数字逻辑设计

王鸿鹏
计算机科学与技术学院
wanghp@hit.edu.cn

同步时序逻辑设计

- 状态机基础
- 原始状态图和状态表
- 状态表化简
- 状态分配

状态机(State machine)定义

- 是一个有向图形，由一组节点和一组相应的转移函数组成。
- 状态机通过响应一系列事件而“运行”。
- 每个事件都在属于“当前”节点的转移函数的控制范围内，其中函数的范围是节点的一个子集。函数返回“下一个”（也许是同一个）节点。这些节点中至少有一个必须是终态。当到达终态，状态机停止。

状态机(State machine)基础

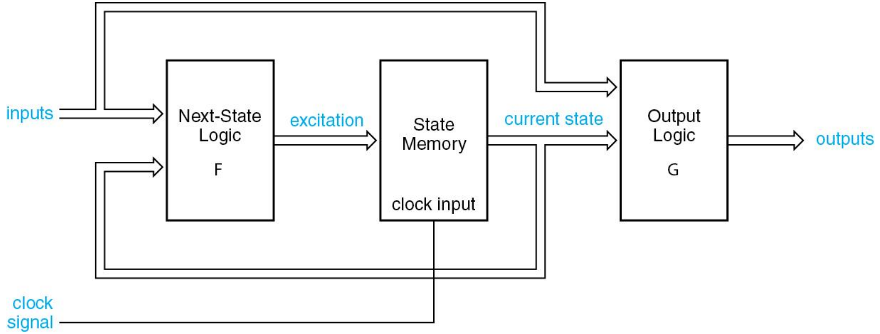
- 包含一组**状态集**(states)、一个起始状态(start state)、一组**输入符号集**(alphabet)、一个映射输入符号和当前状态到下一状态的**转换函数**（transition function）的计算模型
- 状态是**状态变量(state variable)的集合**
  - 状态变量的值包含决定未来行为的所有信息
- 状态变化：从一个状态变为另一个状态
  - 变化的时间、如何变化

时序逻辑电路——状态机描述

- 时序电路的**状态（state）**
  - 具有n位二进制状态变量的电路有2^n种可能状态。
  - 状态有限，可称为有限状态机(FSM：Finite State Machine)
  - 大多时序电路和几乎所有的状态机都会使用边沿触发的D触发器存储状态变量
- 时序电路状态变化
  - 大多数时序电路状态发生变化的时间由**时钟信号**决定
  - 状态变化函数——次态方程

Mealy状态机 vs Moore状态机

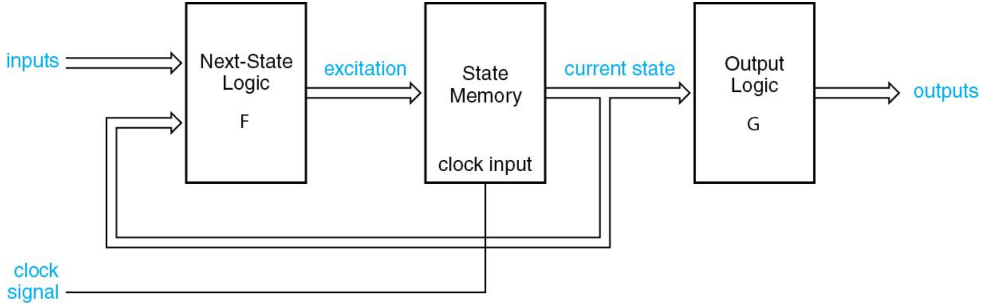
- 状态机结构
  - 状态存储器是存储状态机现态的一组触发器。
  - 次态:由次态逻辑（next-state logic）F确定
  - 输出:由输出逻辑（output logic）G确定



Mealy状态机

次态=F(现态，输入)

输出=G(现态，输入)



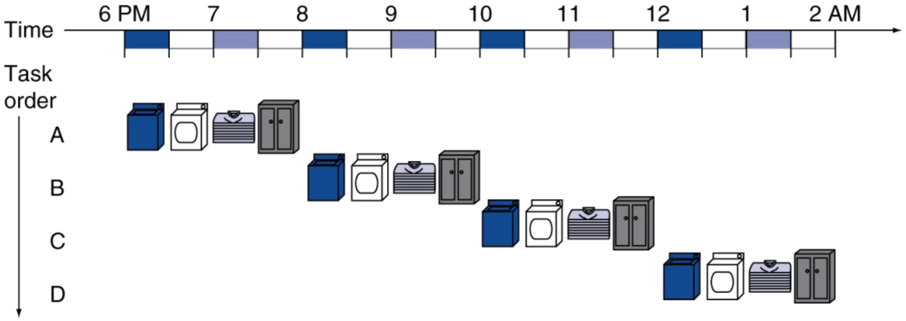
Moore状态机

次态=F(现态，输入)

输出=G(现态)

