10. 时序电路设计

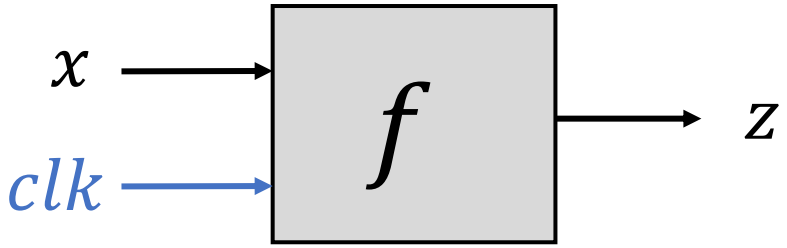
xgsun@fudan.edu.cn

孙晓光

2024-11-3

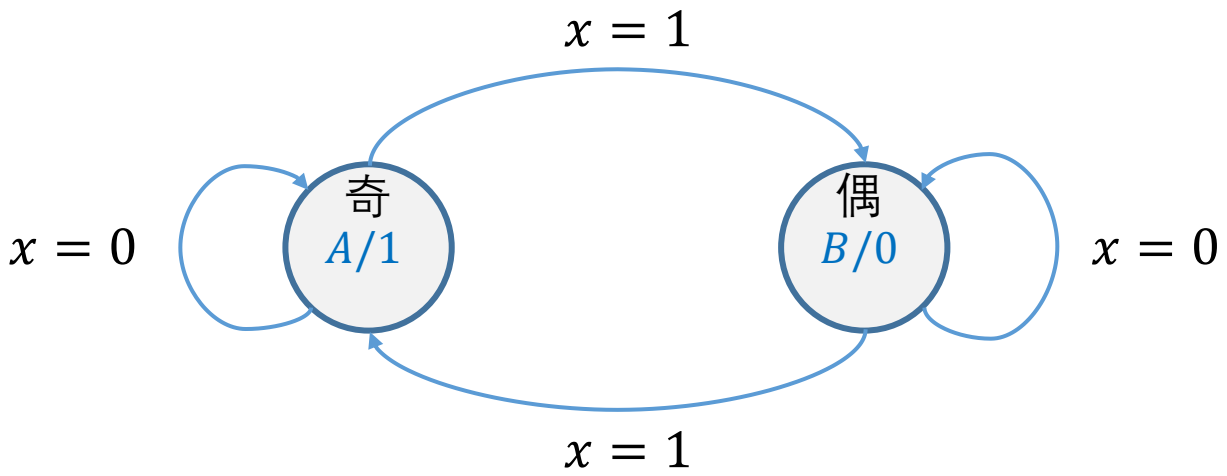
设计：输入**序列**中1的数目为奇数时，输出为1

00100110111



输入 x：0 0 1 _ 0 0 1 1 _ 0 1 1 1

输出 z：**0 0 0 _ 1 1 1 0 _ 1 1 0 1**



状态可对输入序列进行记忆，也可对输入序列产生的结果进行记忆。

关键：弄清有多少信息需要记忆，从而确定需要多少个状态。

状态图的**直接构图法**

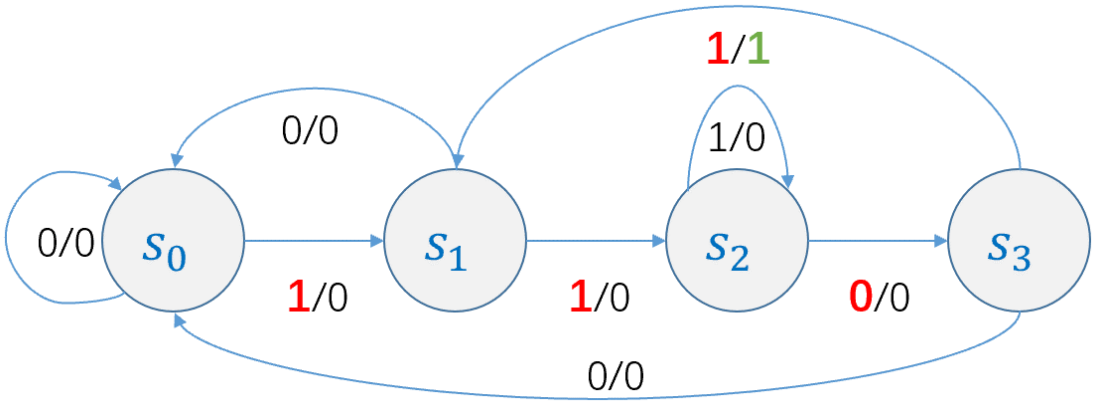
① 先假定一个**初态**

② 然后每加入一个**输入**，就可确定其**次态**

该**次态**：　可能就是现态本身，

也可能是已有的另一个状态，

还可能是新增状态.



③ 重复步骤②，直至每一个现态向其次态转换都被考虑，且不再增加新状态。

1

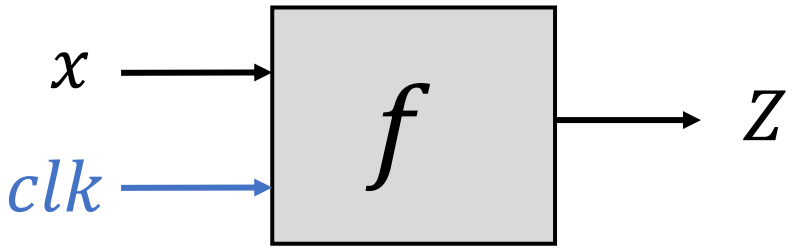
110检测器

【例1】设计检测是否输入“110” (1)

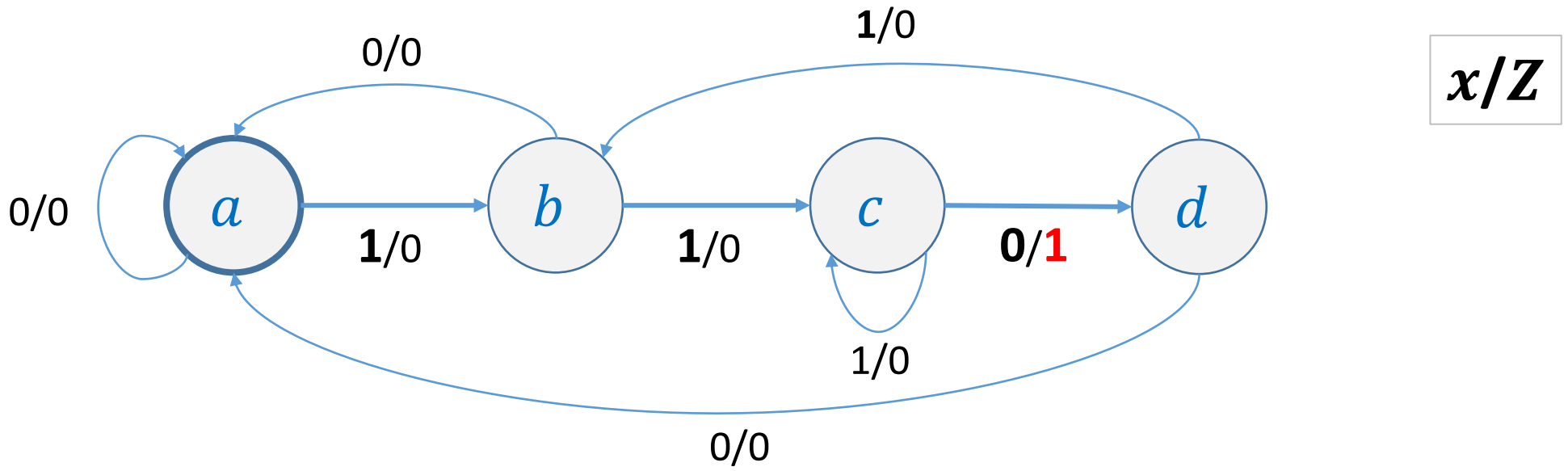
① 确定输入变量、输出变量

输入变量： x（二进制序列 “000011 **110** 000”）

输出变量： Z（=1: 当连续输入“**110**”）

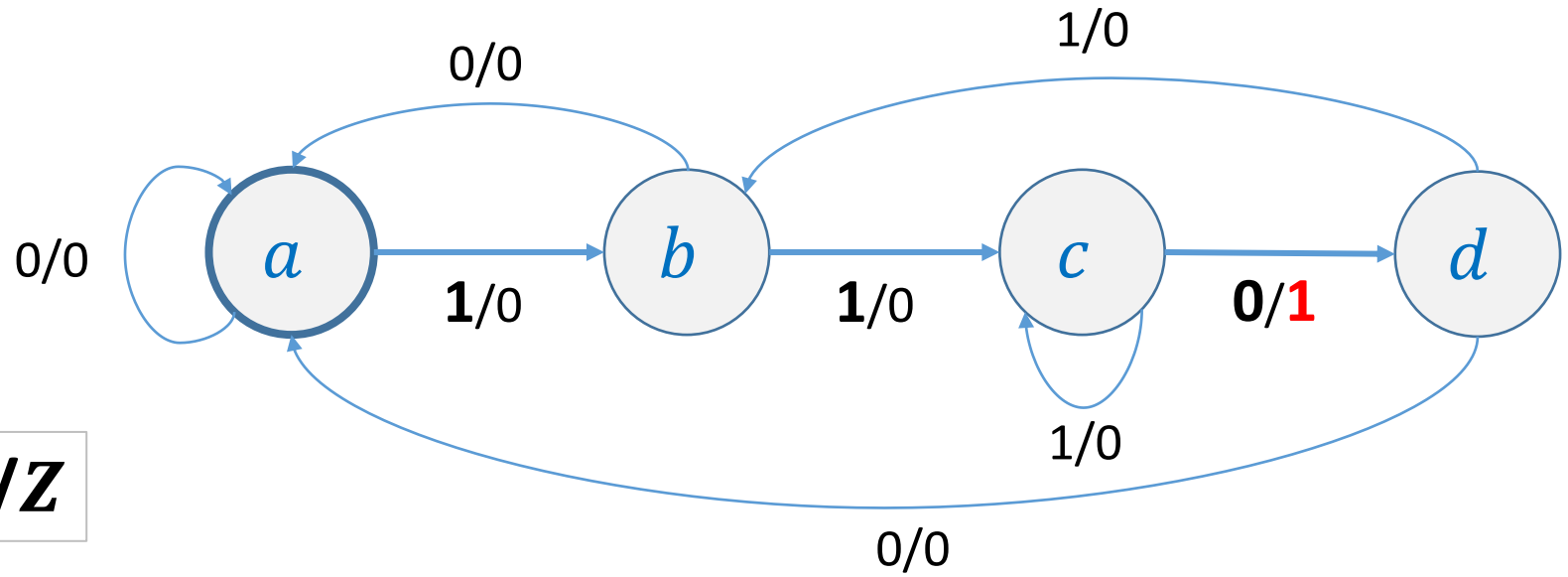


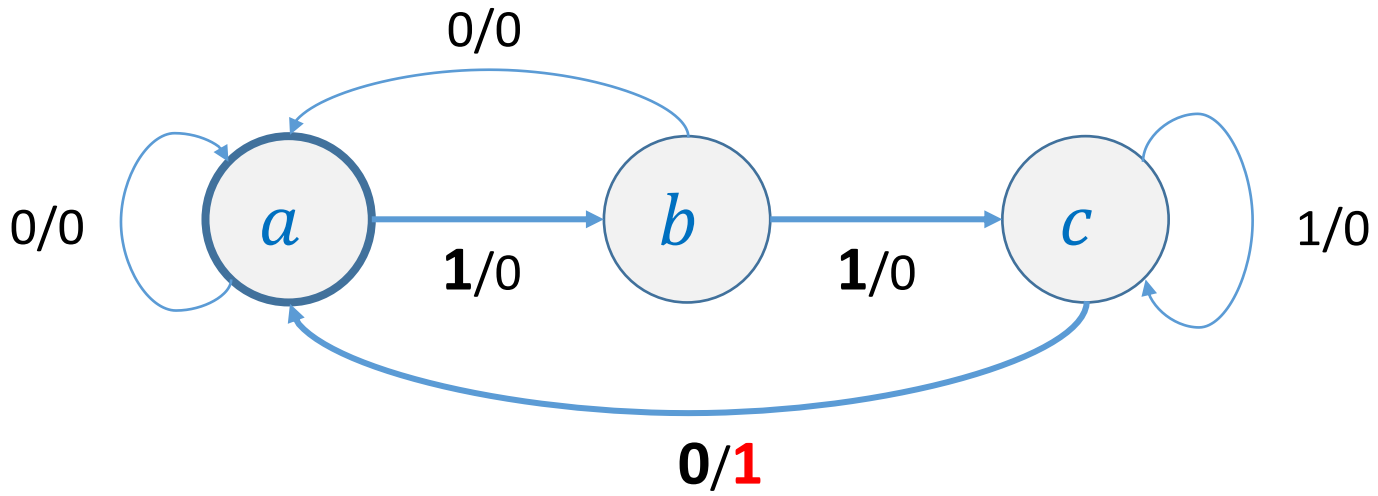
② 画出状态图



x/Z

【例1】设计检测是否输入“110” (2)





