数字逻辑

第四章 时序电路分析与设计

北京理工大学计算机学院

提纲

1. 时序电路简介

2. 基本存储单元

3. 时序电路分析

4. 时序电路设计

5. 状态机设计

1.1 概念

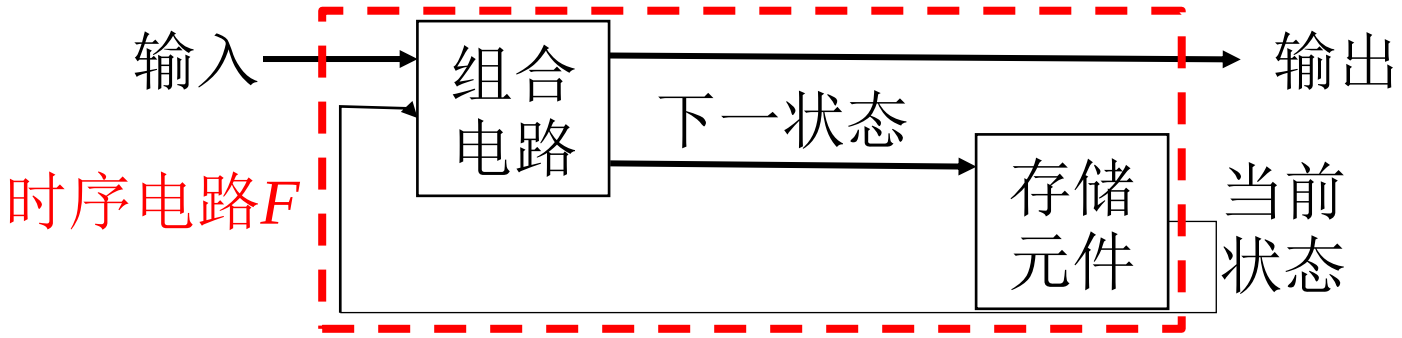
- 时序电路
  - 电路任意时刻的稳态输出，不仅取决于该时刻的输入，而且与前一时刻电路的状态有关
  - 由存储元件和组合电路连接而成
    - 存储元件：存储二进制信息（当前状态）的电路
    - 组合电路：多输出（下一状态等）的转换电路

1.1 概念

- 时序电路

  输出=*F*(输入，状态)

  - 组合电路
    - 输入：从外部输入信号+当前状态
    - 输出：往外部输出信号+下一状态
  - 存储单元
    - 输入：下一状态
    - 输出：新的当前状态