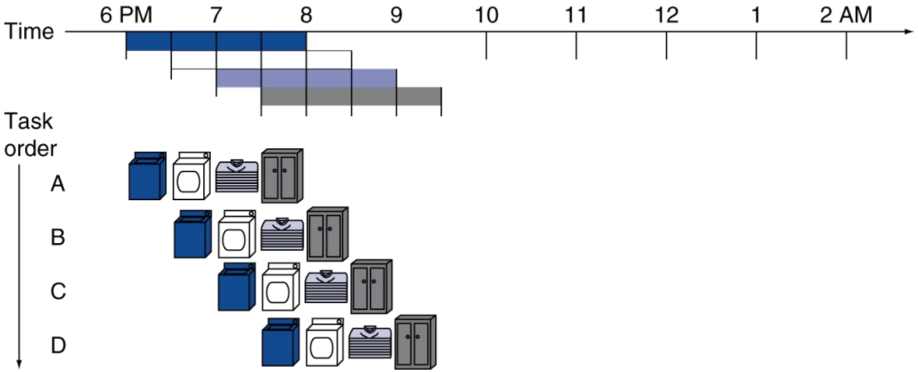
生活中的流水线

- 假设洗衣包括**四个**步骤：洗衣机中洗衣、烘干机中烘干、叠衣服、收纳到柜子中，每个步骤0.5小时。



- 洗衣任务为4，**加速比**

  $= 2*4/(2+0.5*3) \approx 2.3$



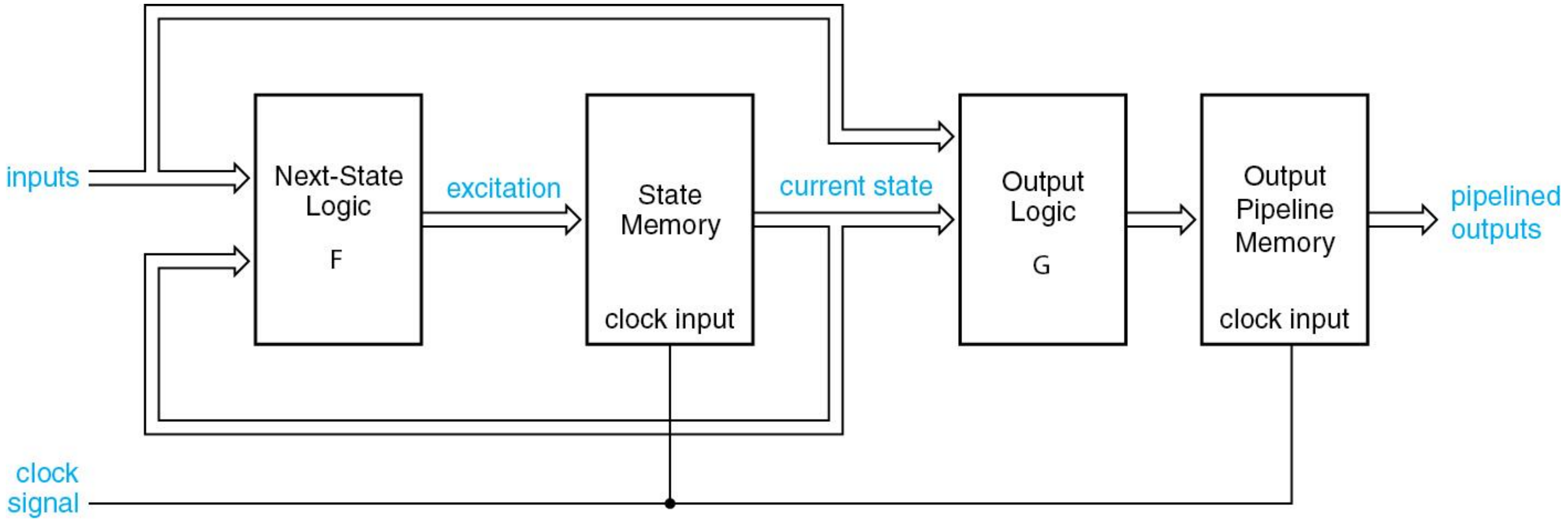
- 洗衣任务数为n，加速比

  $= 2n/(2+0.5*(n-1))$

  $\approx$ **4** = 流水线中的步骤数

状态机结构和分析

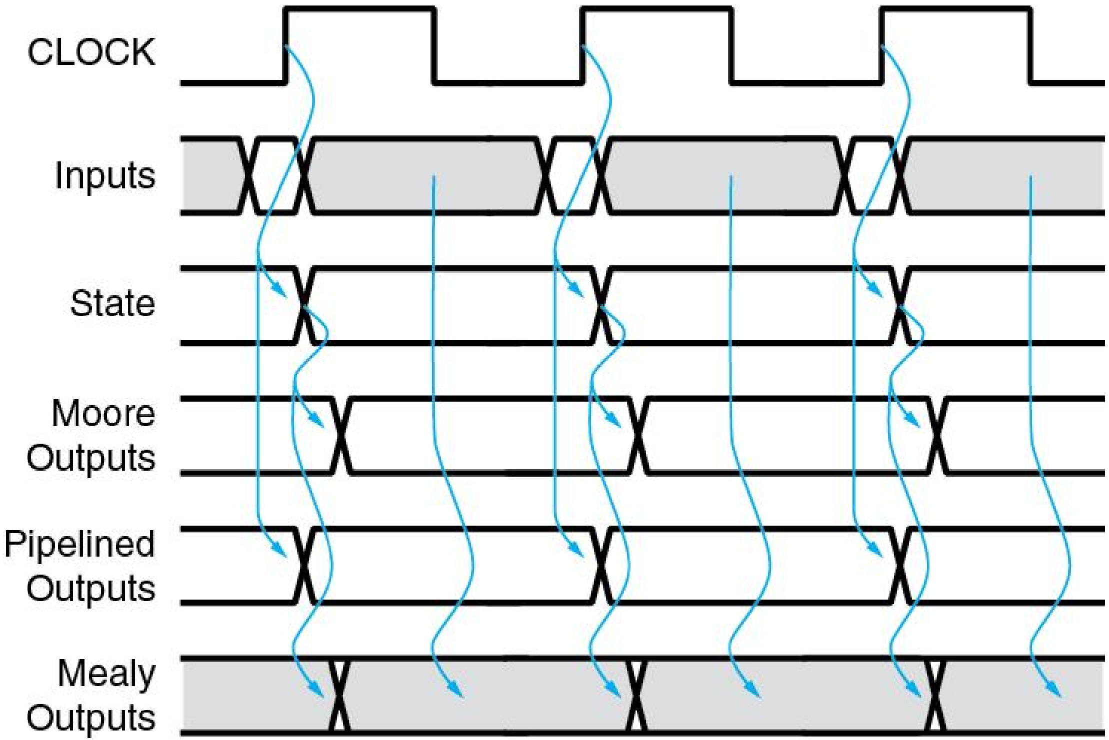
- 将Mealy机中流水线输出的触发器看作状态存储元件一部分，可得到具有**输出编码状态赋值**的Moore机



具有流水线输出的Mealy机

状态机的时序

- 图中使用上升沿触发D触发器
- 箭头表示变化时的因果关系



状态机的定义和分析

- 状态机的形式定义
  - 次态=F(现态，输入信号)
  - 输出=G(现态，*输入信号*)
- 状态机分析的**基本步骤**：
  - 确定次态函数F和输出函数G
  - 用F和G构造状态/输出表
  - 画出状态图，表示上述信息

使用D触发器的状态机分析

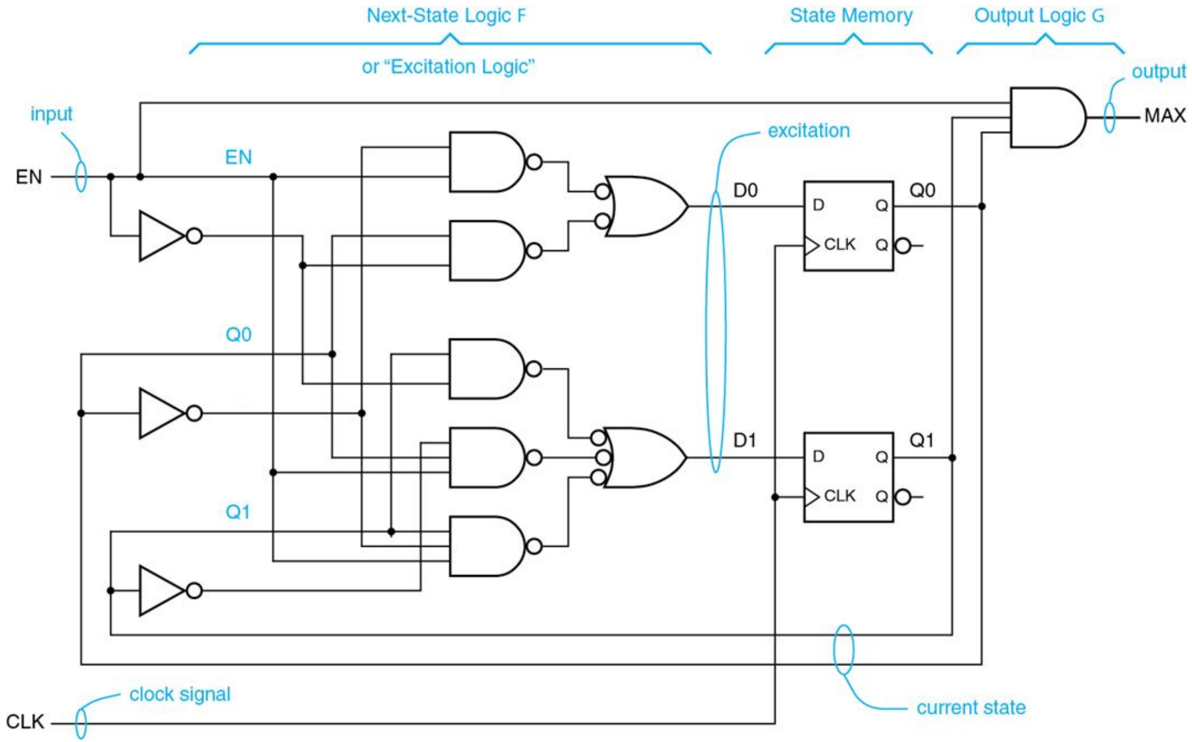
D触发器的输出Q0和Q1为状态变量，记录状态机当前状态值。

输入信号D0、D1在时钟触发沿向D触发器提供**激励(excitation)**。

激励方程：

$$D0=Q0\cdot EN'+Q0'\cdot EN$$

$$D1=Q1\cdot EN' +Q1'\cdot Q0\cdot EN +Q1\cdot Q0'\cdot EN$$



使用D触发器的状态机分析

D触发器的次态$Q^*=D$

状态转移方程：$Q0^*=D0=Q0\cdot EN'+Q0'\cdot EN$

$Q1^*=Q1\cdot EN'+Q1'\cdot Q0\cdot Q1'\cdot EN+Q1\cdot Q0'\cdot EN$

输出方程：$MAX=Q1\cdot Q0\cdot EN$

状态转移表

	EN	
Q1 Q0	0	1
00	00	01
01	01	10
10	10	11
11	11	00
	Q1* Q0*	

状态S(Q_1Q_0的组合)