原始状态表

S	S^* / Z	
	$x=0$	$x=1$
a	a / 0	b / 0
b	a / 0	c / 0
c	d / 1	c / 0
d	a / 0	b / 0

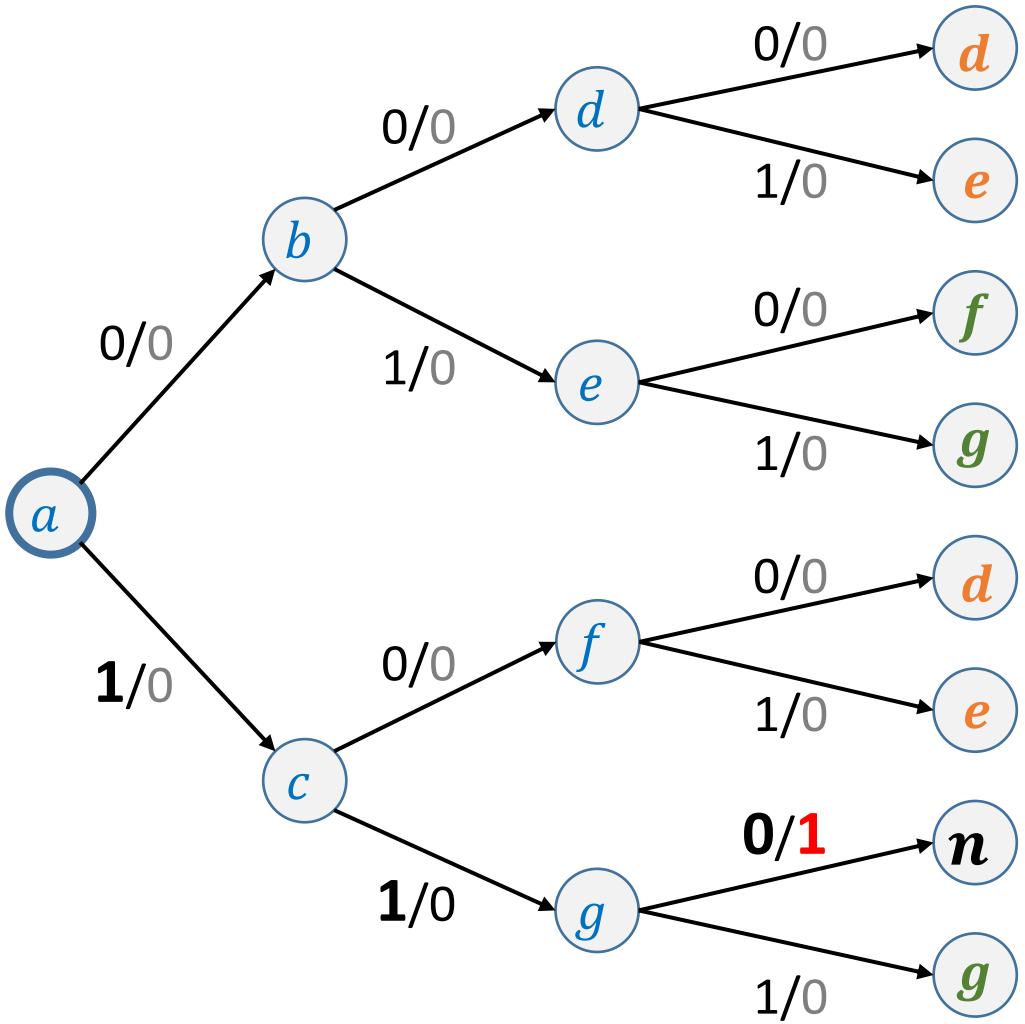
S	S^* / Z	
	$x=0$	$x=1$
a	a / 0	b / 0
b	a / 0	c / 0
c	a / 1	c / 0

状态化简

- **状态化简**：在时序电路中，减少**触发器**数目的过程。
- **化简原则**：电路的输入、输出关系保持不变。
- **副 作 用**：可能会使电路中增加更多的组合逻辑。
- 当对状态进行**化简**时，使用**状态表**比**状态图**更方便。
- 当两个**状态等价**时，其中的一个状态可以被另一个代替，此时输入-输出关系不变。
- **两个状态等价的条件**：对所有可能的输入、输出都相同，且电路的次态**相同**或**等效**。

【例1】设计检测是否输入“110” (3)

二叉树搜索法



S	S^* / Z	
	$x=0$	$x=1$
a	b / 0	c / 0
b	d / 0	e / 0
c	f / 0	g / 0
d	d / 0	e / 0
e	f / 0	g / 0
f	d / 0	e / 0
g	n / 1	g / 0

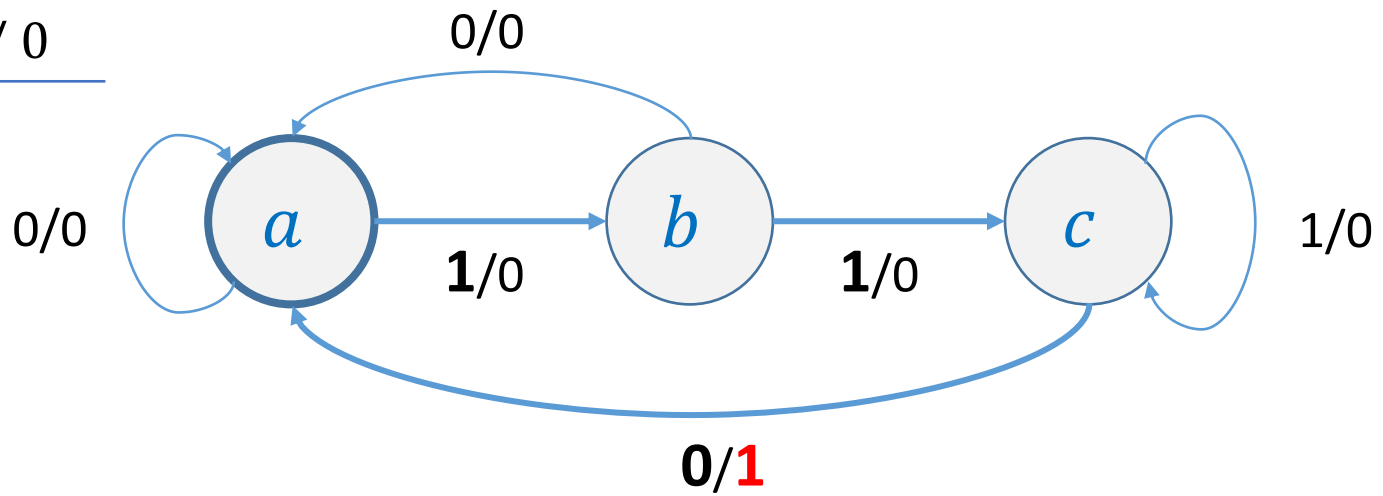
【例1】设计检测是否输入“110” (3)

S	S^* / Z	
	$x=0$	$x=1$
a	b / 0	c / 0
b	d / 0	e / 0
c	f / 0	g / 0
d	d / 0	e / 0
e	f / 0	g / 0
f	d / 0	e / 0
g	n / 1	g / 0

S	S^* / Z	
	$x=0$	$x=1$
a	b / 0	c / 0
b	**b** / 0	e / 0
c	**b** / 0	g / 0
e	**b** / 0	g / 0
g	n / 1	g / 0

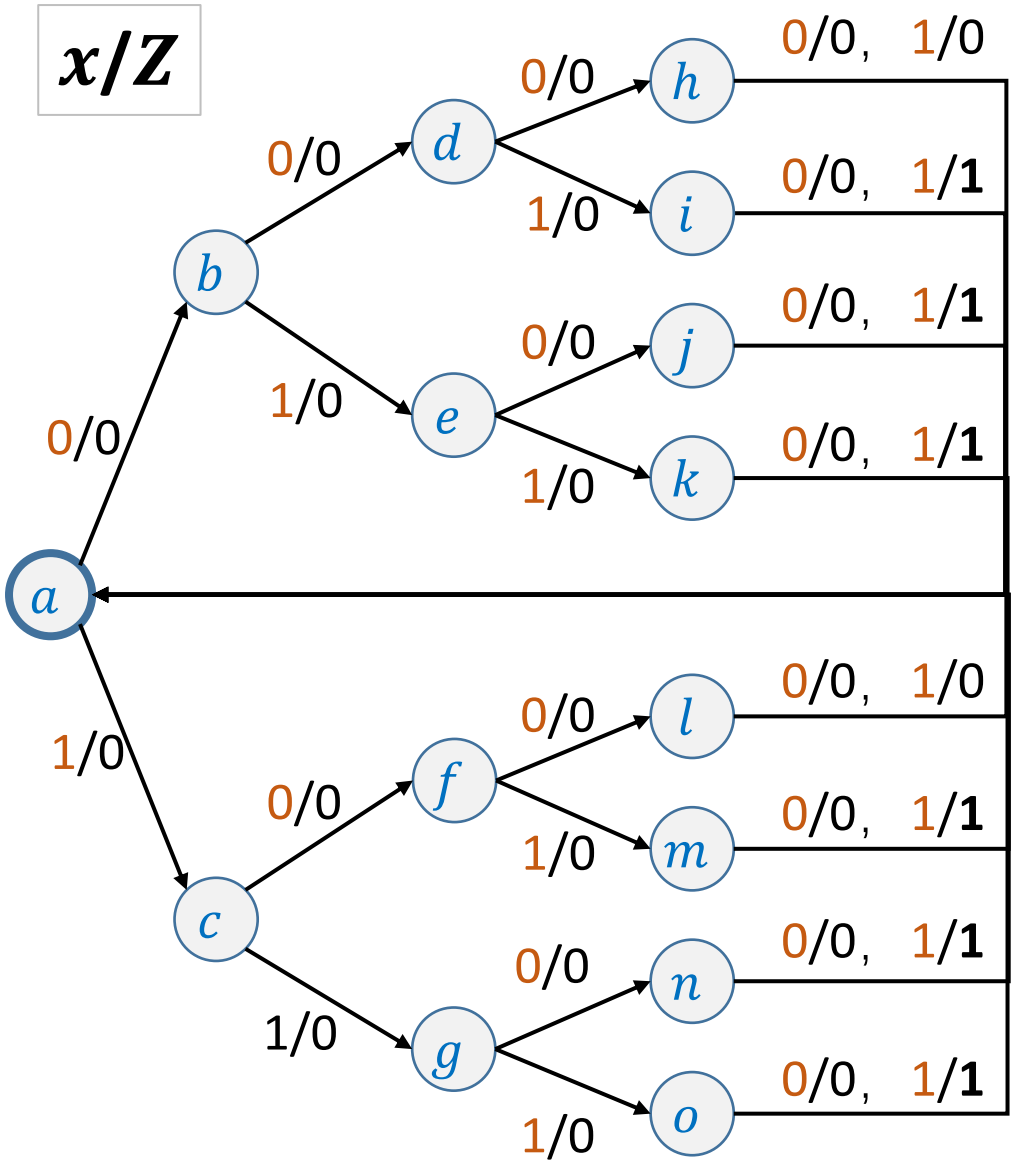
S	S^* / Z	
	$x=0$	$x=1$
a	b / 0	c / 0
b	b / 0	**c** / 0
c	b / 0	g / 0
g	n / 1	g / 0

S	S^* / Z	
	$x=0$	$x=1$
a	**a** / 0	c / 0
c	**a** / 0	g / 0
g	**a / 1**	g / 0



S	S^* / Z	
	$x=0$	$x=1$
a	**a** / 0	b / 0
b	**a** / 0	c / 0
c	**a / 1**	c / 0

【练习】检测串行输入8421BCD码的状态图



S	S^*/Z	
	$x=0$	$x=1$
a	$b/0$	$c/0$
b	$d/0$	$e/0$
c	$f/0$	$g/0$
d	$h/0$	$i/0$
e	$j/0$	$k/0$
f	$l/0$	$m/0$
g	$n/0$	$o/0$
h	$a/0$	$a/0$
i	$a/0$	$a/1$
j	$a/0$	$a/1$
k	$a/0$	$a/1$
l	$a/0$	$a/0$
m	$a/0$	$a/1$
n	$a/0$	$a/1$
o	$a/0$	$a/1$

S	S^*/Z	
	$x=0$	$x=1$
a	$b/0$	$c/0$
b	$d/0$	$e/0$
c	$f/0$	$g/0$
d	$h/0$	$i/0$
e	$\boldsymbol{i}/0$	$\boldsymbol{i}/0$
f	$\boldsymbol{h}/0$	$\boldsymbol{i}/0$
g	$\boldsymbol{i}/0$	$\boldsymbol{i}/0$
h	$a/0$	$a/0$
i	$a/0$	$a/1$

S	S^*/Z	
	$x=0$	$x=1$
a	$b/0$	$c/0$
b	$d/0$	$e/0$
c	$\boldsymbol{d}/0$	$e/0$
d	$h/0$	$i/0$
e	$i/0$	$i/0$
h	$a/0$	$a/0$
i	$a/0$	$a/1$

非法数字：1010、1011、1100

输出为 1。1101、1110、1111

输入顺序：**先低后高**

等效状态

等效状态：对于**所有可能的**输入序列，分别从状态a和状态b出发，所得到的输出响应序列完全相同。

等效状态具有传递性：若a与b等效，b与c等效，则a与c等效。

判断方法：

- 输出相同、次态**相同**
- 输出相同、次态**交错**
- 输出相同、次态**循环**

现态	次态 / 输出	
	$x=0$	$x=1$
a	$\boldsymbol{b}/0$	$c/1$
b	$\boldsymbol{b}/0$	$d/1$
c	$d/0$	$a/0$
d	$c/0$	$b/0$

等效类

等效类

- 由若干个彼此等效的状态构成的集合。
- 一个状态也可以是等效类。
- 一个等效类中任意两个状态都是等效的。

最大等效类：不被任何别的等效类所包含的等效类。

最大：不是状态最多，而是独立性。

即使一个状态，只要它不被包含在别的等效类中，

也是最大等效类。

2018年俄罗斯世界杯C小组循环赛

	法国	澳大利亚	丹麦	秘鲁
法国				
澳大利亚	**1-2**			
丹麦	**0-0**	**1-1**		
秘鲁	**0-1**	**2-0**	**0-1**	

对原始状态表进行化简

次态交错

S	S^* / Z	
	$x=0$	$x=1$
a	c / 0	b / 1
b	f / 0	a / 1
c	f / 0	g / 0
d	d / 1	e / 0
e	c / 0	e / 1
f	c / 0	g / 0
g	c / 1	d / 0

S	S^* / Z	
	$x=0$	$x=1$
a	c / 0	$\boldsymbol{a}$ / 1
c	$\boldsymbol{c}$ / 0	g / 0
d	d / 1	$\boldsymbol{a}$ / 0
g	c / 1	d / 0

隐含表

	a	b	c	d	e	f
b	cf					
c	×	×				
d	×	×	×			
e	be	cf ae	×	×		
f	×	×	√	×	×	
g	×	×	×	cd de	×	×

$ae \longrightarrow be \longrightarrow cf$

次态循环

	a	b	c	d	e	f
b	√					
c	×	×				
d	×	×	×			
e	√	√	×	×		
f	×	×	√	×	×	
g	×	×	×	×	×	×

最大等效对:

$\{\boldsymbol{c}, f\}$ $\{\boldsymbol{a}, b, e\}$ $\{\boldsymbol{d}\}$ $\{\boldsymbol{g}\}$

【例1】设计检测是否输入“110” (4)

S	S^* / Z	
	$x=0$	$x=1$
a	a / 0	b / 0
b	a / 0	c / 0
c	a / 1	c / 0

Q_1Q_0	$Q_1^*Q_0^*$ / Z	
	$x=0$	$x=1$
0 0	00 / 0	10 / 0
1 0	00 / 0	11 / 0
1 1	00 / 1	11 / 0