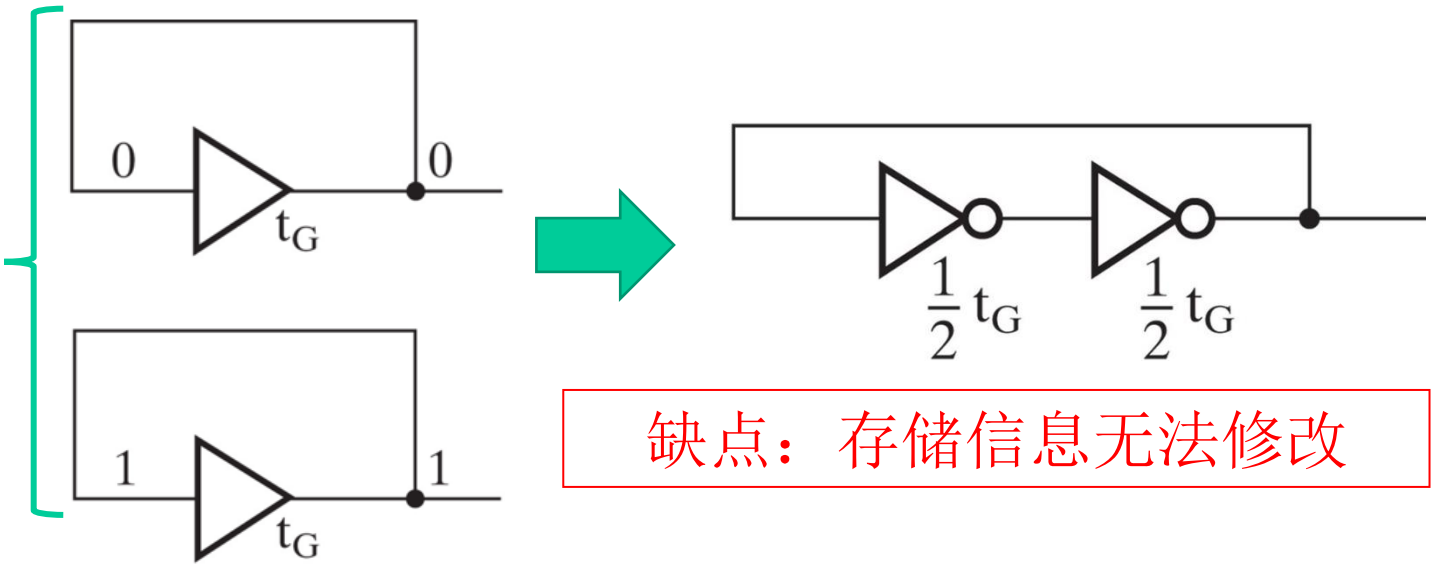


1.1 概念

- 简单例子：缓冲器实现的存储器
  - 缓冲器：将输入值直接输出（假设传播延迟t_G）

缺点：只在t_G时间内有效存储

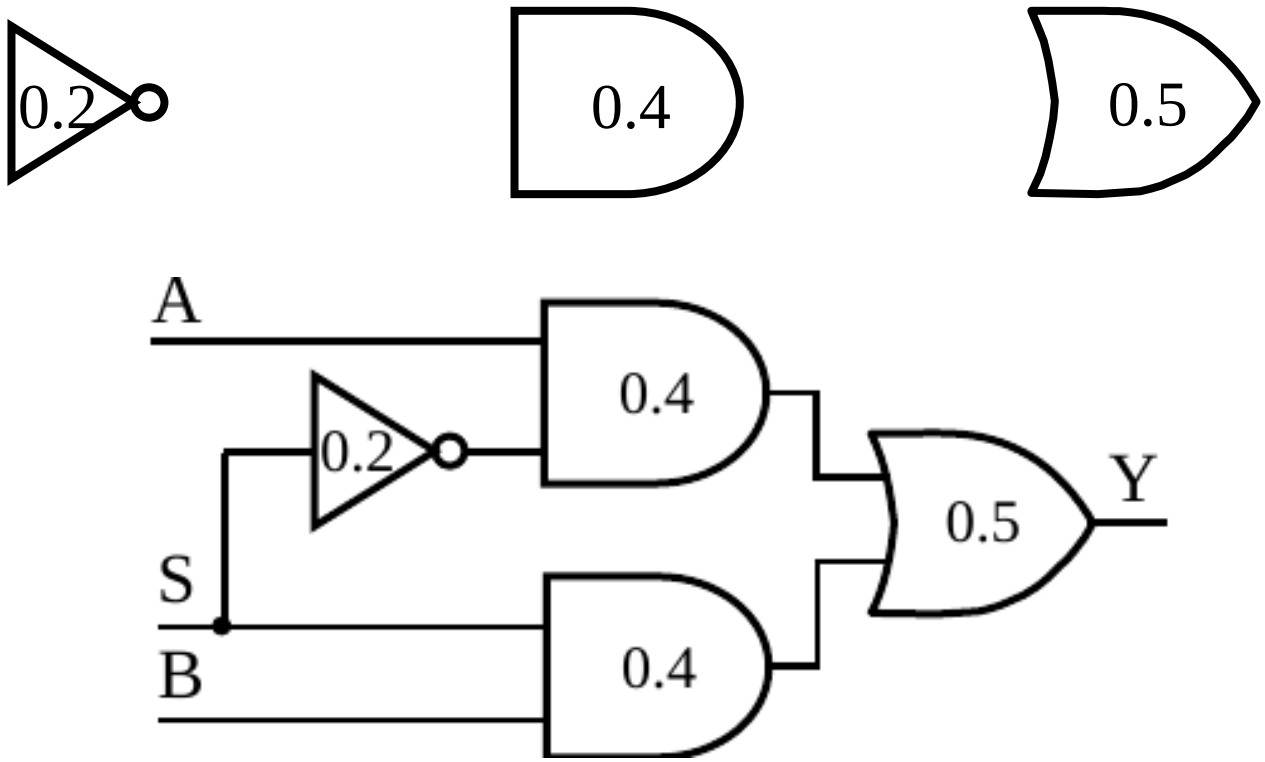
  - 存储器



缺点：存储信息无法修改

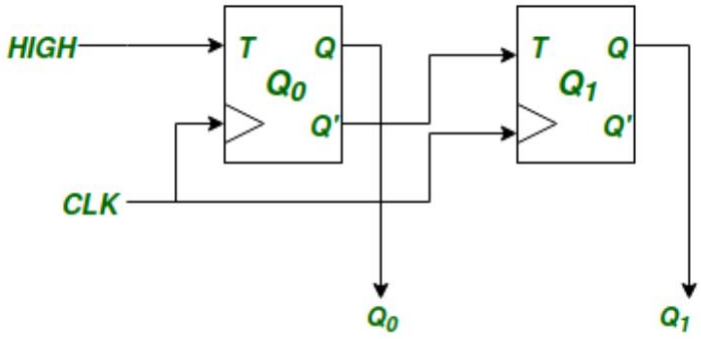
1.1 概念

- 简单例子：缓冲器实现的存储器
  - 补充：逻辑门的延迟，一般以*ns*为单位

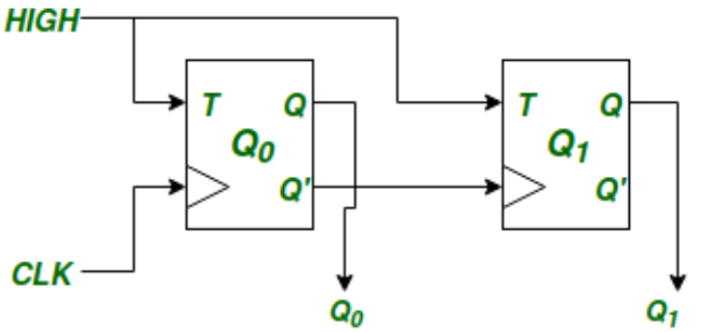


1.2 时序电路类型

- 同步时序电路
  - 由一个时钟统一控制存储元件，状态只会在时钟上升/下降的边沿发生
  - **触发器**是基本存储元件



- 异步时序电路
  - 状态可以在任意时间发生变化
  - **锁存器**是基本存储元件

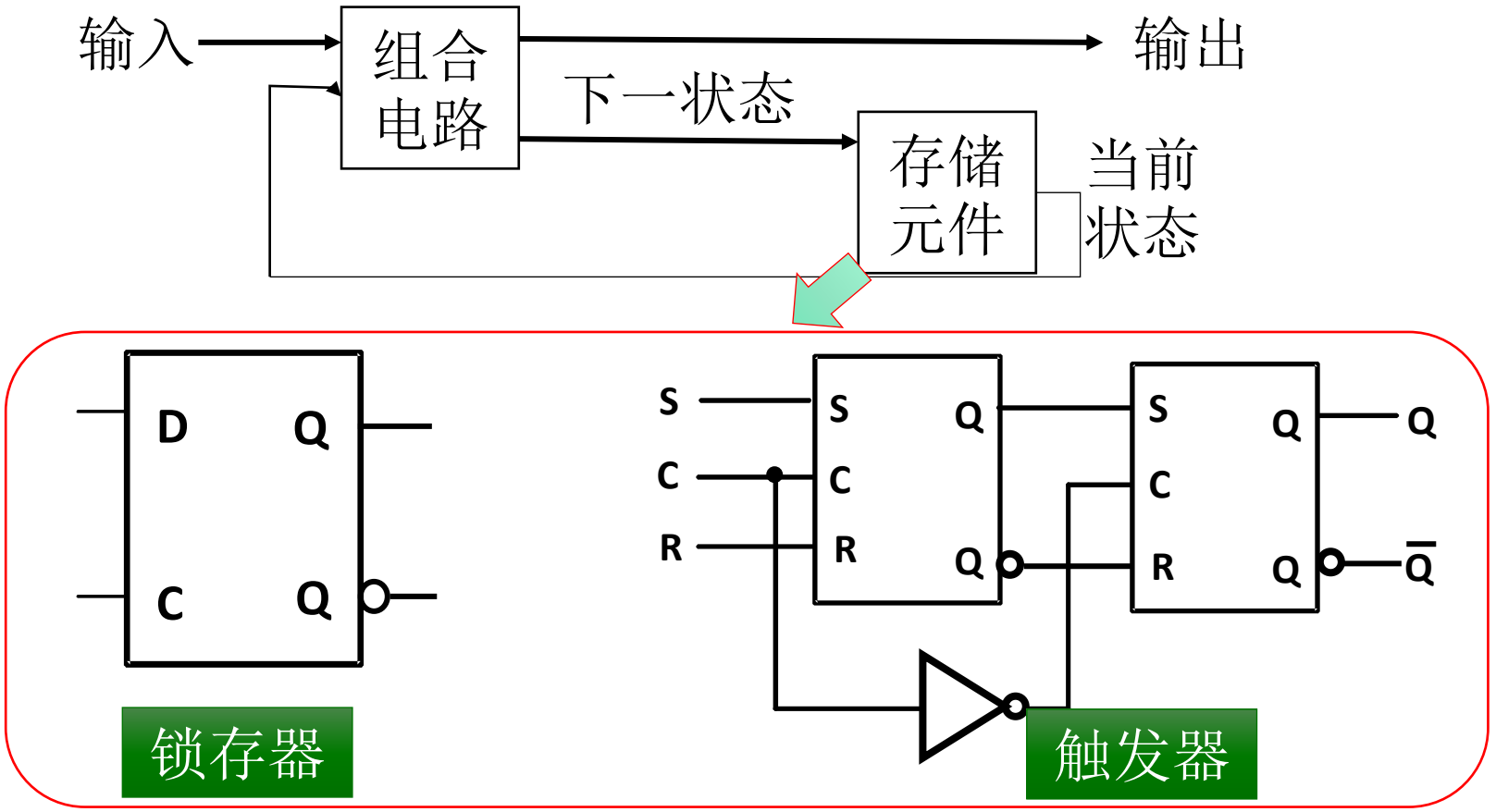


提纲

1. 时序电路简介
2. **基本存储单元**
   - **2.1 锁存器**
   - **2.2 触发器**
3. 时序电路分析
4. 时序电路设计
5. 状态机设计

基本存储单元

- 在电路供电状态下能够无限期保存二进制数据，实现时序电路中存储元件的“记忆”功能



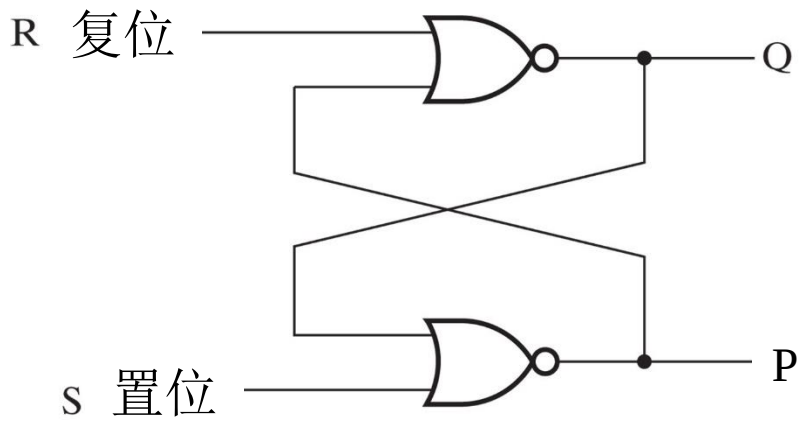
2.1 锁存器

- 锁存器（Latch）是构成存储元件的基础，把信号暂存以维持某种状态
  - S-R锁存器
  - $\overline{S}$-$\overline{R}$锁存器
  - 时钟$\overline{S}$-$\overline{R}$锁存器
  - D锁存器
  - ……
- 通过锁存器可以构成触发器等更复杂的存储元件

2.1 锁存器

2.1.1 S-R锁存器（基本或非门锁存器）

- 交叉耦合的两个或非门得到一个S-R锁存器
  - S=1且R=1是禁止的



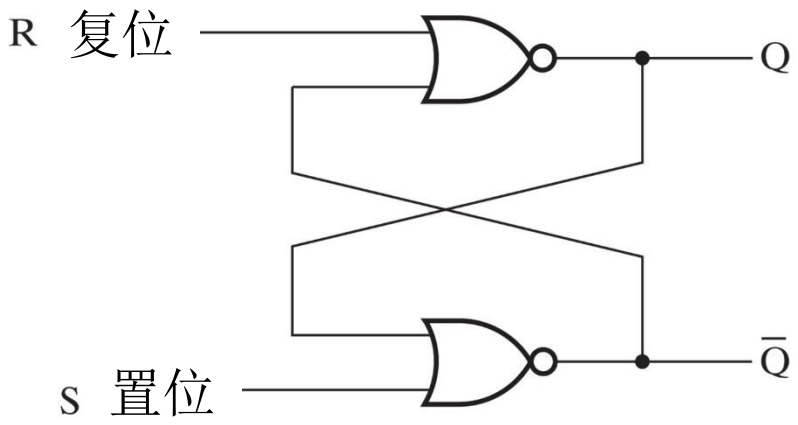
S-R锁存器真值表

S	R	Q	P
1	0	1	0

2.1 锁存器

2.1.1 S-R锁存器（基本或非门锁存器）

- 交叉耦合的两个或非门得到一个S-R锁存器
  - S=1且R=1是禁止的



或非真值表

0	0	1
0	1	0
1	0	0
1	1	0

S-R锁存器真值表

S	R	Q	$\overline{Q}$	
1	0	1	0	置位状态
0	0	1	0	
0	1	0	1	复位状态
0	0	0	1	
1	1	0	0	未定义状态

$$Q = \overline{R + \overline{S + Q}}$$

$$\overline{Q} = \overline{S + \overline{R + \overline{Q}}}$$

2.1 锁存器

2.1.1 S-R锁存器（基本或非门锁存器）

- 交叉耦合的两个或非门得到一个S-R锁存器
- 时序行为分析

Time

R	S	Q	$\overline{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	“Set” Q to 1
0	0	1	0	Now Q “remembers” 1
1	0	0	1	“Reset” Q to 0
0	0	0	1	Now Q “remembers” 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

S-R锁存器真值表

Q_t	S_t	R_t	Q_{t+1}	$\overline{Q}_{t+1}$	
0	1	0	1	0	
1	1	0	1	0	置位状态
1	1	1	1	0	
0	0	1	0	1	
1	0	1	0	1	复位状态
0	0	0	0	1	

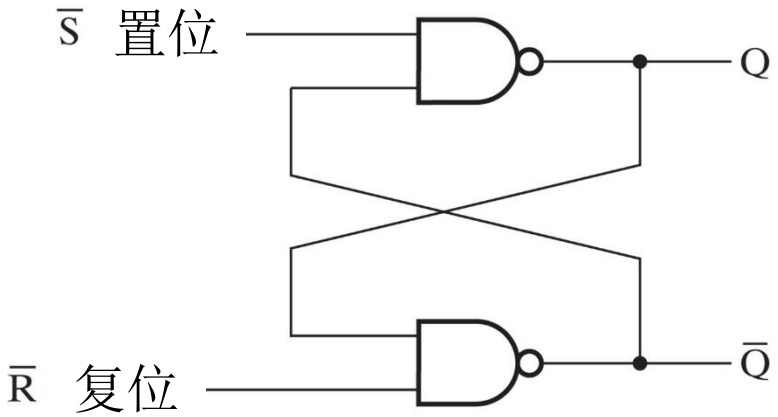
$$Q_{t+1} = \overline{R_t + \overline{S_t + Q_t}}$$

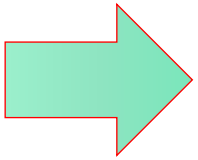
$$\overline{Q}_{t+1} = \overline{S_t + \overline{R_t + \overline{Q}_t}}$$

2.1 锁存器

2.1.2 $\overline{S}$-$\overline{R}$锁存器（基本与非门锁存器）

- 交叉耦合的两个与非门得到一个$\overline{S}$-$\overline{R}$锁存器
  - $\overline{S}$=0且$\overline{R}$=0是禁止的





$\overline{S}$-$\overline{R}$锁存器真值表

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
0	1	1	0	置位状态
1	1	1	0	
1	0	0	1	复位状态
1	1	0	1	
0	0	1	1	未定义状态

与非真值表

0	0	1
0	1	1
1	0	1
1	1	0

$$Q=\overline{\overline{S}\cdot\overline{\overline{R}\cdot Q}}$$

$$\overline{Q}=\overline{\overline{R}\cdot\overline{\overline{S}\cdot\overline{Q}}}$$

2.1 锁存器

2.1.2 $\bar{S}$-$\bar{R}$锁存器（基本与非门锁存器）

- 交叉耦合的两个与非门得到一个$\bar{S}$-$\bar{R}$锁存器
- 时序行为分析

Time

R	S	Q	$\bar{Q}$	Comment
1	1	?	?	Stored state unknown
1	0	1	0	“Set” Q to 1
1	1	1	0	Now Q “remembers” 1
0	1	0	1	“Reset” Q to 0
1	1	0	1	Now Q “remembers” 0
0	0	1	1	Both go high
1	1	?	?	Unstable!