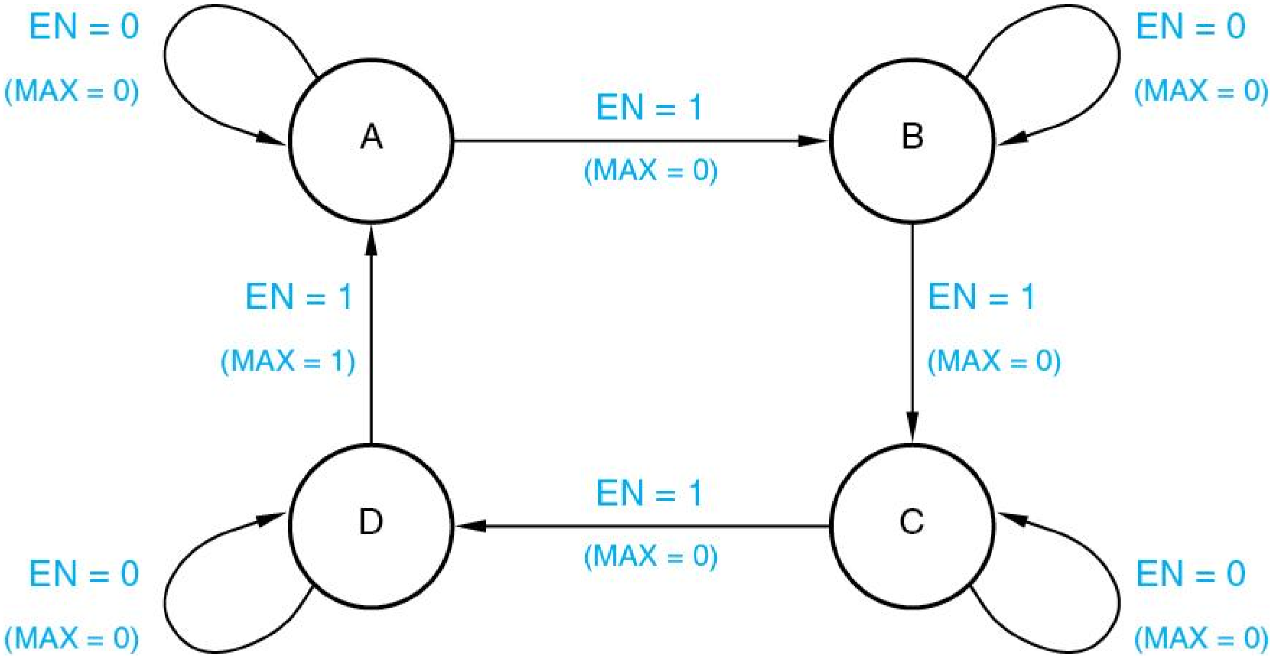
	EN	
S	0	1
A	A	B
B	B	C
C	C	D
D	D	A
	S*	

状态/输出表

	EN	
S	0	1
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	D, 0
D	D, 0	A, 1
	S*, MAX	

Mealy机的状态图

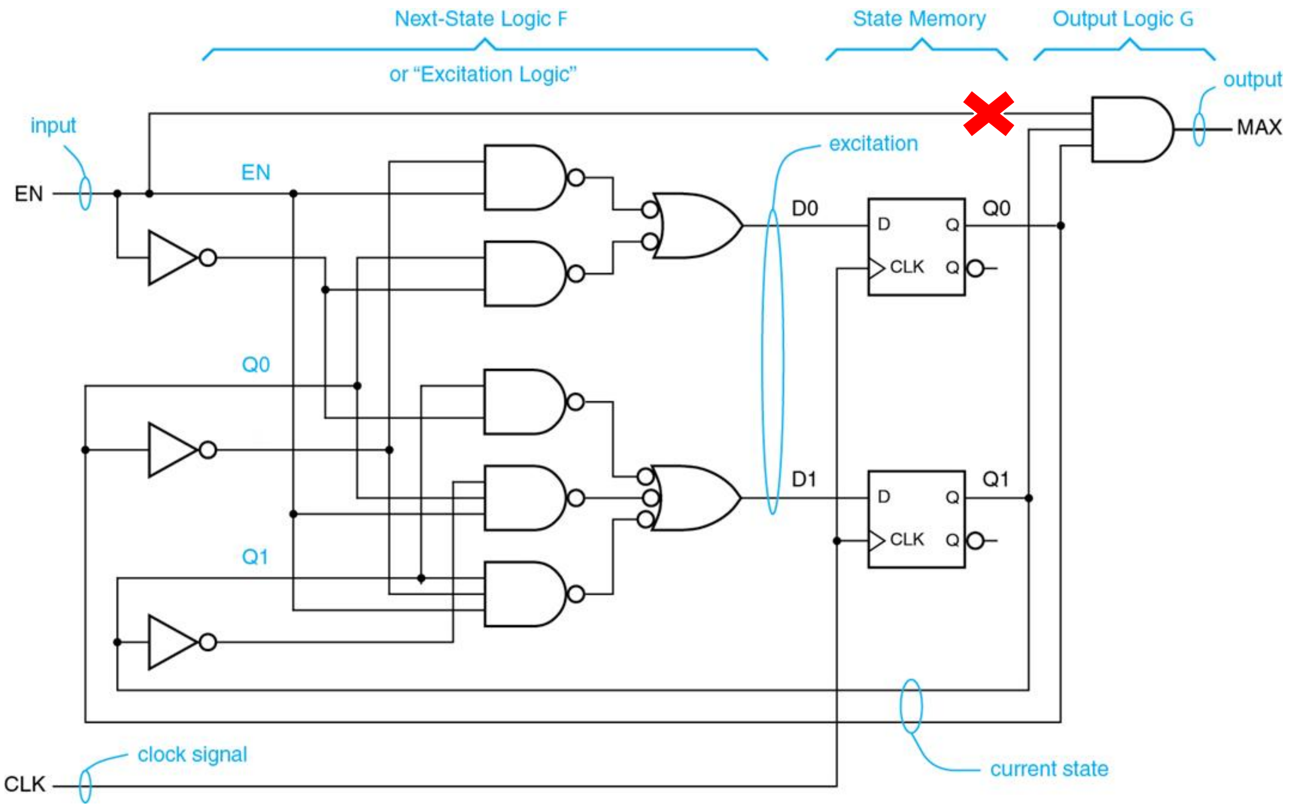
输出方程：MAX=Q1·Q0·EN



	EN	
S	0	1
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	D, 0
D	D, 0	A, 1
	S*, MAX	