如果电路有m个状态，就需要n位二进制，$2^n \geq m$

状态编码：要利于函数的化简

原则

① **次态相同，现态相邻**

② 同一现态，次态相邻

③ 输出相同，现态相邻

$$\text{编码方案数} = \frac{2^n!}{(2^n-m)!}$$

m：状态数

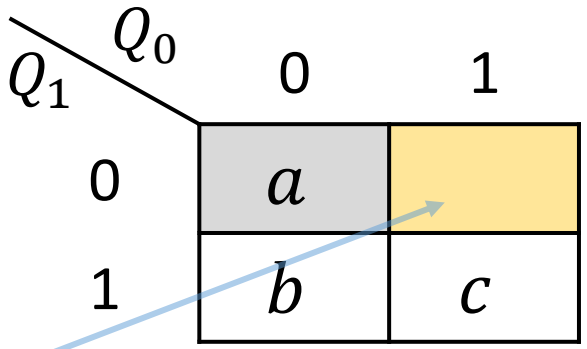
n：二进制位数

当$m=3, n=2$，有24种方案

由**原则①**：a和b，b和c相邻

由原则②：a和b，a和c相邻

由原则③：a和b相邻



不完全确定同步时序逻辑电路

【例1】设计检测是否输入“110”（5）

Q_1Q_0	$Q_1^*Q_0^*$ / Z	
	$x=0$	$x=1$
0 0	00 / 0	10 / 0
1 0	00 / 0	11 / 0
1 1	00 / 1	11 / 0
0 1	dd / d	dd / d

没有用的码字可认为是**任意态**，有助于简化电路。

x	Q_1	Q_0	Q_1^*	Q_0^*	Z
0	**0**	**0**	0	0	0
0	**1**	**0**	0	0	0
0	**1**	**1**	0	0	**1**
1	**0**	**0**	1	0	0
1	**1**	**0**	1	1	0
1	**1**	**1**	1	1	0
0	**0**	**1**	d	d	d
1	**0**	**1**	d	d	d

选D触发器

$$Q_1^* = D_1 = x$$

$$Q_0^* = D_0 = xQ_1$$

$$Z = \bar{x}Q_0$$

Q_1^*

x \ Q_1Q_0	00	01	11	10
0		d		
1	1	d	1	1

Q_0^*

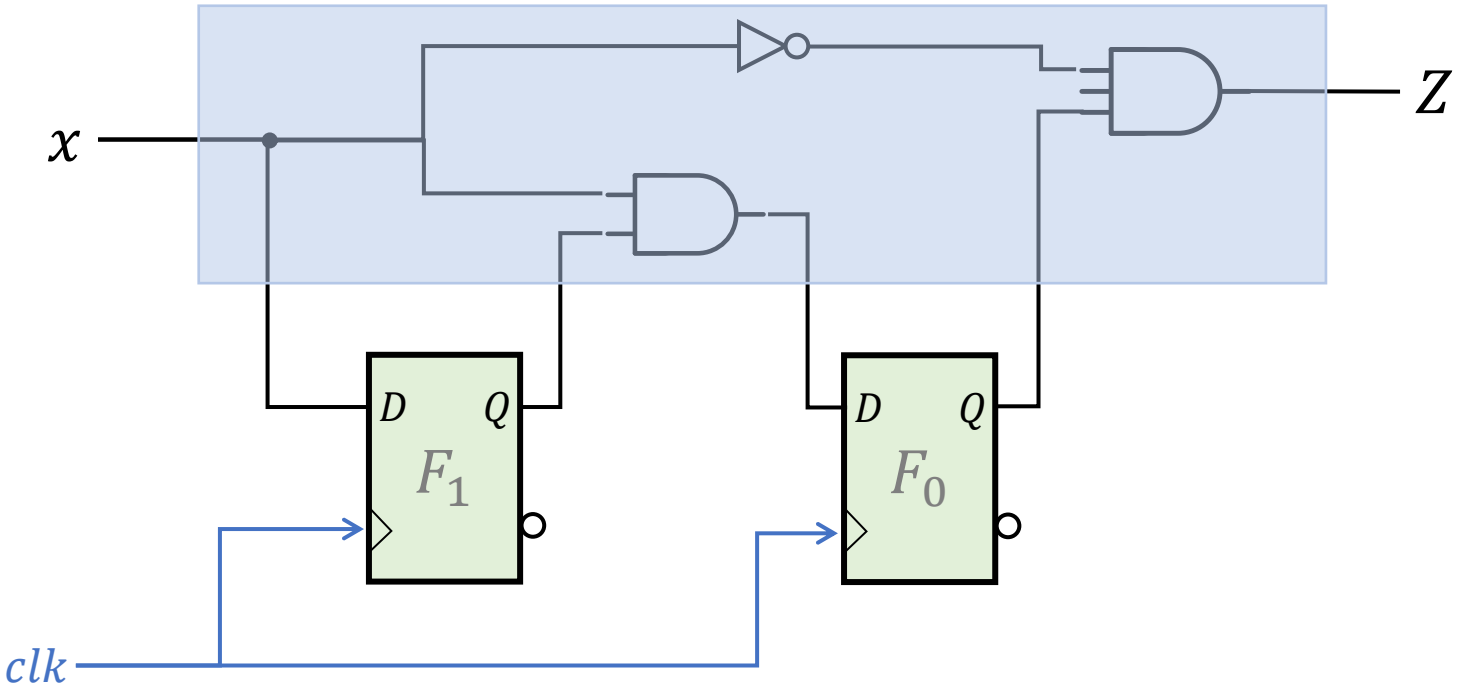
x \ Q_1Q_0	00	01	11	10
0		d		
1		d	1	1

Z

x \ Q_1Q_0	00	01	11	10
0		d	1	
1		d		

【例1】设计检测是否输入“110” (6)

x	Q_1	Q_0	Q_1^*	Q_0^*	Z
0	**0**	**0**	0	0	0
0	**1**	**0**	0	0	0
0	**1**	**1**	0	0	**1**
1	**0**	**0**	1	0	0
1	**1**	**0**	1	1	0
1	**1**	**1**	1	1	0
0	**0**	**1**	d	d	d
1	**0**	**1**	d	d	d

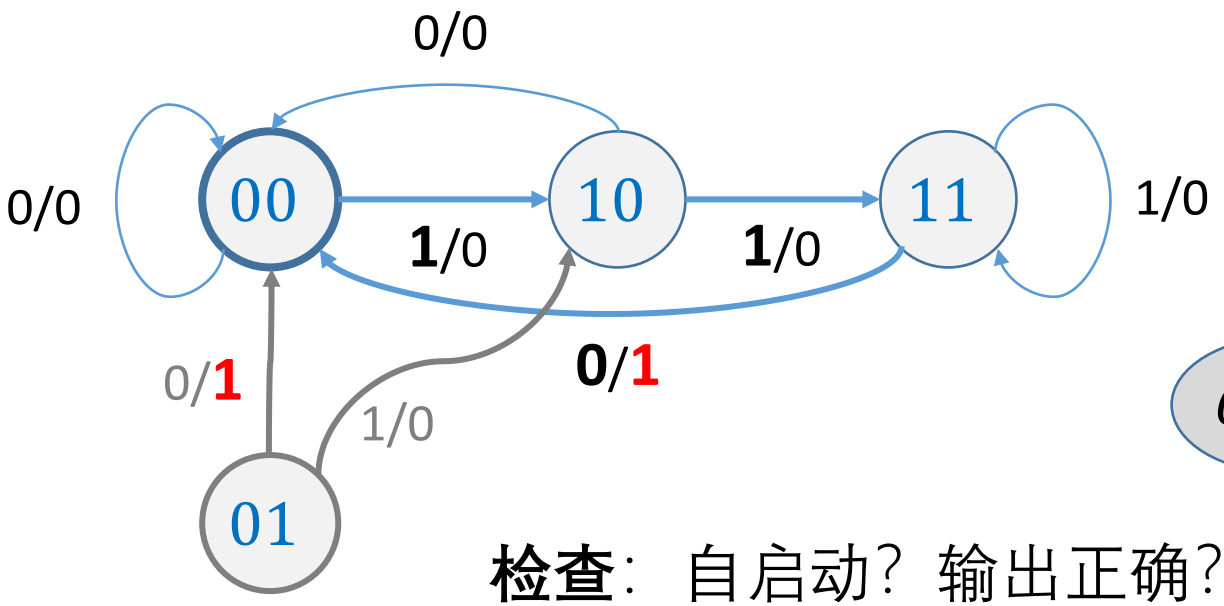


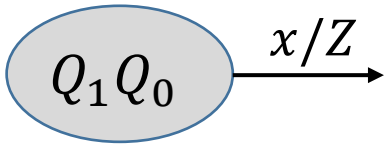
$$Q_1^* = D_1 = x$$

$$Q_0^* = D_0 = xQ_1$$

$$Z = \bar{x}Q_0$$

【例1】设计检测是否输入“110” (7)





检查：自启动？输出正确？

$$Z = \bar{x}Q_0 \quad \Longrightarrow \quad Z = \bar{x}Q_0Q_1$$

Z

x \ Q_1Q_0	00	01	11	10
0		d	1	
1		d		

x	Q_1	Q_0	Q_1^*	Q_0^*	Z
0	**0**	**0**	0	0	0
0	**1**	**0**	0	0	0
0	**1**	**1**	0	0	**1**
1	**0**	**0**	1	0	0
1	**1**	**0**	1	1	0
1	**1**	**1**	1	1	0
0	**0**	**1**	d **0**	d **0**	d **1**
1	**0**	**1**	d **1**	d **0**	d 0

【例1】设计检测是否输入“110” (8)

x	Q_1	Q_0	Q_1^*	Q_0^*	Z
0	0	0	0	0	0
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	1	1	1	1	0
0	0	1	0	0	0
1	0	1	1	0	0

选JK触发器

$Q^* = J\bar{Q} + \bar{K}Q$

公式法

$$Q_1^* = x$$
$$= x(\bar{Q}_1 + Q_1)$$
$$= x\bar{Q}_1 + xQ_1$$

$J_1 = x, \quad K_1 = \bar{x}$

$$Q_0^* = xQ_1$$
$$= xQ_1\bar{Q}_0 + xQ_1Q_0$$

$J_0 = xQ_1, \quad K_0 = \overline{xQ_1}$

卡诺图法

Q_1^*

x \ Q_1Q_0	00	01	11	10
0				
1	1	1	1	1

$Q_1^* = x\bar{Q}_1 + xQ_1$

Q_0^*

x \ Q_1Q_0	00	01	11	10
0				
1			1	1

$Q_0^* = xQ_1\bar{Q}_0 + xQ_1Q_0$

【例1】设计检测是否输入“110” (9)

x	Q_1	Q_0	Q_1^*	Q_0^*	Z	J_1	K_1	J_0	K_0
0	**0**	**0**	0	0	0	0	d	0	d
0	**1**	**0**	0	0	0	d	1	0	d
0	**1**	**1**	0	0	**1**	d	1	d	1
1	**0**	**0**	1	0	0	1	d	0	d
1	**1**	**0**	1	1	0	d	0	1	d
1	**1**	**1**	1	1	0	d	0	d	0
0	**0**	**1**	0	0	0	0	d	d	1
1	**0**	**1**	1	0	0	1	d	d	1

激励表

Q	Q^*	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

Q	Q^*	J	K
0	0	0	0
		0	1
0	1	1	0
		1	1
1	0	0	1
		1	1
1	1	0	0
		1	0

状态表法

J_1

x \ Q_1Q_0	00	01	11	10
0			d	d
1	1	1	d	d

$J_1 = x$

K_1

x \ Q_1Q_0	00	01	11	10
0	d	d	1	1
1	d	d		

$K_1 = \bar{x}$

J_0

x \ Q_1Q_0	00	01	11	10
0		d	d	
1		d	d	1

$J_0 = xQ_1$

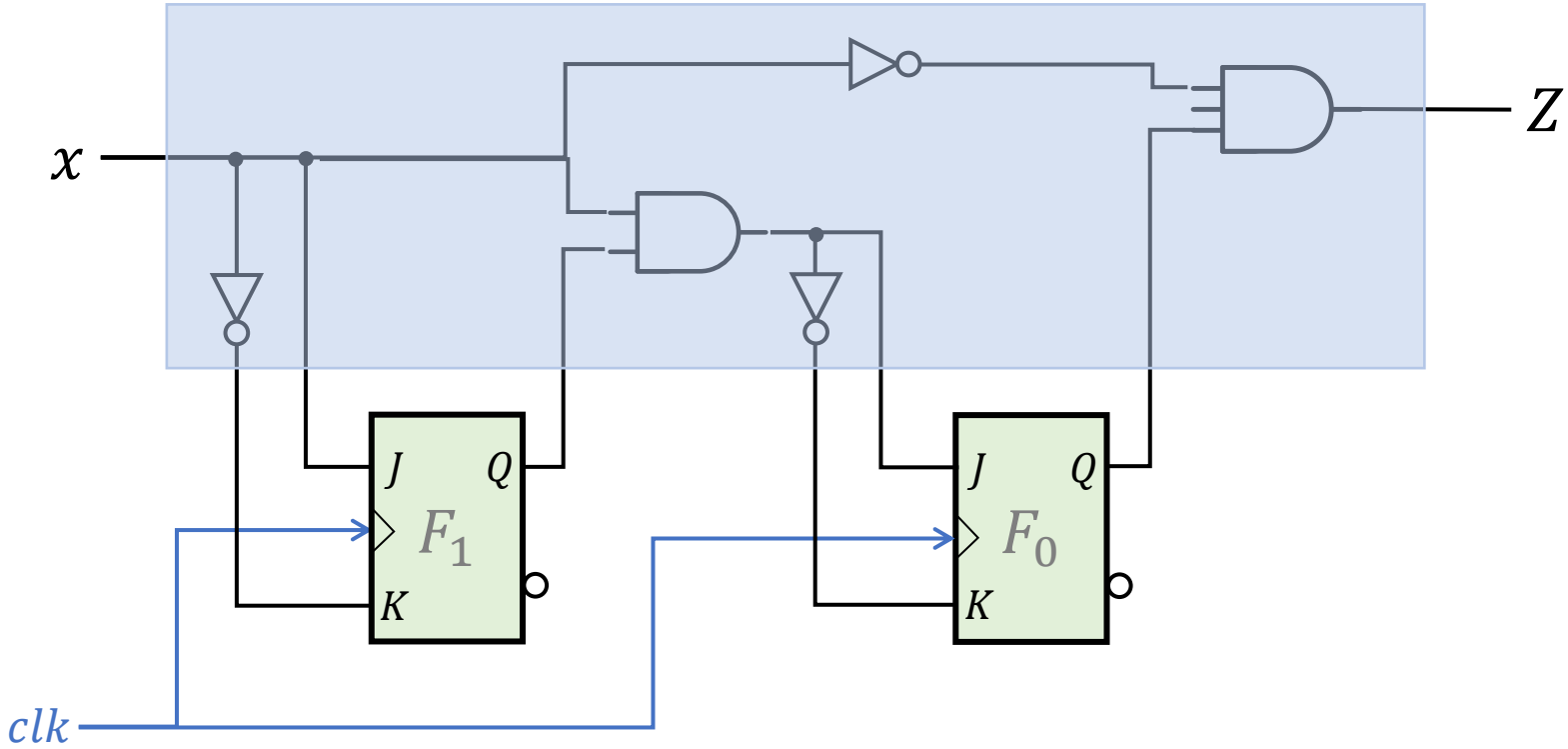
K_0

x \ Q_1Q_0	00	01	11	10
0	d	1	1	d
1	d	1		d

$K_0 = \overline{xQ_1}$

【例1】设计检测是否输入“110” (10)