$\bar{S}$-$\bar{R}$锁存器真值表

Q_t	$\bar{S}_t$	$\bar{R}_t$	Q_{t+1}	$\bar{Q}_{t+1}$	
0	0	1	1	0	
1	0	1	1	0	置位状态
1	1	1	1	0	
0	1	0	0	1	
1	1	0	0	1	复位状态
0	1	1	0	1	

$$Q_{t+1} = \overline{\bar{S}_t \cdot \overline{\bar{R}_t \cdot Q_t}}$$

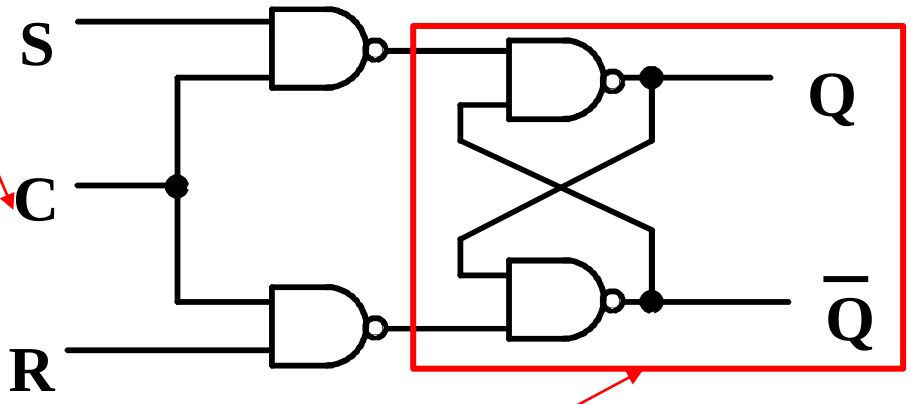
$$\bar{Q}_{t+1} = \overline{\bar{R}_t \cdot \overline{\bar{S}_t \cdot \bar{Q}_t}}$$

2.1 锁存器

2.1.3 时钟S-R锁存器（带控制的S-R锁存器）

- 在$\overline{S}$-$\overline{R}$锁存器基础上添加额外控制信号C，以此开启或关闭（触发）锁存器的功能
  - C=1，S-R锁存器的功能
  - C=0，状态Q不受S和R输入的影响

clock或者控制信号



$\overline{S}$-$\overline{R}$锁存器

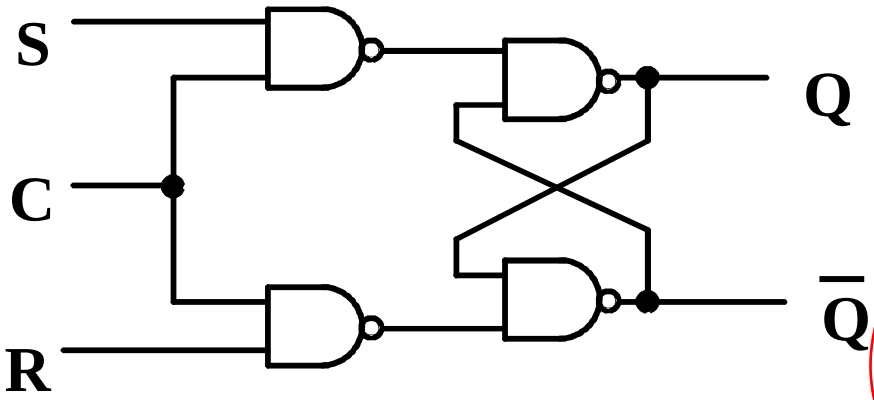
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

S-R锁存器功能

2.1 锁存器

2.1.3 时钟S-R锁存器（带控制的S-R锁存器）

- C=1时序行为分析
  - 输入：S、R和当前时刻状态Q(t)
  - 输出：下一状态Q(t+1)



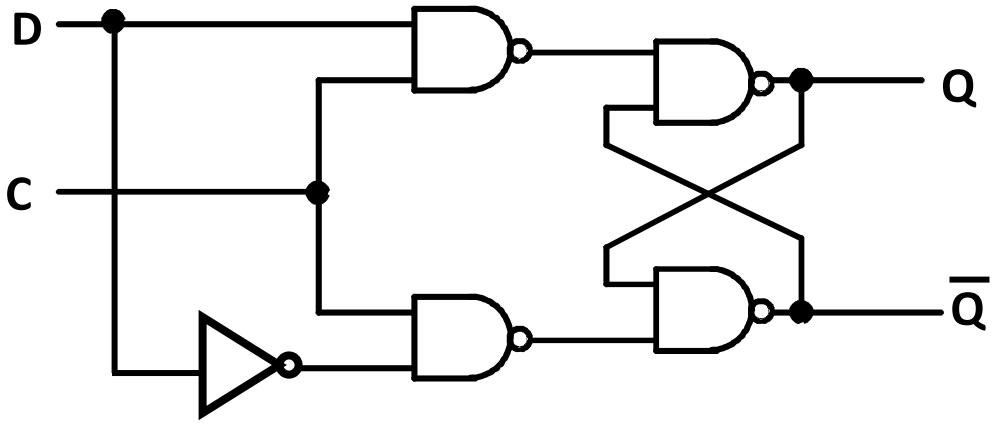
Q(t)	S	R	Q(t+1)	Comment
0	0	0	0	No change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	Indeterminate
1	0	0	1	No change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	Indeterminate

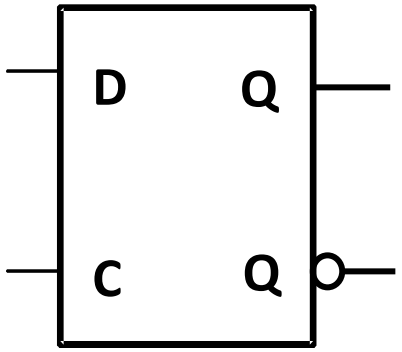
未定义状态

2.1 锁存器

2.1.4 D锁存器

- 在时钟S-R锁存器的基础上，添加一个反相器





D锁存器的图形符号表示

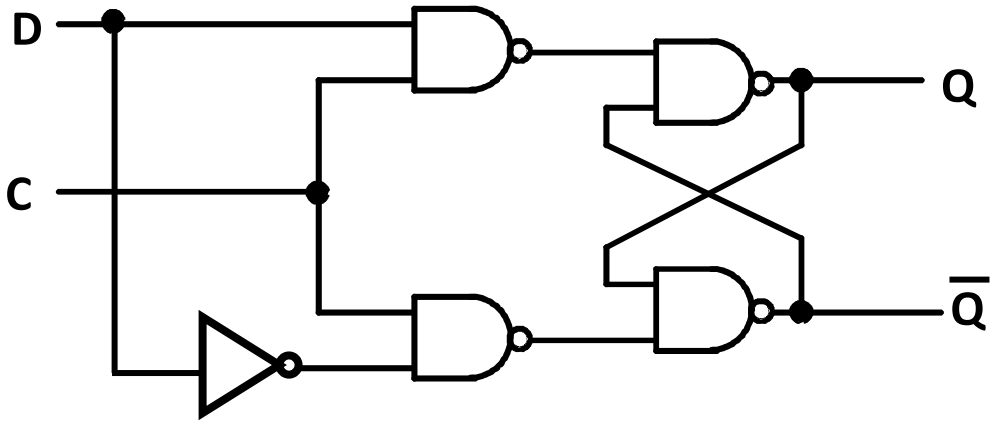
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

D锁存器不存在未定义状态！

2.1 锁存器

2.1.4 D锁存器

- D：数据信号
- C：控制信号



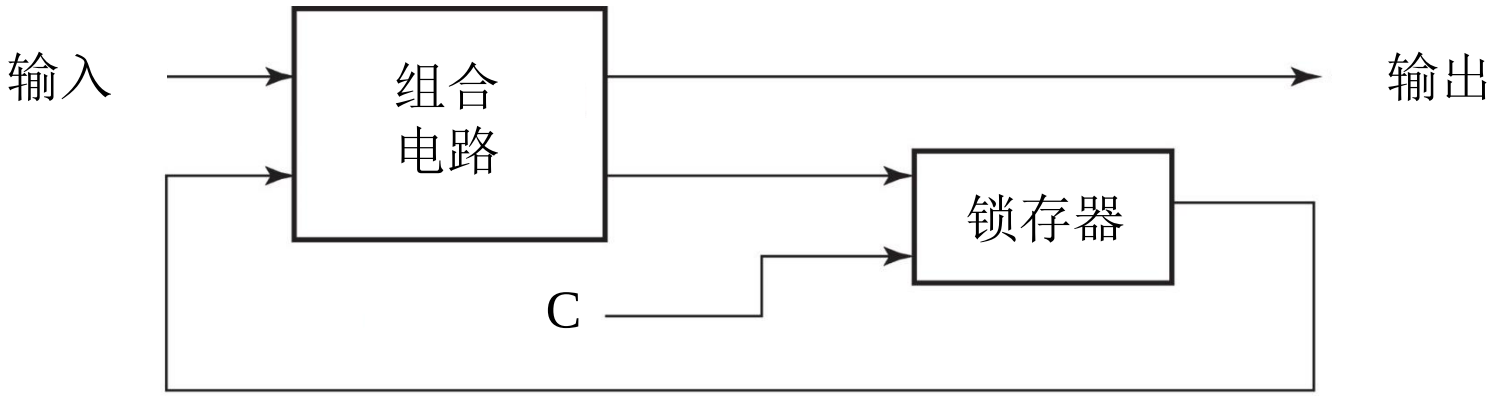
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

C=1时，Q=D

锁存器是“透明”的！

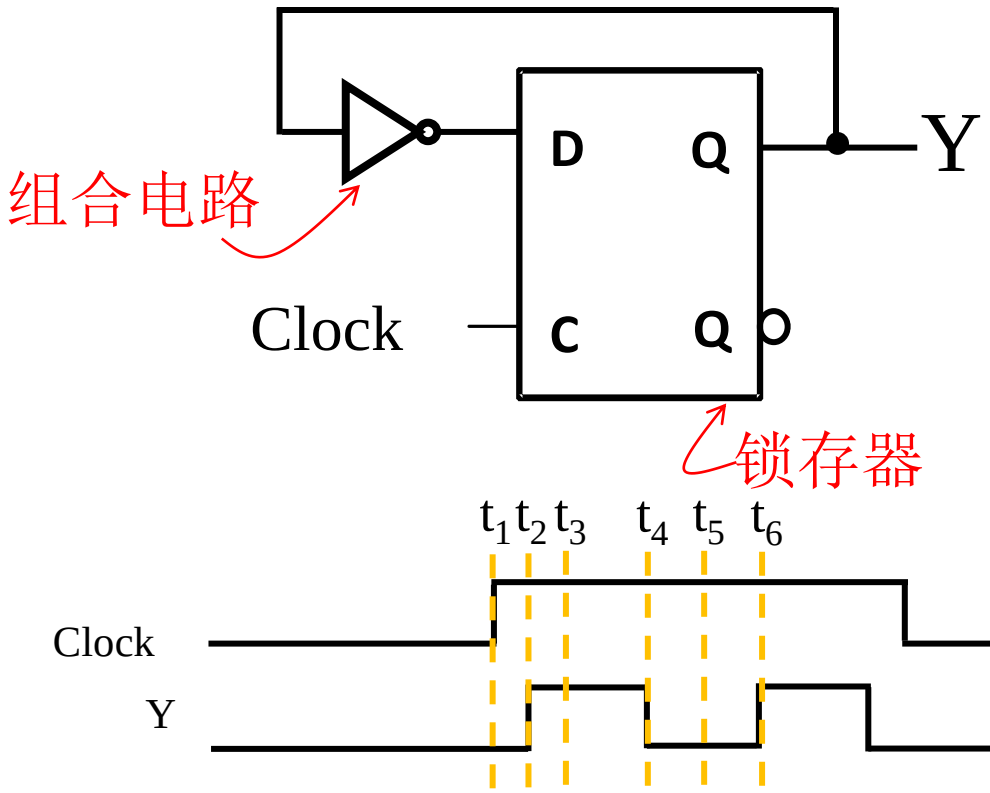
2.1 锁存器

- 锁存器作为存储单元的问题
  - 时钟信号C=1时，由锁存器输出到组合电路的反馈通路使得锁存器的输入存在不断变化的可能，使得存储元件进入不稳定状态



2.1 锁存器

- 锁存器作为存储单元的问题
  - 例子：组合电路+D锁存器



假设锁存器电路传播延迟$d_L=t_2-t_1$，组合电路传播延迟$d_C=t_3-t_2$，**那么**

时序行为分析：

- 初始时C=0, Y=0, D=1
- t_1时刻，C->1, Y=0, D=1
- t_2时刻，C=1, Y=1, D=1
- t_3时刻，C=1, Y=1, D=0
- t_4时刻，C=1, Y=0, D=0
- t_5时刻，C=1, Y=0, D=1
- t_6时刻，C=1, Y=1, D=1
- …

