第六讲 有限状态机
Finite State Machine (FSM)

佟冬

tongdong@pku.edu.cn

微处理器研究开发中心（MPRC）

计算机科学技术系

北京大学

时序逻辑电路回顾

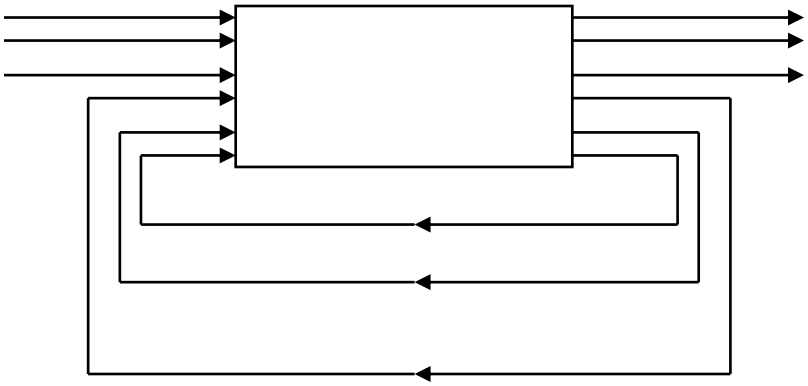
- 实现电路状态的基本模块
  - 锁存器和触发器
  - R-S锁存器, R-S主从触发器, D 主从触发器, D边沿触发器
- 电路定时方法Timing methodologies
  - 使用时钟
  - 级联触发器正确工作的条件
  - 关注时钟扭斜
- 异步输入及其危害性
  - 同步器
- 基本的寄存器
  - 移位寄存器shift registers
  - 计数器counters
- 硬件描述语言和时序逻辑电路

有限状态机Finite State Machines

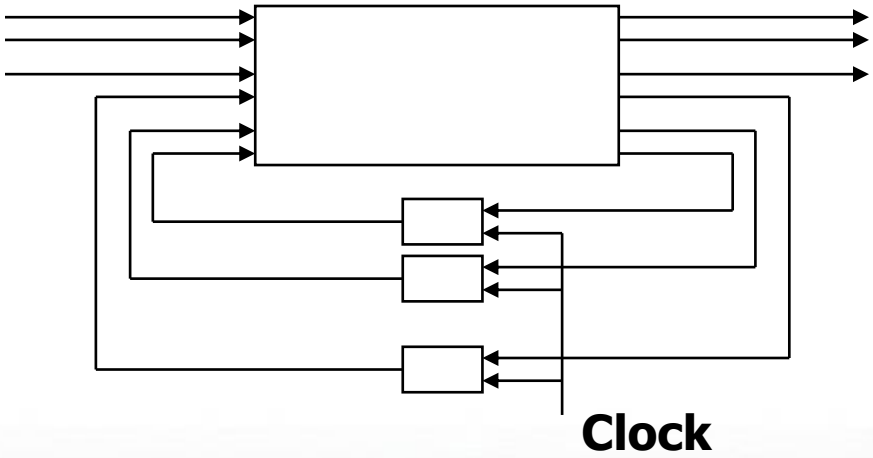
- 时序电路
  - 基本时序单元：锁存器、触发器、寄存器
  - 组合电路
- 描述序电路的模型
  - 有限状态机finite-state machines (Moore and Mealy)
- 基本的时序电路
  - 移位寄存器
  - 计数器
- 设计流程
  - 状态图/状态表
  - 状态转换表
  - 次态/输出函数
- 硬件描述语言

同步时序电路模型

- 同步(Synchronous)时序电路
  - 时序电路中状态的改变由系统统一时钟控制
- 异步(Asynchronous)时序电路
  - 时序电路中状态的改变不受统一时钟的控制，由输入变化引起改变



时序电路模型

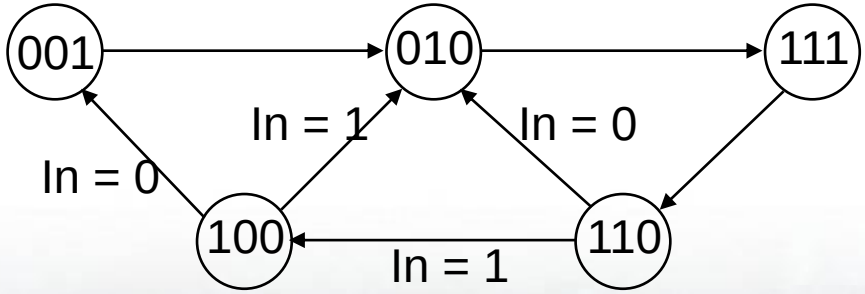


同步时序电路模型

有限状态机的表示

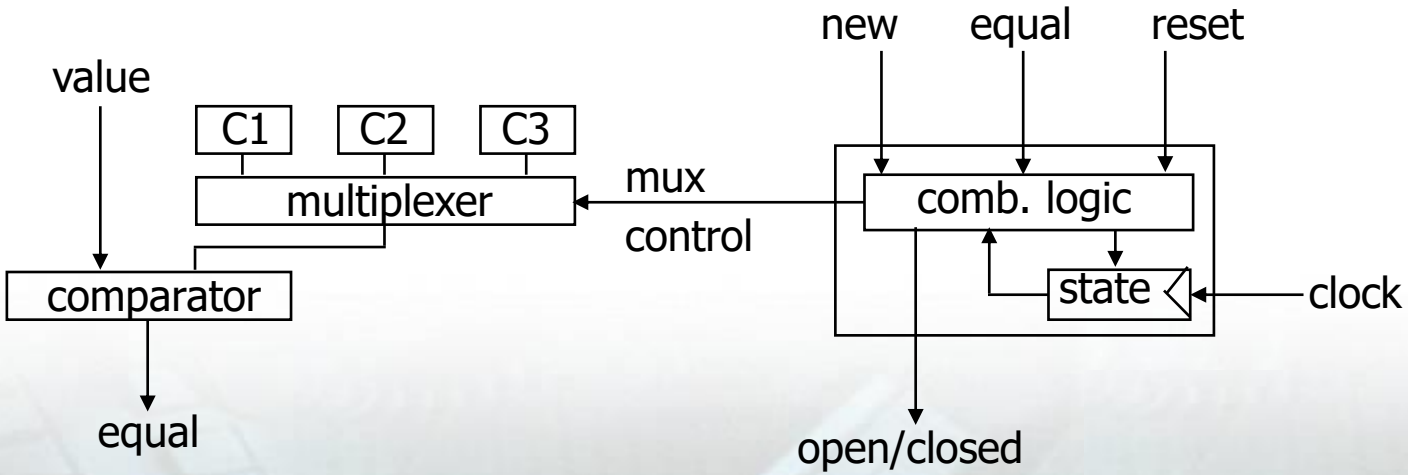
- 状态state：由时序存储原件的可能取值确定
- 转换Transitions：状态的改变
- 时钟Clock：控制状态原件改变状态的控制信号

- 时序电路
  - 由有一系列状态组成的状态序列
  - 基于输入信号的值改变的状态序列
  - 序列时钟周期定义为状态序列变化的时间单位



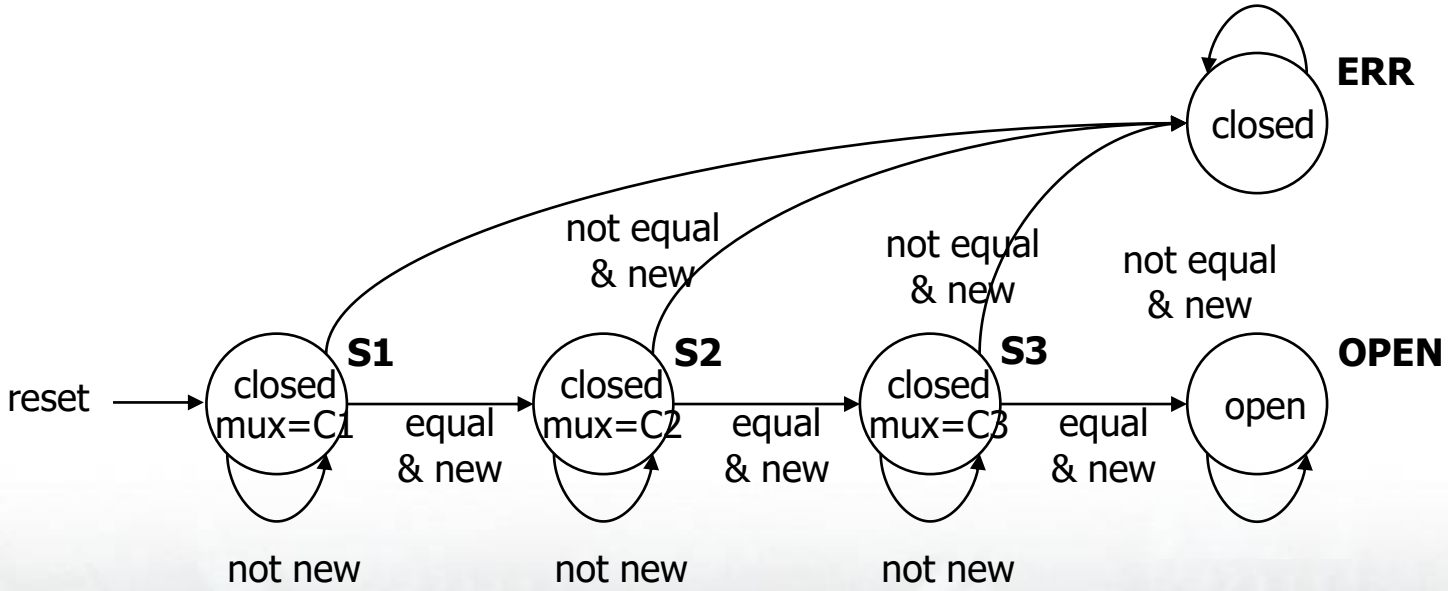
课程回顾：时序电路

- 带反馈（feedback）的电路
  - outputs = f(inputs, past inputs, past outputs)
  - 在逻辑电路中构造“记忆”的基础
  - 门组合锁例子
    - 状态是记忆
    - 状态是组合逻辑的输出和输入
    - 存储单元的组合



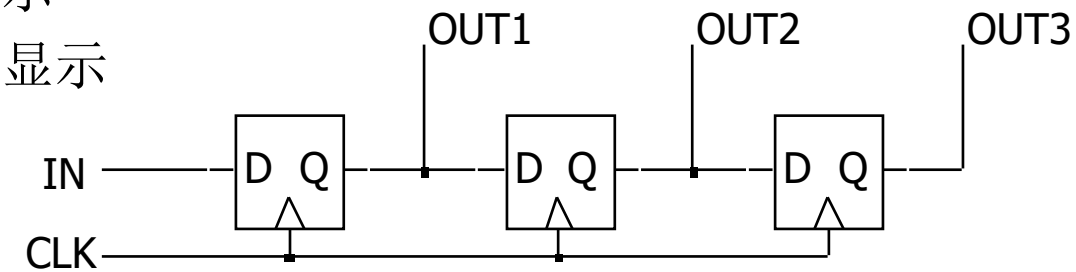
有穷状态自动机FSM：举例

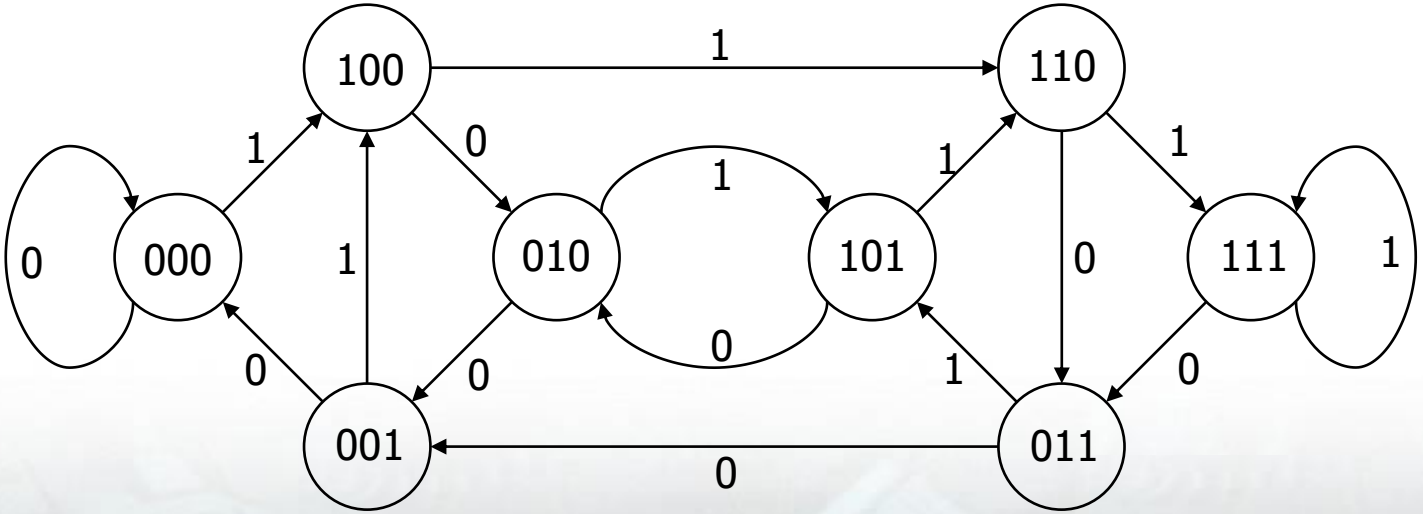
- 组合锁
  - 5个状态
  - 5个状态自转换
  - 6 个状态间转换
  - 1个复位reset转换到状态S1



任何时序电路都能表示为状态图吗？

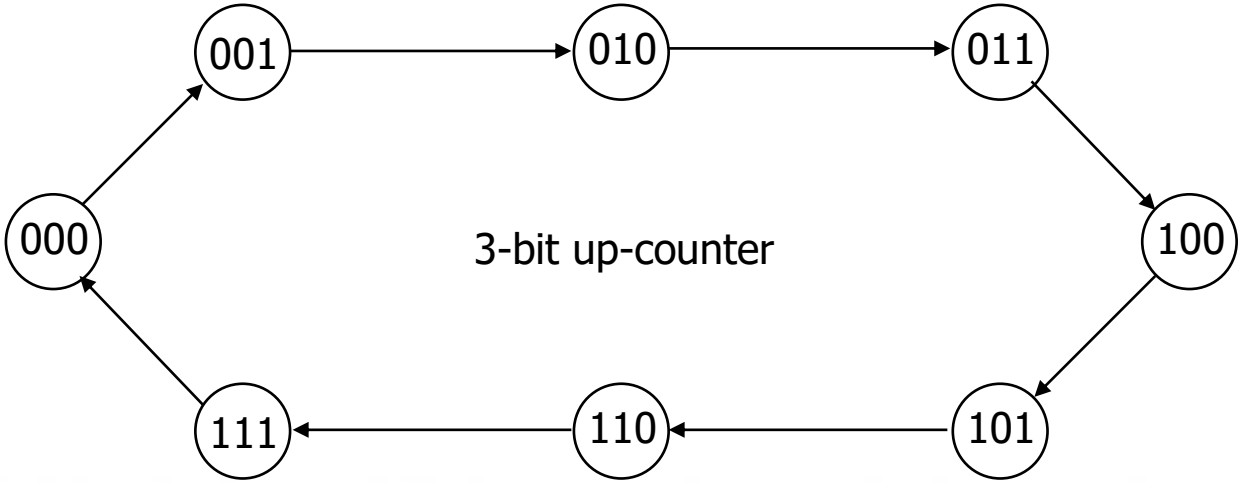
- 移位寄存器
  - 输入值在转换线上显示
  - 输出值在状态节点中显示





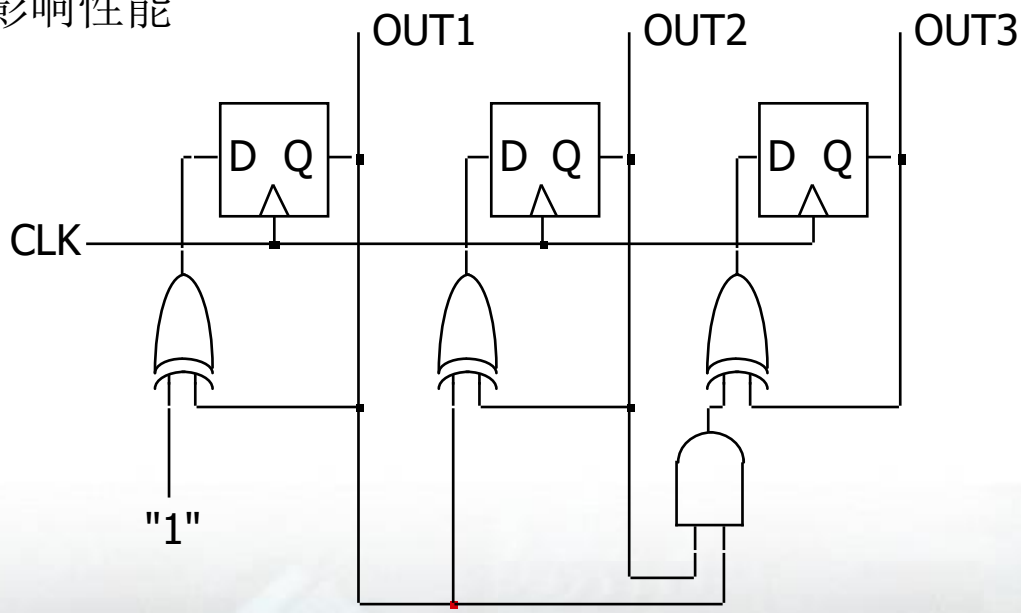
计数器是简单的有限状态机

- 计数器
  - proceed through well-defined sequence of states in response to enable
- 多种类型计数器：二进制计数器，BCD计数器，格雷码计数器
  - 3位向上计数器：000, 001, 010, 011, 100, 101, 110, 111, 000, ...
  - 3位向下计数器：111, 110, 101, 100, 011, 010, 001, 000, 111, ...