$J_1 = x,\quad K_1 = \bar{x}$ $\qquad J_0 = x\,Q_1,\quad K_0 = \overline{xQ_1}$ $\qquad \boldsymbol{Z} = \bar{x}Q_0$



【例1】设计检测是否输入“110”（11）

x	Q_1	Q_0	Q_1^*	Q_0^*	Z	T_1	T_0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	1	1	0	0	1	1	1
1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	0	0
0	0	1	0	0	0	0	1
1	0	1	1	0	0	1	1

选T触发器

激励表

Q	Q^*	T
0	0	0
0	1	1
1	0	1
1	1	0

Q_1^*

Q_1Q_0 / x	00	01	11	10
0				
1	1	1	1	1

$$Q^* = T\bar{Q} + \bar{T}Q$$

$$Q_1^* = x\bar{Q}_1 + xQ_1$$

$$Q_0^* = xQ_1$$

T_1

Q_1Q_0 / x	00	01	11	10
0			1	1
1	1	1		

T_0

Q_1Q_0 / x	00	01	11	10
0		1	1	
1		1		1

$$Z = \bar{x}Q_0$$

$$T_1 = x\bar{Q}_1 + \bar{x}Q_1$$

$$T_0 = \bar{Q}_1Q_0 + \bar{x}Q_0 + xQ_1\bar{Q}_0$$

同步时序逻辑电路设计步骤

设计：画出实现给定逻辑功能的**电路图**。

① 分析**电路类型**(组合？时序？同步？异步？)

② 确定**输入变量**、**输出变量**

③ 根据设计要求建立**原始状态表**

④ 如有必要**化简**原始状态表

⑤ **状态编码** 每个状态用一个符号代表，进而用一个二进制来表示。**状态图、状态(转换)表**

⑥ 选择触发器，建立**激励方程**、**输出方程**

一旦触发器类型和数目确定后，就从**时序电路**问题转化成了**组合电路**问题。

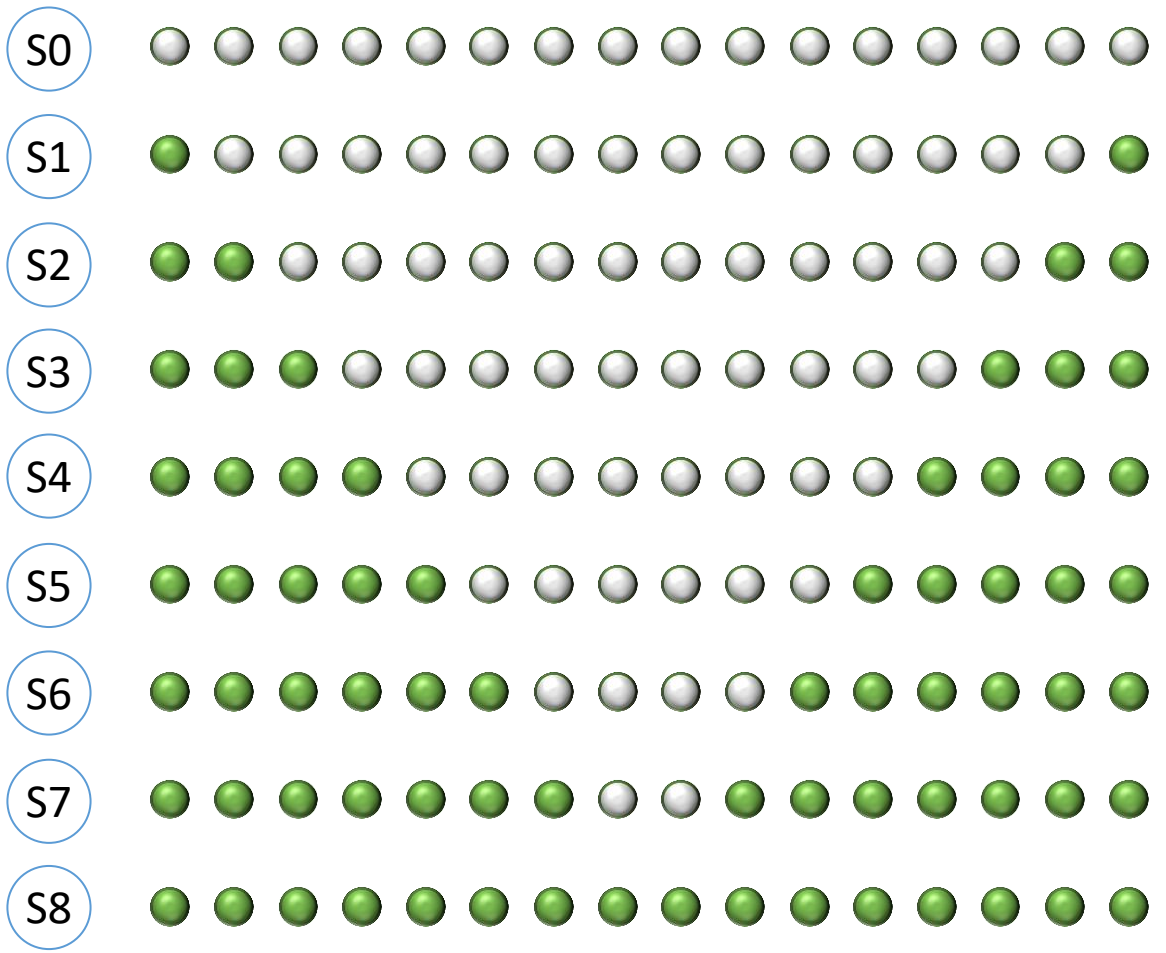
⑦ 画出**电路图**，检测**自启动**、**输出逻辑**是否正确？

2

流水灯

【例2】流水灯电路设计

	Q_3	Q_2	Q_1	Q_0	Q_3^*	Q_2^*	Q_1^*	Q_0^*	L_{15}	L_{14}	…	L_1	L_0
0	0	0	0	0	0	0	0	1	1	0	…	0	1
1	0	0	0	1	0	0	1	0	1	1	…	1	1
2	0	0	1	0	0	0	1	1	1	1	…	1	1
3	0	0	1	1	0	1	0	0	1	1	…	1	1
4	0	1	0	0	0	1	0	1	1	1	…	1	1
5	0	1	0	1	0	1	1	0	1	1	…	1	1
6	0	1	1	0	0	1	1	1	1	1	…	1	1
7	0	1	1	1	1	0	0	0	1	1	…	1	1
8	1	0	0	0	0	0	0	0	0	0	…	0	0
9	1	0	0	1	d	d	d	d	d	d	…	d	d
10	1	0	1	0	d	d	d	d	d	d	…	d	d
11	1	0	1	1	d	d	d	d	d	d	…	d	d
12	1	1	0	0	d	d	d	d	d	d	…	d	d
13	1	1	0	1	d	d	d	d	d	d	…	d	d
14	1	1	1	0	d	d	d	d	d	d	…	d	d
15	1	1	1	1	d	d	d	d	d	d	…	d	d



【例2】流水灯电路设计-2

	Q_3	Q_2	Q_1	Q_0	Q_3^*	Q_2^*	Q_1^*	Q_0^*	L_{15}	L_{14}	…	L_1	L_0
0	0	0	0	0	0	0	0	1	1	0	…	0	1
1	0	0	0	1	0	0	1	0	1	1	…	1	1
2	0	0	1	0	0	0	1	1	1	1	…	1	1
3	0	0	1	1	0	1	0	0	1	1	…	1	1
4	0	1	0	0	0	1	0	1	1	1	…	1	1
5	0	1	0	1	0	1	1	0	1	1	…	1	1
6	0	1	1	0	0	1	1	1	1	1	…	1	1
7	0	1	1	1	1	0	0	0	1	1	…	1	1
8	1	0	0	0	0	0	0	0	0	0	…	0	0
9	1	0	0	1	d	d	d	d	d	d	…	d	d
10	1	0	1	0	d	d	d	d	d	d	…	d	d
11	1	0	1	1	d	d	d	d	d	d	…	d	d
12	1	1	0	0	d	d	d	d	d	d	…	d	d
13	1	1	0	1	d	d	d	d	d	d	…	d	d
14	1	1	1	0	d	d	d	d	d	d	…	d	d
15	1	1	1	1	d	d	d	d	d	d	…	d	d

$Q_3^*Q_2^*Q_1^*Q_0^*$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0001	0010	0100	0011
01	0101	0110	1000	0111
11	$dddd$	$dddd$	$dddd$	$dddd$
10	0000	$dddd$	$dddd$	$dddd$

Q_3^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	d	d	d	d
10	0	d	d	d

$$Q_3^* = Q_2Q_1Q_0$$

$$D_3 = Q_2Q_1Q_0$$

【例2】流水灯电路设计-3

Q_3^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	d	d	d	d
10	0	d	d	d

$$Q_3^* = Q_2Q_1Q_0 = D_3$$

Q_2^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	d	d	d	d
10	0	d	d	d

$$Q_2^* = Q_2\overline{Q}_1 + Q_2\overline{Q}_0 + \overline{Q}_2Q_1Q_0 = D_2$$

Q_1^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	d	d	d	d
10	0	d	d	d

$$Q_1^* = \overline{Q}_1Q_0 + Q_1\overline{Q}_0 = D_1$$

Q_0^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	d	d	d	d
10	0	d	d	d

$$Q_0^* = \overline{Q}_3\overline{Q}_0 = D_0$$

【例2】流水灯电路设计-4

Q_3^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	0	1	0
10	0	d	0	0

$Q_3^*Q_2^*Q_1^*Q_0^*$

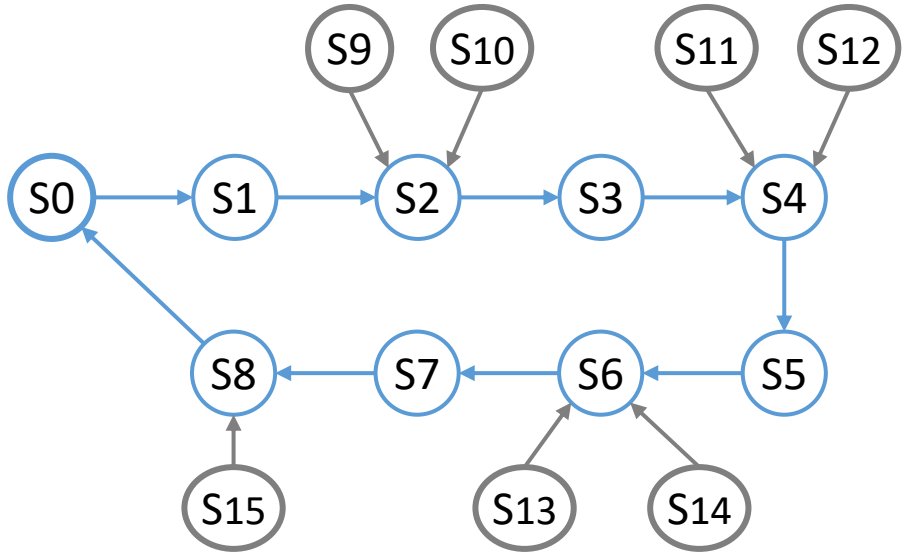
Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0001	0010	0100	0011
01	0101	0110	1000	0111
11	0100	0110	1000	0110
10	0000	0010	0100	0010

Q_2^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

Q_1^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1



Q_0^*

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	0	0	0	0

【例2】流水灯电路设计-5

	Q_3	Q_2	Q_1	Q_0	Q_3^*	Q_2^*	Q_1^*	Q_0^*	L_{15}	L_{14}	…	L_1	L_0
0	0	0	0	0	0	0	0	1	**1**	0	…	0	1
1	0	0	0	1	0	0	1	0	**1**	1	…	1	1
2	0	0	1	0	0	0	1	1	**1**	1	…	1	1
3	0	0	1	1	0	1	0	0	**1**	1	…	1	1
4	0	1	0	0	0	1	0	1	**1**	1	…	1	1
5	0	1	0	1	0	1	1	0	**1**	1	…	1	1
6	0	1	1	0	0	1	1	1	**1**	1	…	1	1
7	0	1	1	1	1	0	0	0	**1**	1	…	1	1
8	1	0	0	0	0	0	0	0	**0**	0	…	0	0
9	1	0	0	1	d	d	d	d	$\boldsymbol{d}$	d	…	d	d
10	1	0	1	0	d	d	d	d	$\boldsymbol{d}$	d	…	d	d
11	1	0	1	1	d	d	d	d	$\boldsymbol{d}$	d	…	d	d
12	1	1	0	0	d	d	d	d	$\boldsymbol{d}$	d	…	d	d
13	1	1	0	1	d	d	d	d	$\boldsymbol{d}$	d	…	d	d
14	1	1	1	0	d	d	d	d	$\boldsymbol{d}$	d	…	d	d
15	1	1	1	1	d	d	d	d	$\boldsymbol{d}$	d	…	d	d