不稳定状态！

2.2 触发器

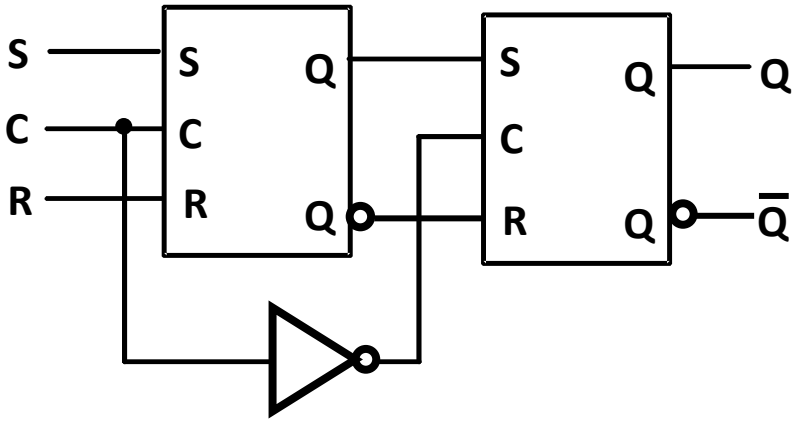
- **触发（trigger）**
  - 输入信号值的改变可以控制内部锁存器的状态，这种现象称为触发。
- **触发器**
  - 泛指用于记忆1位二进制信号的基本单元电路
  - 锁存器是触发器的基本构成部分
    - 脉冲式触发触发器：主从S-R触发器、主从D触发器
    - 边沿触发式触发器：边沿触发的D触发器、…
    - 直接输入
    - …

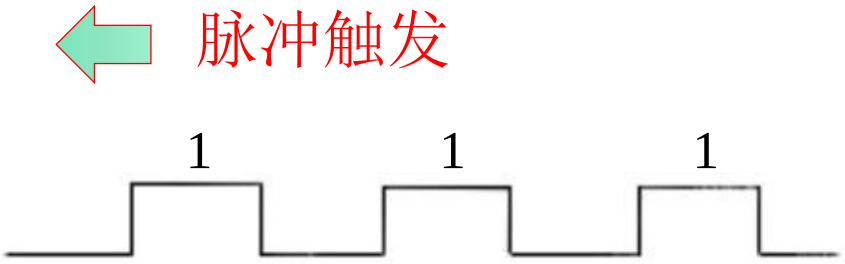
2.2 触发器

2.2.1 脉冲触发式触发器（主从触发器）

主从S-R触发器

- 由两个时钟S-R锁存器串联而成, 其中第二个锁存器的时钟信号由反相器取反
- 从左到右的输入到输出路径被时钟信号不同取值切断
  - 左：主锁存器，C=1时能够根据输入改变状态
  - 右：从锁存器，C=0时能够根据输入改变状态

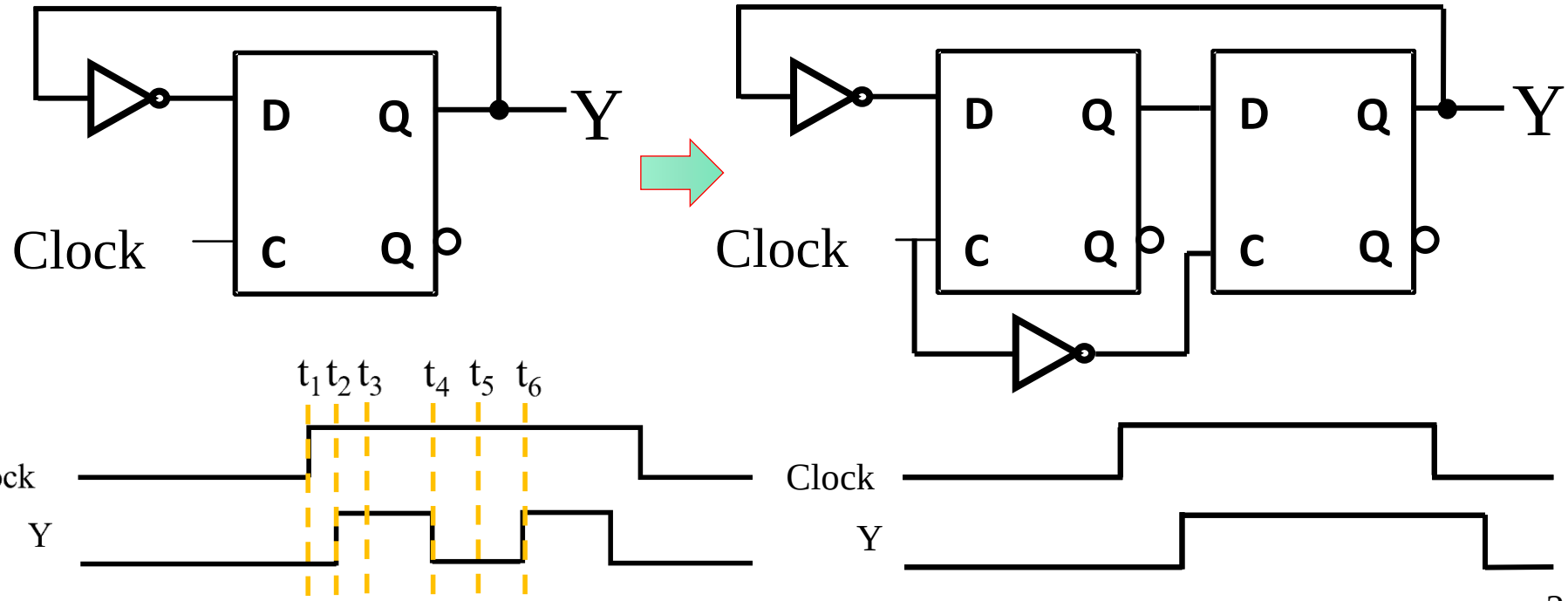




2.2 触发器

2.2.1 脉冲触发式触发器（主从触发器）

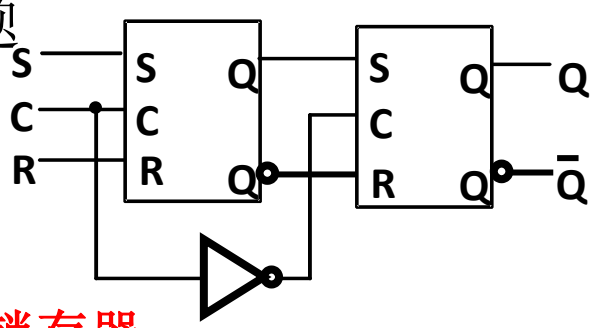
- 优点：
  - 如下例子中Y驱动D的行为在主从D触发器中会被阻止



2.2 触发器

2.2.1 脉冲触发式触发器（主从触发器）

- 缺点：
  - 触发器输出的改变被脉冲宽度延迟，使得整个电路变慢
  - C=1时1箝位（1s-catching）问题
    - 假设右端输出Q=0, 而后S变成1后回到0, R保持为0
      - 主锁存器置位为1
      - 当C=0时，**一个1被传输到从锁存器**
    - 假设右端输出Q=0, 而后S变成1后回到0, 而后R变成1后回到0
      - 主锁存器置位后复位
      - 当C=0时，**一个0被传输到从锁存器**

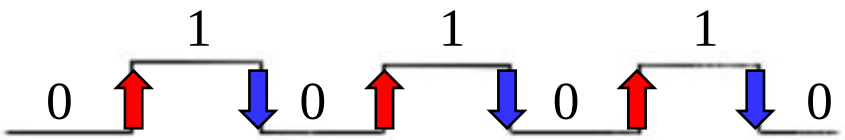


主锁存器置位为1的操作丢失，发生1箝位

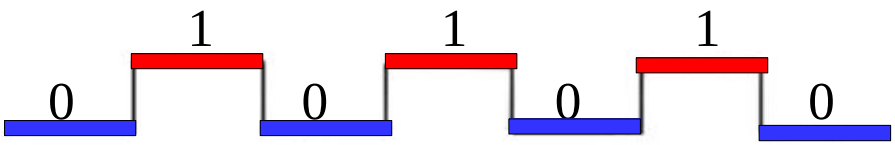
2.2 触发器

2.2.2 边沿触发式触发器

- 采用边沿触发来代替主从触发
- 边沿触发器忽略在常数水平的脉冲，只有在**时钟信号转换**的时候被触发
- 边沿触发式D触发器是目前使用最广泛的触发器



边沿触发式：靠控制信号“边沿”的变化来实现数据的保存

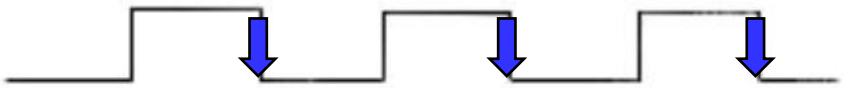


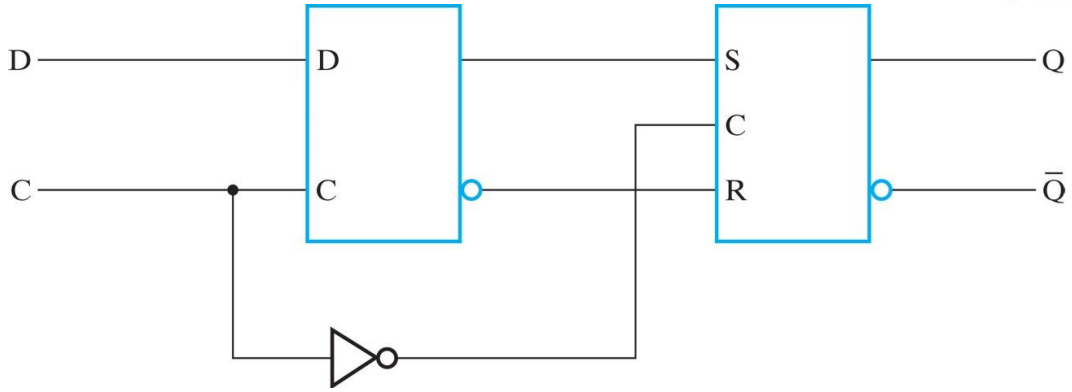
脉冲触发式：靠控制信号电平的高低来实现数据的保存

2.2 触发器

2.2.2 边沿触发式触发器

- 负边沿触发的D触发器：1跳变到0触发
  - 将主从S-R触发器的第一个时钟S-R锁存器换成时钟D锁存器
    - 触发器输出的改变和脉冲的负边沿联结在一起
    - 用D代替S和R，S-R主从触发器的延迟被避免，消除了1s-catching行为