将状态图转换成逻辑电路

- 计数器
  - 3个触发器保存状态
  - 逻辑电路计算次态
  - 时钟信号控制何时触发器存储值变化
    - 等待足够长的时间确保组合电路计算出正确的新值
    - 等待时间不能太长，会影响性能

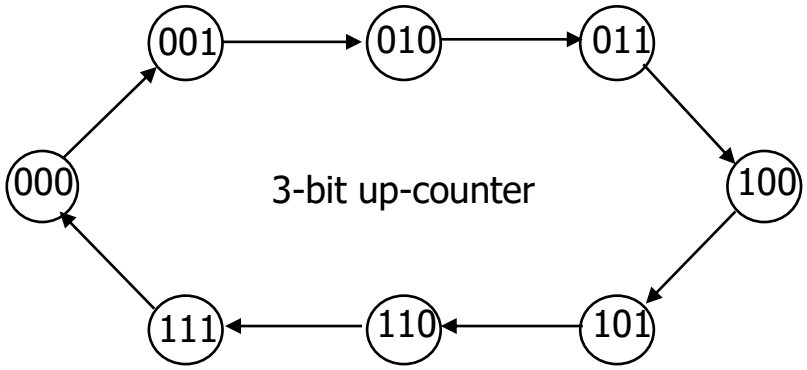


FSM设计流程

- 从简单计数器开始
  - 输出值就是状态值本身
  - 次态计算不依赖输入信号
- 状态图到状态转换表
  - 状态图的表格形式
  - 类似真值表
- 状态编码
  - 确定状态的二进制表示
  - 计数器的状态就是计数器的值
- 电路实现
  - 每个状态位一个触发器
  - 基于译码器的组合电路

FSM设计流程：状态图到状态转换表

- 状态图的表格形式
- 类似真值表（指定输入组合和输出的对应关系）
- 状态的编码：对于计数器很简单，就是存储的值



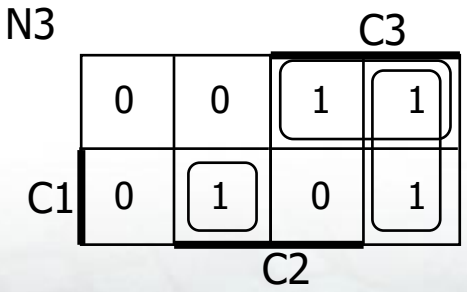
present state		next state	
0	000	001	1
1	001	010	2
2	010	011	3
3	011	100	4
4	100	101	5
5	101	110	6
6	110	111	7
7	111	000	0

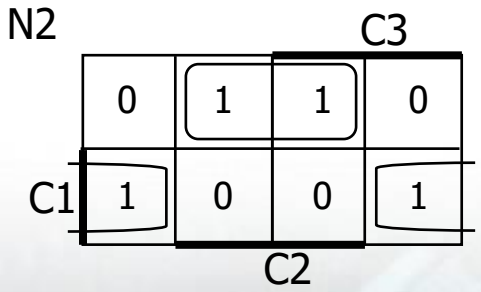
电路实现

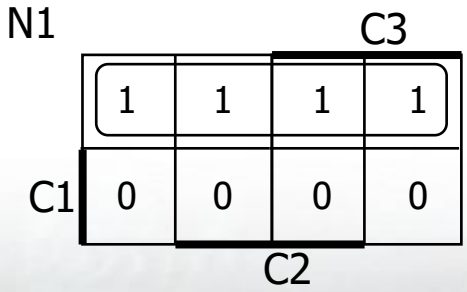
- 每个状态位一个D触发器
- 基于编码设计组合电路

C3	C2	C1	N3	N2	N1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

N1 <= C1′
N2 <= C1C2′ + C1′C2
<= C1 xor C2
N3 <= C1C2C3′ + C1′C3 + C2′C3
<= (C1C2)C3′ + (C1′ + C2′)C3
<= (C1C2)C3′ + (C1C2)′C3
<= (C1C2) xor C3



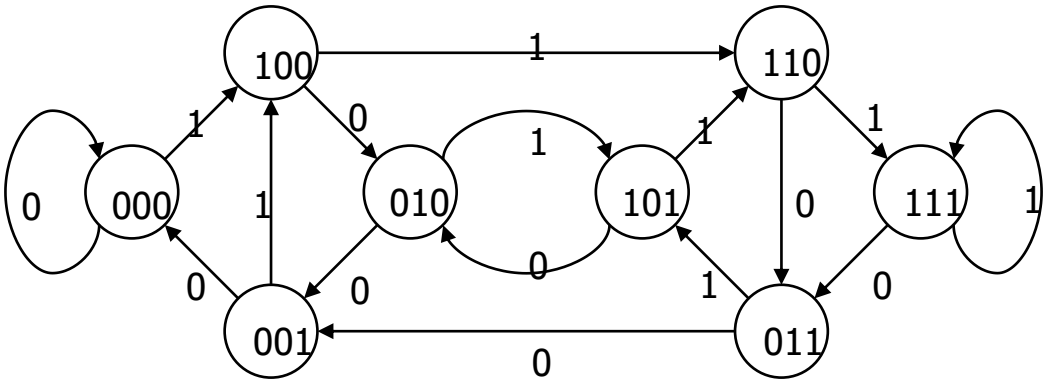




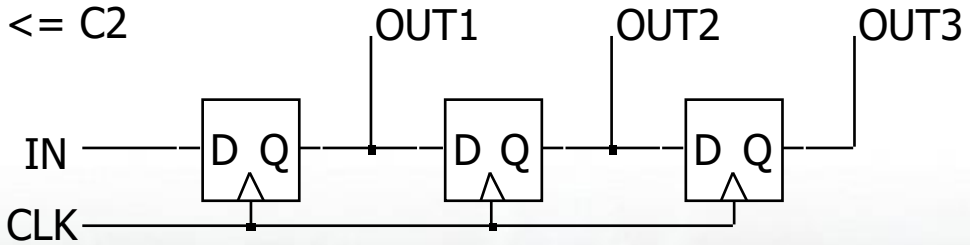
移位寄存器

输入决定次态

In	C1	C2	C3	N1	N2	N3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

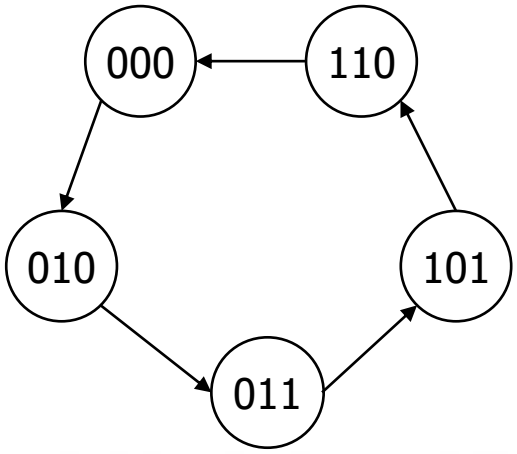


N1 <= In
N2 <= C1
N3 <= C2



复杂计数器和自启动计数器

- 复杂计数器
  - 顺序重复5个状态
  - 非顺序二进制表示
- 第一步：给出状态转换图
  - 计数序列: 000, 010, 011, 101, 110
- 第二步：从状态转换图给出状态转换表



Present State			Next State		
C	B	A	C+	B+	A+
0	0	0	0	1	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	–	–	–
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	–	–	–

注：因为存在无用的状态码，存在无关项。

复杂状态机

- 第三步：次态函数的卡诺图化简

C+

A \ BC	00	01	11	10
0	0	0	0	X
1	X	1	X	1

B+

A \ BC	00	01	11	10
0	1	1	0	X
1	X	0	X	1

A+

A \ BC	00	01	11	10
0	0	1	0	X
1	X	1	X	0

C+ <= A

B+ <= B′ + A′C′

A+ <= BC′

自启动计数器

- 根据化简后的函数重新得到状态转换表

C+

A \ BC	00	01	11	10
0	0	0	0	0
1	1	1	1	1

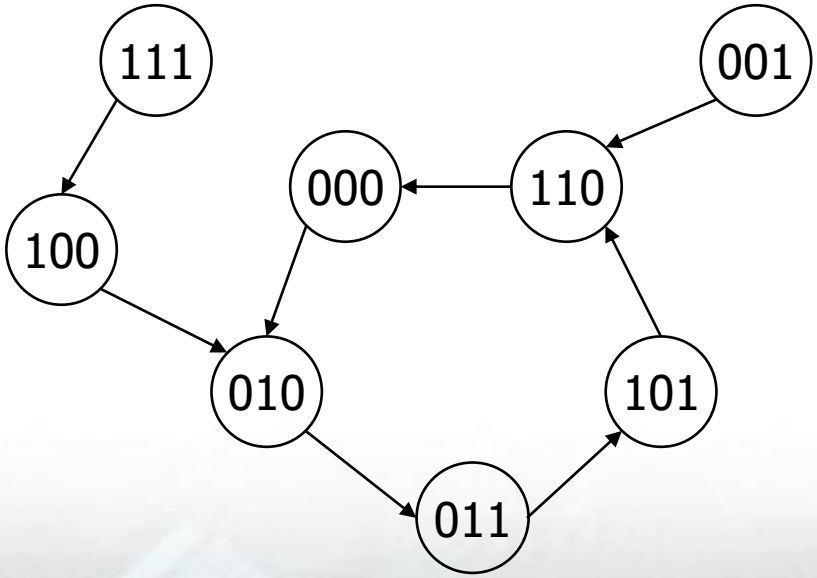
B+

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	1

A+

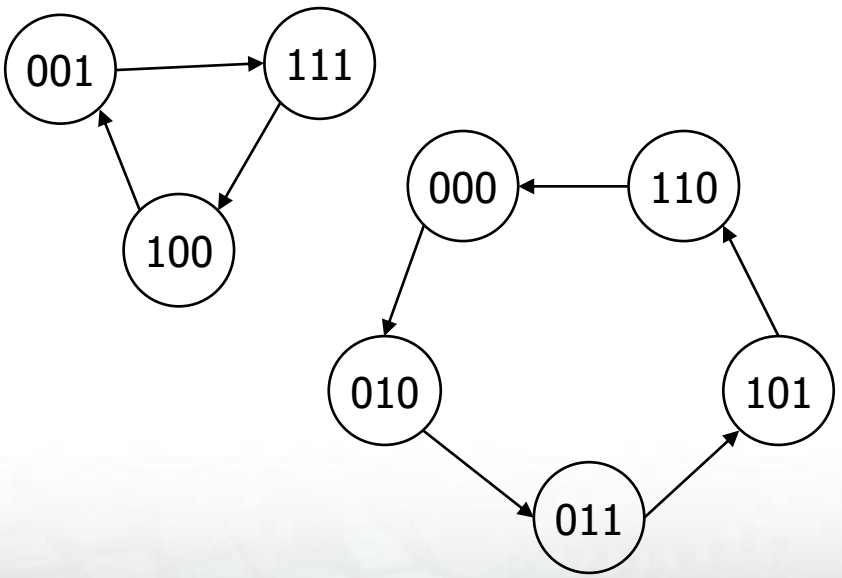
A \ BC	00	01	11	10
0	0	1	0	0
1	0	1	0	0

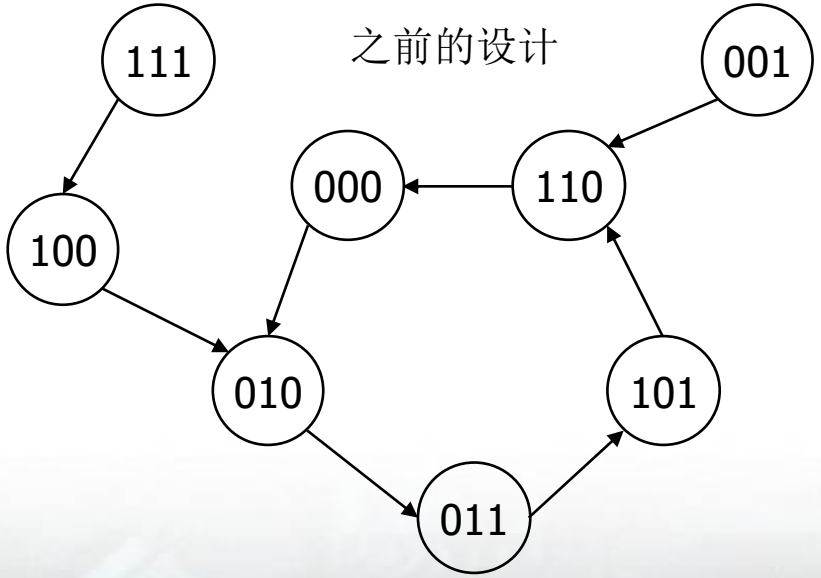
Present State			Next State		
C	B	A	C+	B+	A+
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	1	0	0



自启动计数器

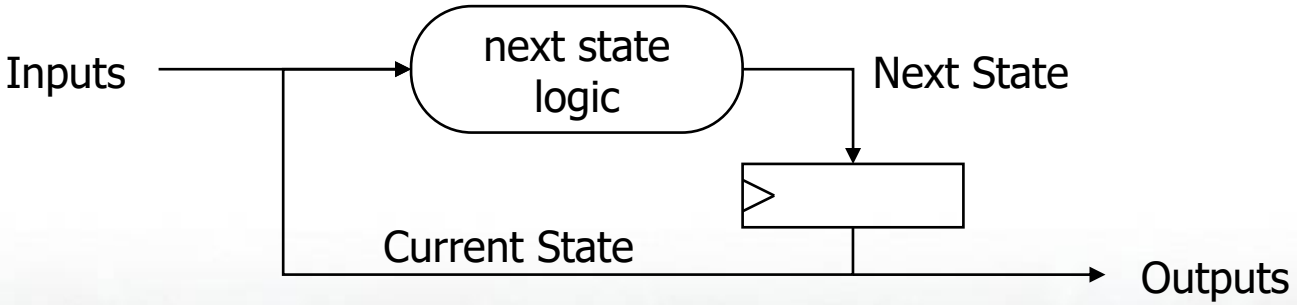
- 起始状态
  - 在上电时，计数器可能处在无用或者无效的状态
  - 设计者必须确保计数器可以进入正确的状态
- 自启动解决方案
  - 设计计数器应该使每个无效状态都能自行进入有效状态
  - 可能会限制对无关项的利用，通常采用全局复位信号





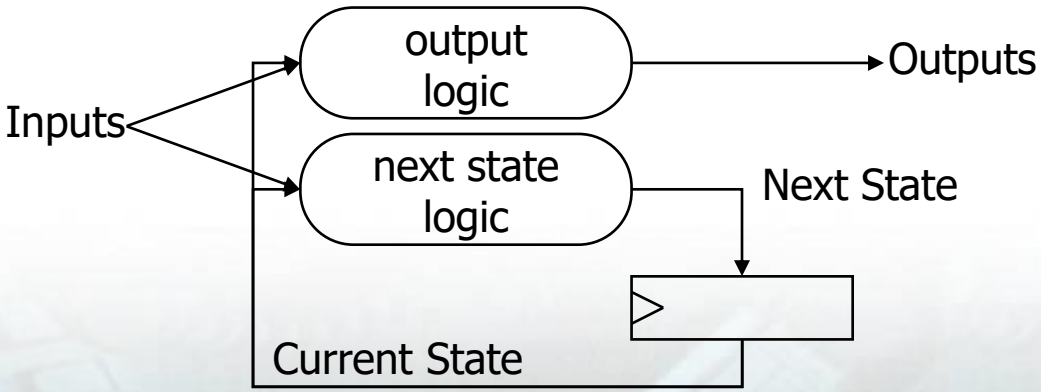
计数器/移位寄存器模型

- 存在寄存器中的值表示电路的状态
- 组合电路计算：
  - 次态（Next State）
    - 先态和输入的函数
  - 输出
    - 各触发器的输出值



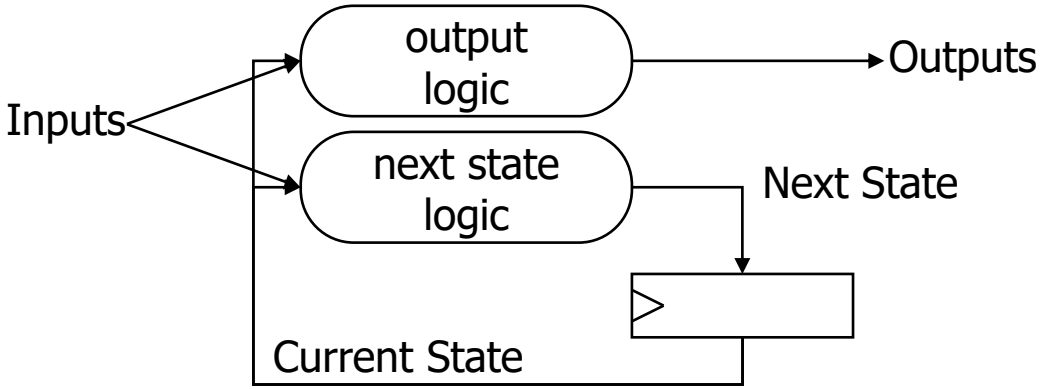
一般的状态机模型

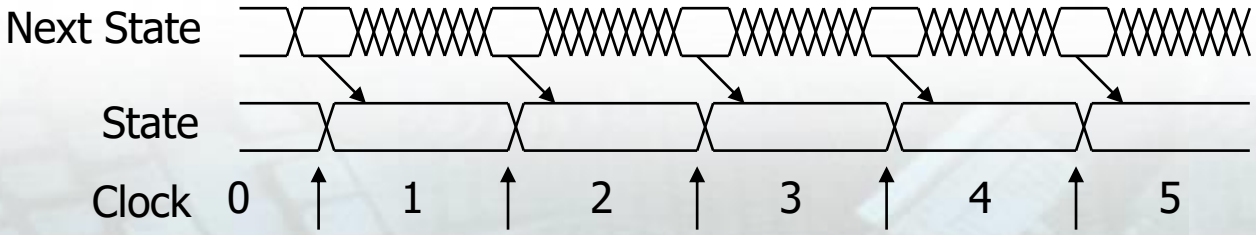
- 存在寄存器中的值表示电路的状态
- 组合电路计算：
  - 次态（Next State）
    - 先态和输入的函数
  - 输出
    - 现态和输入的函数 (Mealy机)
    - 现态的函数 (Moore机)



状态机模型

- 状态（State）: $S_1, S_2, ..., S_k$
- 输入（Inputs）: $I_1, I_2, ..., I_m$
- 输出（Outputs）: $O_1, O_2, ..., O_n$
- 转换函数（Transition function）: $F_s(S_i, I_j)$
- 输出函数（Output function）: $F_o(S_i)$ or $F_o(S_i, I_j)$