L_{15}

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	d	d	d	d
10	0	d	d	d

$$\boldsymbol{L_{15}} = \bar{Q}_3$$

$$\boldsymbol{L_{14}} = \ldots\ldots$$

…… …… ……

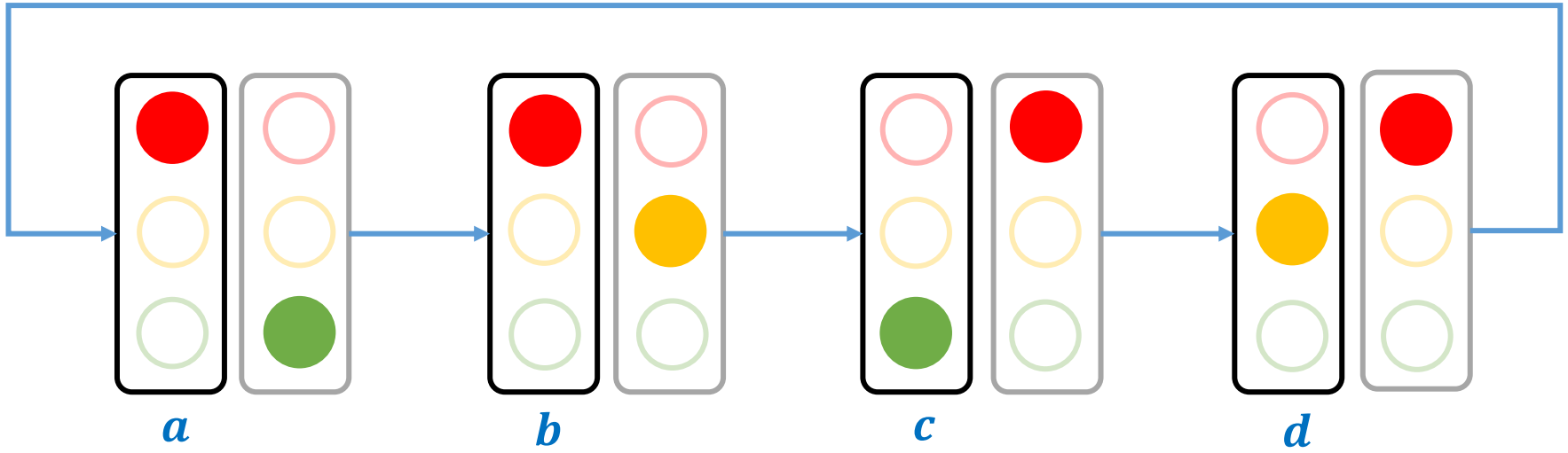
$$\boldsymbol{L_0} = \ldots\ldots$$

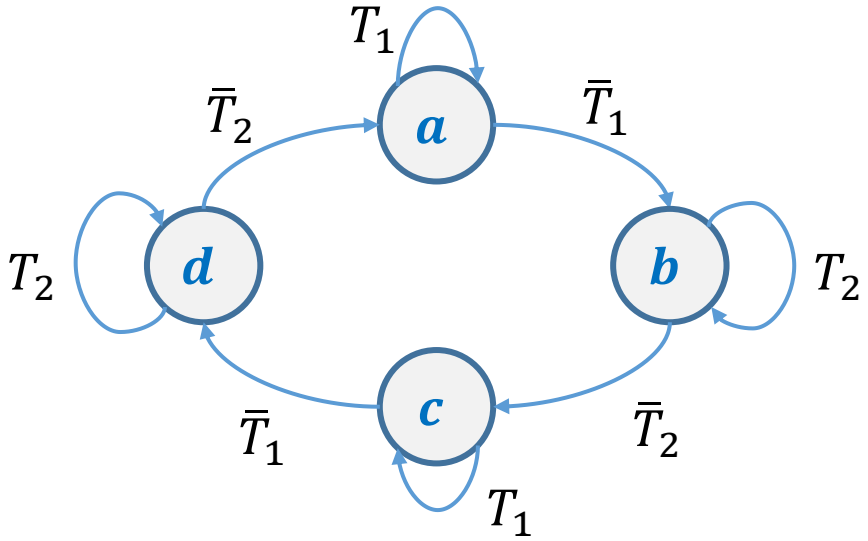
3

交通灯

十字路口交通灯，画出状态图

红灯显示 T_1 =25s，绿灯显示 T_1 =25s，黄灯显示 T_2 =4s



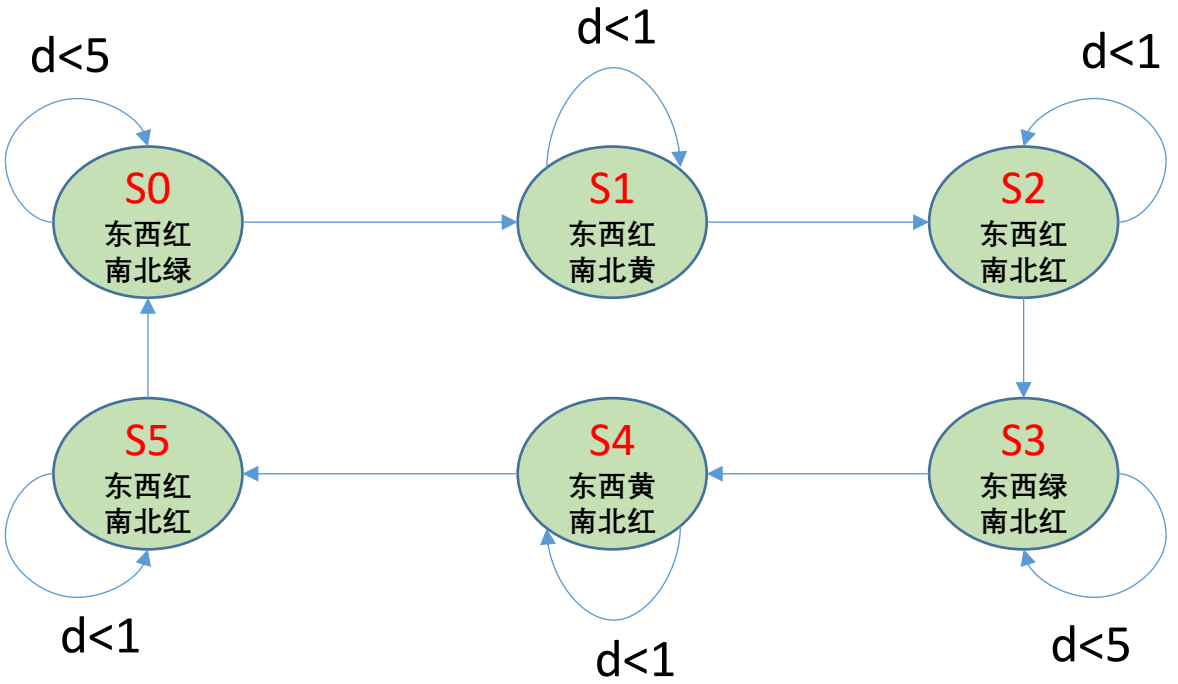


Example56_Traffic

交通红绿灯

状态	东西	南北	延迟d(秒)
S0	红	绿	5
S1	红	黄	1
S2	红	红	1
S3	绿	红	5
S4	黄	红	1
S5	红	红	1

Moore状态机




```
1  module Traffic_Top(
2      input  logic CLK100MHZ,
3      input  logic BTNC,
4      output logic [5:0]LED );
5
6      logic clk3Hz;
7
8      clkdiv  U1(.clk(CLK100MHZ), .clr(BTNC), .clk3(clk3Hz));
9      Traffic U2(.clk3(clk3Hz), .clr(BTNC), .lights(LED));
10 endmodule
```

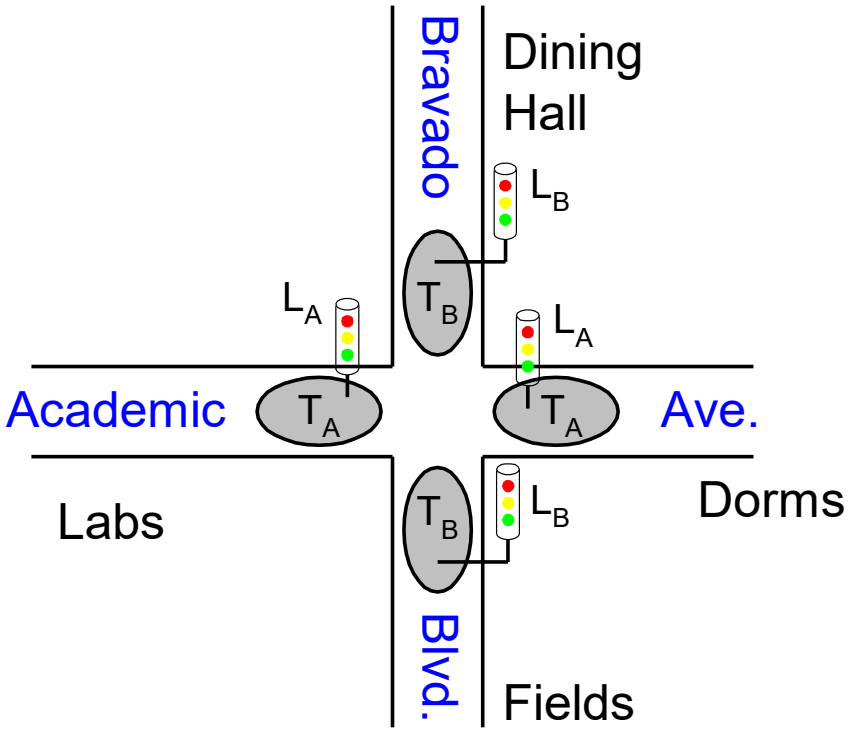
```
// 十字路口交通灯：
module Traffic(
    input  logic clk3, //3Hz时钟
    inout  logic clr,
    output logic [5:0] lights );
    // 定义变量
    logic [2:0] state;  //reg，状态
    logic [3:0] count;  //reg，延迟计数
    // 定义常量
    parameter  S0=3'b000, S1=3'b001, S2=3'b010,
               S3=3'b011, S4=3'b100, S5=3'b101;
    parameter  Sec1=4'b0010, //1秒用3个时钟得到
               Sec5=4'b1110; //5秒用15个时钟得到
```

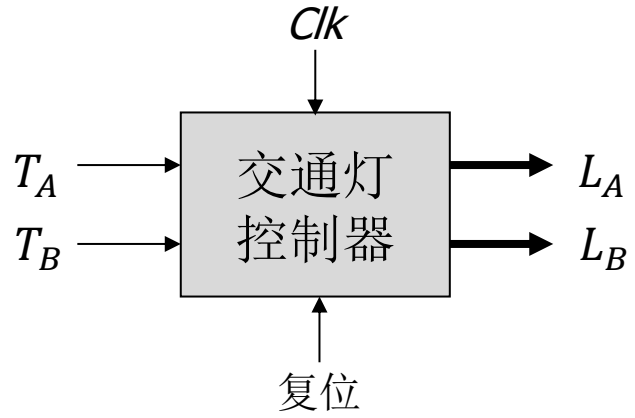
```
    always_ff @(posedge clk3, posedge clr)
    begin
        if(clr==1)                  begin  state<=S0; count<=0;          end
        else
            case(state)
                S0: if(count<Sec5)  begin  state<=S0; count<=count+1;    end
                    else            begin  state<=S1; count<=0;          end
                S1: if(count<Sec1)  begin  state<=S1; count<=count+1;    end
                    else            begin  state<=S2; count<=0;          end
                S2: if(count<Sec1)  begin  state<=S2; count<=count+1;    end
                    else            begin  state<=S3; count<=0;          end
                S3: if(count<Sec5)  begin  state<=S3; count<=count+1;    end
                    else            begin  state<=S4; count<=0;          end
                S4: if(count<Sec1)  begin  state<=S4; count<=count+1;    end
                    else            begin  state<=S5; count<=0;          end
                S5: if(count<Sec1)  begin  state<=S5; count<=count+1;    end
                    else            begin  state<=S0; count<=0;          end
                default             begin  state<=S0; count<=0;          end
            endcase
    end
```

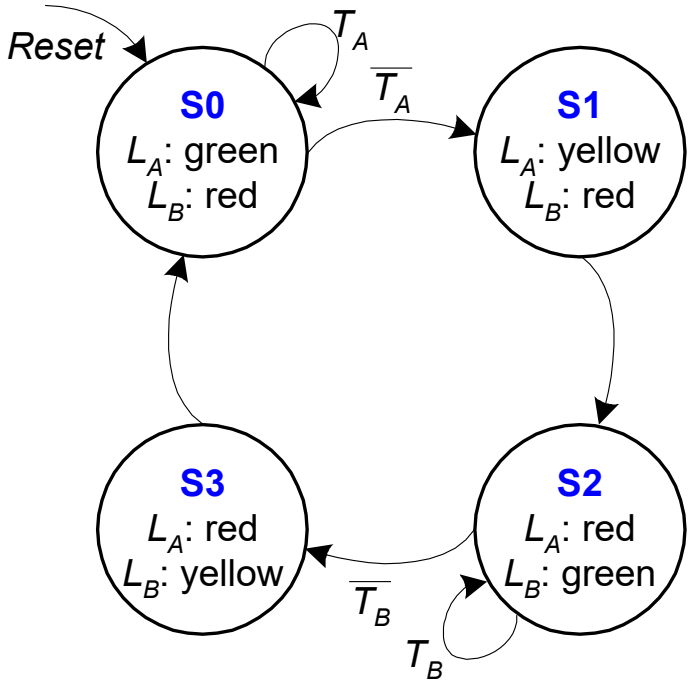
```
    // 输出逻辑
    always_comb     //控制不同状态下的红绿灯
    begin                       // 东-西 南-北
        case(state)             //红黄绿红黄绿
            S0:     lights = 6'b1_0_0_0_0_1;
            S1:     lights = 6'b1_0_0_0_1_0;
            S2:     lights = 6'b1_0_0_1_0_0;
            S3:     lights = 6'b0_0_1_1_0_0;
            S4:     lights = 6'b0_1_0_1_0_0;
            S5:     lights = 6'b1_0_0_1_0_0;
            default lights = 6'b1_0_0_0_0_1;
        endcase
    end
endmodule
```

状态	东西	南北	延迟(s)
S0	红	绿	5
S1	红	黄	1
S2	红	红	1
S3	绿	红	5
S4	黄	红	1
S5	红	红	1

【例3】交通灯控制器 (P70)