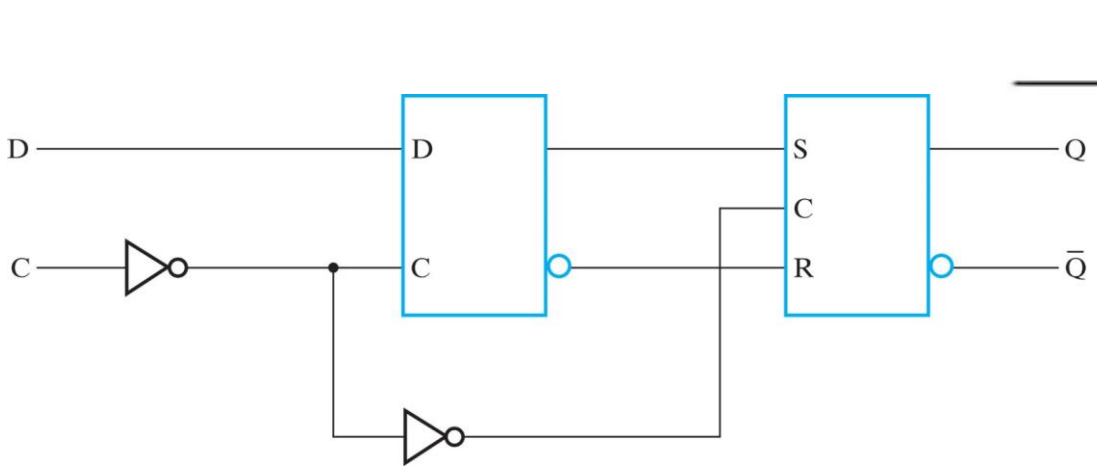


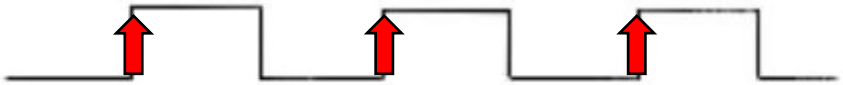
时序行为：
C: 1->0, S=D
C=0, 主锁存器关闭，
从锁存器打开, Q=D

2.2 触发器

2.2.2 边沿触发式触发器

- 正边沿触发的D触发器：0跳变到1触发
  - 给负边沿触发的D触发器的时钟信号增加一个反相器
    - 输出端Q变成在正时钟边沿符合时间限制的D值
    - 对大多数时序电路, 被看做是标准触发器





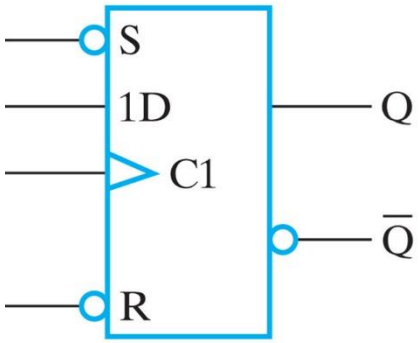
时序行为：
C: 0->1, S=D
C=1, 主锁存器关闭，
从锁存器打开, Q=D

2.2 触发器

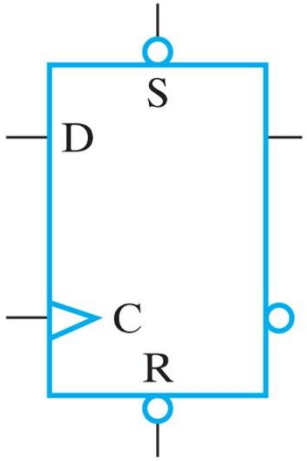
2.2.3 直接输入

- 直接置位或者预置，以及直接复位或者清零
- 在时钟正常运行之前，将触发器设置成一个初始状态

具备直接置位和复位功能的正边沿D触发器



S	R	C	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	Undefined	
1	1	↑	0	0	1
1	1	↑	1	1	0

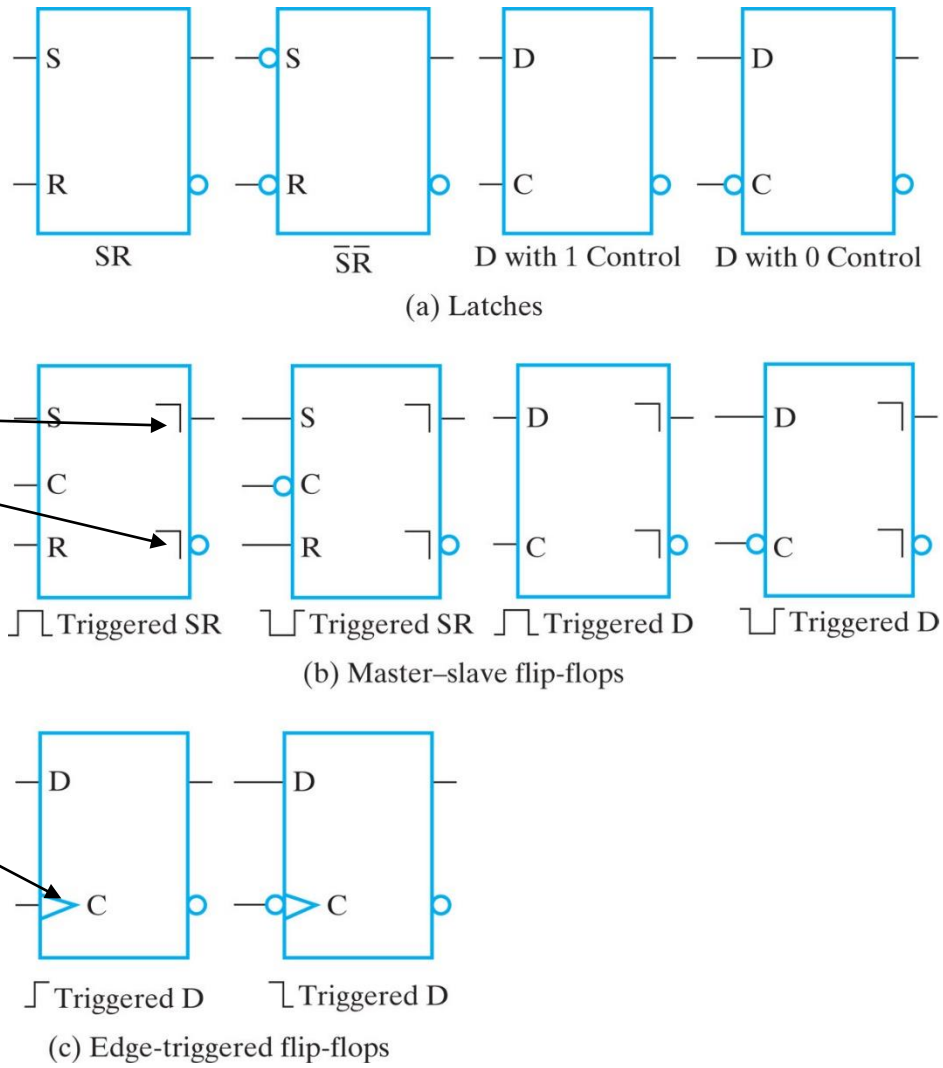


标准图形符号

简化的符号

存储单元的标准图形符号

- 主从:
  延迟输出指示
- 边沿触发:
  动态输入指示

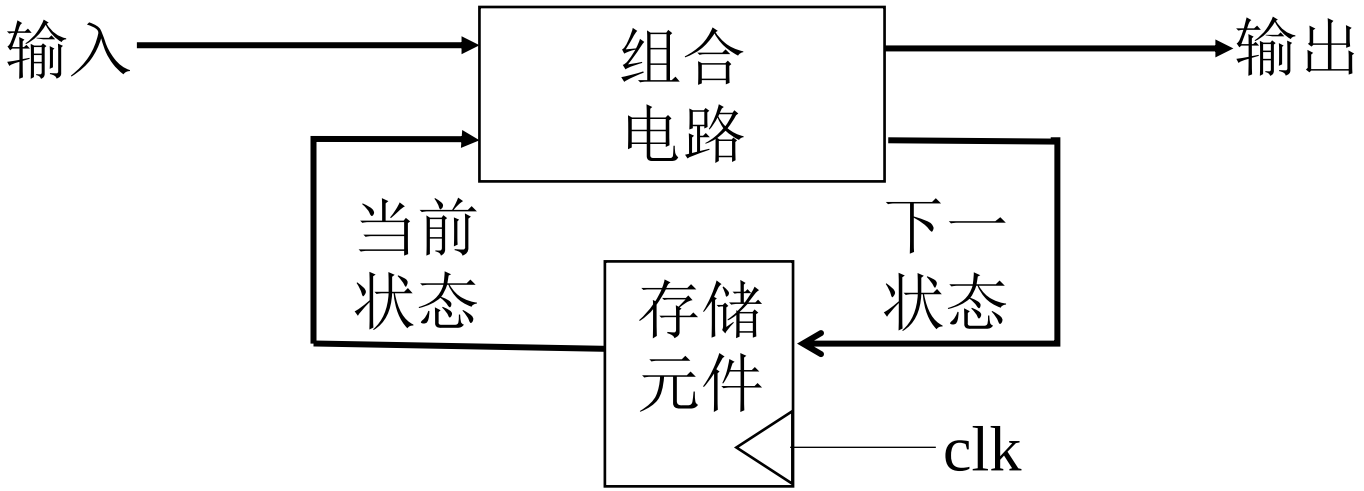


提纲

1. 时序电路简介
2. 基本存储单元
3. **时序电路分析**
4. 时序电路设计
5. 状态机设计

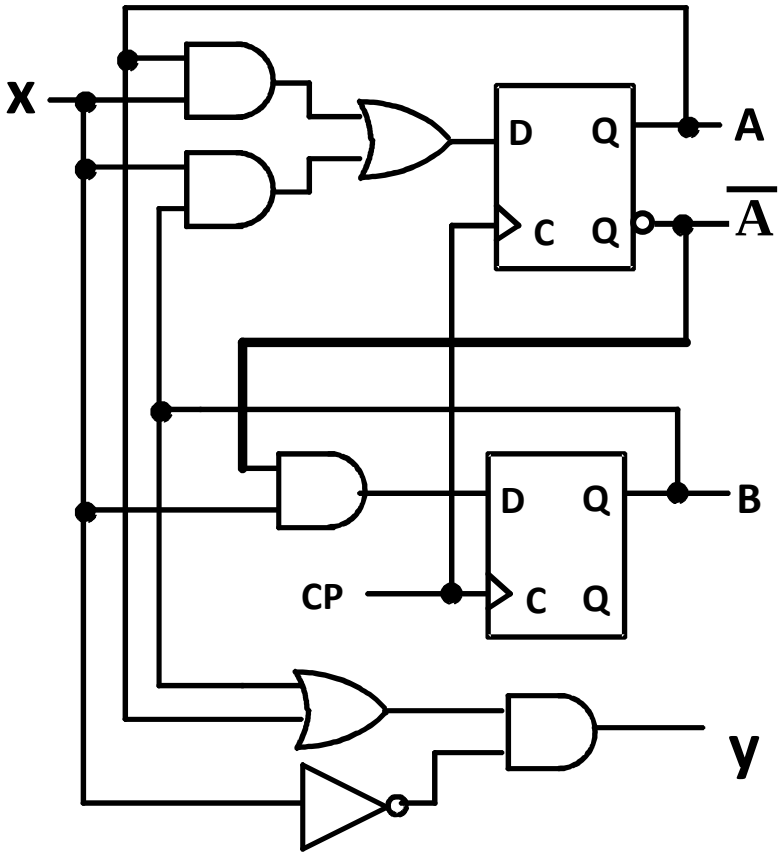
3.1 基本模型

- 电路输入、输出及当前状态的时序模型
  - 当前状态（时间t）存储在触发器存储元件
  - 下一状态（时间t+1）是当前状态和输入的布尔函数
  - 时间t时的输出是t时状态（或有t时输入）的布尔函数



3.1 基本模型

- 电路输入、输出及当前状态的时序模型
  - 例子
    - 输入: x(t)
    - 输出: y(t)
    - 状态: (A(t), B(t))
    - 输出函数?
    - 下一状态函数?



