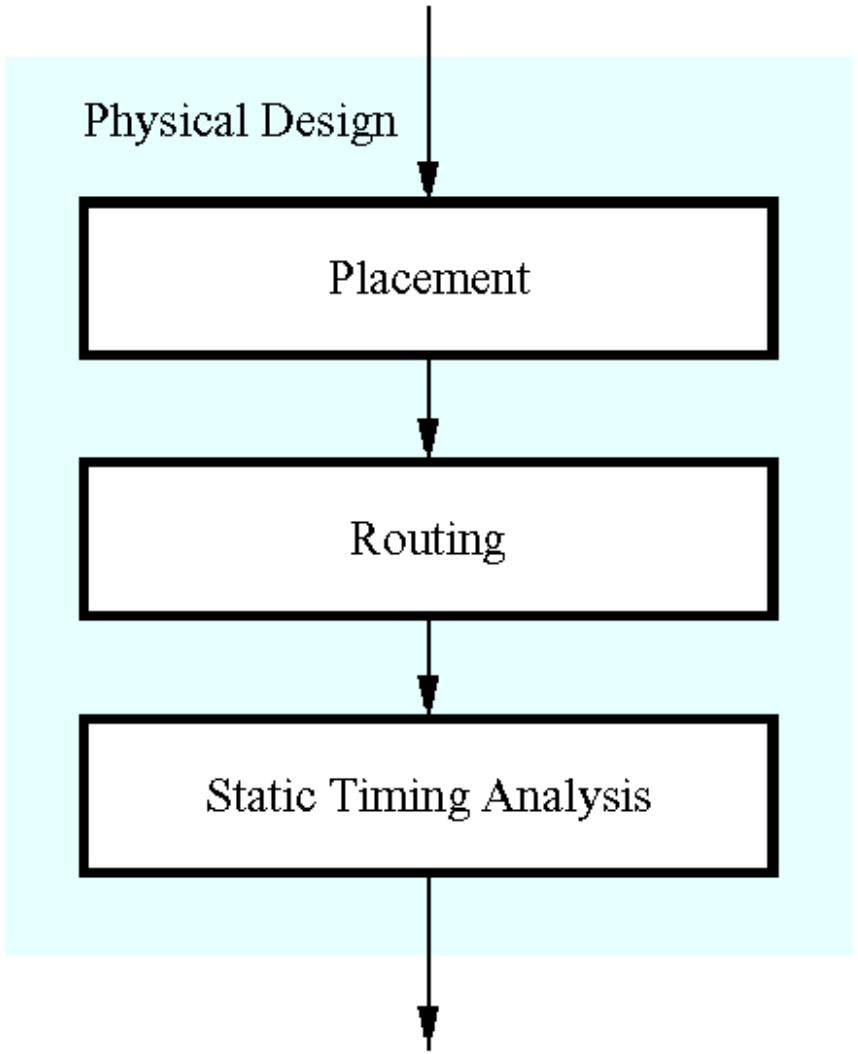
物理设计

布局：为网表中的逻辑模块选择合适的位置；
布线：把放置在芯片中的模块用线连接起来；
静态时序分析：检测电路延迟信息。

Parameter	Actual	Required	Slack	From	To
f_{max}	261.1 MHz	200 MHz	1.17 ns	*AddSub*	*Overflow*
t_{su}	2.356 ns	10.0 ns	7.644 ns	b_0	$breg_0$
t_{co}	6.772 ns	10.0 ns	3.228 ns	$zreg_0$	z_0
t_h	0.240 ns	10.0 ns	9.76 ns	b_1	$breg_1$

Table 10.2 Major CAD tool products.

Vendor Name	WWW Locator	Product Names
Altera	altera.com	Quartus II
Mentor Graphics	mentorgraphics.com	ModelSim, Precision
Synopsys	synopsys.com	Design Compiler, VCS
Xilinx	xilinx.com	ISE, Vivado



第五章作业：

1、课后习题：5.3、5.4、5.5。

2、如何用数字电路实现超越函数计算，学习CORDIC算法。