Moore机的状态图

- 假设从产生MAX输出的与门输入去掉EN

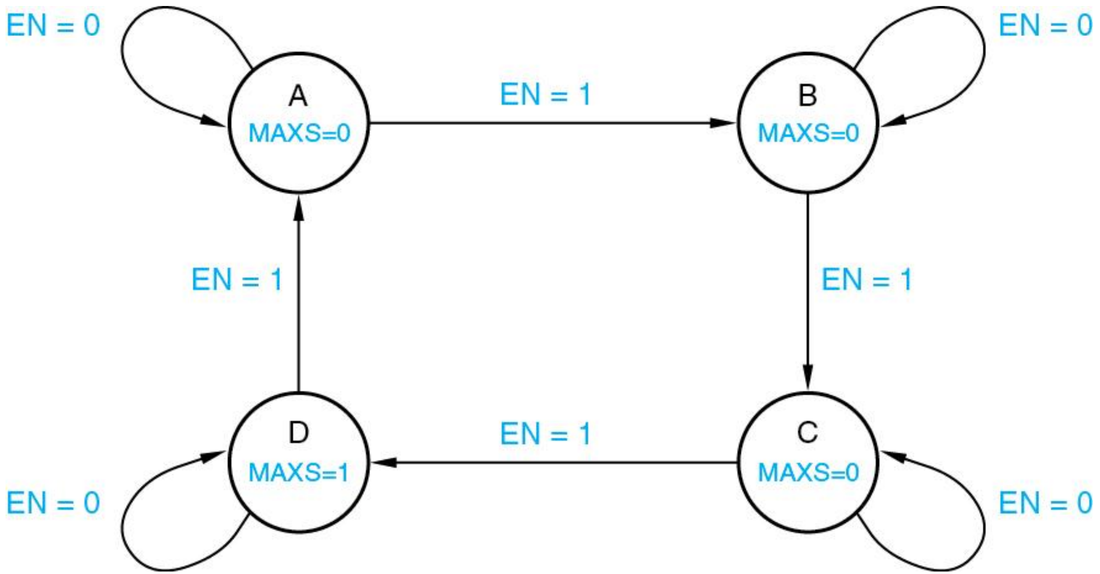


Moore机的状态图

- 假设从产生MAX输出的与门输入去掉EN
- 输出方程：MAXS=Q1·Q0（Moore型）

	EN		
S	*0*	*1*	*MAXS*
A	A	B	0
B	B	C	0
C	C	D	0
D	D	A	1
	S*		

Moore机的状态/输出表



同步时序逻辑设计

- 状态机基础
- 原始状态图和状态表
- 状态表化简
- 状态分配

利用触发器设计时序逻辑的方法

1 根据需求——> 获得原始状态图、状态表

2 最小化状态图、状态表

3 状态编码（分配）——>获得状态转移表

4 状态转移表
触发器特征 } ——>触发器激励表

5 卡诺图化简——> { 激励（输入）函数表达式
输出函数表达式 }

6 电路实现 7 检查无关状态

直接构图法（适于简单应用）

1 根据文字描述的设计要求，先假定一个初态；

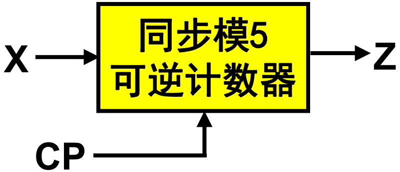
2 从初态开始，每加入一个输入取值，就可确定其次态和输出；

3 该次态可能是现态本身，也可能是已有的另一个状态，或是新增加的一个状态。

4 这个过程持续下去，直至每一个现态向其次态的转换都已被考虑，并且不再构成新的状态。

现态 Q^n	Q^{n+1}/ Z	
	X=0	X=1
a	b / 0	e / 1
b	c / 0	a / 0
c	d / 0	b / 0
d	e / 0	c / 0
e	a / 1	d / 0

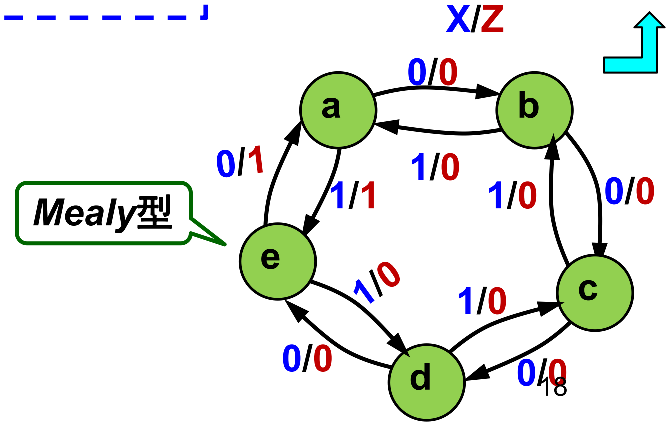
例1：给出同步模5可逆计数器的状态表



X=0： 加计数

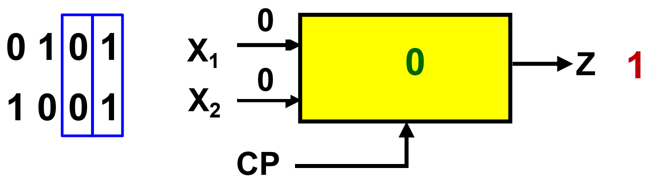
X=1： 减计数

Z: 进位、借位输出标志

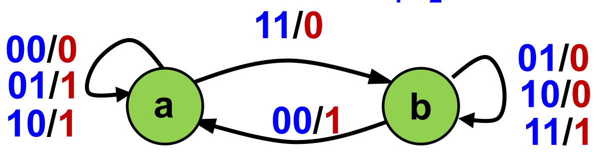


直接构图法——实例

给出同步二进制串行加法器的状态表



设加法器内部状态
a—— 无进位
b—— 有进位



直接构图法

1）根据文字描述的设计要求，先假定一个初态；

2）从这个初态开始，每加入一个输入取值，就可确定其次态和输出；

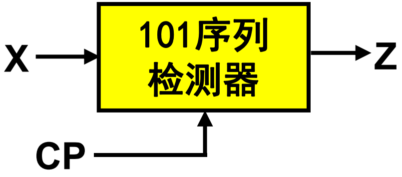
3）该次态可能是现态本身，也可能是已有的另一个状态，或是新增加的一个状态。

4）这个过程持续下去，直至每一个现态向其次态的转换都已被考虑，并且不再构成新的状态。

现态 Q^n	Q^{n+1}/ Z			
	X_1X_2=00	X_1X_2=01	X_1X_2=10	X_1X_2=11
a	a / 0	a / 1	a / 1	b / 0
b	a / 1	b / 0	b / 0	b / 1

序列检测—101序列检测器

例：给出同步*Mealy*型101序列检测器的状态表



X:　0 1 0 1 0 1 1 0 1

Z:　0 0 0 1 0 1 0 0 1

可重叠检测

X:　0 1 0 1 0 1 0 1 1

Z:　0 0 0 1 0 0 0 1 0

不可重叠检测

（1）状态设定

S_0——初始状态，表示收到1位数据：“0”

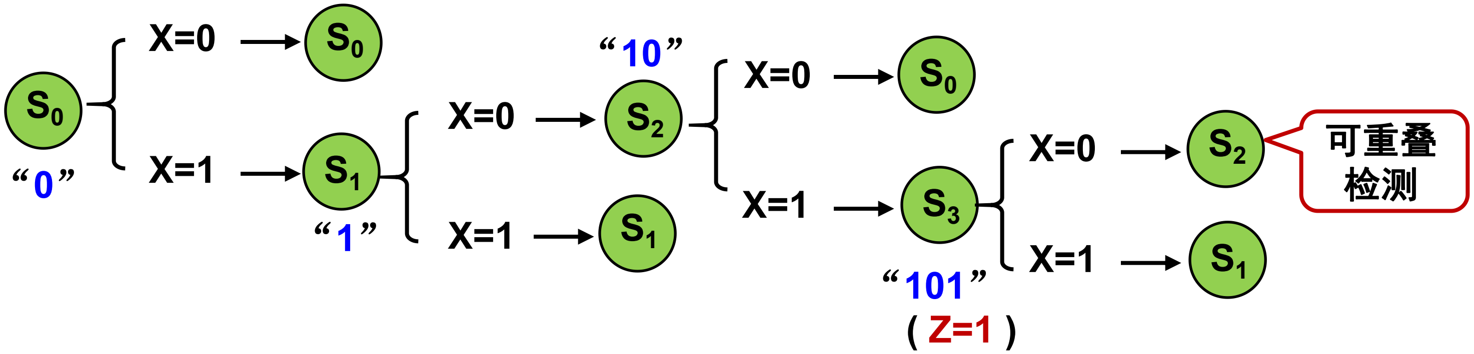
S_1——表示收到1位数据：“1”