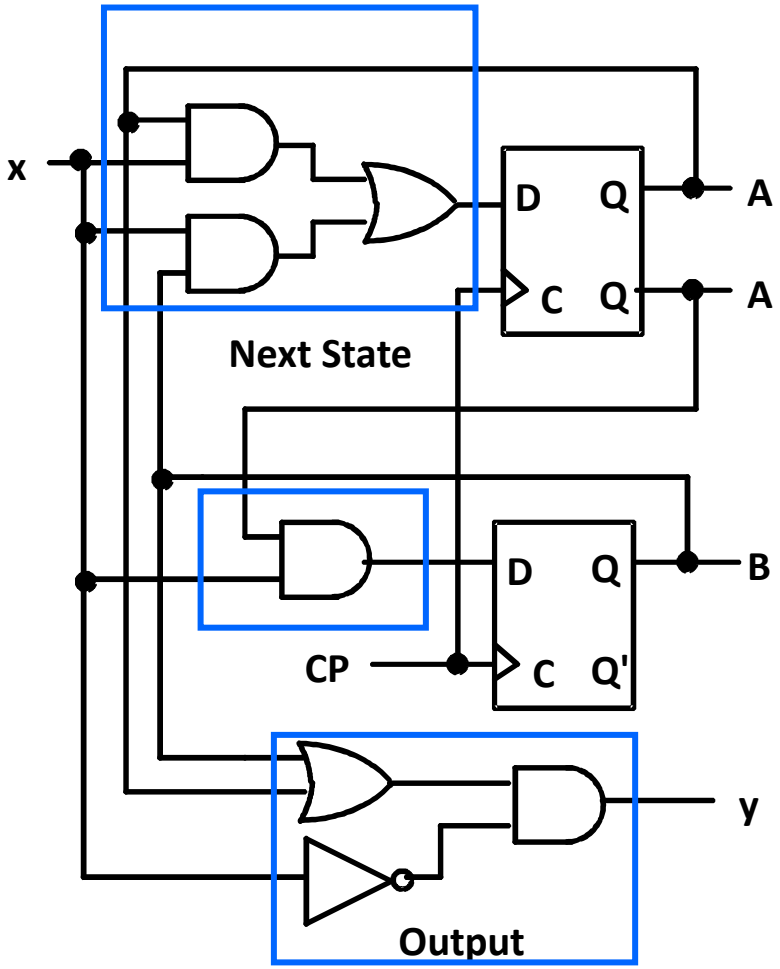
3.1 基本模型

- 电路输入、输出及当前状态的时序模型
  - 例子
    - $A(t+1) = A(t)x(t) + B(t)x(t)$
    - $B(t+1) = \overline{A}(t)x(t)$
    - $y(t) = x(t)(B(t) + A(t))$

状态函数：

$$\begin{cases} D_A = AX + BX \\ D_B = \overline{A}X \end{cases}$$

输出函数：$Y = (A + B)\overline{X}$



3.2 状态表

- **输入、输出、状态构成的多变量表**
  - 由以下四个部分组成:
    - 当前状态：状态变量的合法值
    - 输入：合法的输入组合
    - 下一状态：基于当前状态和输入的(t+1)时的状态值
    - 输出：基于当前状态和输入(有时)的输出值
  - 从真值表的角度理解:
    - 输入：输入+当前状态
    - 输出：输出+下一状态

状态表可以用下一状态和输出函数推导！

3.2 状态表

- 一维状态表
  - 输出栏是在当前时刻状态和输入的组合下的输出值；当前状态和输入合并在一个组合栏中
  - 例子：

$$D_A = AX + BX$$

$$D_B = \bar{A}X$$

$$Y = (A + B)\bar{X}$$

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

3.2 状态表

- 二维状态表
  - 当前状态列在表的左侧，输入则从左到右列在表的第一行组成二维表描述下一状态及输出
  - 例子：

$$D_A = AX + BX$$

$$D_B = \bar{A}X$$

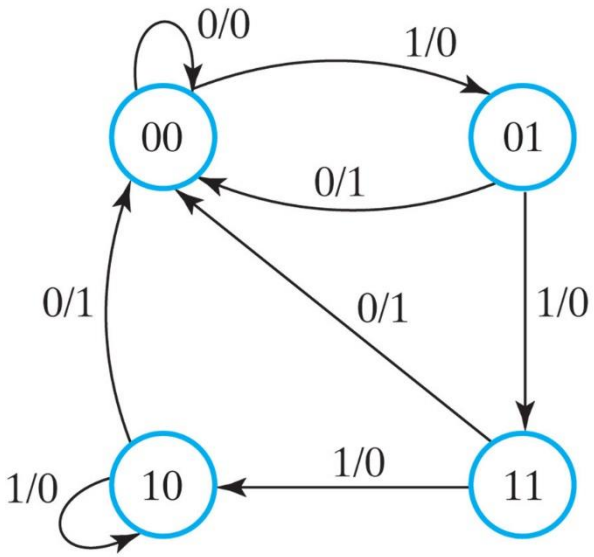
$$Y = (A + B)\bar{X}$$

Present State		Next State				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

如果交换后两行顺序，状态行和输入列符合格雷码的顺序，与卡诺图匹配！

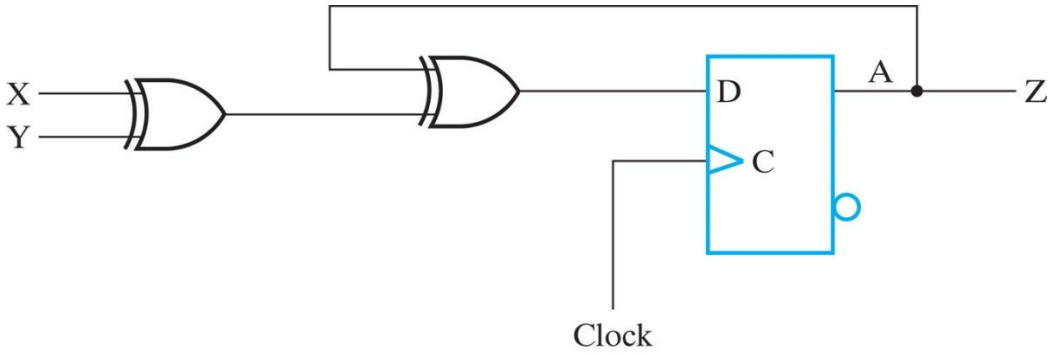
3.3 状态图

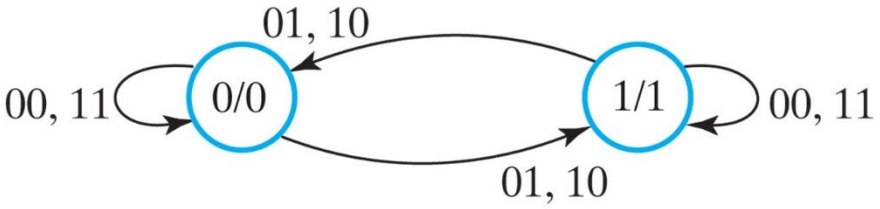
- 采用图形式表示状态表中的输入输出信息
  - 每个状态用一个圆圈表示
  - 每个状态转移用当前状态和下一状态之间的有向线段表示
  - 在有向线段上标注出输入表示造成状态转移的原因



3.3 状态图

- 采用图形式表示状态表中的输入输出信息
  - 例子





Present State	Inputs		Next State	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3.3 状态图

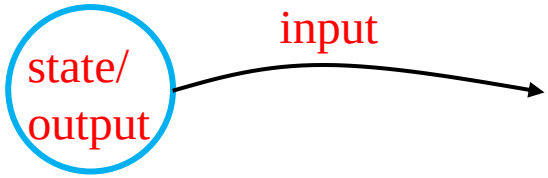
3.3.1 两种类型

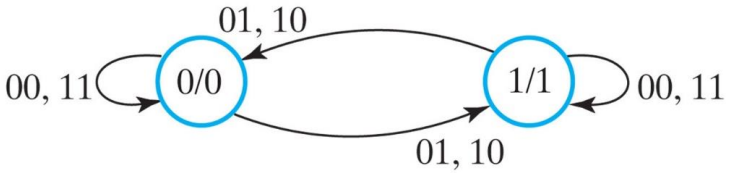
① Moore型电路状态图

- 输出仅仅是状态的函数

  output = *F*(state)

- 在表示状态的圆圈内标识输出



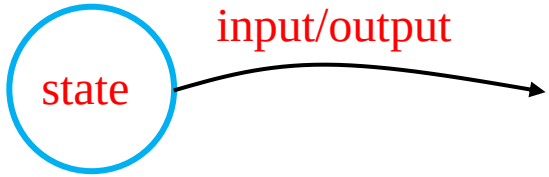


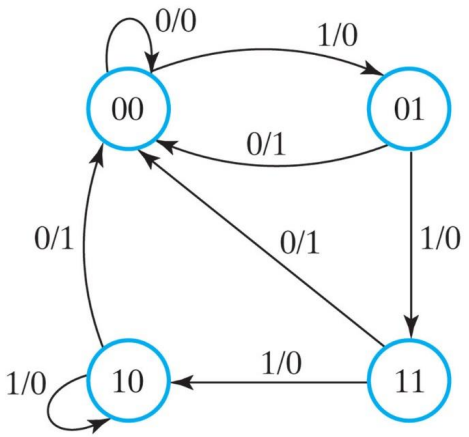
② Mealy型电路状态图

- 输出是输入和状态的函数

  output = *F*(input, state)