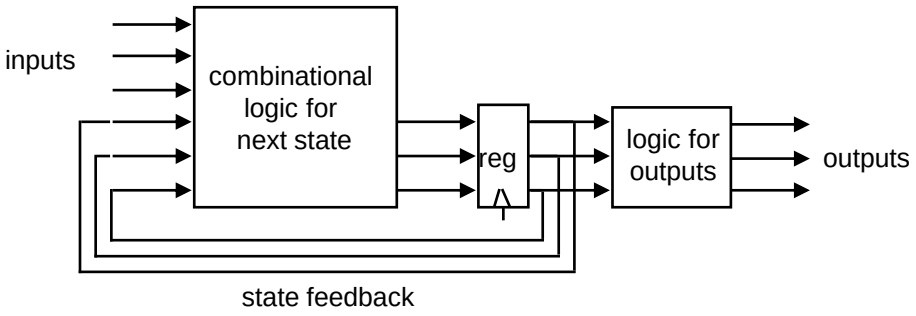


Mealy机和 Moore机的比较

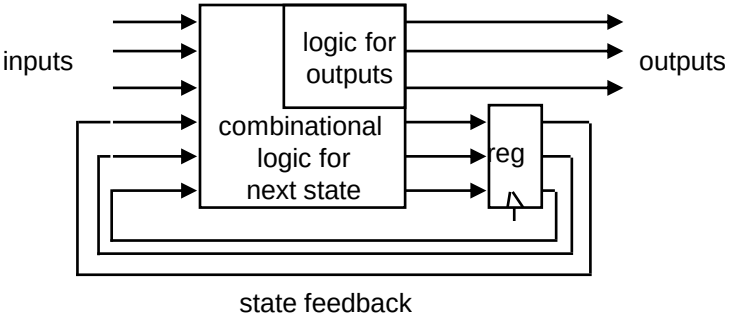
- Mealy机的状态个数较少
- Moore机更安全
  - 输出在时钟边沿变化（在时钟周期后）
  - 在Mealy机中，输入的变化能直接导致输出的变化，当两个状态机互连时非常危险，可能会因为设计得不小心导致异步的组合回路（feed-back）
- Mealy机对输入的反应更快
  - 在同一个周期反应—不需要等待时钟
  - 在Moore中，需要更多的逻辑对状态译码计算输出，时钟沿后需要更多的门延迟

Mealy机和 Moore机的比较

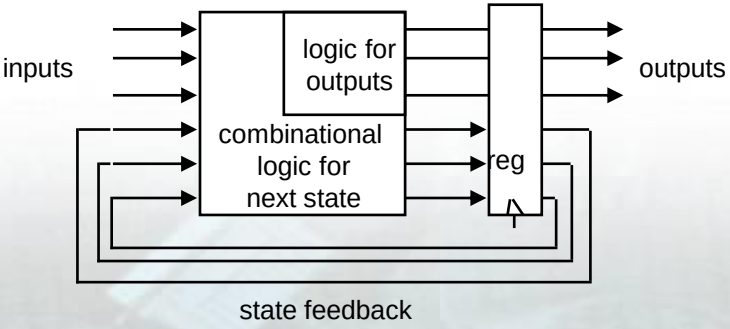
- Moore机



- Mealy机



- 同步Mealy机

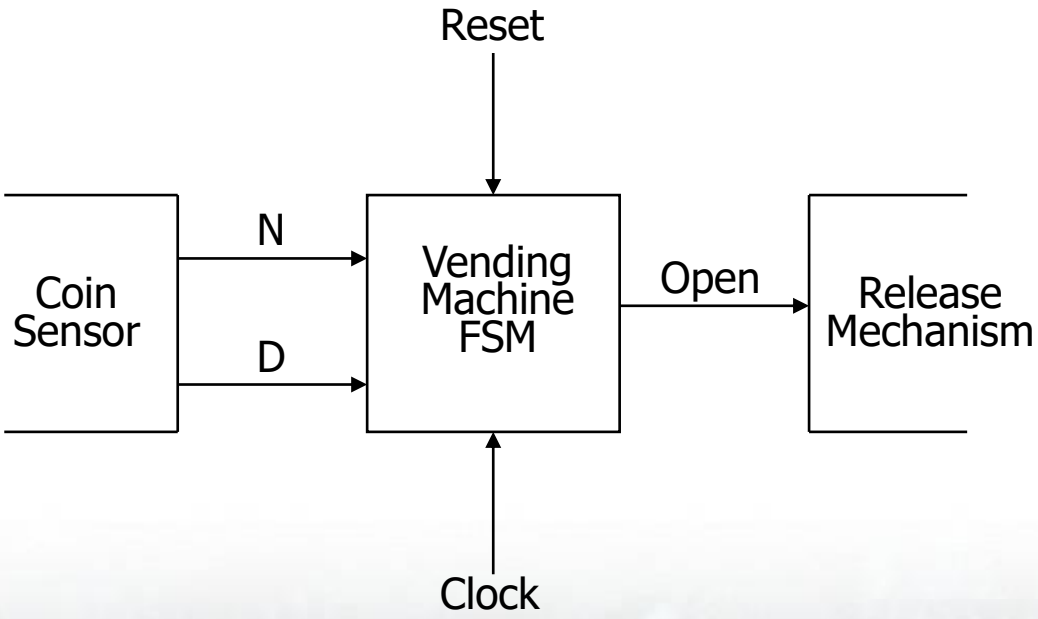


电路的初始状态

- 电路上电后的初始状态需要确定
- 同步时序电路中引入“复位”信号Reset
- 同步时序电路中，上电后先产生一定时间宽度的Reset信号
- Reset信号一般采用异步复位和置位方式，电路系统设置为初始的状态
- Reset信号无效后，系统进入正常工作的状态

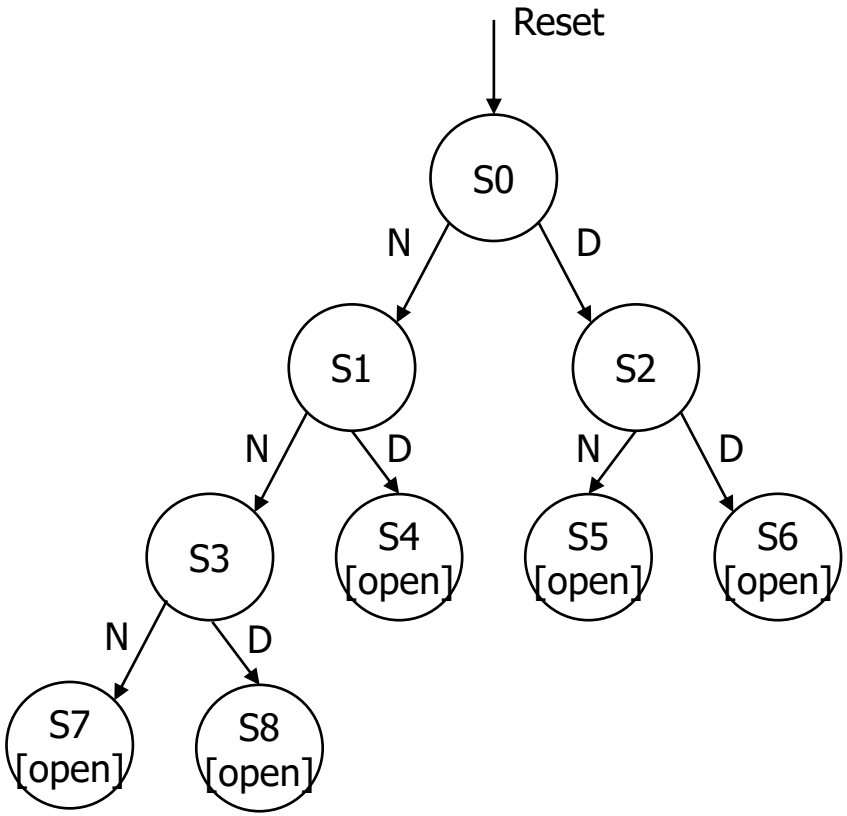
例1：自动售货机

- 投入15分，出一个商品
- 一个投币口，接受10分和5分两种硬币
- 不找零



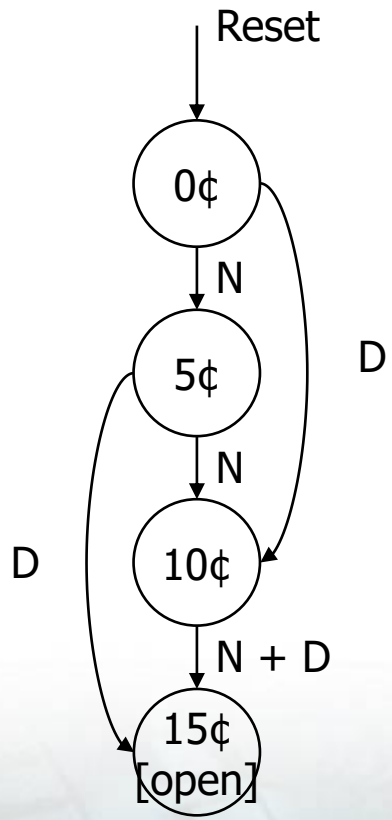
自动售货机（续）

- 适当的抽象表示
  - 列出典型的硬币输入序列：
    - 3个五分
    - 一个五分接着一个十分
    - 一个十分接着一个五分
    - 两个十分
  - 画出状态图：
    - 输入: N, D, reset
    - 输出: open chute
  - 假设：
    - 每个时钟周期已能投入一个5分或者一个10分
    - 如果N = D = 0 （没有硬币）状态不变



自动售货机

- 使状态个数最小，即可能的复用状态



present state	inputs D	N	next state	output open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	–	–
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	–	–
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	–	–
15¢	–	–	15¢	1

symbolic state table

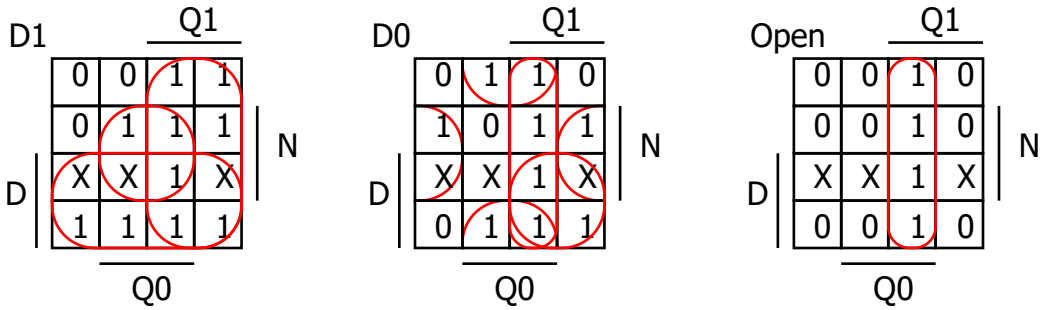
自动售货机

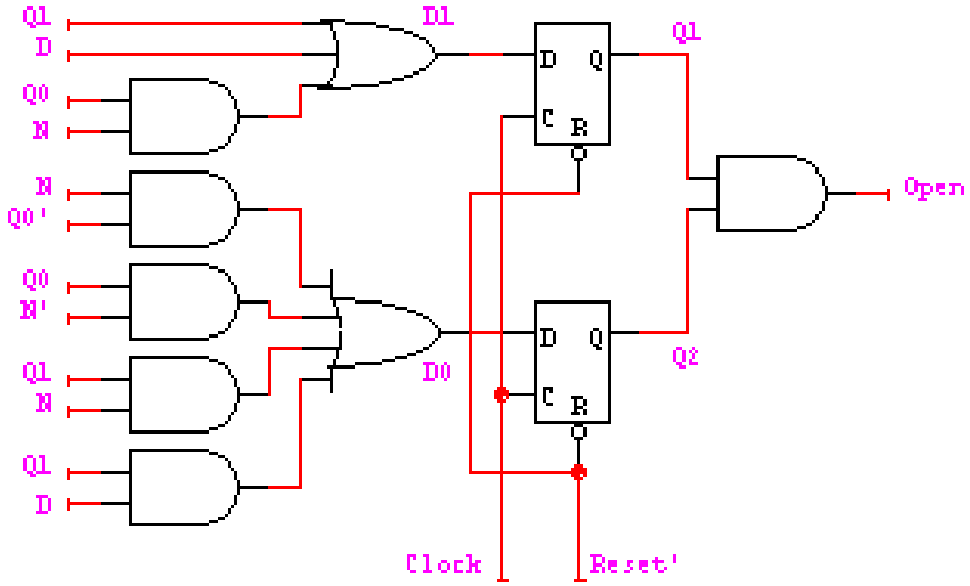
- 状态编码

present state Q1	Q0	inputs D	N	next state D1	D0	output open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	–	–	–
1	1	–	–	1	1	1

例1：Moore机实现

- 映射成逻辑电路





D1 = Q1 + D + Q0 N

D0 = Q0′ N + Q0 N′ + Q1 N + Q1 D

OPEN = Q1 Q0

自动售货机

One-hot编码

present state				inputs		next state				output
Q3	Q2	Q1	Q0	D	N	D3	D2	D1	D0	open
0	0	0	1	0	0	0	0	0	1	0
				0	1	0	0	1	0	0
				1	0	0	1	0	0	0
				1	1	-	-	-	-	-
0	0	1	0	0	0	0	0	1	0	0
				0	1	0	1	0	0	0
				1	0	1	0	0	0	0
				1	1	-	-	-	-	-
0	1	0	0	0	0	0	1	0	0	0
				0	1	1	0	0	0	0
				1	0	1	0	0	0	0
				1	1	-	-	-	-	-
1	0	0	0	-	-	1	0	0	0	1

D0 = Q0 D′ N′

D1 = Q0 N + Q1 D′ N′

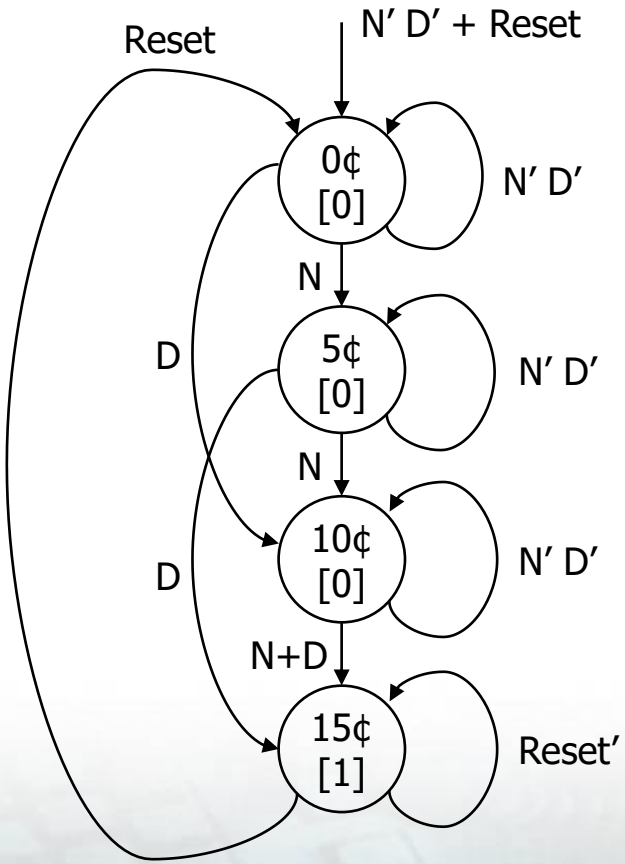
D2 = Q0 D + Q1 N + Q2 D′ N′

D3 = Q1 D + Q2 D + Q2 N + Q3

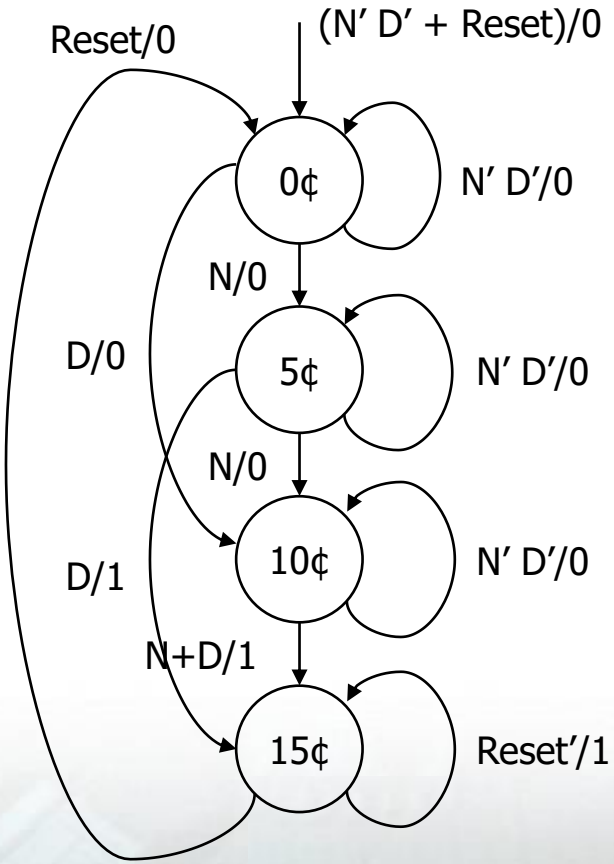
OPEN = Q3

等价的Mealy机和Moore机

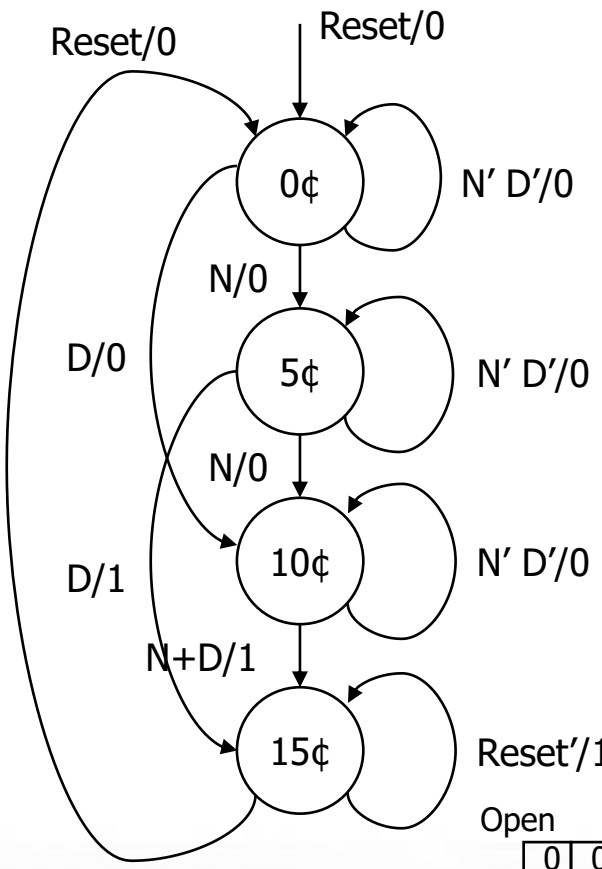
Moore机



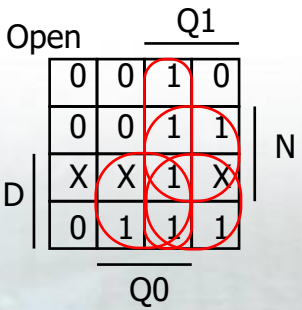
Mealy机



例2：Mealy机实现



present state Q1	Q0	inputs D	N	next state D1	D0	output open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	1
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	1
		1	0	1	1	1
		1	1	–	–	–
1	1	–	–	1	1	1