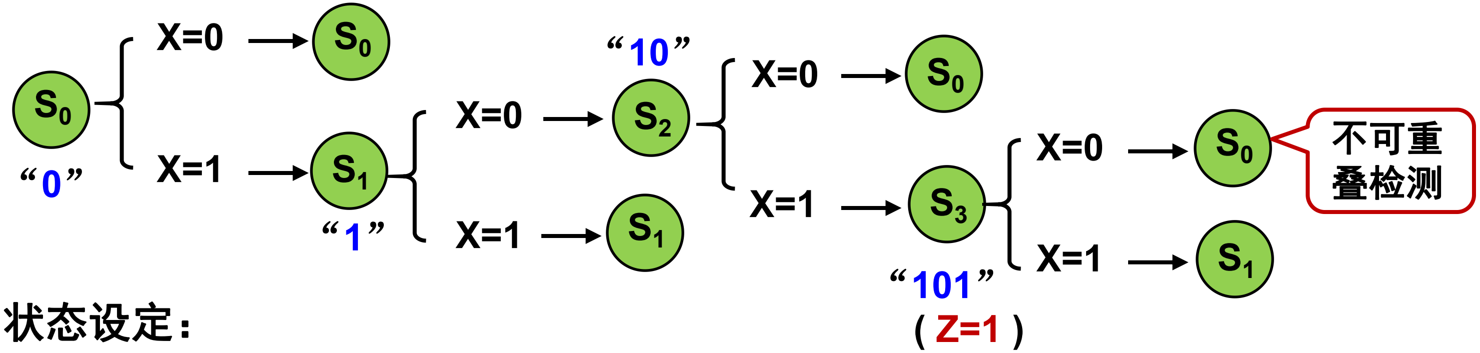
S_2——表示收到2位数据：“10”

S_3——表示收到3位数据：“101”，此时输出标志 Z=1.

只标记感兴趣的子串

101序列检测器



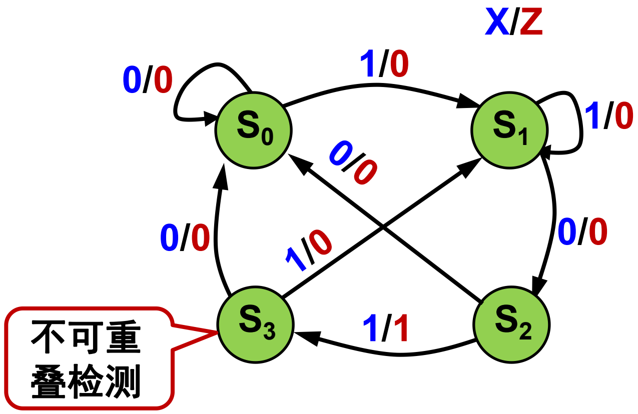


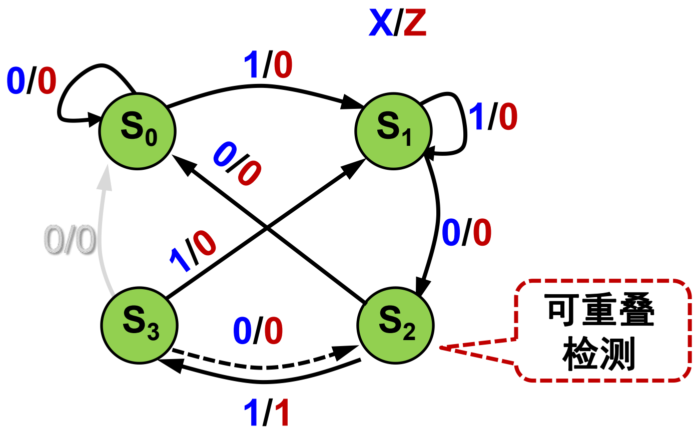
状态设定：

S_0——0； S_1—— 1;

S_2—— 10； S_3—— 101， 且Z=1

构造原始状态图和状态表





现态 Q^n	Q^{n+1}/ Z	
	X=0	X=1
S_0	S_0 / 0	S_1 / 0
S_1	S_2 / 0	S_1 / 0
S_2	S_0 / 0	S_3 / 1
S_3	S_0 / 0	S_1 / 0

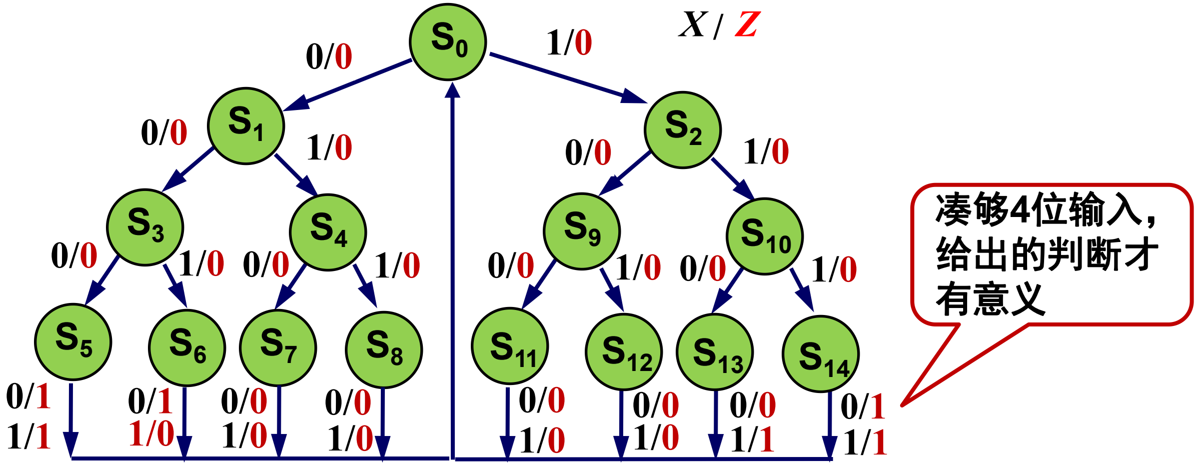
现态 Q^n	Q^{n+1}/ Z	
	X=0	X=1
S_0	S_0 / 0	S_1 / 0
S_1	S_2 / 0	S_1 / 0
S_2	S_0 / 0	S_3 / 1
S_3	S_2 / 0	S_1 / 0

序列检测的原始状态图构造方法总结

- 检测器输入端收到1位数据时，有两种可能：0或1，分别用S_0和S_1标记这两个状态，通常用S_0表示初始状态。
- 收到2位数据，只标记感兴趣的子串，用S_2表示(如10)
- 同理，收到3位数据，只标记感兴趣的子串，用S_3表示(例如101)……，直到把感兴趣的完整子串标记完为止。
- 从初始状态开始，采用直接构图法，直到将每一个当前状态在所有取值下的次态转换及输出情况都已经考虑到，并且没有遗漏为止。

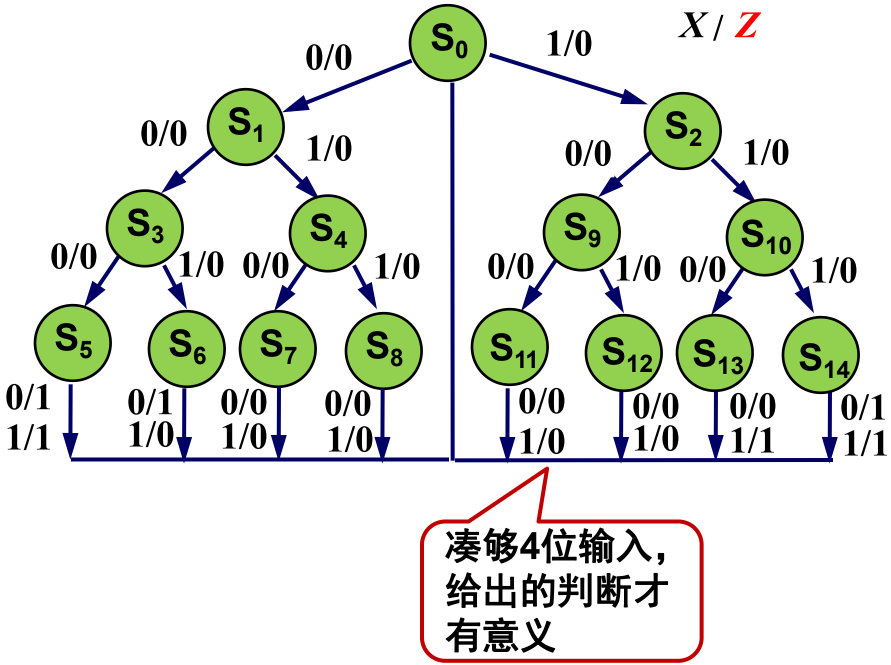
码制检测电路设计

- 建立一个余3码误码检测器的原始状态图和原始状态表。要求：余3码高位在前、低位在后串行地加到检测器的输入端；电路每接收一组代码（即在收到第4位代码时）判断。若是错误代码，则输出为1；否则输出为0，电路又回到初始状态并开始接收下一组代码。