- 在表示状态转移的有向线段上标识输出

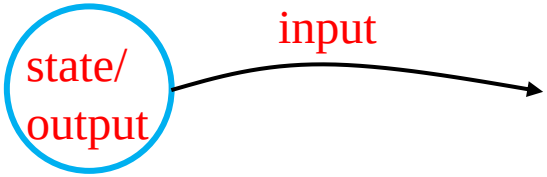




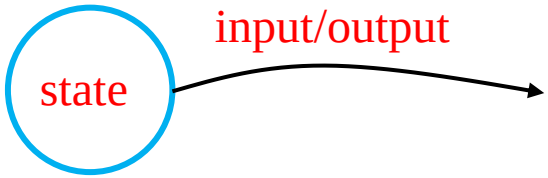
3.3 状态图

3.3.1 两种类型

- **包含输出的圆圈上:**
  - 状态/输出
  - **Moore 型**输出只依赖于状态
- **包含输出的有向线段上:**
  - 输入/输出
  - **Mealy型**输出决定于状态和输入



Moore型

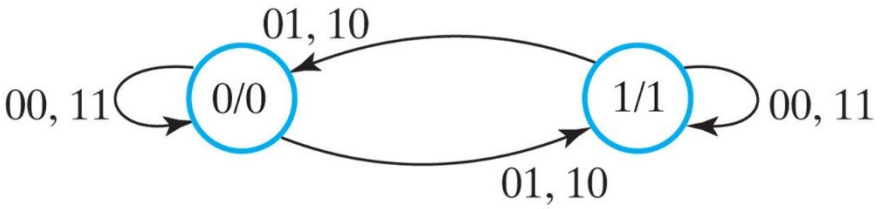


Mealy型

3.3 状态图

3.3.1 两种类型

- Moore型电路输出只依赖于当前状态，状态图对应于一维状态表



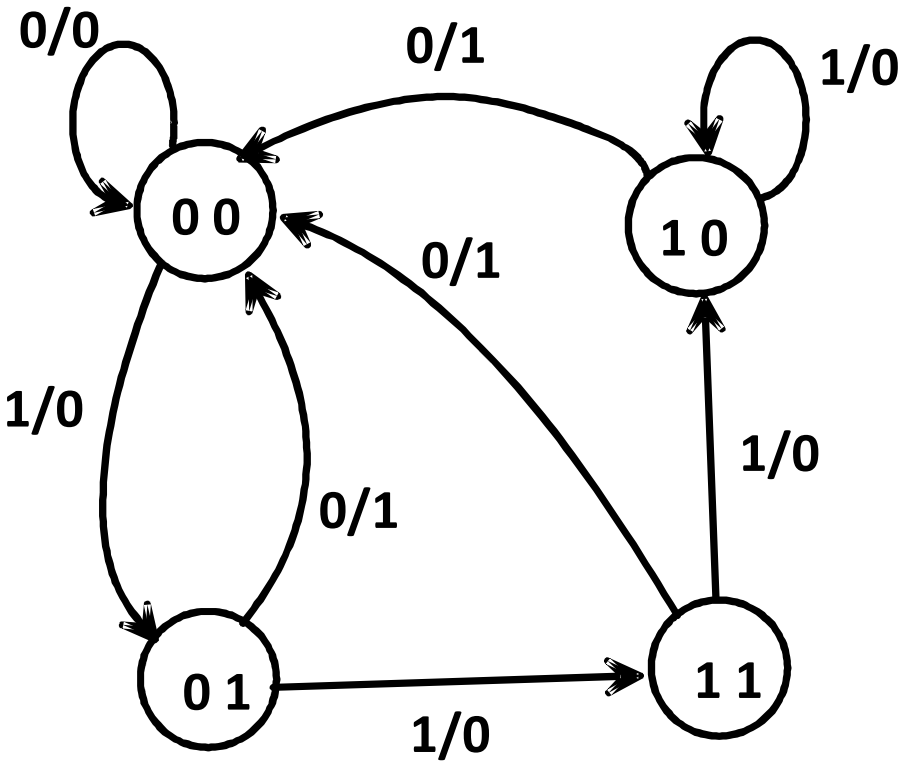
output = *F*(state)

Present State	Inputs		Next State	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3.3 状态图