D0 = Q0′N + Q0N′ + Q1N + Q1D

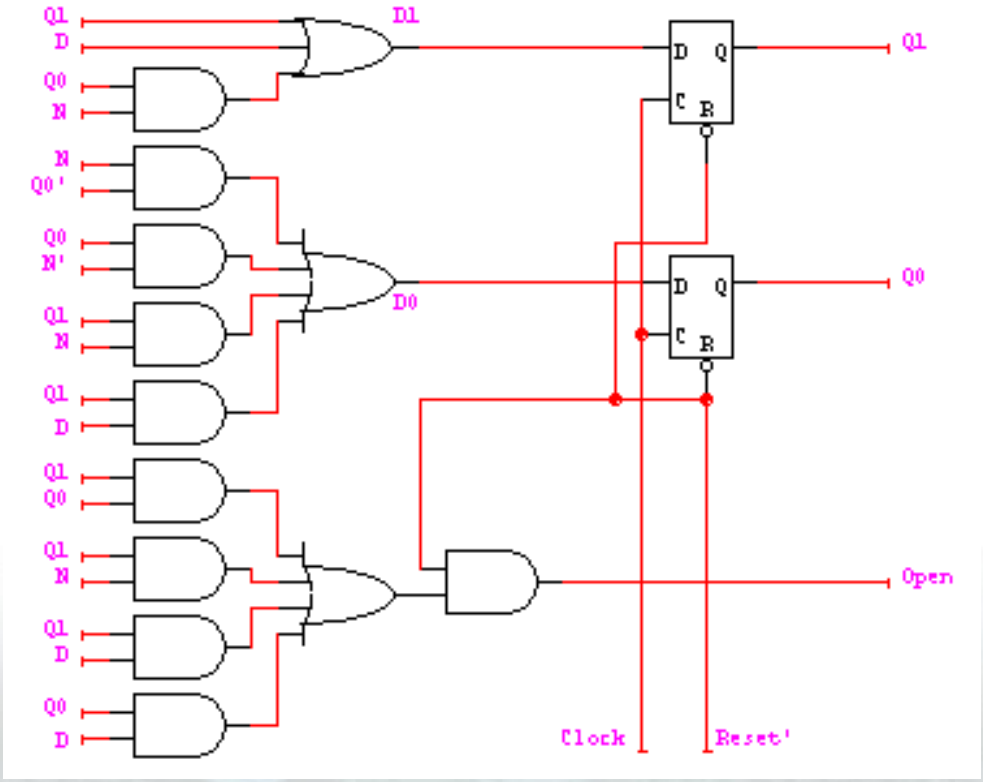
D1 = Q1 + D + Q0N

OPEN = Q1Q0 + Q1N + Q1D + Q0D

例2：Mealy机实现

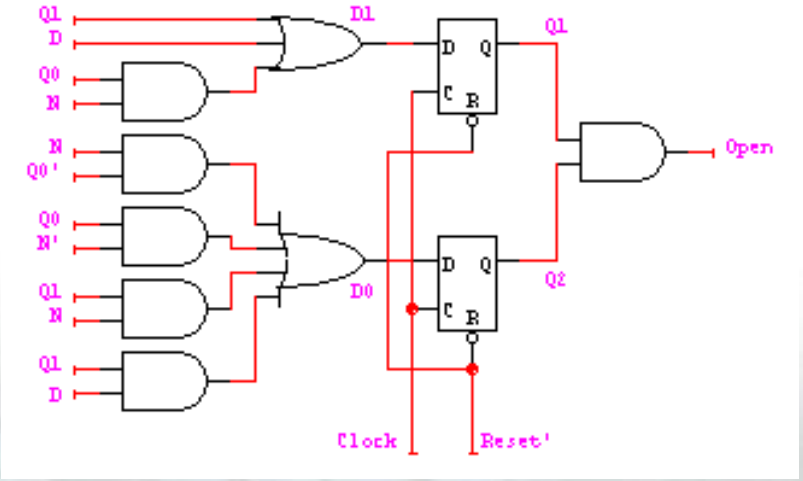
D0 = Q0'N + Q0N' + Q1N + Q1D
D1 = Q1 + D + Q0N
OPEN = Q1Q0 + Q1N + Q1D + Q0D

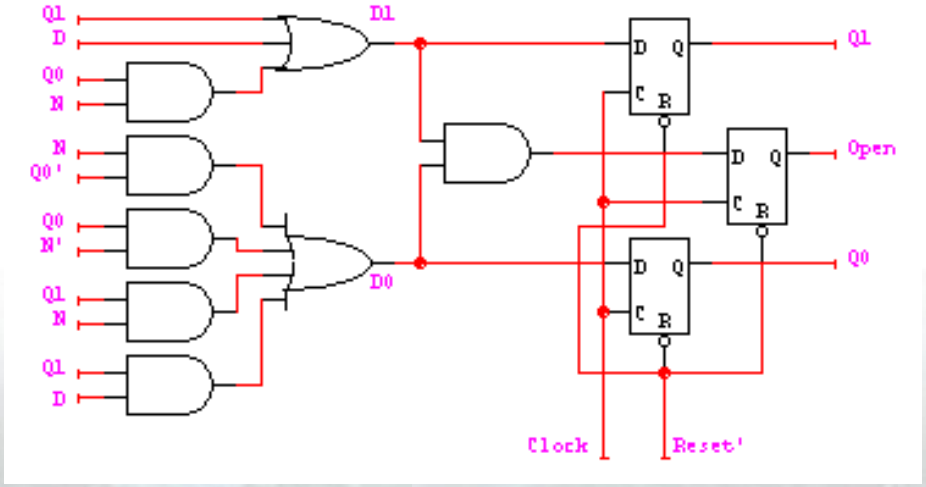
保证当Reset时OPEN为0



售货机Moore机到同步Mealy机

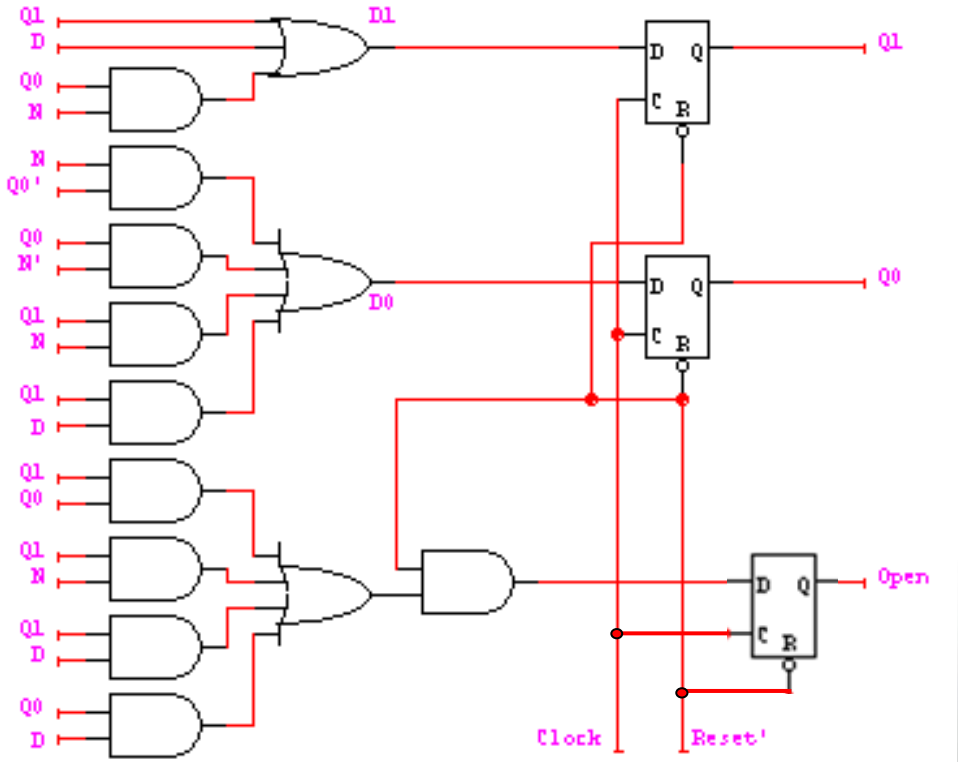
- 在Moore机中OPEN = Q1Q0 产生组合延迟在Q1和Q0变化之后
- 可以通过移动寄存器和逻辑来改善Delay
- OPEN.d = (Q1 + D + Q0N)(Q0'N + Q0N' + Q1N + Q1D)
  = Q1Q0N' + Q1N + Q1D + Q0'ND + Q0N'D
- 这种实现比较象同步的Mealy机



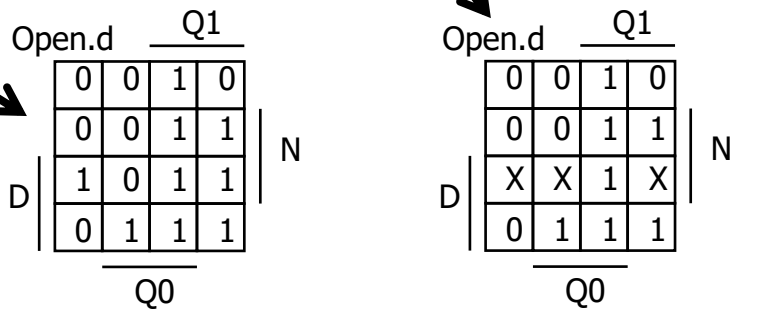


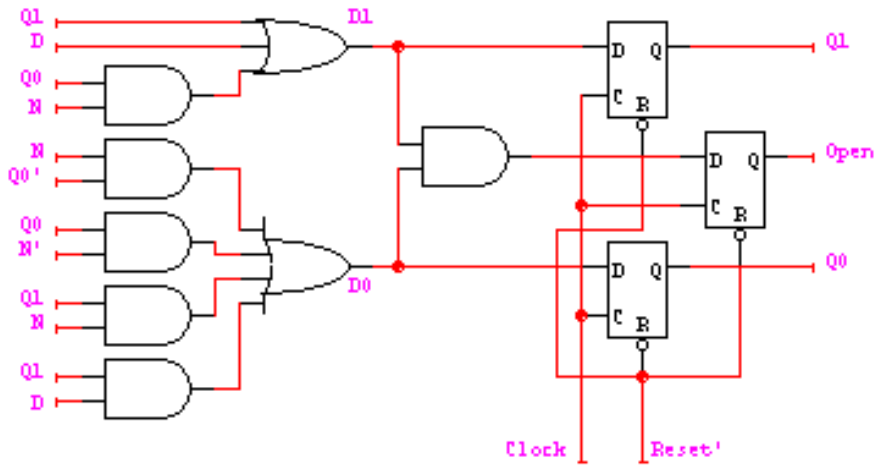
售货机Moore机到同步Mealy机

- OPEN.d = Q1Q0 + Q1N + Q1D + Q0D
- OPEN.d = (Q1 + D + Q0N)(Q0'N + Q0N' + Q1N + Q1D)
  = Q1Q0N' + Q1N + Q1D + Q0'ND + Q0N'D



米利机转同步米利机





摩尔机转同步米利机

有限状态机Finite state machines

- 描述时序电路的模型
  - 时序单元的抽象
  - 有限状态机和状态图
  - 输入和输出
  - 米利机、摩尔机和同步米利机
- 有限状态机的设计流程
  - 给出状态图
  - 给出状态转换表
  - 确定次态和输出函数
  - 实现组合逻辑电路
- 硬件描述语言

例3：有限二进制串模式识别

- 有限二进制串模式识别
  - 一个输入(X) 和一个输出 (Z)
  - 输出有效的条件：在输入序列中发现…010…，当序列…100…出现后停止识别

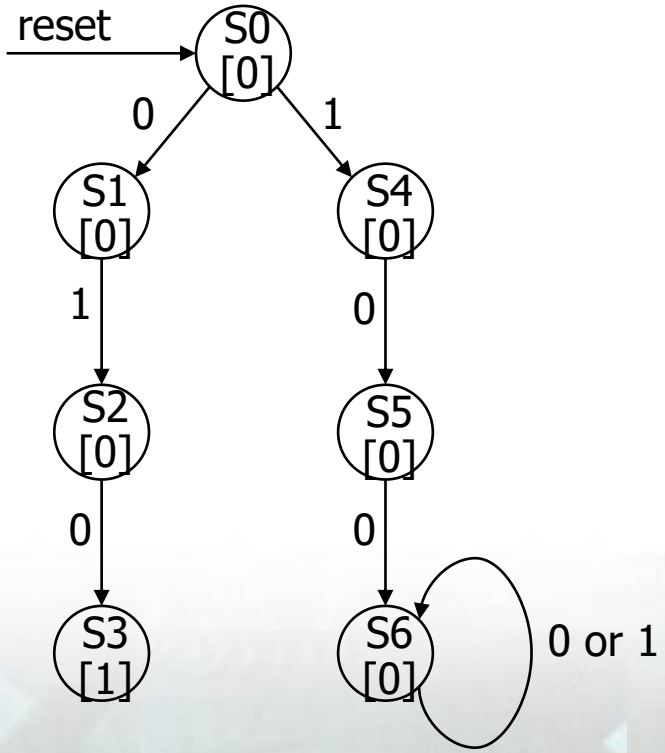
- 第1步：理解问题
  - 简单的输入/输出行为：

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

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

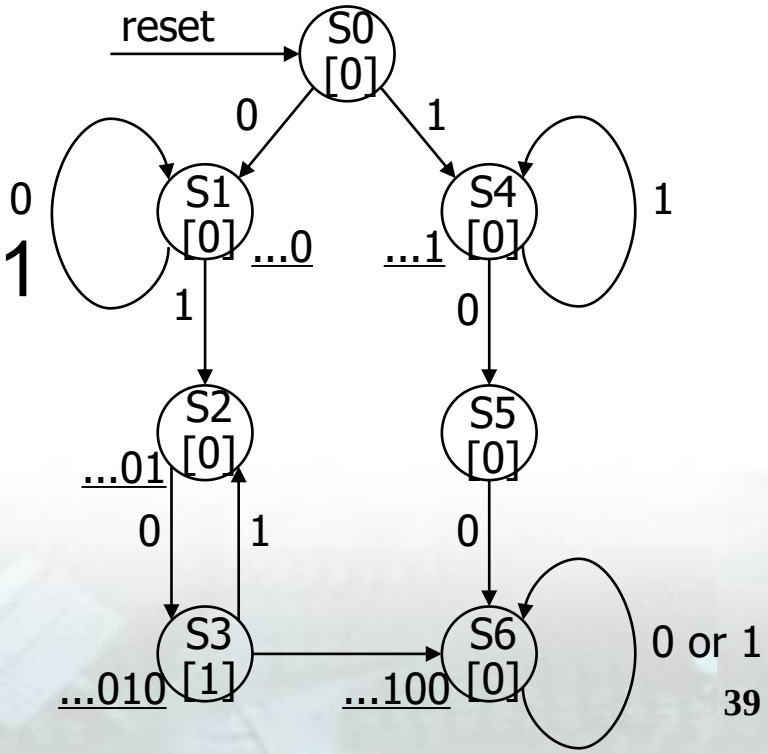
有限二进制串模式识别 (第二步)

- 画状态图
  - 必须识别的二进制串010 and 100
  - Moore机实现



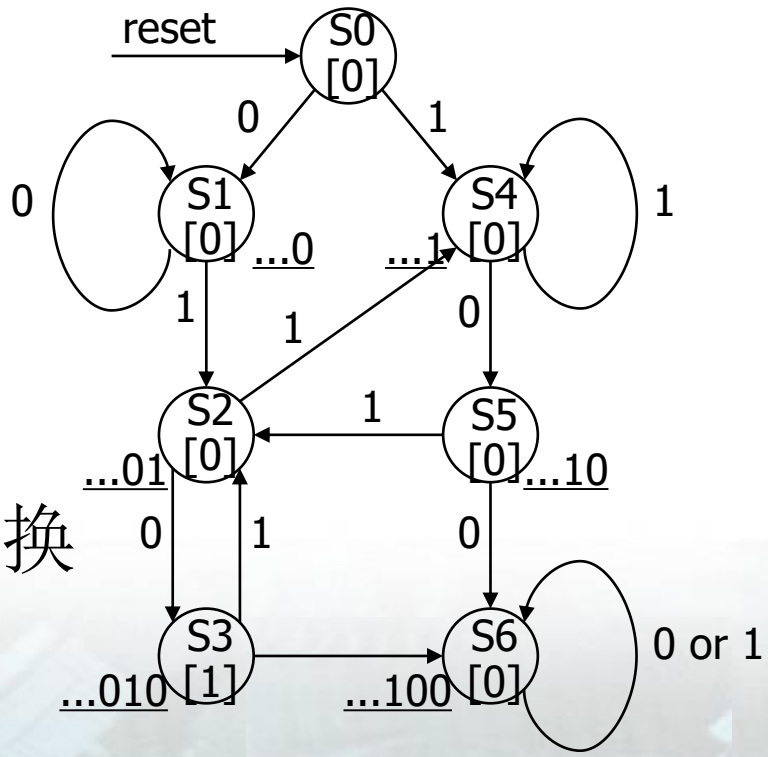
有限二进制串模式识别 (第二步)