原始状态图和状态表

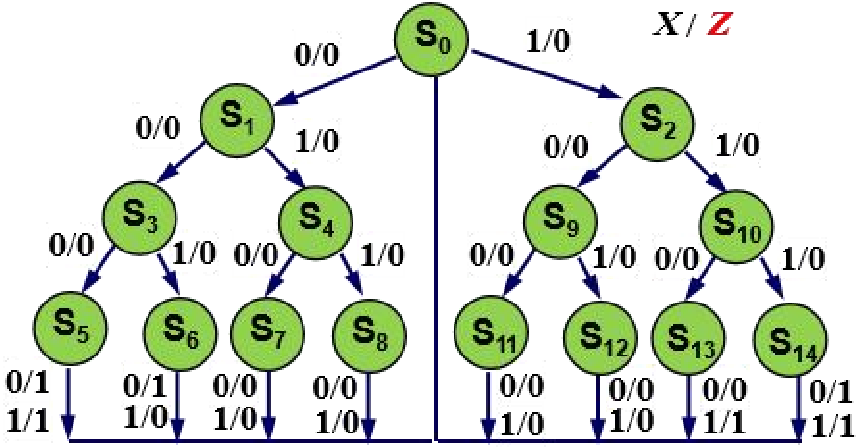
原始状态图



现态 Q^n	Q^{n+1}/ Z	
	X=0	X=1
S_0	S_1 / 0	S_2 / 0
S_1	S_3 / 0	S_4 / 0
S_2	S_9 / 0	S_{10} / 0
S_3	S_5 / 0	S_6 / 0
S_4	S_7 / 0	S_8 / 0
S_5	S_0 / 1	S_0 / 1
S_6	S_0 / 1	S_0 / 0
S_7	S_0 / 0	S_0 / 0
S_8	S_0 / 0	S_0 / 0
S_9	S_{11} / 0	S_{12} / 0
S_{10}	S_{13} / 0	S_{14} / 0
S_{11}	S_0 / 0	S_0 / 0
S_{12}	S_0 / 0	S_0 / 0
S_{13}	S_0 / 0	S_0 / 0
S_{14}	S_0 / 0	S_0 / 1
S_{15}	S_0 / 1	S_0 / 1

N位码制检测电路的原始状态图构造方法总结

- 从初始状态S_0开始（这个初始状态没有特殊含义，仅仅代表一个起点），每来一个输入，次态总是分成左右两种情况。
- 状态图由上至下分为N层：第一层代表起点；第二层代表检测器收到1位数据时，电路的状态情况；第三层代表检测器收到2位数据时，电路的状态情况……；直到第N层，代表检测器收到 N-1位数据时，电路的状态情况。再来一位输入数据，则构成了N位待检测码制。此时，检测器可以给出判断，该码制正确还是错误。
- 一轮检测结束，回到初始状态，等待下一组输入。



同步时序逻辑设计

- 状态机基础
- 原始状态图和状态表
- 状态表化简
- 状态分配

利用触发器设计时序逻辑的方法

1 根据需求——> 获得原始状态图、状态表

2 **最小化状态图、状态表**

3 状态编码（分配）——>获得状态转移表

4 状态转移表 \
触发器特征 } ——>触发器激励表

5 卡诺图化简——> { 激励（输入）函数表达式 \
输出函数表达式

6 电路实现　　　　7 检查无关状态

状态表化简

- 时序电路的两个状态S_i和S_j，如果它们对每个输入所产生的输出完全相同，且它们的次态等价，则这两个状态是等价的（可以合并为一个状态）——状态化简
- 完全定义状态表的化简——隐含（蕴含）表法
  - 两两比较原始状态表中的**所有**状态，找出能合并、不能合并、能否合并待定的状态对。
  - 追踪**能否合并待定**的状态对，直至确定它们能不能合并，从而找到原始状态表中的所有**等价状态对**。
  - 基于这些等价状态对确定**最大等价状态类**，获得原始状态表的**最小覆盖集**，建立最简状态表

状态表化简——等价状态的判定条件

- 状态表中的任意两个状态 S_i 和 S_j **同时**满足下列两个条件，它们可以合并为一个状态
  - 对所有不同的输入，**输出**分别相同
  - 对所有不同的输入，**次态**为下列情况之一：
    1. 两个次态完全相同
    2. 两个次态为其现态本身或**交错**
    3. 两个次态为**状态对封闭链**中的一个状态对
    4. 两个次态的某一后续状态对可以合并

状态合并的必要条件

隐含表法化简状态表

等价状态的判定条件

状态表中的任意两个状态 S_i 和 S_j 同时满足下列两个条件，它们可以合并为一个状态

1. 在所有不同的现输入下，现输出分别相同

状态合并的必要条件

2. 在所有不同的现输入下，次态分别为下列情况之一

（1）两个次态完全相同

（2）两个次态为其现态本身或交错

（3）两个次态为状态对封闭链中的一个状态对

（4）两个次态的某一后续状态对可以合并

例1：化简如下状态表

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1
a	c / 0	b / 1
b	f / 0	a / 1
c	d / 0	g / 0
d	d / 1	e / 0
e	c / 0	e / 1
f	d / 0	g / 0
g	c / 1	d / 0

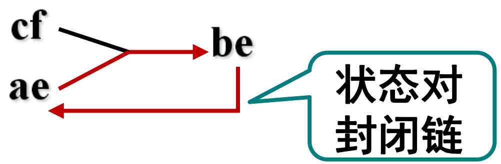
① 建立隐含表

② 比较

③ 追踪

	a	b	c	d	e	f
b	cf ✓					
c	X	X				
d	X	X	X			
e	be ✓	ae cf ✓	X	X		
f	X	X	✓	X	X	
g	X	X	X	de ed X	X	X

竖列横排
掐头去尾



等价状态对

{ a ,b }、{ a ,e }

{ b ,e }、{ c ,f }

隐含表法化简状态表——续

④ 获得最大等价状态类

等价状态类的定义——

If： $S_i \equiv S_j$， $S_j \equiv S_m$

Then： $S_i \equiv S_j \equiv S_m$, 即 $\{ S_i , S_j , S_m \}$

最大等价状态类——某一等价状态类不属于其他任何等价状态类

等价状态对：

{ a ,b }、 { a ,e }

{ b ,e }、 { c ,f }

最大等价状态类：

{ a , b ,e }、 { c ,f }

令
q_1 = { a , b ,e }
q_2 = { c ,f }
q_3 = d
q_4 = g

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1
a	c / 0	b / 1
b	f / 0	a / 1
c	d / 0	g / 0
d	d / 1	e / 0
e	c / 0	e / 1
f	d / 0	g / 0
g	c / 1	d / 0

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1
q_1	q_2 / 0	q_1 / 1
q_1	q_2 / 0	q_1 / 1
q_2	q_3 / 0	q_4 / 0
q_3	q_3 / 1	q_1 / 0
q_1	q_2 / 0	q_1 / 1
q_2	q_3 / 0	q_4 / 0
q_4	q_2 / 1	q_3 / 0

化简后的状态表

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1
q_1	q_2 / 0	q_1 / 1
q_2	q_3 / 0	q_4 / 0
q_3	q_3 / 1	q_1 / 0
q_4	q_2 / 1	q_3 / 0