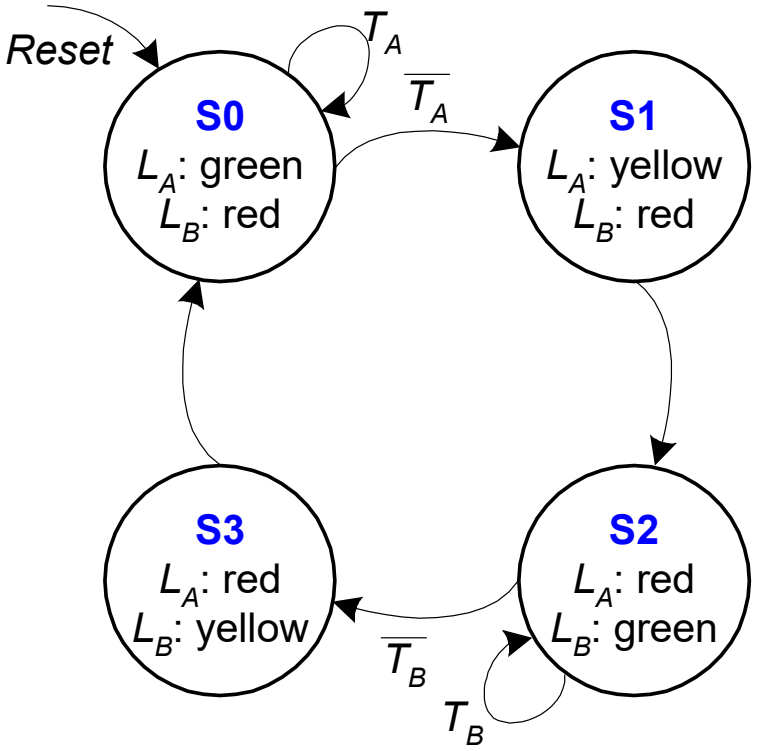






- T_A, T_B : Traffic sensors
  TRUE when 有学生出现
  FALSE when 没有学生
- L_A, L_B: Lights (5秒时钟)

【例3】交通灯控制器 (P70)-2



状态	编码
S0	00
S1	01
S2	10
S3	11

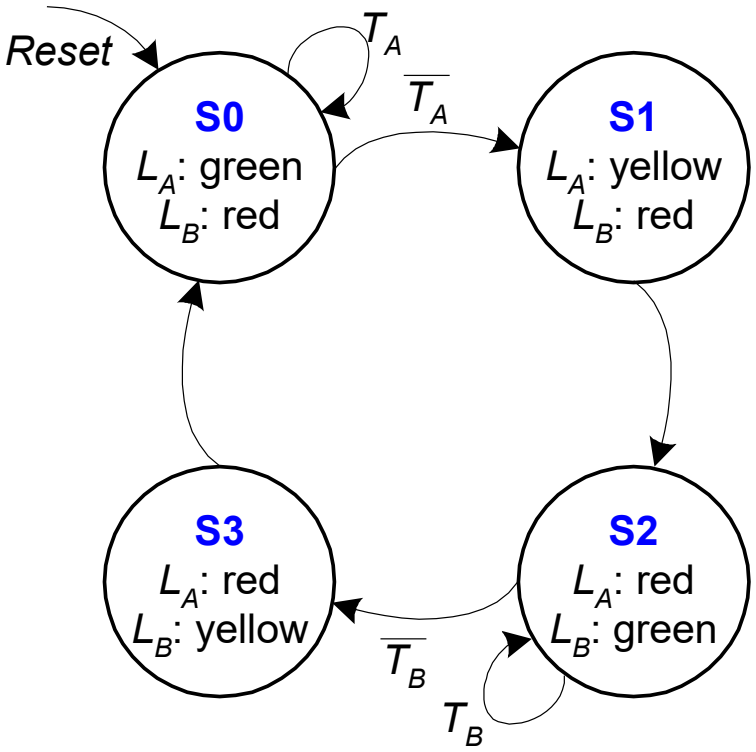
现状态 S	输 入 T_A	T_B	次状态 S^*
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

S_1	S_0	T_A	T_B	S_1^*	S_0^*
0	**0**	0	X	**0**	**1**
0	**0**	1	X	**0**	**0**
0	**1**	X	X	**1**	**0**
1	**0**	X	0	**1**	**1**
1	**0**	X	1	**1**	**0**
1	**1**	X	X	**0**	**0**

$$S_1^* = \bar{S}_1 S_0 + S_1 \bar{S}_0 \bar{T}_B + S_1 \bar{S}_0 T_B = S_0 \oplus S_1$$

$$S_0^* = \bar{S}_1 \bar{S}_0 \bar{T}_A + S_1 \bar{S}_0 \bar{T}_B$$

【例3】交通灯控制器 (P70)-3



输出	编 码
green	00
yellow	01
red	10

现态		输出			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	**0**	**0**	1	0
0	1	**0**	**1**	1	0
1	0	**1**	**0**	0	0
1	1	**1**	**0**	0	1

$$\boldsymbol{L_{A1}} = S_1$$

$$\boldsymbol{L_{A0}} = \bar{S}_1 S_0$$

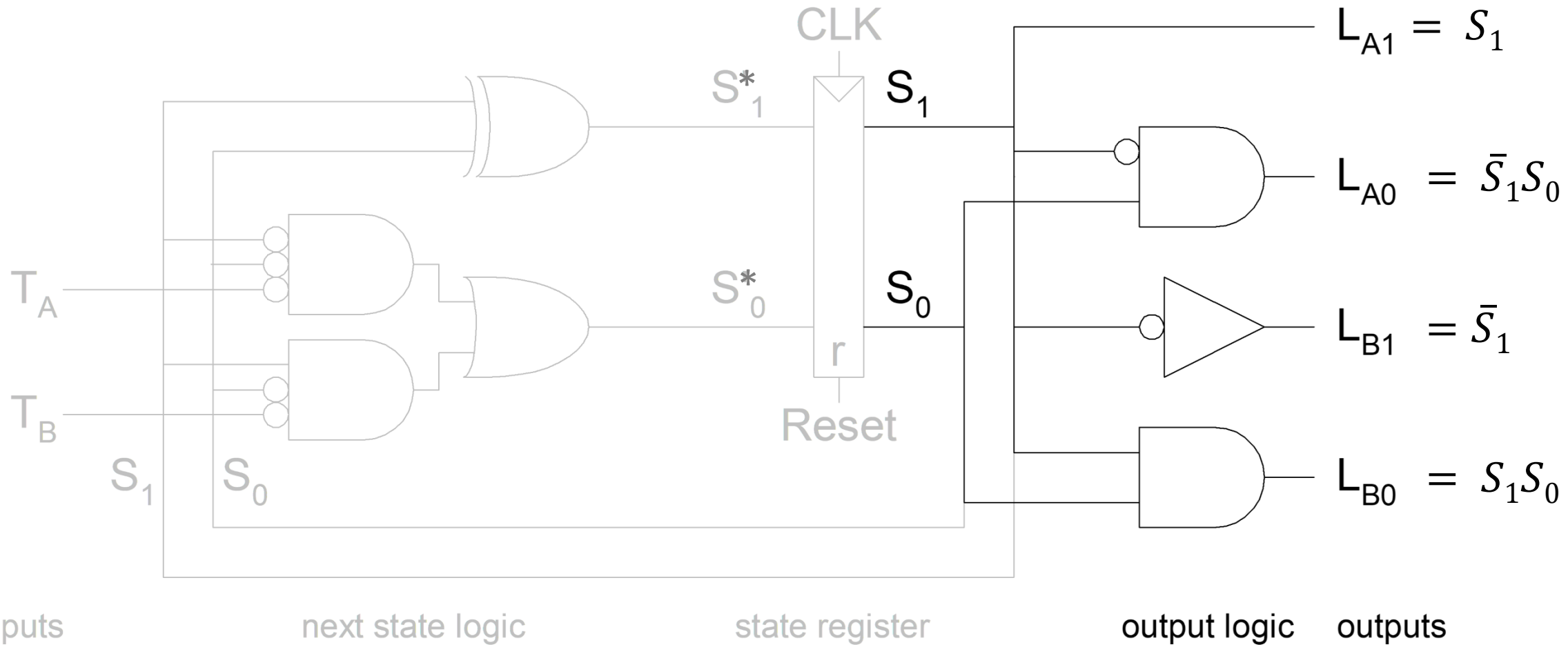
$$\boldsymbol{L_{B1}} = \bar{S}_1$$

$$\boldsymbol{L_{B0}} = S_1 S_0$$

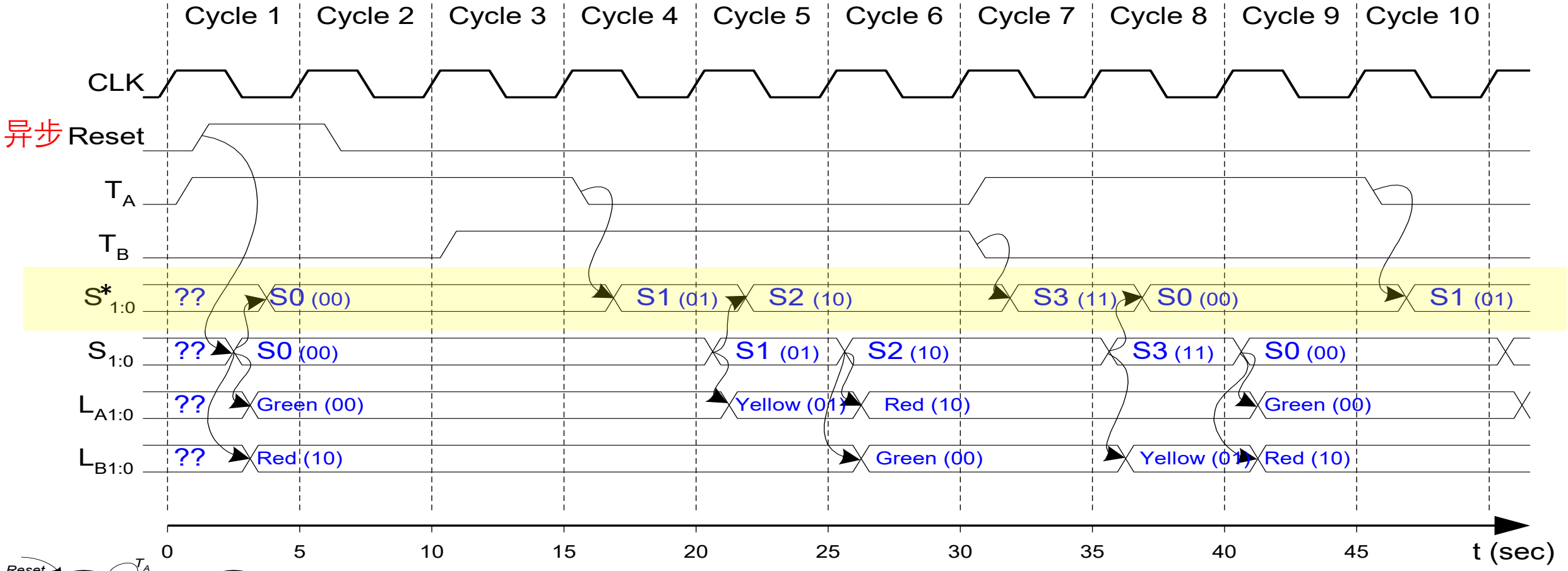
【例3】交通灯控制器 (P70)-4

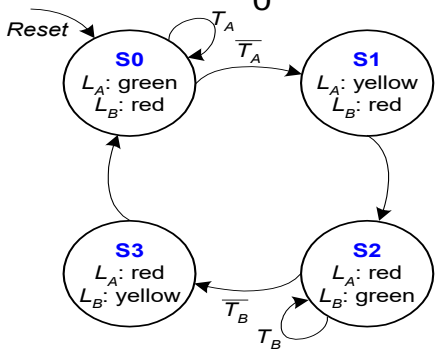
$$S_1^* = S_0 \oplus S_1$$

$$S_0^* = \bar{S}_1 \bar{S}_0 \bar{T}_A + S_1 \bar{S}_0 \bar{T}_B$$



【例3】交通灯控制器 (P70)-5

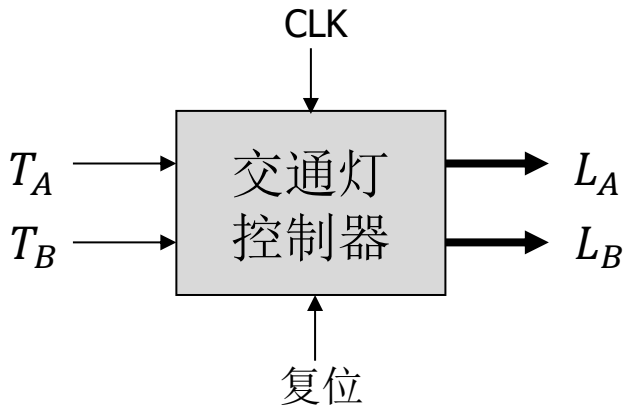


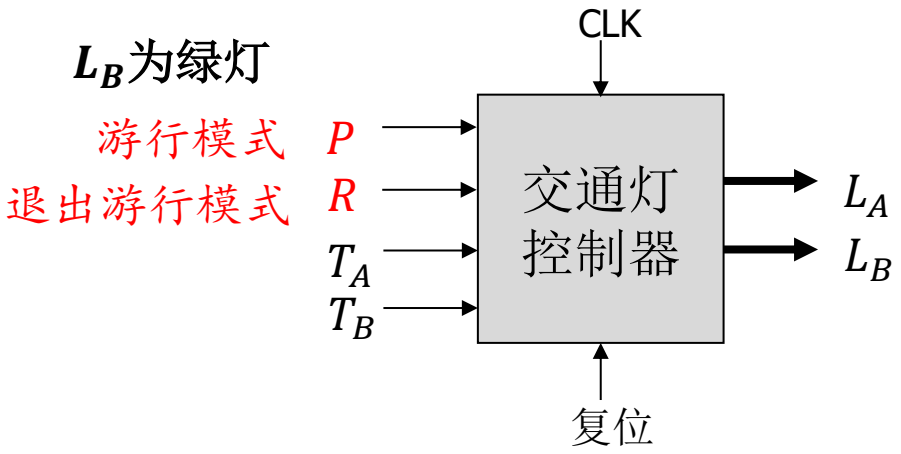


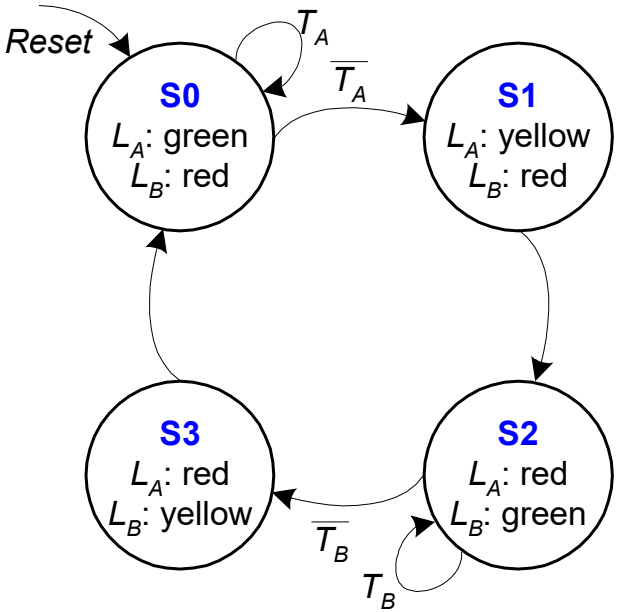
$$S_1^* = S_0 \oplus S_1 \qquad L_{A1} = S_1 \qquad L_{B1} = \bar{S}_1$$