- S3的退出条件：识别…010
  - 如果下一个输入时0： …0100 = ...100 （S6）
  - 如果下一个输入时1： …0101 = …01 （S1）
- S1的退出条件：识别多0串…0
  - 没有1出现
  - 如果输入 0，循环跳回S1
- S4的退出条件：识别多1串…1
  - 没有0出现
  - 如果输入1，循环跳回S4



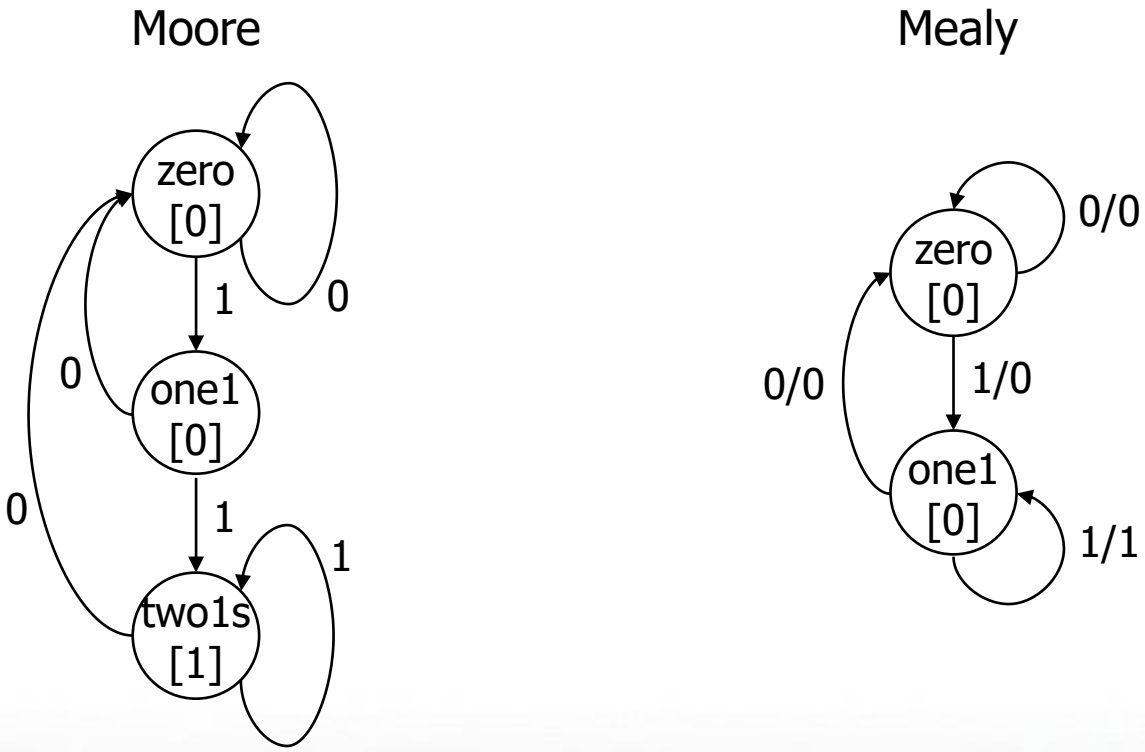
有限二进制串模式识别 (第二步)

- S2 和 S5 还没有完全的跳转
  - S2 = …01；如果下一个输入1
    输入串 (01)1(00)
    S4符合这种情况
  - S5 = …10；如果下一个输入1
    输入串 (10)1(0)
    S2符合这种情况
- 尽可能复用状态
  - 寻找具有相同含义的状态
  - 减少状态数
- 所有的状态具有完全的状态转换
  则完成状态机设计



硬件描述语言和时序电路

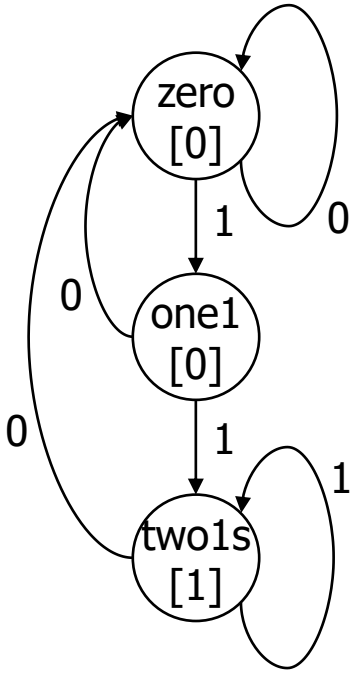
- 例4：将输入连续出现的1中去掉一个1



Verilog FSM

Moore机

```
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;

  parameter zero  = 2'b00;
  parameter one1  = 2'b01;
  parameter two1s = 2'b10;

  reg out;
  reg [2:1] state;        // state variables
  reg [2:1] next_state;

  always @(posedge clk)
    if (reset) state = zero;
    else       state = next_state;
```



Moore Verilog FSM

```
always @(in or state)

  case (state)
    zero:
   // last input was a zero
    begin
      if (in) next_state = one1;
      else    next_state = zero;
    end
     one1:
   // we've seen one 1
    begin
      if (in) next_state = two1s;
      else    next_state = zero;
    end
     two1s:
   // we've seen at least 2 ones
    begin
      if (in) next_state = two1s;
      else    next_state = zero;
    end
  endcase
```

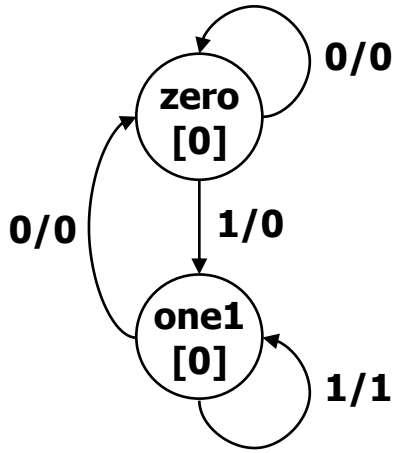
状态转换

电路输出，输出只与状态相关

```
  always @(state)
    case (state)
      zero: out = 0;
      one1: out = 0;
     two1s: out = 1;
    endcase

endmodule
```

Mealy Verilog FSM

```
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;
  reg out;
  reg state; // state variables
  reg next_state;

  always @(posedge clk)
    if (reset) state = zero;
    else       state = next_state;

  always @(in or state)
    case (state)
      zero:              // last input was a zero
     begin
       out = 0;
       if (in) next_state = one;
       else    next_state = zero;
     end
      one:               // we've seen one 1
     if (in) begin
        next_state = one; out = 1;
     end else begin
        next_state = zero; out = 0;
     end
    endcase
endmodule
```



输出和输入相关

同步 Mealy 机

```
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;
  reg out;
  reg state; // state variables

  always @(posedge clk)
    if (reset) state = zero;
    else
     case (state)
      zero:       // last input was a zero
     begin
       out = 0;
       if (in) state = one;
       else    state = zero;
     end
      one:        // we've seen one 1
     if (in) begin
         state = one; out = 1;
     end else begin
         state = zero; out = 0;
     end
    endcase
endmodule
```

例3—精确的“0010”识别器

设计同步序电路:
当输入准确的00串后输入为10,
输出1；否则输出0。

例如：
x=001001000010010
z=000100100000001

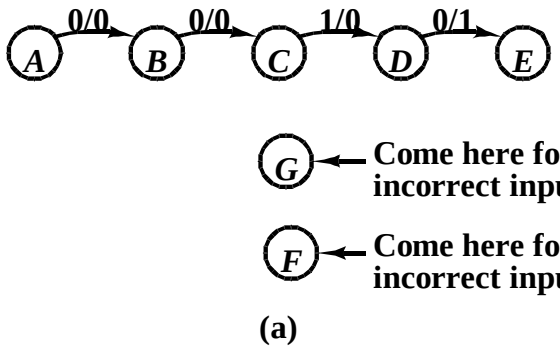
A：初始状态
B：识别出0
C：识别出00
D：识别出001
E：识别出0010

添加出错状态：G, F
G：出现多于两个‘0’
F：’1’的位置出错，重新开始

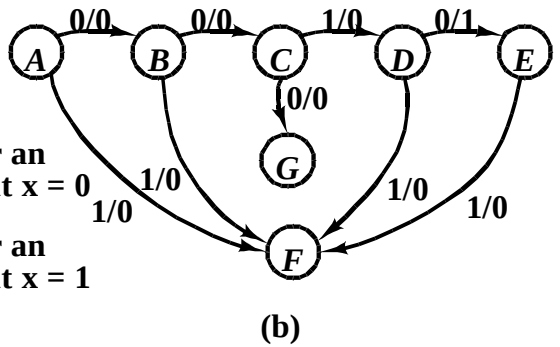
例3

其中：
E状态和B状态等价
A状态和F状态等价

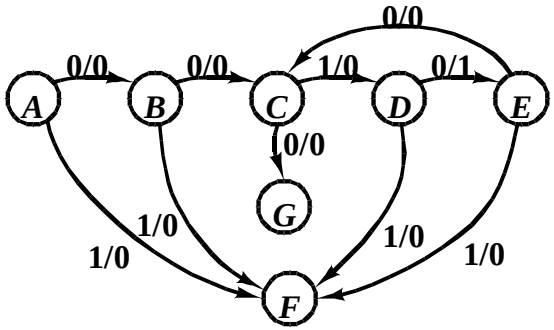
新的状态图(g)中
B状态：识别出0
A状态：初始状态，重新开始



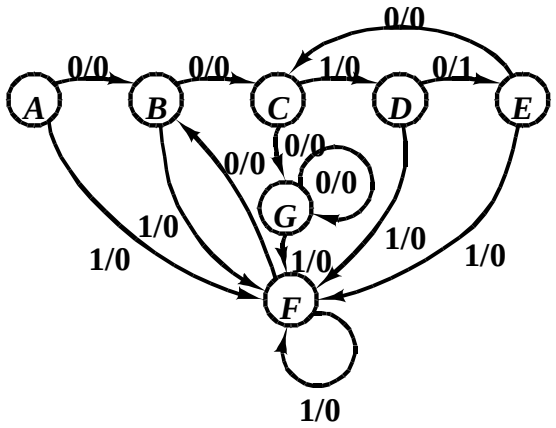
(a)



(b)



(c)



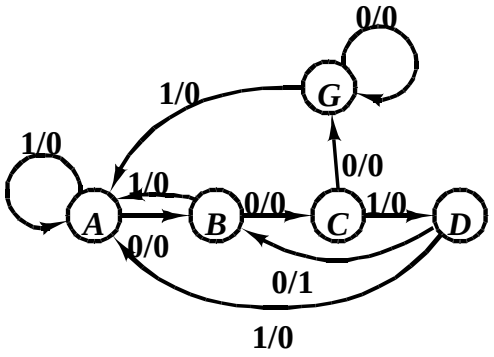
(d)

	x = 0	x = 1
A	B/0	F/0
B	C/0	F/0
C	G/0	D/0
D	E/1	F/0
E	C/0	F/0
F	B/0	F/0
G	G/0	F/0

(e)

	x = 0	x = 1
A	B/0	A/0
B	C/0	A/0
C	G/0	D/0
D	B/1	A/0
G	G/0	A/0

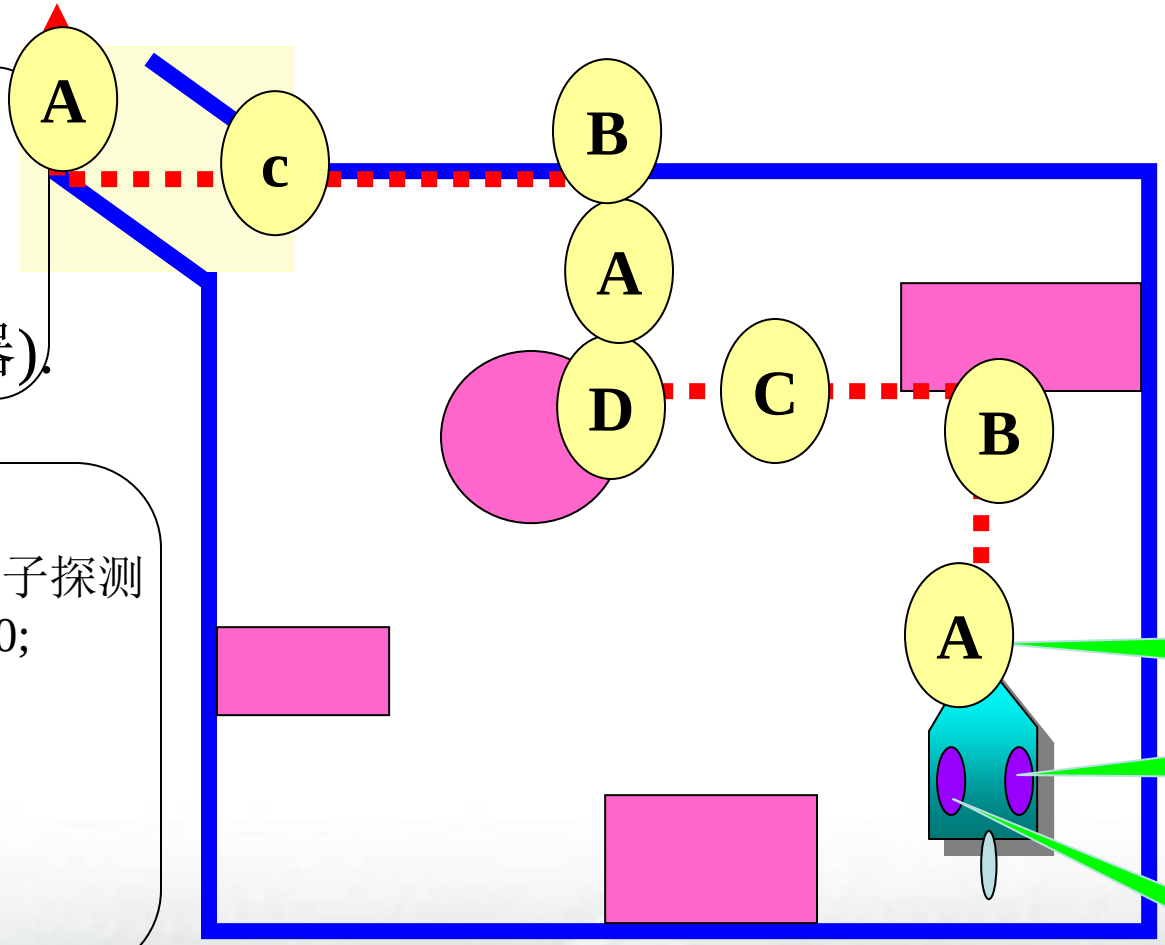
(f)



(g)

例4—机器人控制器

设计电路实现:
使得该机器人
能顺利找到
出口(用JK触发器).



机器人的操作:
(1)当遇到障碍物时,电子探测器输出x为1,否则输出0;
(2)z1,z2两控制线:
z1z2=10时向左转;
z1z2=01时向右转;
z1z2=00时前进;
z1，z2不能同时为1。

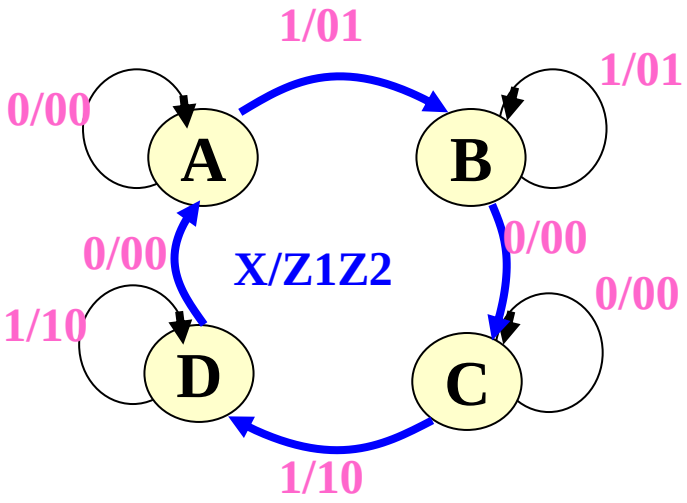
例4—机器人控制器

状态A:没有遇到障碍物,上次是左转;
状态B:监测到障碍物,正在右转;
状态C:没有遇到障碍物,上次是右转;
状态D:检测到障碍物,正在左转.