最小覆盖集： $\{q_1 , q_2 , q_3 , q_4\}$

隐含表法化简状态表——实例

化简如下状态表

现态 Q^n	Q^{n+1}/Z $X_1X_2=00$	$X_1X_2=01$	$X_1X_2=10$	$X_1X_2=11$
a	b / 0	c / 0	b / 1	a / 0
b	e/ 0	c / 0	b/ 1	d / 1
c	a / 0	b / 0	c / 1	d / 1
d	c / 1	d / 0	a / 1	b / 0
e	c / 0	c / 0	c / 1	e / 0

b	X			
c	X	ae ✓		
d	X	X	X	
e	bc ✓	X	X	X
	a	b	c	d

bc → ae → bc

状态对封闭链

等价状态对:

{a,e} , {b,c}

令 $\begin{cases} q_1 = \{ a ,e \} \\ q_2 = \{ b ,c \} \\ q_3 = d \end{cases}$

现态 Q^n	Q^{n+1}/Z $X_1X_2=00$	$X_1X_2=01$	$X_1X_2=10$	$X_1X_2=11$
q_1	q_2 / 0	q_2 / 0	q_2 / 1	q_1 / 0
q_2	q_1/ 0	q_2 / 0	q_2/ 1	q_3 / 1
q_2	q_1 / 0	q_2 / 0	q_2 / 1	q_3 / 1
q_3	q_2 / 1	q_3 / 0	q_1 / 1	q_2 / 0
q_1	q_2 / 0	q_2 / 0	q_2 / 1	q_1 / 0

现态 Q^n	Q^{n+1}/Z $X_1X_2=00$	$X_1X_2=01$	$X_1X_2=10$	$X_1X_2=11$
q_1	q_2 / 0	q_2 / 0	q_2 / 1	q_1 / 0
q_2	q_1 / 0	q_2 / 0	q_2 / 1	q_3 / 1
q_3	q_2 / 1	q_3 / 0	q_1 / 1	q_2 / 0

同步时序逻辑设计

- 状态机基础
- 原始状态图和状态表
- 状态表化简
- 状态分配

利用触发器设计时序逻辑的方法

1 根据需求——> 获得原始状态图、状态表

2 最小化状态图、状态表

3 状态编码（分配）——>获得状态转移表

4 状态转移表
触发器特征 } ——>触发器激励表

5 卡诺图化简——> { 激励（输入）函数表达式
输出函数表达式

6 电路实现　　　7 检查无关状态

状态分配（编码）

化简110序列检测器的原始状态表，并用JK触发器实现。

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1	
S_0	S_0 / 0	S_1 / 0	√
S_1	S_0 / 0	S_2 / 0	
S_2	S_3 / 1	S_2 / 0	
S_3	S_0 / 0	S_1 / 0	√

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1
S_0	S_0 / 0	S_1 / 0
S_1	S_0 / 0	S_2 / 0
S_2	S_0 / 1	S_2 / 0

状态分配（$S—Y_2Y_1$）：

S_0 —— 00

S_1 —— 10

S_2 —— 11

输入 X	现态 Y_2^n	现态 Y_1^n	次态 Y_2^{n+1}	次态 Y_1^{n+1}	触发器 J_2	K_2	J_1	K_1	输出 Z
0	0	0	0	0	0	X	0	X	0
0	1	0	0	0	X	1	0	X	0
0	1	1	0	0	X	1	X	1	1
1	0	0	1	0	1	X	0	X	0
1	1	0	1	1	X	0	1	X	0
1	1	1	1	1	X	0	X	0	0
0	0	1	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X

X \ $Y_2^nY_1^n$	00	01	11	10
0	0	X	X	0
1	0	X	X	1

$J_1 = XY_2^n$

X \ $Y_2^nY_1^n$	00	01	11	10
0	X	X	1	X
1	X	X	0	X

$K_1 = \overline{X}$

X \ $Y_2^nY_1^n$	00	01	11	10
0	0	X	1	0
1	0	X	0	0

$Z = \overline{X}Y_1^n$

X \ $Y_2^nY_1^n$	00	01	11	10
0	0	X	X	X
1	1	X	X	X

$J_2 = X$

X \ $Y_2^nY_1^n$	00	01	11	10
0	X	X	1	1
1	X	X	0	0

$K_2 = \overline{X}$

状态分配（编码）

分配方案(1)

S_0 —— 00

S_1 —— 10

S_2 —— 11

简单

$$\begin{cases} J_1 = XY_2^n \\ K_1 = \overline{X} \\ J_2 = X \\ K_2 = \overline{X} \\ Z = \overline{X}Y_1^n \end{cases}$$

分配方案(2)

S_0 —— 00

S_1 —— 11

S_2 —— 10

复杂

$$\begin{cases} J_1 = X\overline{Y_2^n} \\ K_1 = 1 \\ J_2 = X \\ K_2 = \overline{X}+\overline{Y_1^n} \\ Z = \overline{X}\, Y_2^n\, \overline{Y_1^n} \end{cases}$$

电路实现代价与状态分配密切相关

状态分配

需要解决两个问题：

①确定需要的触发器数量K

$$2^{K-1} \leq N \leq 2^K$$

K —— 触发器数量

N —— 最简状态数量

② 为每一个状态分配二进制编码

力图获得一个最小代价的实现方案

状态分配（编码）规则

1.同一输入下，相同的次态所对应的**现态**应该给予相邻编码

2.同一现态在不同输入下所对应的**次态**应给予相邻编码

3.给定输入下，输出完全相同，**现态**编码应相邻

● 是一种经验法，目的是尽量使卡诺图中更多的1（或0）相邻

注意：

- 初始状态一般可以放在卡诺图的 0号单元格里
- 优先满足规则1和规则2
- 状态编码尽量按照相邻原则给予
- 对于多输出函数,规则3可以适当调高优先级

状态分配（编码）规则——例子

- 规则1：同一输入，次态相同，现态编码应相邻

x=0时, 次态 (c,c) →现态 a,b
x=1时, 次态 (d,d) →现态 a,c
ab,ac 应相邻

现态 Q^n	Q^{n+1}/ Z	
	X=0	X=1
a	c / 0	d / 0
b	c / 0	a / 0
c	b / 0	d / 0
d	a / 1	b / 1

- 规则2：同一现态对应的次态应给予相邻编码

现态		次态
a	→	(c,d)
b	→	(c,a)
c	→	(b,d)
d	→	(a,b)

cd,ca,bd,ab应相邻

- 规则3：输出相同，现态编码应相邻

现态	输出
a ,b ,c	0

ab,ac,bc应相邻

(a,b), (a,c) 应相邻, 满足规则1,2,3

a——00, b——01
c—— 10, d——11

	0	1
0	a	b
1	c	d

规 则

1.同一输入下，相同的次态所对应的现态应该给予相邻编码
2.同一现态在不同输入下所对应的次态应给予相邻编码
3.给定输入下，输出完全相同，现态编码应相邻

很难找到一个最佳的状态分配方案

利用触发器设计时序逻辑的方法

1 根据需求——> 获得原始状态图、状态表

2 最小化状态图、状态表

3 状态编码（分配）——>获得状态转移表

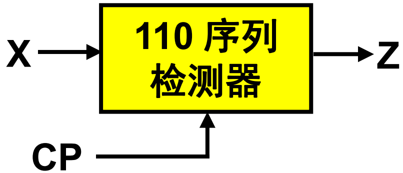
4 状态转移表
触发器特征
——>触发器激励表

5 卡诺图化简——>
激励（输入）函数表达式
输出函数表达式

6 电路实现　　7 检查无关状态（是否可自启动）

完整电路设计过程示例

例：利用JK触发器设计110序列检测器



X:	0	1	1	0	1	1	1	0	1
Z:	0	0	0	1	0	0	0	1	0

1. 获得原始状态图和原始状态表

（1）状态设定

S_0——初始状态，表示收到1位数据：“0”

S_1——表示收到1位数据：“1”