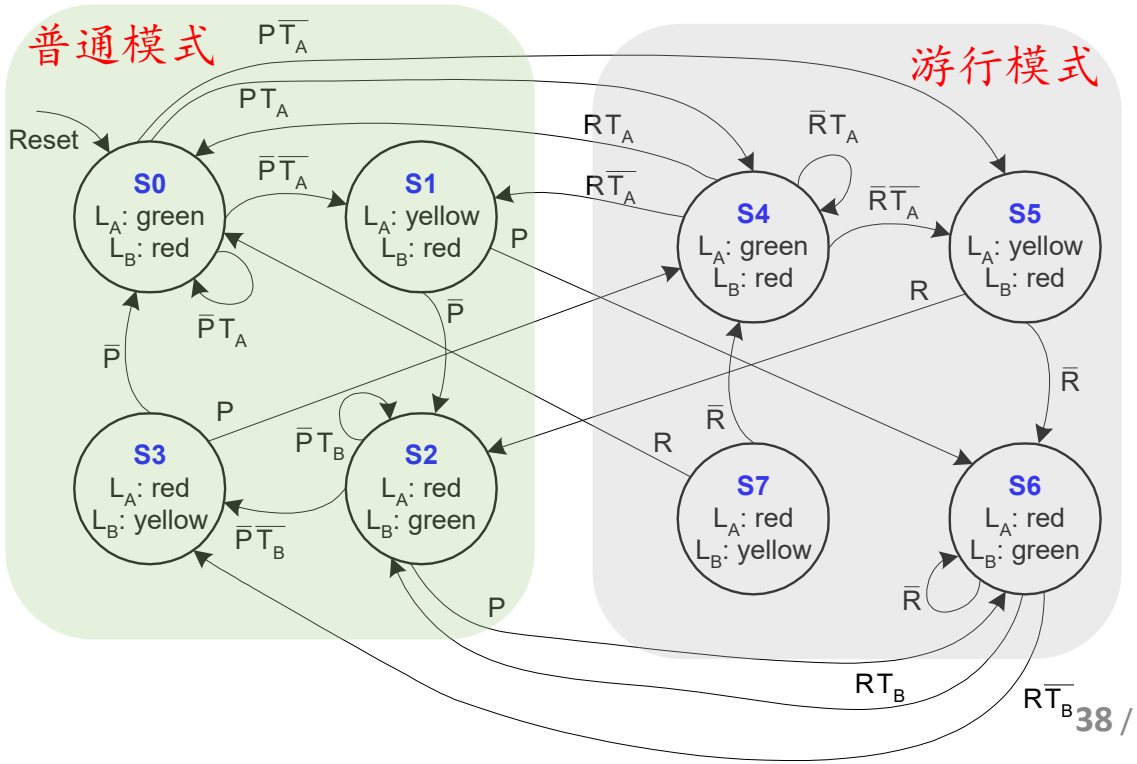
$$S_0^* = \bar{S}_1 \bar{S}_0 \bar{T}_A + S_1 \bar{S}_0 \bar{T}_B \qquad L_{A0} = \bar{S}_1 S_0 \qquad L_{B0} = S_1 S_0$$

【例3】交通灯控制器：**状态机的分解** (P79)

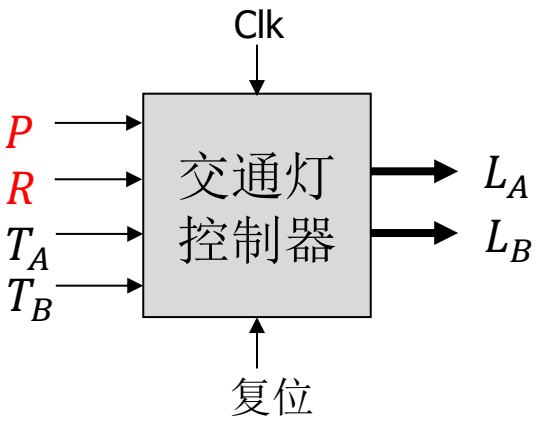


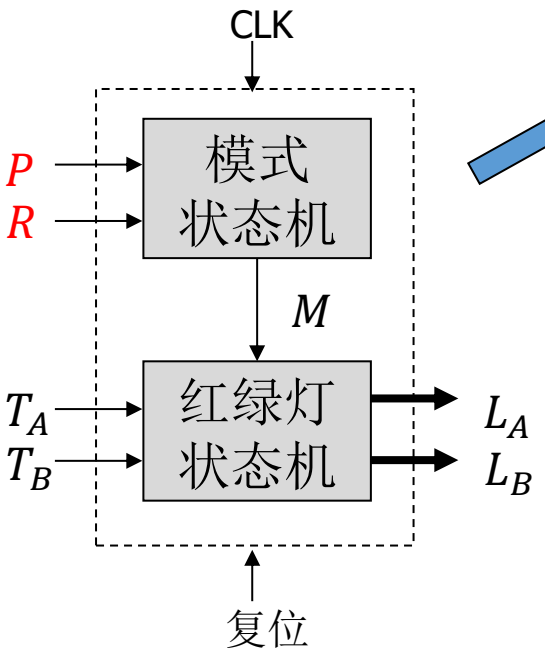


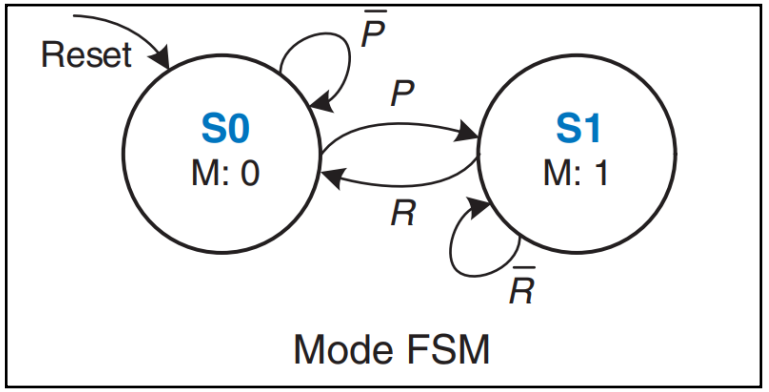


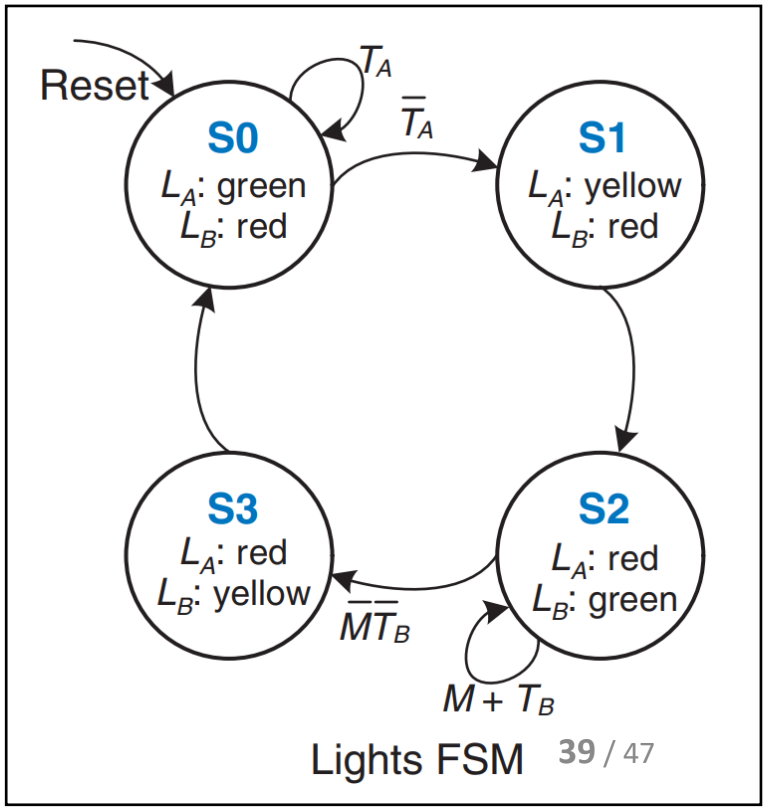


【例3】交通灯控制器：**状态机的分解** (P79)





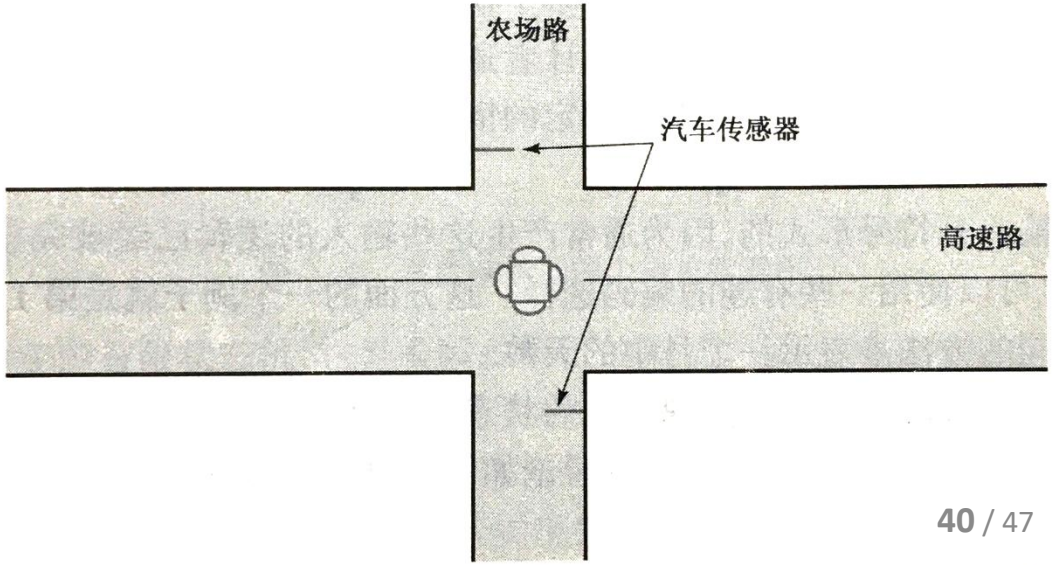




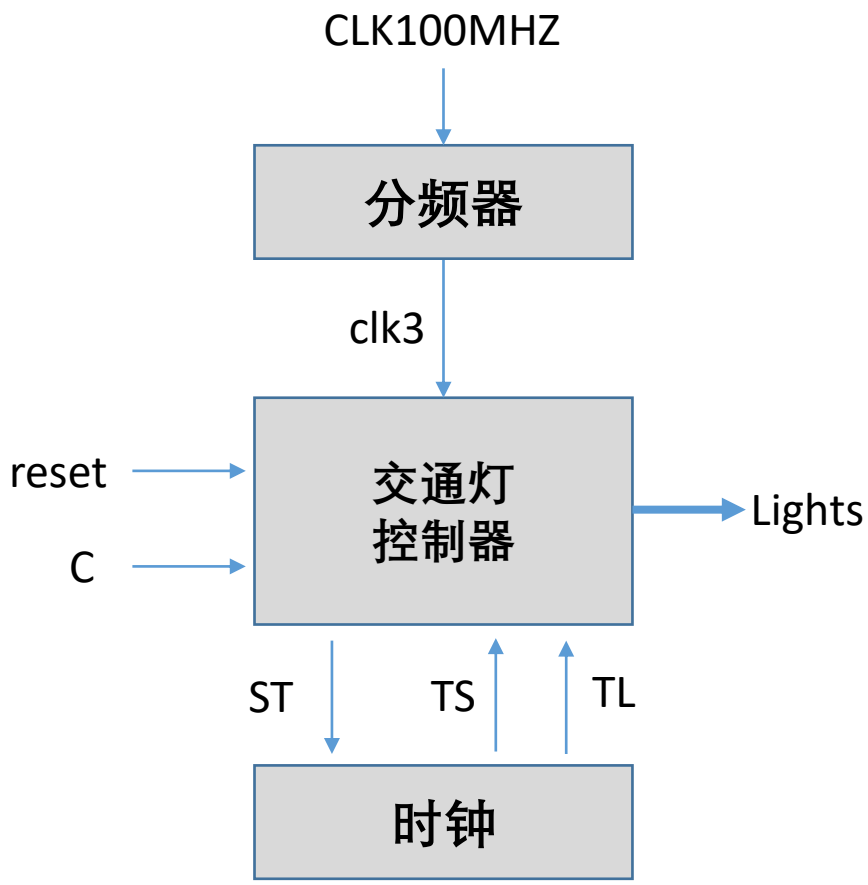
农场-高速公路 交通红绿灯*

繁忙的高速路与较少使用的农场路相交

- 只要农场路上的探测器没有发现车辆，高速公路上的交通灯就应该保持为绿灯。
- 只要有车辆在**农场路**上等待穿越十字路口，探测器就发出**信号C**。
  高速公路上的交通灯就由**绿→黄→红**，同时农场路上的交通灯变绿。
- 只要在农场路上探测到车辆，就保持一段**时间(TL)为绿灯**。
- 若农场路上没有车辆，或绿灯超过限定时间，农场路上的交通灯由**绿→黄→红**，
  同时高速公路上的交通灯变回到**绿灯**。
- 高速公路上的**绿灯**应保持一段**时间(TL)**。
- **黄灯的保持时间为TS**。