(1)上次向右转,当遇到障碍物时向左转;
(2)上次向左转,当遇到障碍物时向右转;

例4——机器人控制器

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



状态分配
状态A：00
状态B：01
状态D：10
状态C：11

y_1y_2	x=0	x=1
A	A/00	B/01
B	C/00	B/01
C	C/00	D/10
D	A/00	D/10

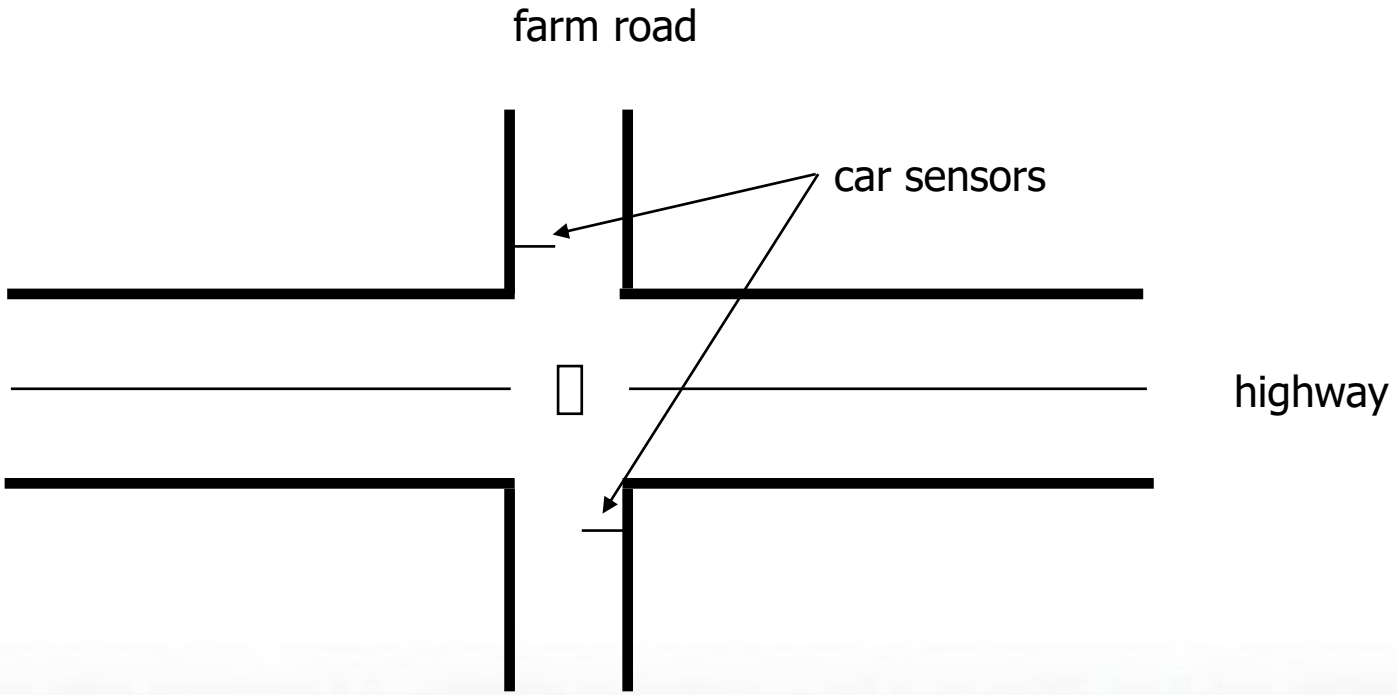
y_1y_2	x=0	x=1
00	00/00	01/01
01	11/00	01/01
11	11/00	10/10
10	00/00	10/10

例5：交通信号灯控制

- 繁忙的公路和一条很少使用的乡村路交叉
- 感应器C探测乡村路上是否有车载等待
  - 如果乡村路上没车，继续保持公路方向上的绿灯
  - 如果乡村路上有车，公路方向的信号灯由绿变黄再变红，乡村路上的信号灯变绿
  - 如果一直探测乡村路上有车，保持该方向绿信号灯，但不能长于一个特定的时间间隔
  - 否则，乡村路方向的信号灯由绿变黄再变红，公路方向信号灯变绿
  - 即使乡村路上有车载在等待，公路方向信号灯也要至少保持一定的时间
- 假设外部产生间隔时间:
  - 短时间脉冲(TS)
  - 长时间脉冲(TL)
  - 时间间隔表示信号 (ST)
  - TS是黄灯保持时间（3个时间单位）
  - TL是绿灯/红灯保持时间（7个时间单位）

交通信号灯控制

- 公路/乡村路交叉



交通信号灯控制

- 输入/输出列表

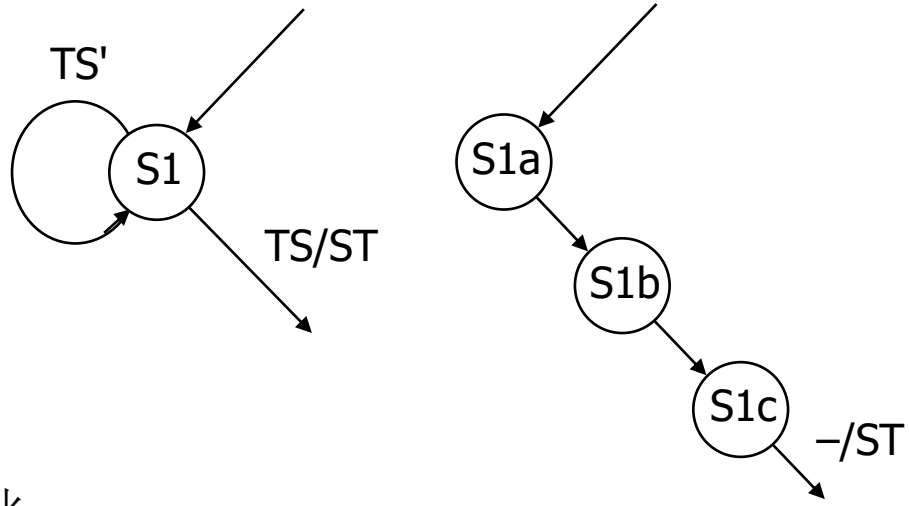
inputs	description	outputs	description
reset	place FSM in initial state	HG, HY, HR	assert green/yellow/red highway lights
C	detect vehicle on the farm road	FG, FY, FR	assert green/yellow/red highway lights
TS	short time interval expired	ST	start timing a short or long interval
TL	long time interval expired		

- 状态列表

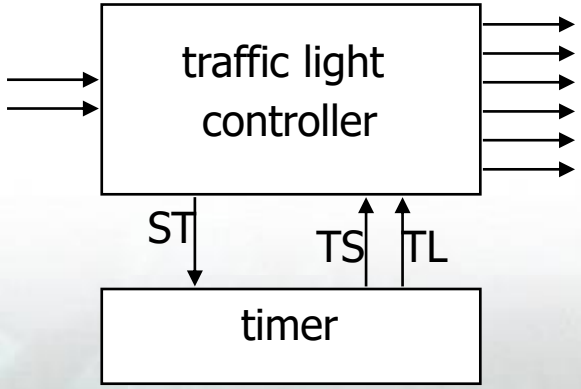
state	description
HG	highway green (farm road red)
HY	highway yellow (farm road red)
FG	farm road green (highway red)
FY	farm road yellow (highway red)

交通信号灯控制两FSM间通讯

- 如果没有计时器
  - S0 需要 7 状态
  - S1 需要 3 状态
  - S2 需要 7 状态
  - S3 需要 3 状态
  - S1 和 S3 比较简单
  - S0 和 S2 需要非常多的边
    - C 可能在任意7个状态变化

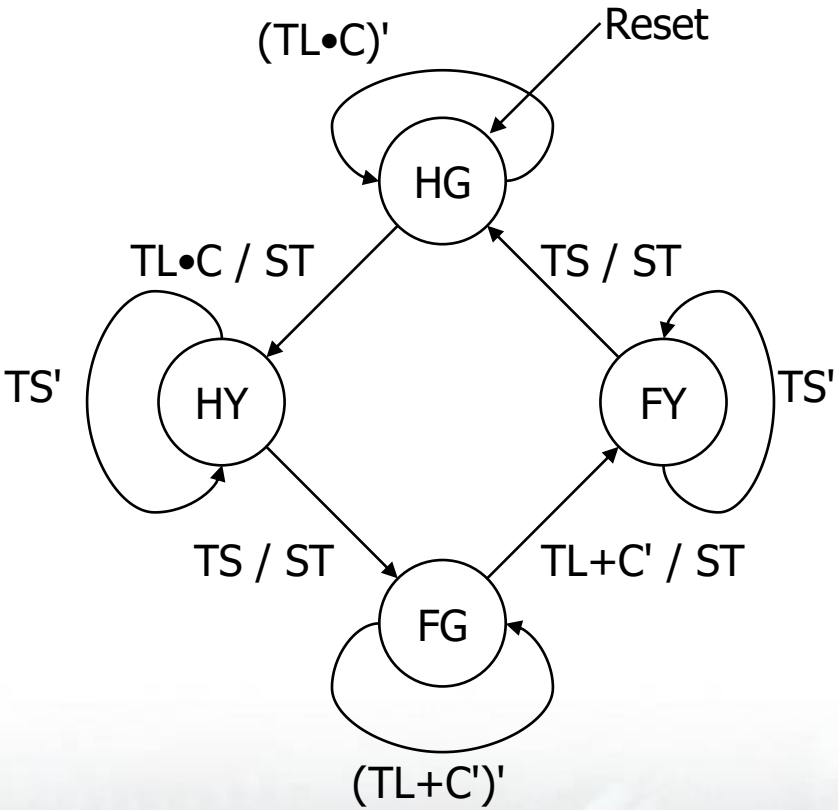


- 采用计时器
  - 极大的减少状态数
    - 由20个状态减为4个状态
  - 计时器只需要7-8个状态
    - 由20个状态减为12个状态



交通信号灯控制

- 状态图



交通信号灯控制

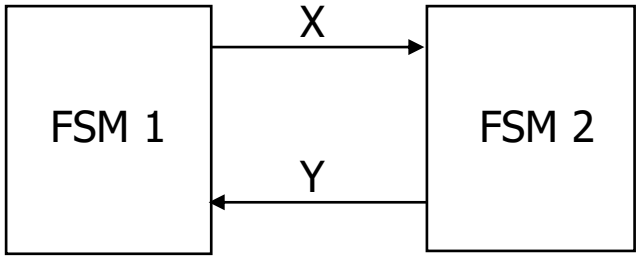
- 状态表
- 考虑状态分配

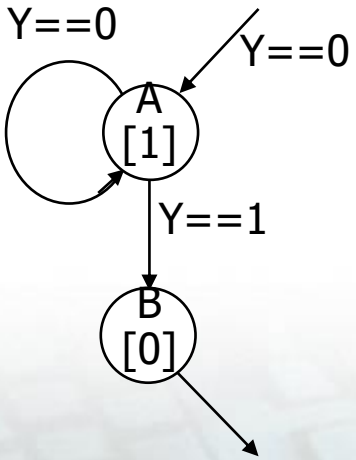
Inputs C	TL	TS	Present State	Next State	Outputs ST	H	F
0	–	–	HG	HG	0	Green	Red
–	0	–	HG	HG	0	Green	Red
1	1	–	HG	HY	1	Green	Red
–	–	0	HY	HY	0	Yellow	Red
–	–	1	HY	FG	1	Yellow	Red
1	0	–	FG	FG	0	Red	Green
0	–	–	FG	FY	1	Red	Green
–	1	–	FG	FY	1	Red	Green
–	–	0	FY	FY	0	Red	Yellow
–	–	1	FY	HG	1	Red	Yellow

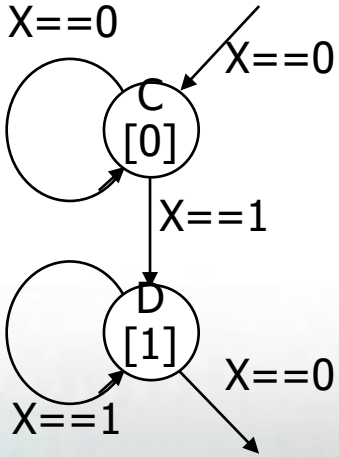
SA1:	HG = 00	HY = 01	FG = 11	FY = 10	
SA2:	HG = 00	HY = 10	FG = 01	FY = 11	
SA3:	HG = 0001	HY = 0010	FG = 0100	FY = 1000	(one-hot)

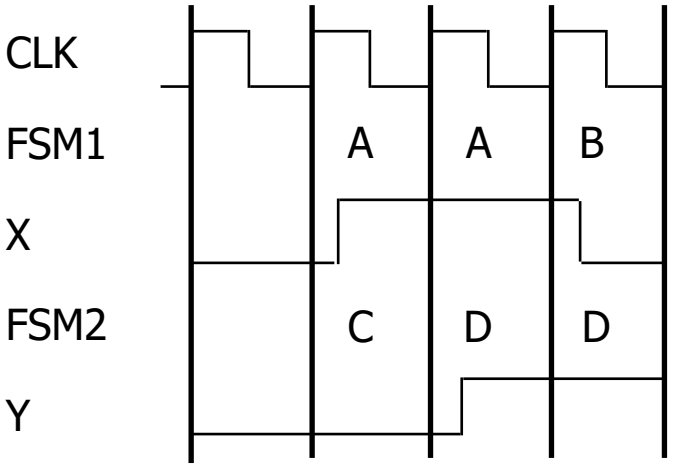
FMS间通讯

- 一个状态机的输出是另一个状态机的输入



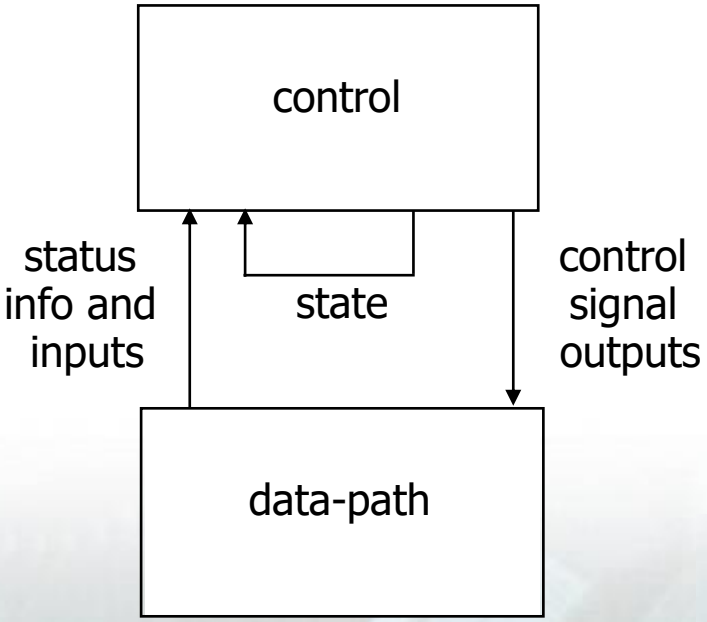






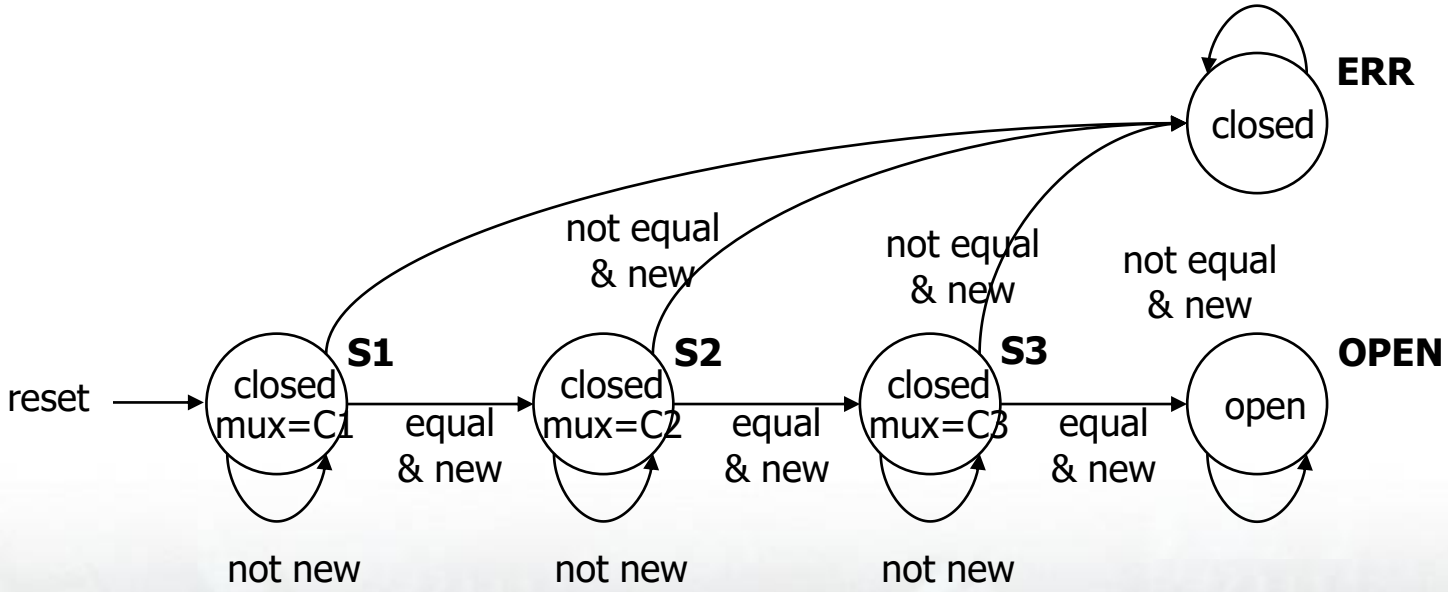
数据通路和控制

- 数字硬件系统 = 数据通路 + 控制
  - 数据通路：寄存器，计数器，组合逻辑（ALU），通讯（总线）
  - 控制：FSM产生控制信号的序列，控制数据通路的行为