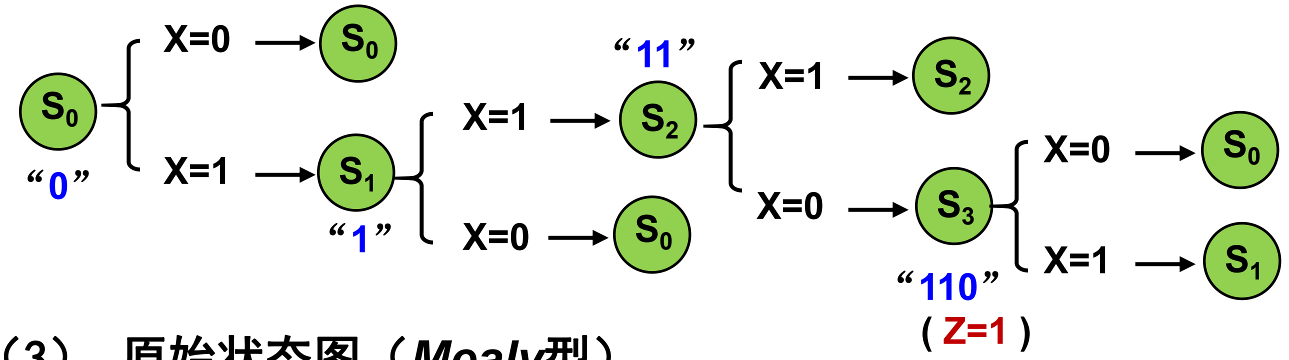
S_2——表示收到2位数据：“11”

S_3——表示收到3位数据：“110”，此时输出标志 Z=1.

只标记感兴趣的子串

利用JK触发器设计110序列检测器

（2）分析状态转换情况



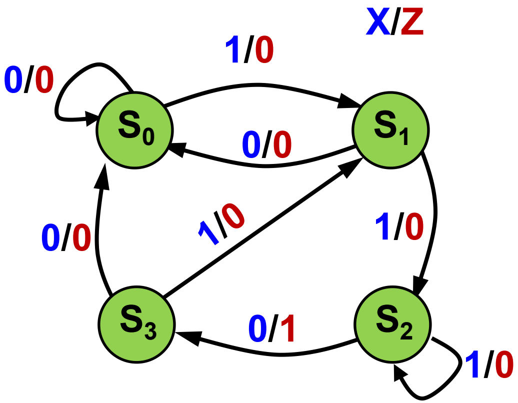
S_0： 0

S_1： 1

S_2： 11

S_3： 110, 且Z=1.

（3） 原始状态图（*Mealy*型）



（4） 原始状态表

现态 Q^n	Q^{n+1}/ Z	
	X=0	X=1
S_0	S_0 / 0	S_1 / 0
S_1	S_0 / 0	S_2 / 0
S_2	S_3 / 1	S_2 / 0
S_3	S_0 / 0	S_1 / 0

利用JK触发器设计110序列检测器

2. 状态化简

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1	
S_0	S_0 / 0	S_1 / 0	√
S_1	S_0 / 0	S_2 / 0	
S_2	S_3 / 1	S_2 / 0	
S_3	S_0 / 0	S_1 / 0	√

现态 Q^n	Q^{n+1}/ Z X=0	Q^{n+1}/ Z X=1
S_0	S_0 / 0	S_1 / 0
S_1	S_0 / 0	S_2 / 0
S_2	S_0 / 1	S_2 / 0

3. 状态分配

使用2个JK触发器

y_2y_1

S_0 —— 00

S_1 —— 10

S_2 —— 11

JK触发器驱动表

Q_n → Q_{n+1}	J	K
0 → 0	0	X
0 → 1	1	X
1 → 0	X	1
1 → 1	X	0

4. 状态转换真值表

J_2 K_2： 看$Q_2^n \rightarrow Q_2^{n+1}$
J_1 K_1： 看$Q_1^n \rightarrow Q_1^{n+1}$

输入 X	现态 Y_2^n	现态 Y_1^n	次态 Y_2^{n+1}	次态 Y_1^{n+1}	触发器 J_2	触发器 K_2	触发器 J_1	触发器 k_1	输出 Z
0	0	0	0	0	0	X	0	X	0
0	1	0	0	0	X	1	0	X	0
0	1	1	0	0	X	1	X	1	1
1	0	0	1	0	1	X	0	X	0
1	1	0	1	1	X	0	1	X	0
1	1	1	1	1	X	0	X	0	0
0	0	1	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X

规 则

1.同一输入下，相同的次态所对应的现态应该给予相邻编码

2.同一现态在不同输入下所对应的次态应给予相邻编码

3.给定输入下，输出完全相同，现态编码应相邻

利用JK触发器设计110序列检测器——续

4. 状态转换真值表

输入	现态		次态		触发器				输出
X	Y_2^n	Y_1^n	Y_2^{n+1}	Y_1^{n+1}	J_2	K_2	J_1	k_1	Z
0	0	0	0	0	0	X	0	X	0
0	1	0	0	0	X	1	0	X	0
0	1	1	0	0	X	1	X	1	1
1	0	0	1	0	1	X	0	X	0
1	1	0	1	1	X	0	1	X	0
1	1	1	1	1	X	0	X	0	0
0	0	1	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X

5. 卡诺图化简

$X \backslash Y_2^nY_1^n$	00	01	11	10
0	0	X	X	X
1	1	X	X	X

$J_2 = X$

$X \backslash Y_2^nY_1^n$	00	01	11	10
0	X	X	1	1
1	X	X	0	0

$K_2 = \overline{X}$

$X \backslash Y_2^nY_1^n$	00	01	11	10
0	0	X	X	0
1	0	X	X	1

$J_1 = XY_2^n$

$X \backslash Y_2^nY_1^n$	00	01	11	10
0	X	X	1	X
1	X	X	0	X

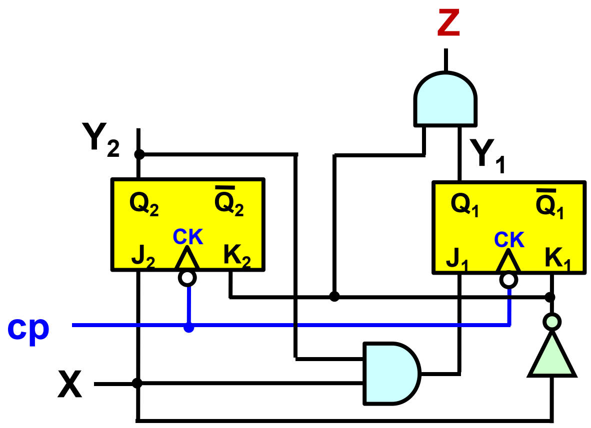
$K_1 = \overline{X}$

$X \backslash Y_2^nY_1^n$	00	01	11	10
0	0	X	1	0
1	0	X	0	0

$Z = \overline{X}Y_1^n$

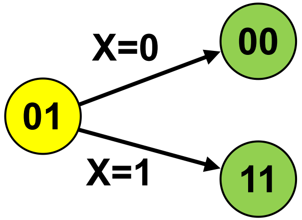
利用JK触发器设计110序列检测器——续

6. 电路实现



7. 检查是否可以自启动

$$\begin{cases} J_1 = XY_2^n \\ K_1 = \overline{X} \\ J_2 = X \\ K_2 = \overline{X} \end{cases} \Rightarrow \begin{cases} Y_1^{n+1} = XY_2^n\overline{Y_1^n} + XY_1^n \\ \qquad\;\; = X(Y_1^n + Y_2^n) \\ Y_2^{n+1} = X\overline{Y_2^n} + XY_2^n \\ \qquad\;\; = X \end{cases}$$



电路可以自启动

利用触发器设计时序逻辑的方法

1 根据需求——> 获得原始状态图、状态表

2 最小化状态图、状态表

3 状态编码（分配）——>获得状态转移表

4 状态转移表
触发器特征
——>触发器激励表

5 卡诺图化简——>
激励（输入）函数表达式
输出函数表达式

6 电路实现 7 检查无关状态