农场-高速公路 交通红绿灯*



C: 在农场路的两个方向上探测到车辆

ST: **复位**定时器，开始对长、短时间间隔进行计时

TS: 相对短的时间已到

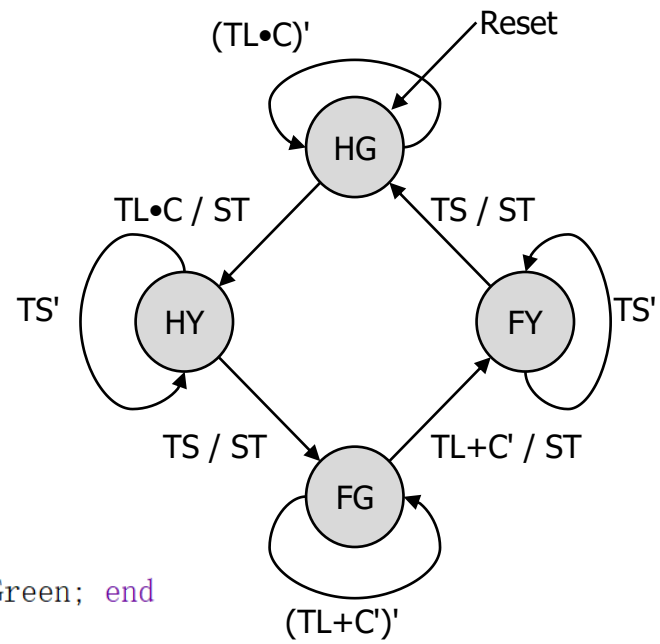
TL: 相对长的时间已到

```
1  module TrafficFarm_Top(
2      input  logic CLK100MHZ,
3      input  logic BTNL,
4      input  logic BTNR,
5      input  logic [0:0] SW,
6      output logic [5:0] LED );
7
8      logic TS, TL, ST;
9      logic clr    = BTNL;
10     logic reset  = BTNR;
11     logic findCar= SW[0];
12     logic clk3Hz;
13
14     clkdiv  U1(.clk(CLK100MHZ), .clr(clr), .clk3(clk3Hz));
15     Timer   T1(.clk(clk3Hz), .ST(ST), .TS(TS), .TL(TL));
16     TrafficFarm T2(.clk(clk3Hz),
17                    .reset(reset),
18                    .C(findCar),
19                    .TS(TS),
20                    .TL(TL),
21                    .Lights(LED),
22                    .ST(ST));
23 endmodule
```

```
2  module Timer(
3      input  logic clk,
4      input  logic ST,   // reset
5      output logic TS, TL);
6
7      integer value;
8
9      always_ff @(posedge clk, posedge ST)
10         if(ST == 1)        value <= 0; //async reset
11         else if(value<19) value <= value + 1;
12              else          value <= 19;
13
14     assign TS = (value>= 4); //5 cycles after reset
15     assign TL = (value>=19);//20 cycles after reset
16 endmodule
```

农场-高速公路*

```
1  // 交通灯控制：高速公路-农场路
2  module TrafficFarm(
3      input  logic clk,
4      input  logic reset,
5      input  logic C,    //探测农场路上是否由车辆
6      input  logic TS,   //相对短的时间已到(高速公路绿灯的最短时间)
7      input  logic TL,   //相对长的时间已到(农场路上绿灯的最长时间)
8      output logic [5:0] Lights,
9      output logic ST ); //复位定时器
10     //                                高-速 农-场
11     // 状态常量                      红黄绿红黄绿
12     parameter HighwayGreen  = 6'b0_0_1_1_0_0;
13     parameter HighwayYellow = 6'b0_1_0_1_0_0;
14     parameter FarmroadGreen = 6'b1_0_0_0_0_1;
15     parameter FarmroadYellow= 6'b1_0_0_0_1_0;
16
17     logic [5:0] state;
18     // 初始化
19     initial
20     begin ST = 0;  state = HighwayGreen; end
```

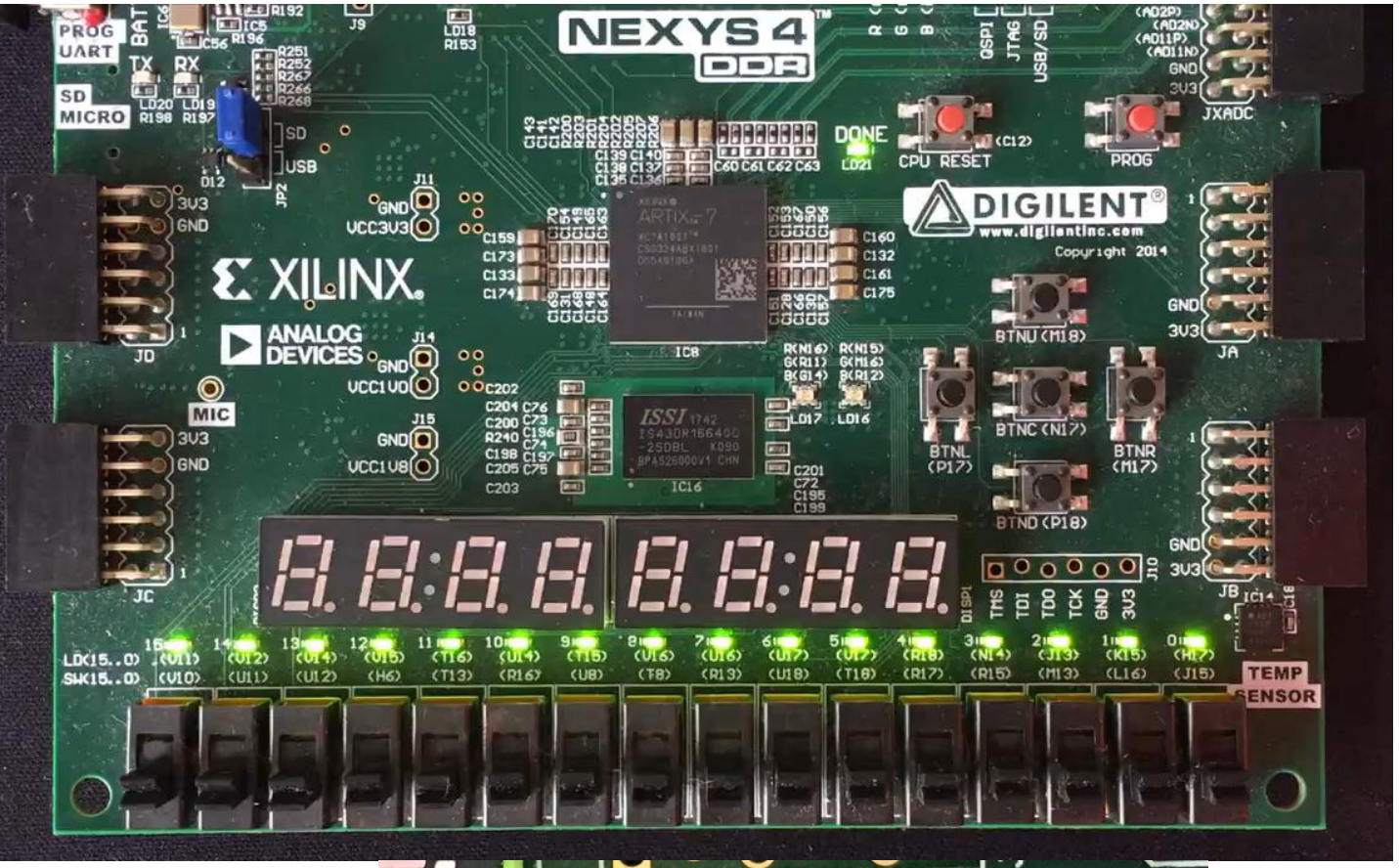


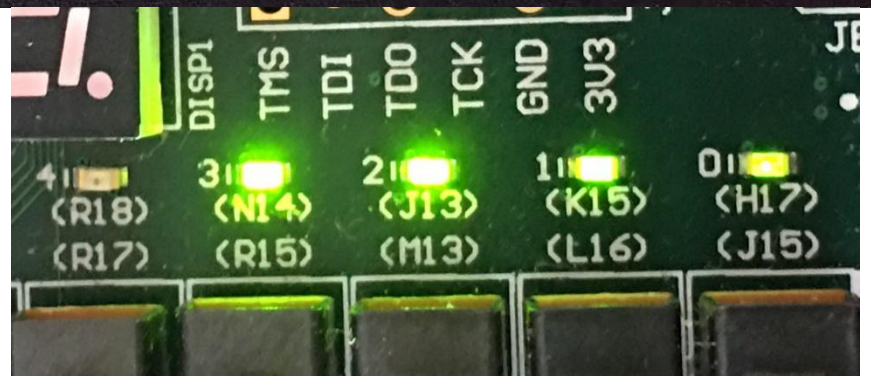

```
22     // 状态转换 + 输出ST
23     always_ff @(posedge clk)
24     begin  // 同步reset
25         if(reset) begin ST<=1; state<=HighwayGreen; end
26         else      begin ST<=0;
27             case(state)
28                 HighwayGreen  : if(TL & C)  begin ST<=1; state<=HighwayYellow;  end
29                 HighwayYellow : if(TS)      begin ST<=1; state<=FarmroadGreen;  end
30                 FarmroadGreen : if(TL | ~C) begin ST<=1; state<=FarmroadYellow; end
31                 FarmroadYellow: if(TS)      begin ST<=1; state<=HighwayGreen;   end
32             endcase   end
33     end
34
35     // 输出逻辑
36     assign Lights = state;
37 endmodule
```

4

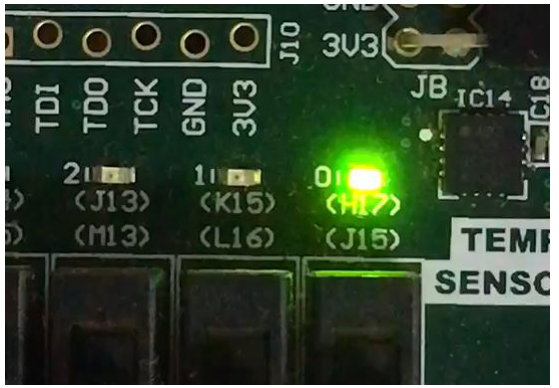
呼吸灯

呼吸灯？



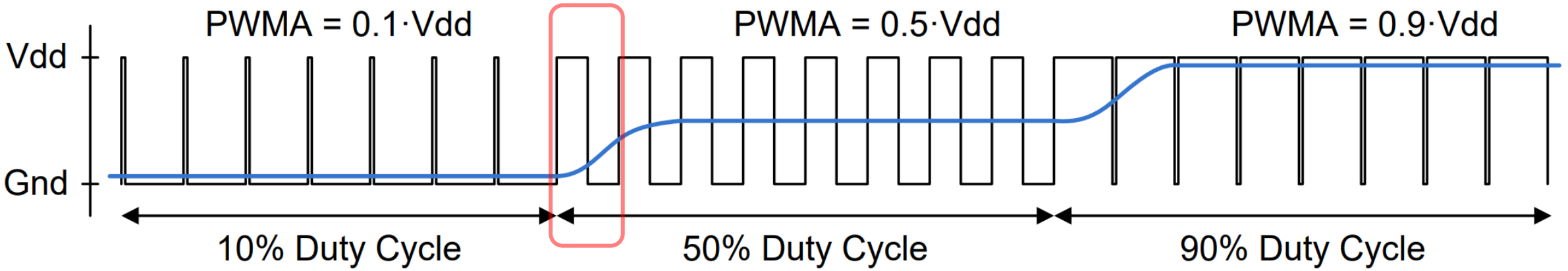



```
1  module blinkLED(
2      input logic CLK100MHZ,
3      output logic [1:0] LED   );
4
5      logic [30:0] blinkCount;
6
7      always @(posedge CLK100MHZ)
8         blinkCount <= blinkCount + 1;
9
10     assign LED[0] = blinkCount[25]; //Slow
11     assign LED[1] = blinkCount[24]; //Fast
12 endmodule
```



脉冲宽度调制PWM (Pulse-Width Modulation)

将模拟信号变换为数字脉冲信号的一种技术。



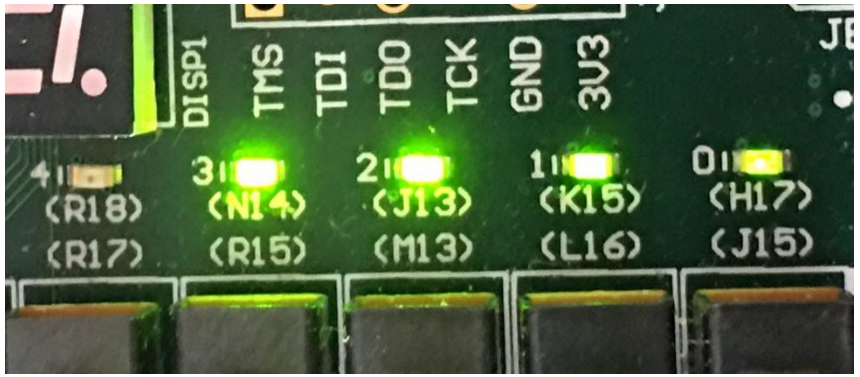
- 占空比(duty cycle)：脉冲高时间除以脉冲窗口时间。

$$\text{占空比} = \frac{\text{脉冲高电平持续时间}}{\text{脉冲周期}} \times 100\%$$

PWM_LED

PWM-LED

```
1  module PWM  #( parameter N=8 )      // 占空比位数
2      //若N=8，则表明：目标PWM频率 = pwmClk的频率/256
3      ( input  logic pwmClk,
4        input  logic [N-1:0] duty, //duty cycle占空比
5        output logic pwmOut   );
6
7      logic [N-1:0] cnt = 0;
8
9      always_ff @(posedge pwmClk)
10     begin
11         cnt <= cnt + 1;
12         // 当cnt<duty时，pwmOut=1;
13         // 否则，          pwmOut=0.
14         pwmOut <= (cnt < duty);
15     end
16 endmodule
```




```
1  module PWM_LED_Top(
2      input  logic CLK100MHZ,
3      output logic [3:0] LED  );
4
5      // 100MHz/(2^8) = 390kHz (目标PWM频率：LED的闪烁频率)
6      PWM #(8)L0(.pwmClk(CLK100MHZ), .duty(10),  .pwmOut(LED[0])); //  11/256= 4%
7      PWM #(8)L1(.pwmClk(CLK100MHZ), .duty(50),  .pwmOut(LED[1])); //  51/256=20%
8      PWM #(8)L2(.pwmClk(CLK100MHZ), .duty(127), .pwmOut(LED[2])); // 128/256=50%
9      PWM #(8)L3(.pwmClk(CLK100MHZ), .duty(255), .pwmOut(LED[3])); // 256/256=100%
10 endmodule
```

呼吸灯代码

```
1  module breathLED_Top(
2      input  logic CLK100MHZ,
3      output logic [15:0] LED  );
4
5      logic [7:0] dutyNumber; // 0~255
6      logic clk100Hz, brightness;
7
8      ClkDiv C1(CLK100MHZ, clk100Hz);
9
10     breathLED B1(clk100Hz, dutyNumber);
11
12     PWM L1(CLK100MHZ, dutyNumber, brightness);
13
14     assign LED[15:0] = {16{brightness}};
15 endmodule
```

```
1  // 呼吸灯：256个亮度等级
2  module breathLED(input logic clk,
3                   output logic [7:0] data );
4
5      logic [8:0] i = 0;   // 512
6
7      always_ff @(posedge clk)
8      begin
9          i <= i + 1;
10         if (i <= 255)
11             data <= i;          //渐增256个亮度
12         else
13             data <= 511 - i; //渐减256个亮度
14     end
15 endmodule
```