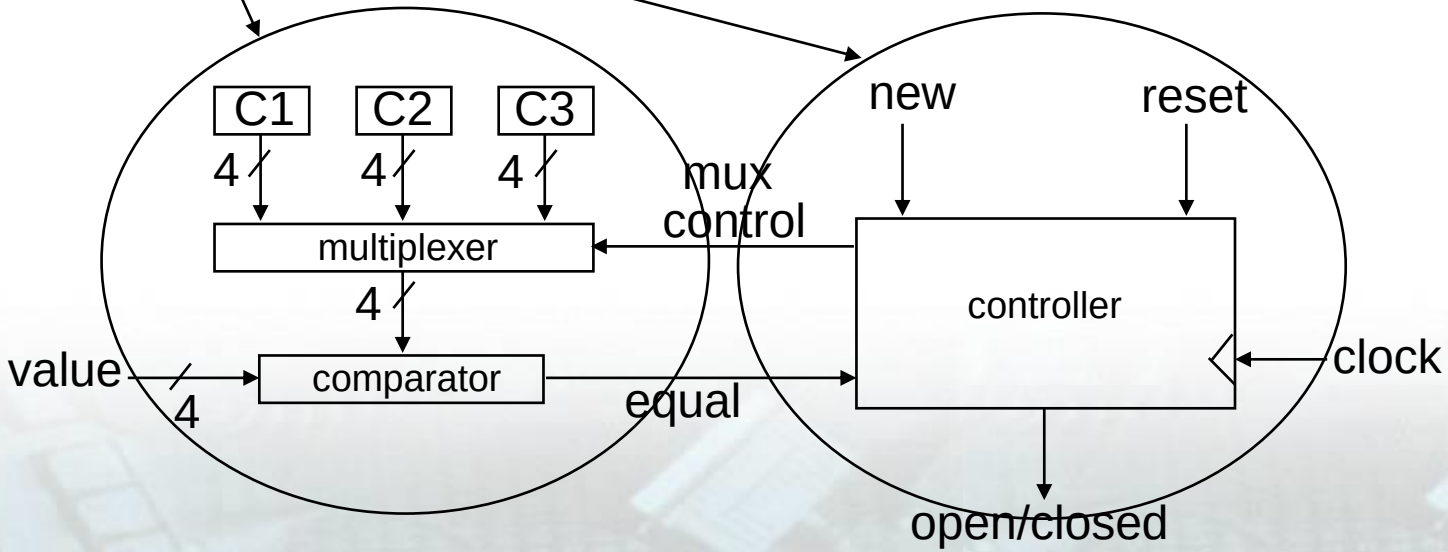
数据通路和控制

- 组合锁
  - 5个状态
  - 5个状态自转换
  - 6 个状态间转换
  - 1个复位reset转换到状态S1



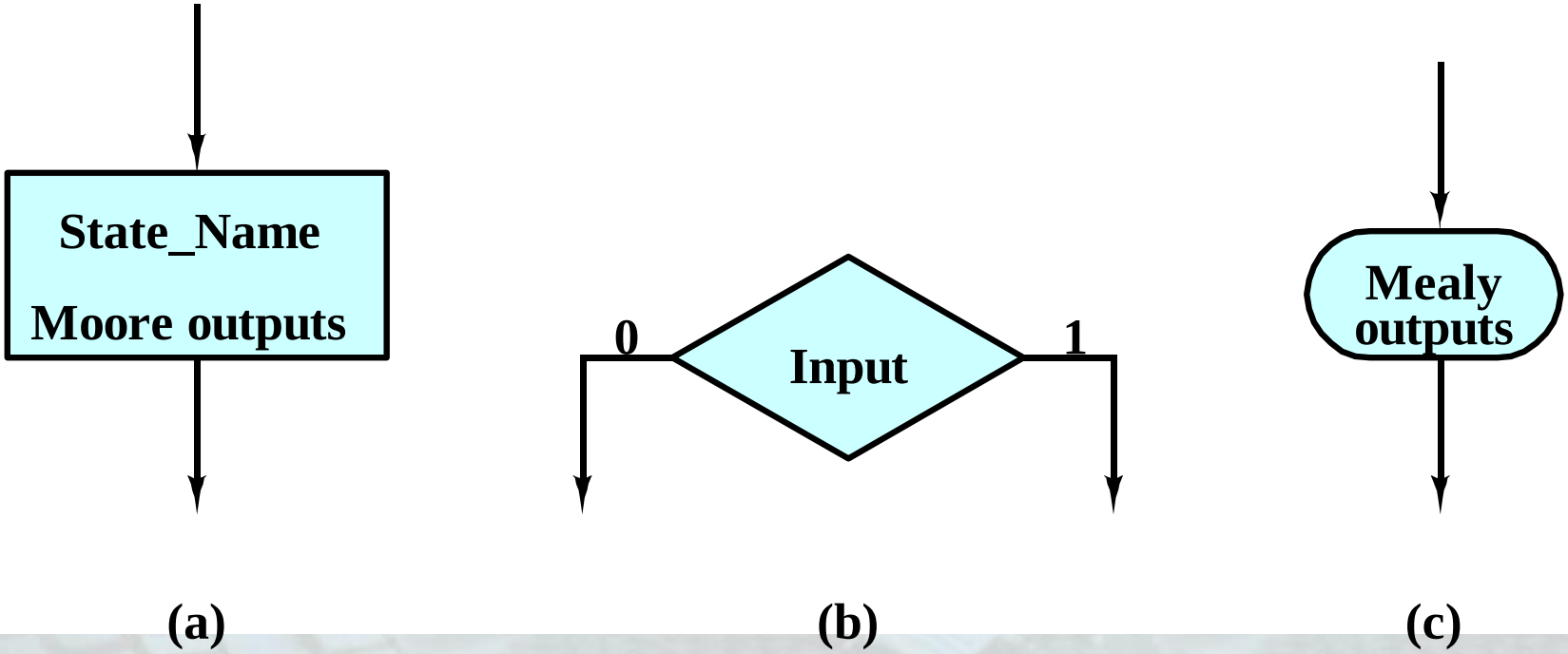
数据通路和控制

- 数据通路
  - 存储寄存器
  - 多选器
  - 比较器
- 控制
  - 有限状态自动机控制
  - 对数据通路控制（选择那个数值用来比较）

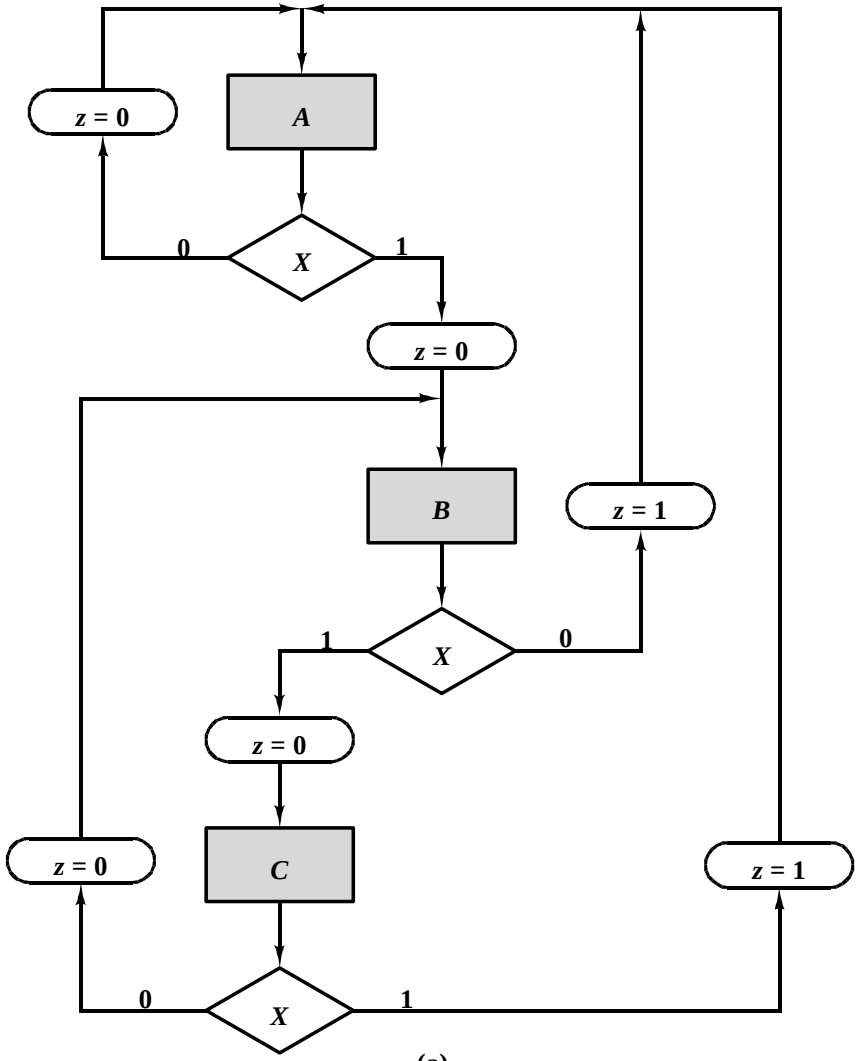


算法形状态机 (ASM)

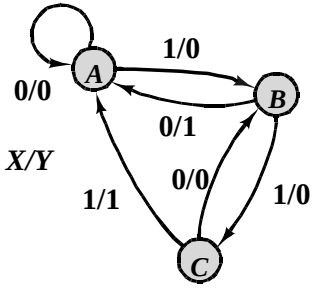
- (a) 状态框：状态名称，Moore机的输出
- (b) 决策框：状态的变换条件，决策框之处理一个输入的判断。
- (c) 条件输出框：Mealy机的输出



Mealy机的ASM表述

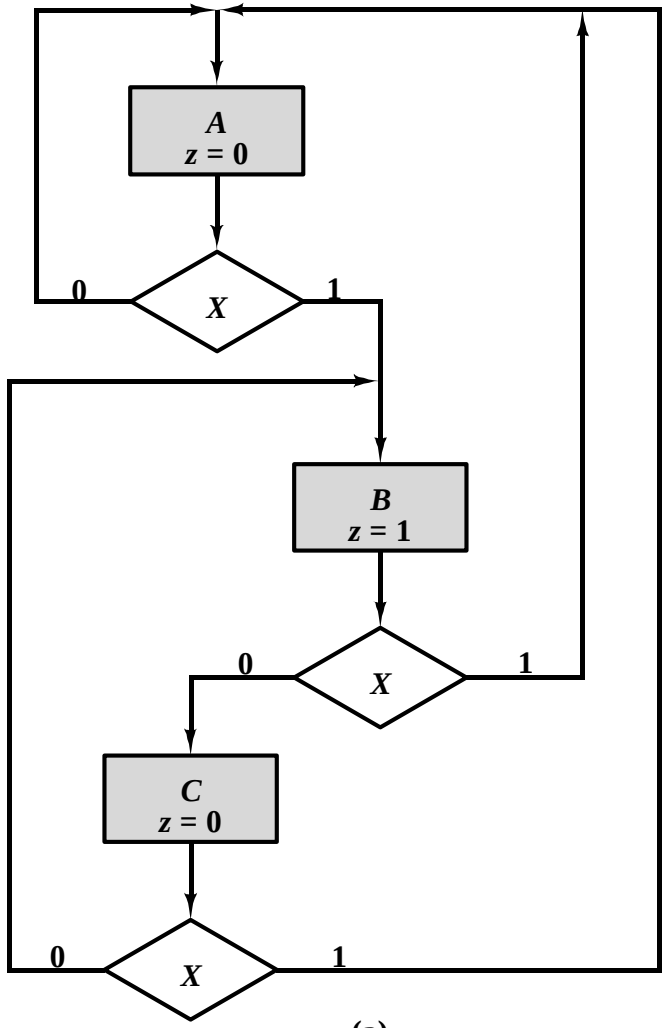


(a)

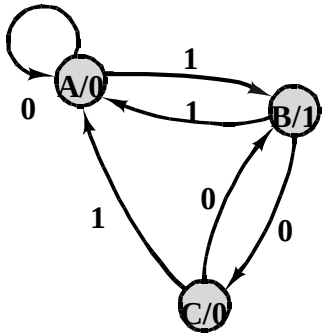


(b)

Moore机的ASM表述

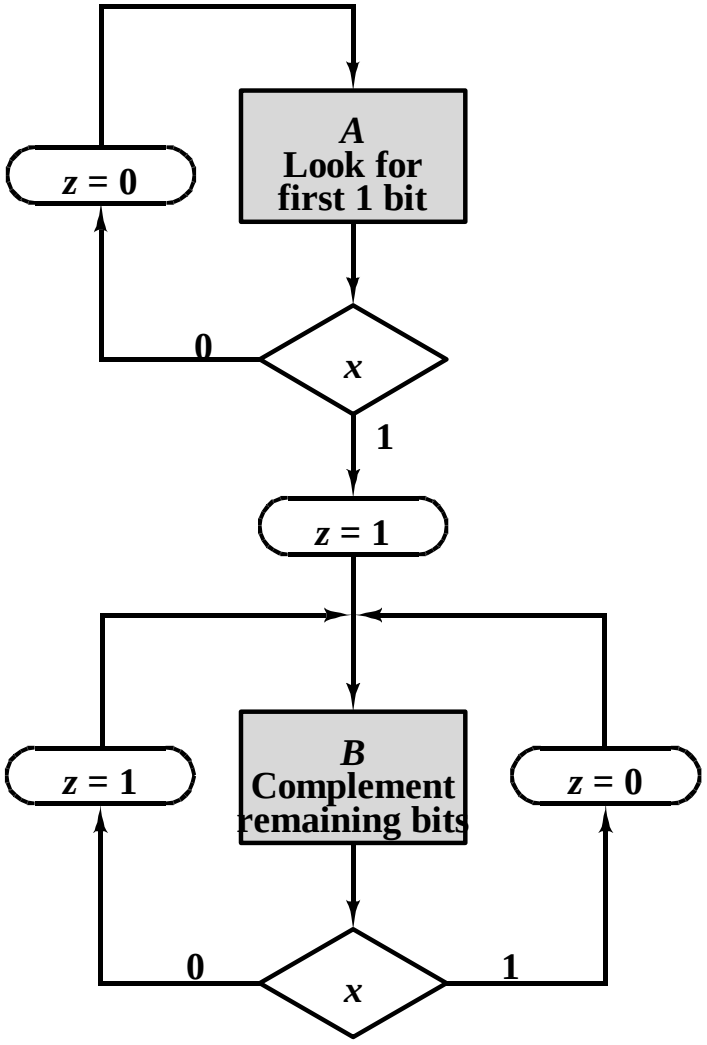


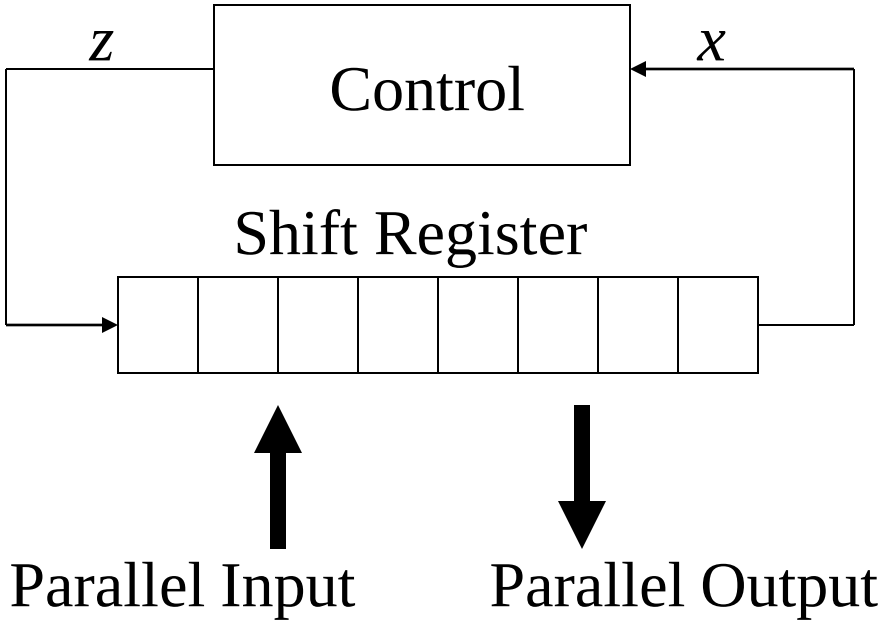
(a)



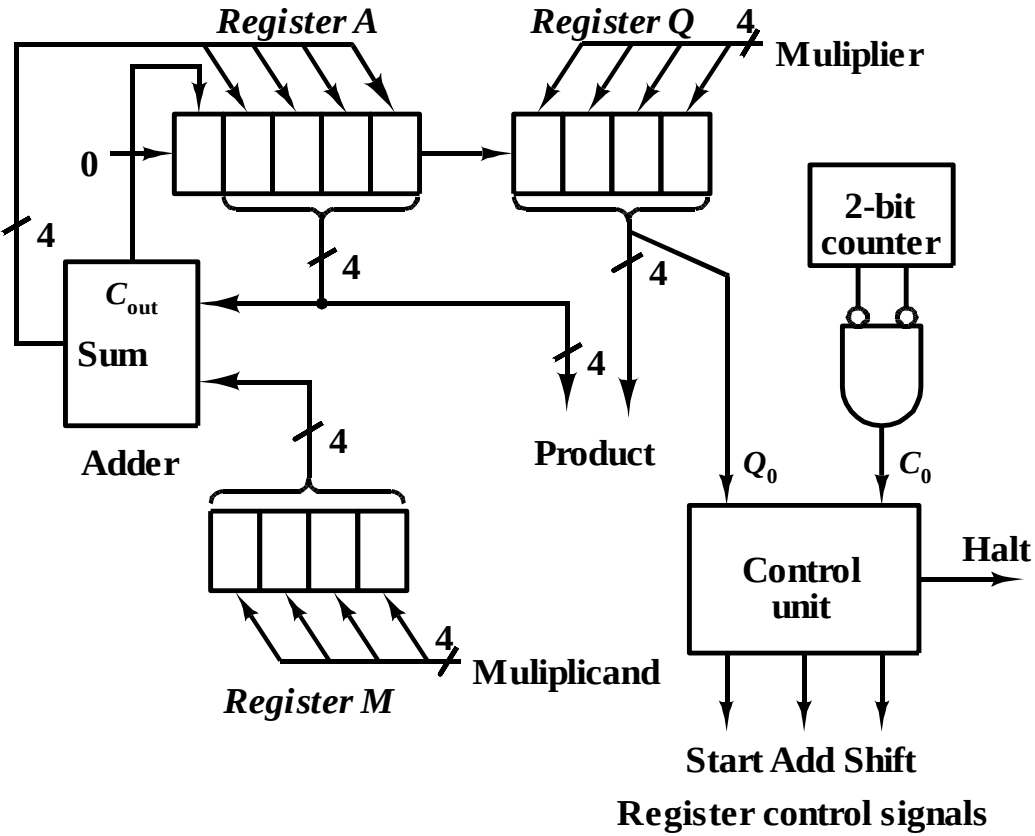
(b)

8-bit补码生成器ASM

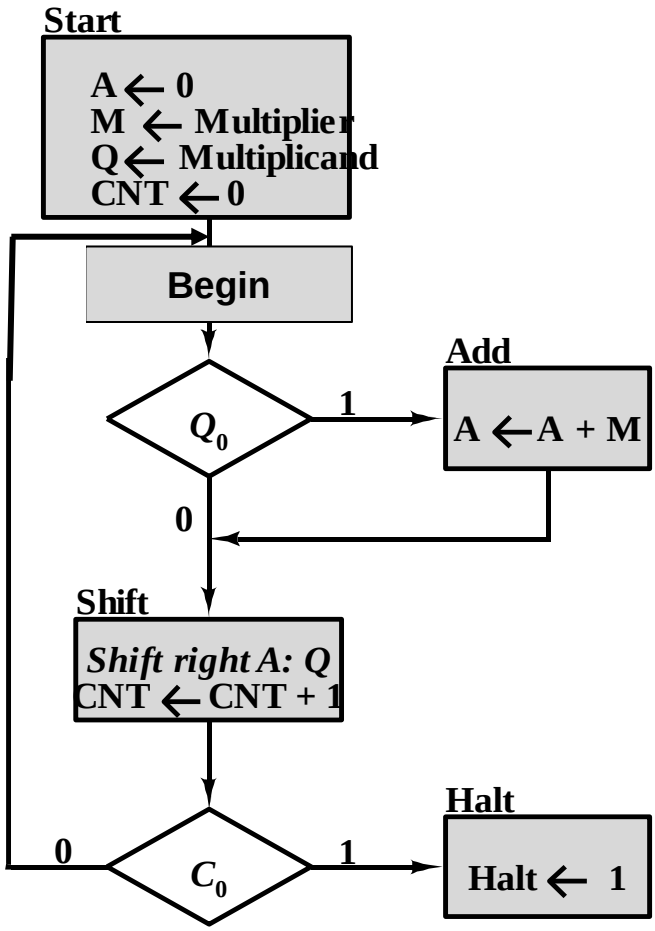




二进制4位乘法器控制器(Lab4)



(a)



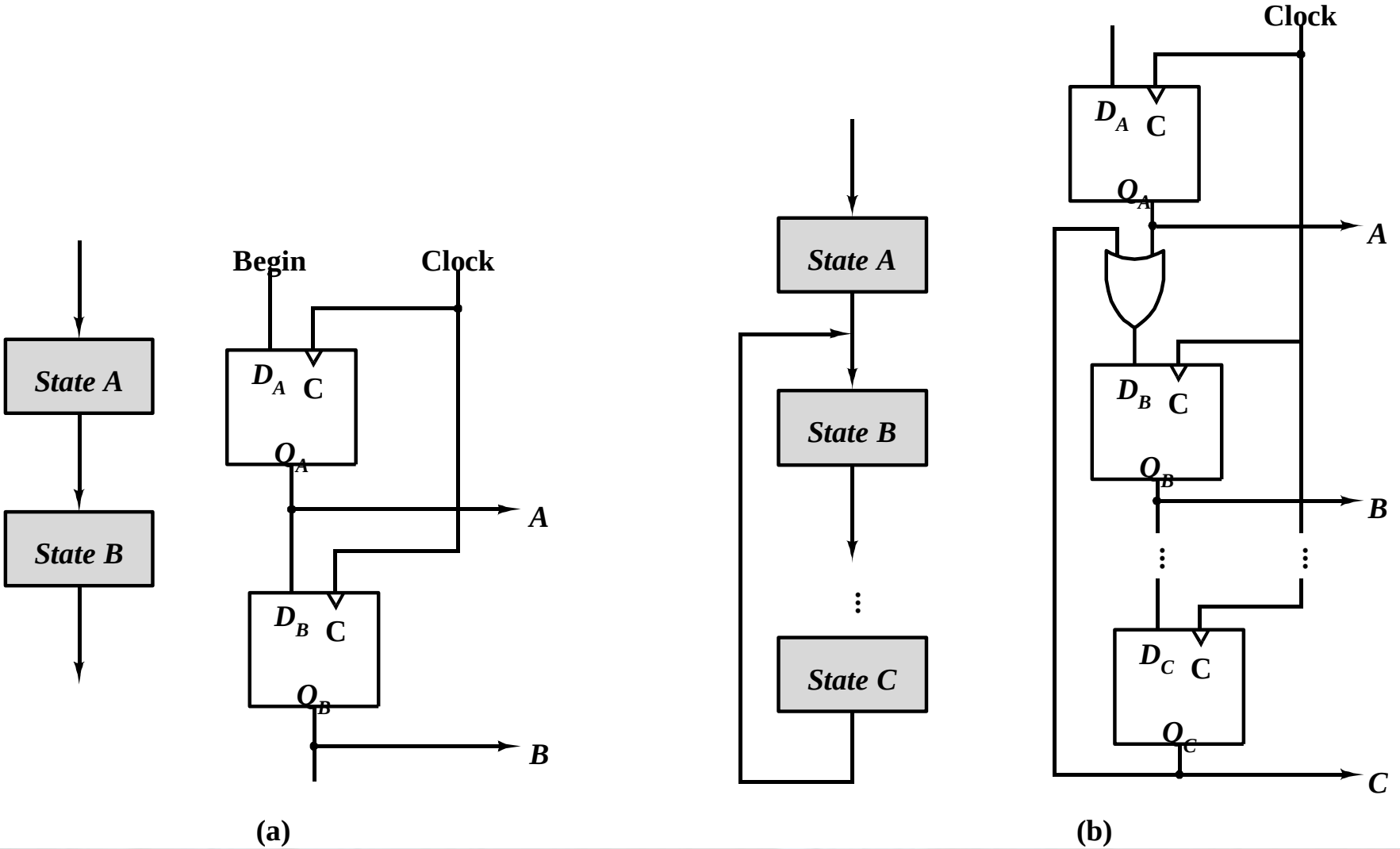
(b)

One-Hot 状态分配

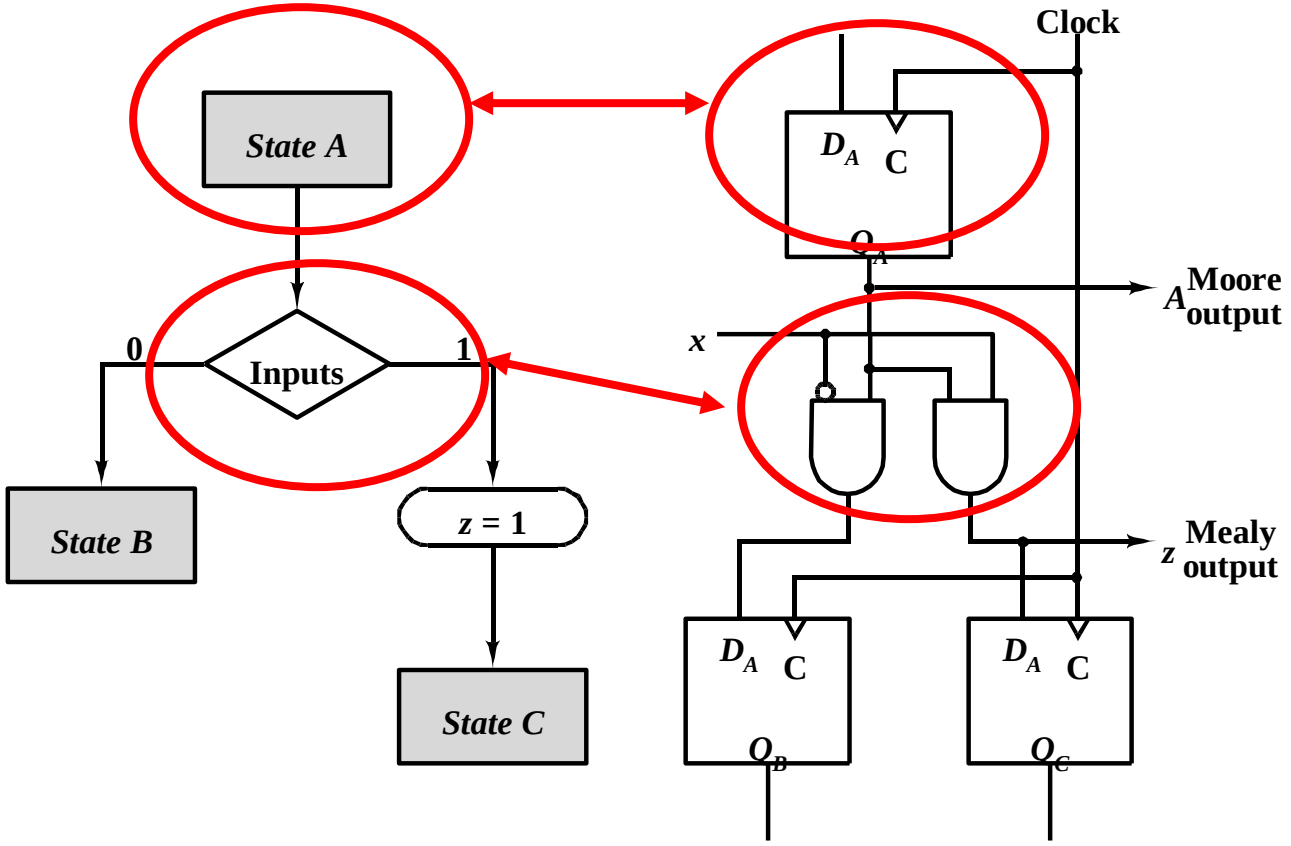
- 两种状态分配形式
  - 二进制编码(binary)
  - One_Hot编码
    - 减少设计时间
    - 实现状态的触发器数目多
    - 可以直接由ASM获得

	Sequential Assignment	One-hot Assignment
State	y_1y_0	$y_3y_2\ y_1y_0$
A	00	0001
B	01	0010
C	10	0100
D	11	1000

采用One-Hot 状态分配的ASM 设计

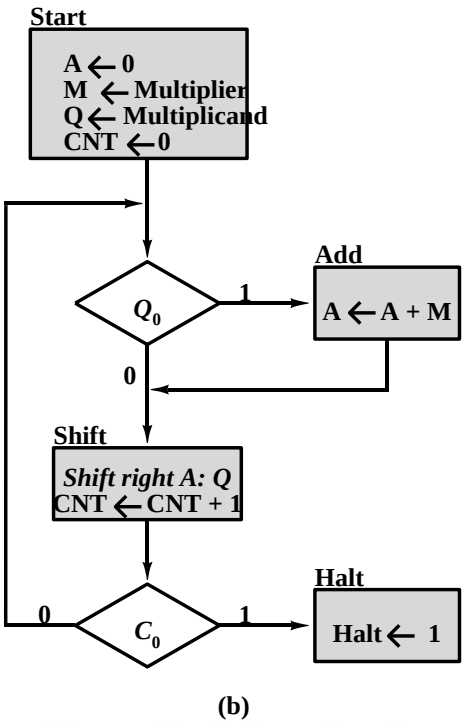


采用One-Hot 状态分配的ASM 设计

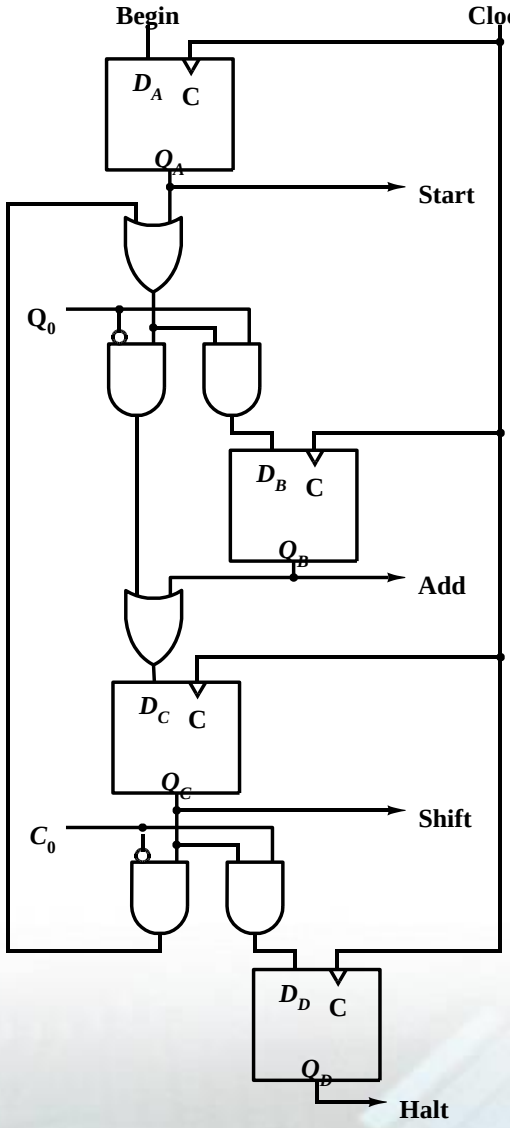


(c)

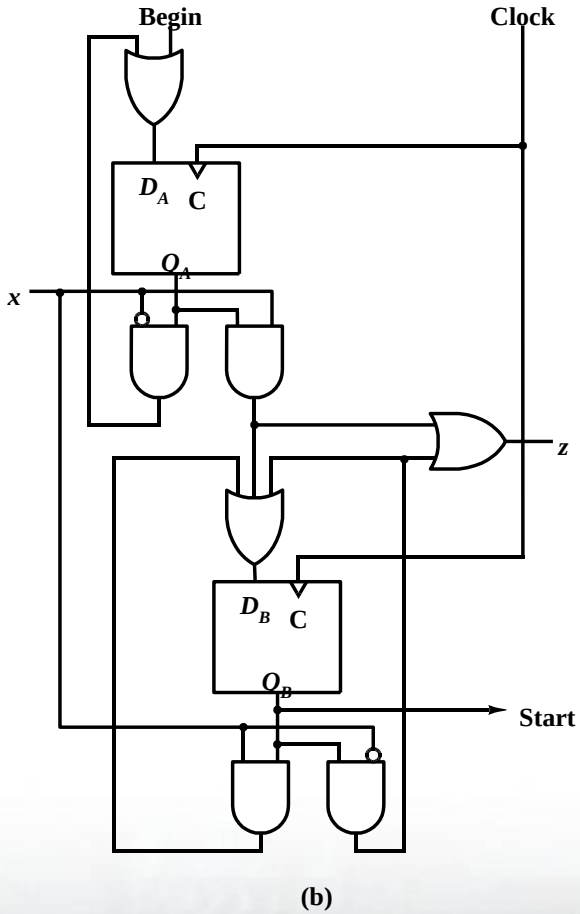
4位乘法控制器的One-hot设计



(b)

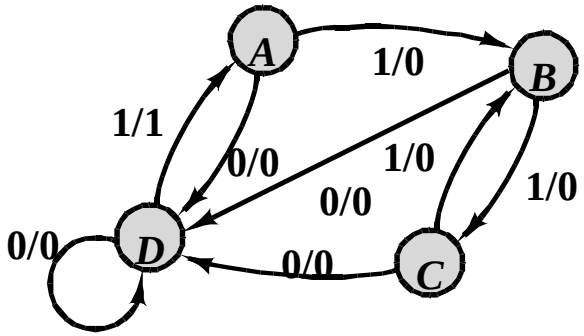


(a)



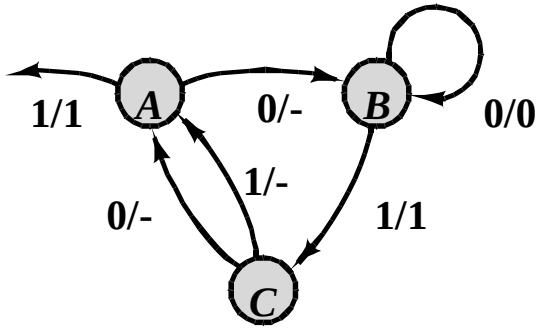
(b)

完全确定和非完全确定的状态机



	x = 0	x = 1
A	D/0	B/0
B	D/0	C/0
C	D/0	B/0
D	D/0	A/1

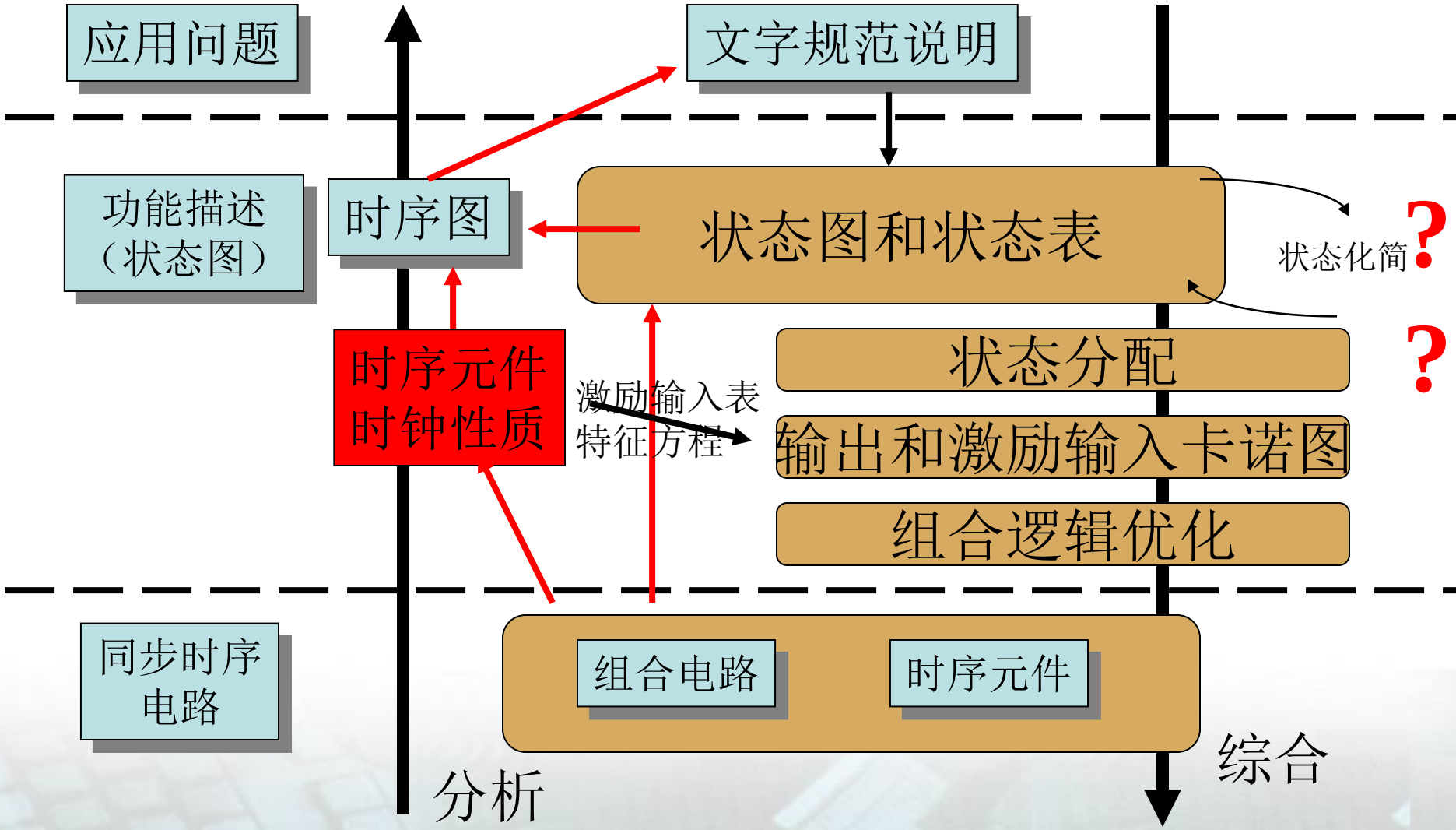
(a) Completely specified circuit



	x = 0	x = 1
A	B/-	-/1
B	B/0	C/1
C	A/-	A/-

(b) Incompletely specified circuit

时序电路分析与综合总结



时序电路分析与综合总结