3.3.1 两种类型

- 下面状态图属于哪种类型？

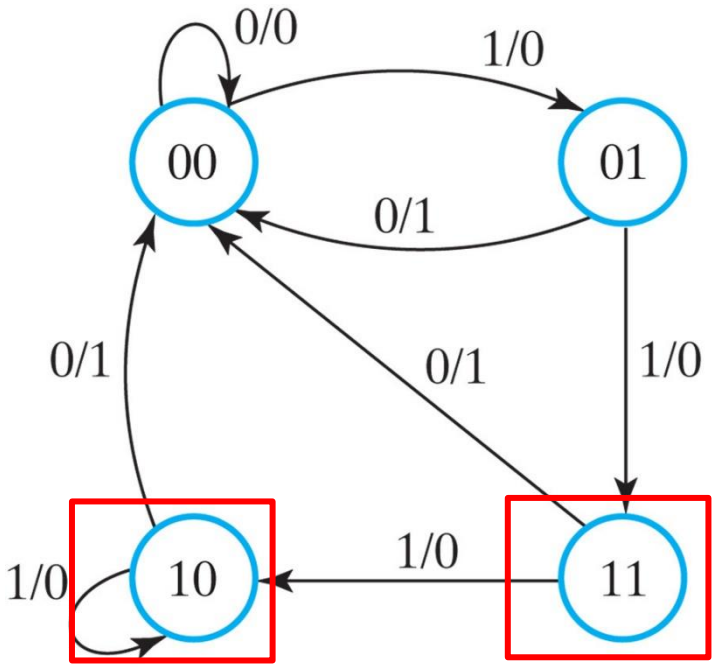


- 对大型电路，状态图容易混淆，
- 对小型电路，状态图比状态表容易看懂

3.3 状态图

3.3.2 等价状态

- 如果两个状态对每一个输入都产生相同的输出, 并且下一状态也是一致的，那么这两个状态是等价的



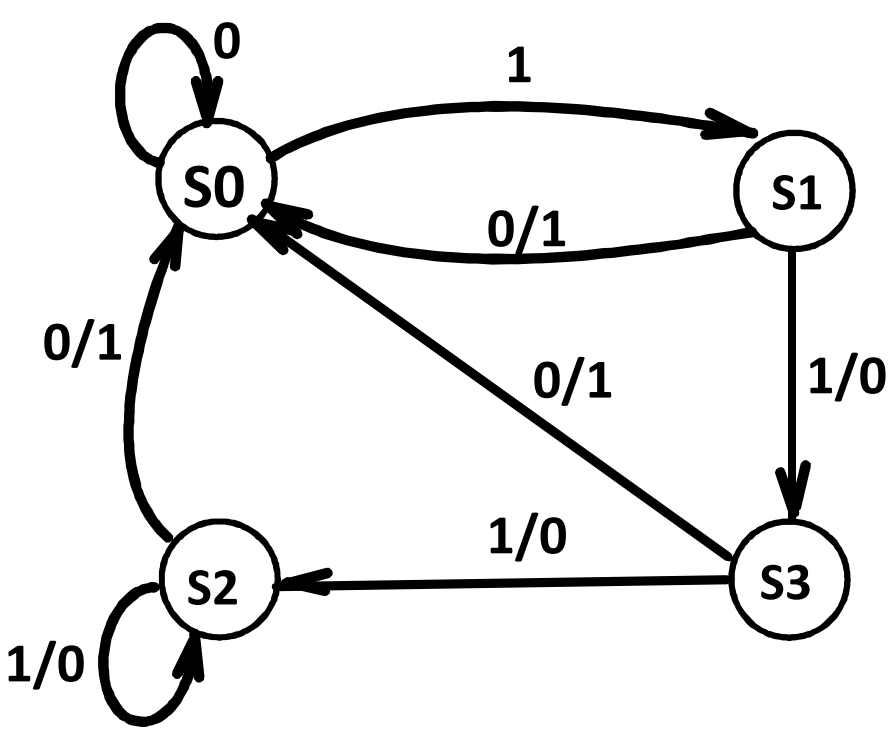
10
- 输入1，输出0，下一状态10
- 输入0，输出1，下一状态00

11
- 输入1，输出0，下一状态10
- 输入0，输出1，下一状态00

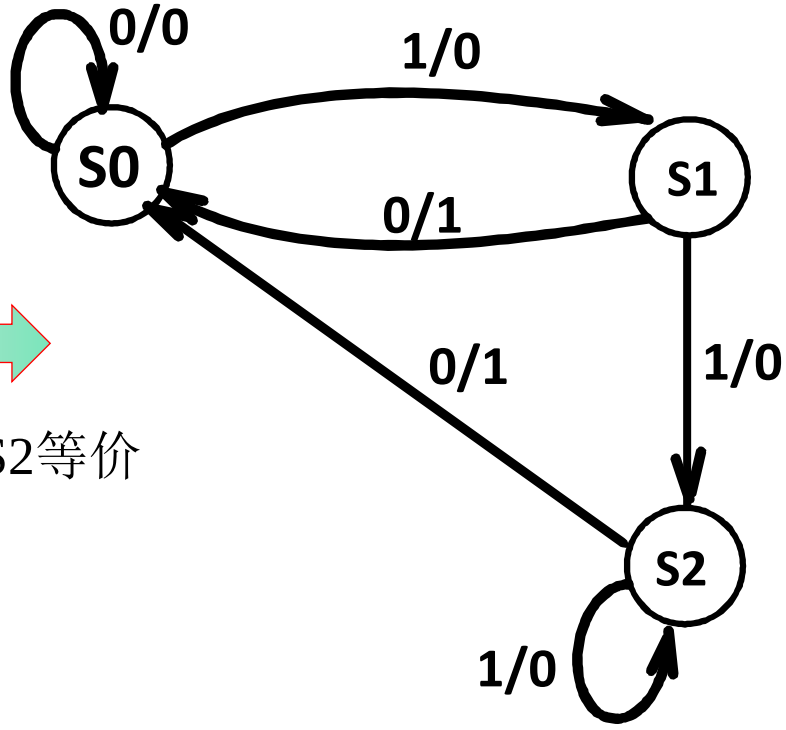
3.3 状态图

3.3.2 等价状态

- 彼此等价的两个状态可以合并为一个新状态



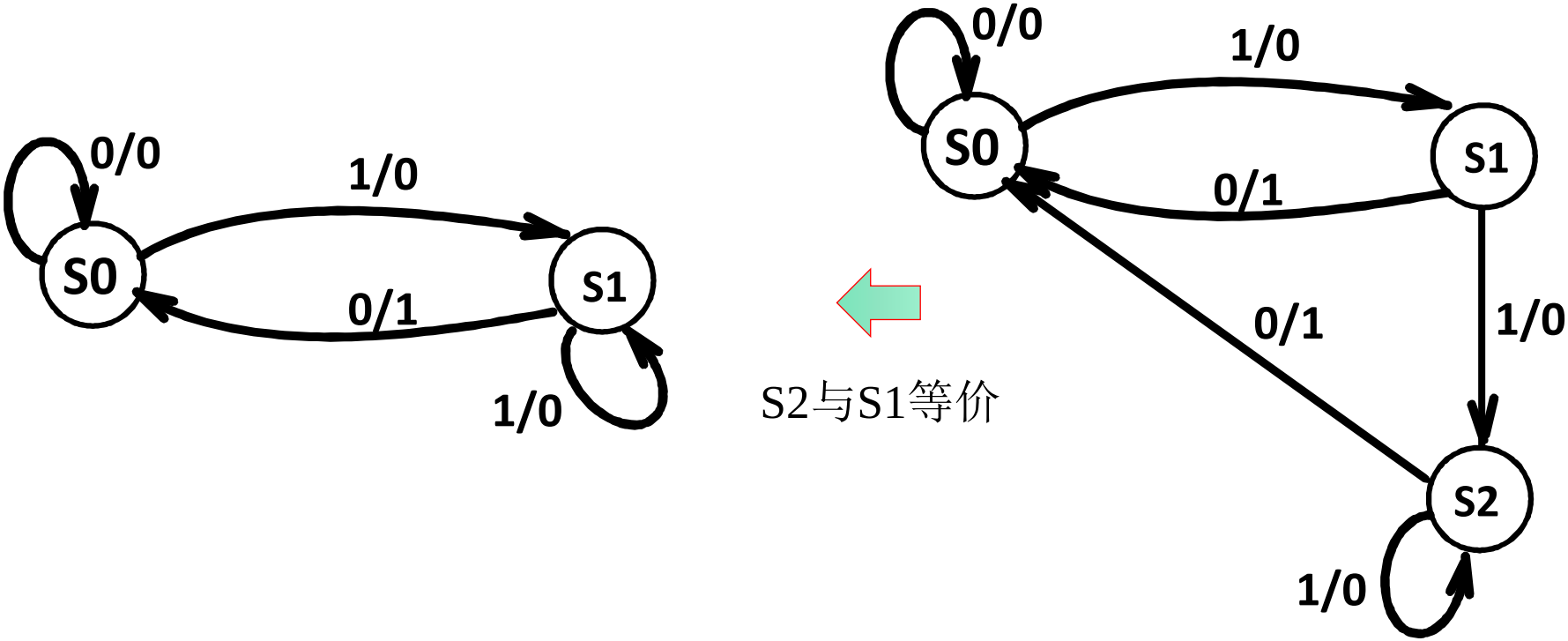
S3与S2等价



3.3 状态图

3.3.2 等价状态

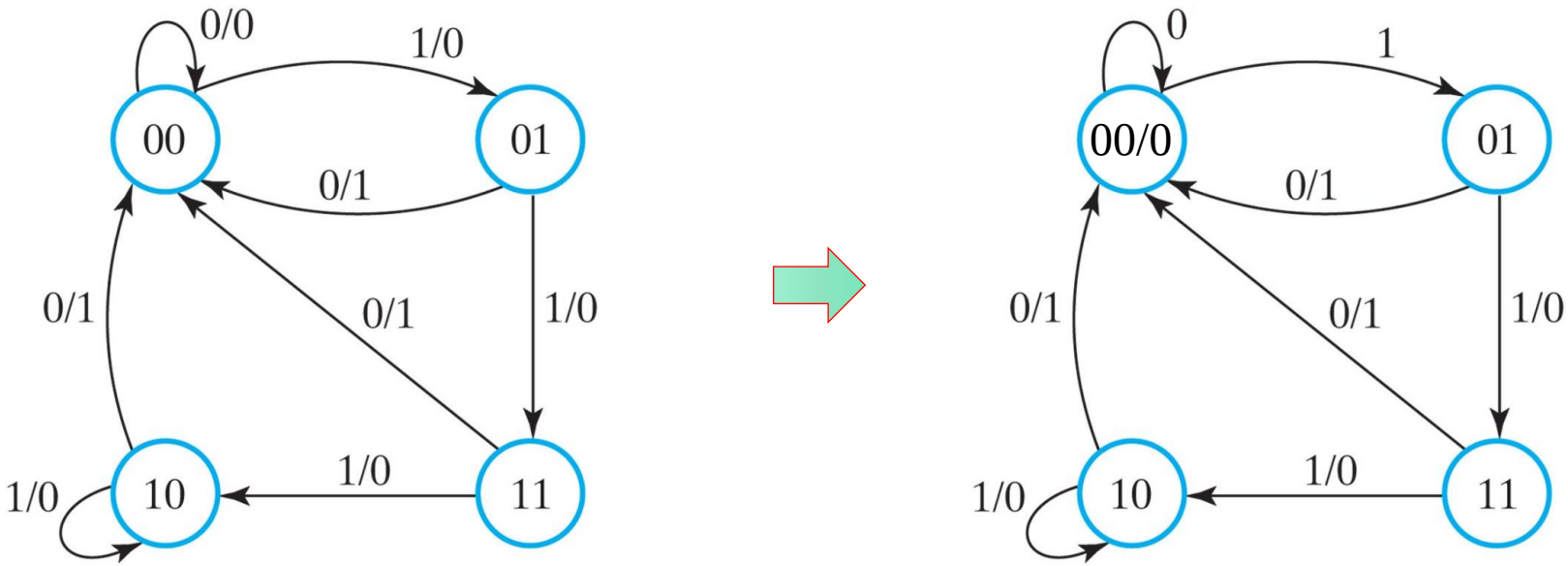
- 彼此等价的两个状态可以合并为一个新状态



3.3 状态图

3.3.3 混合型

- 在实际设计中, 一个电路中可能有些输出是Moore型的，有些是Mealy型
- 两种类型之间转换



状态 00：Moore型；状态 01, 10和11：Mealy型

3.4 时序电路分析

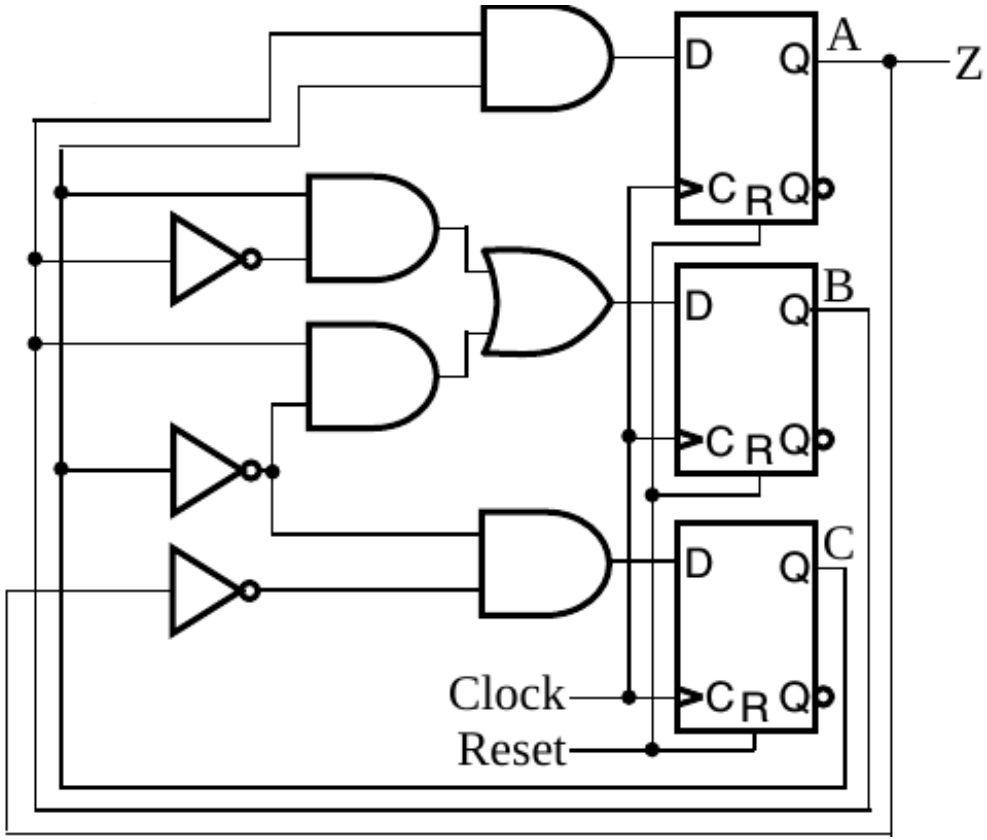
- 例子
  - 变量
    - 输入：无
    - 输出：Z
    - 状态：A、B、C
  - 布尔函数
    - 状态函数：

      A(t+1) = ?

      B(t+1) = ?

      C(t+1) = ?
    - 输出函数：

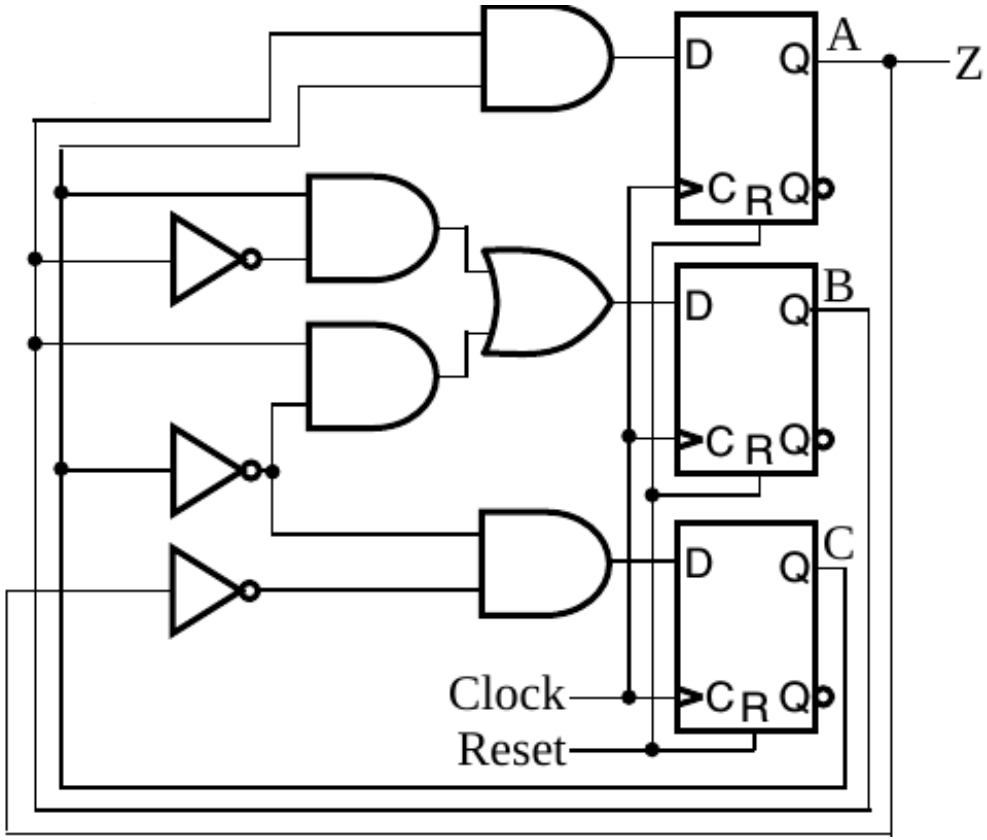
      Z = ?



3.4 时序电路分析

- 例子
  - 状态和输出函数

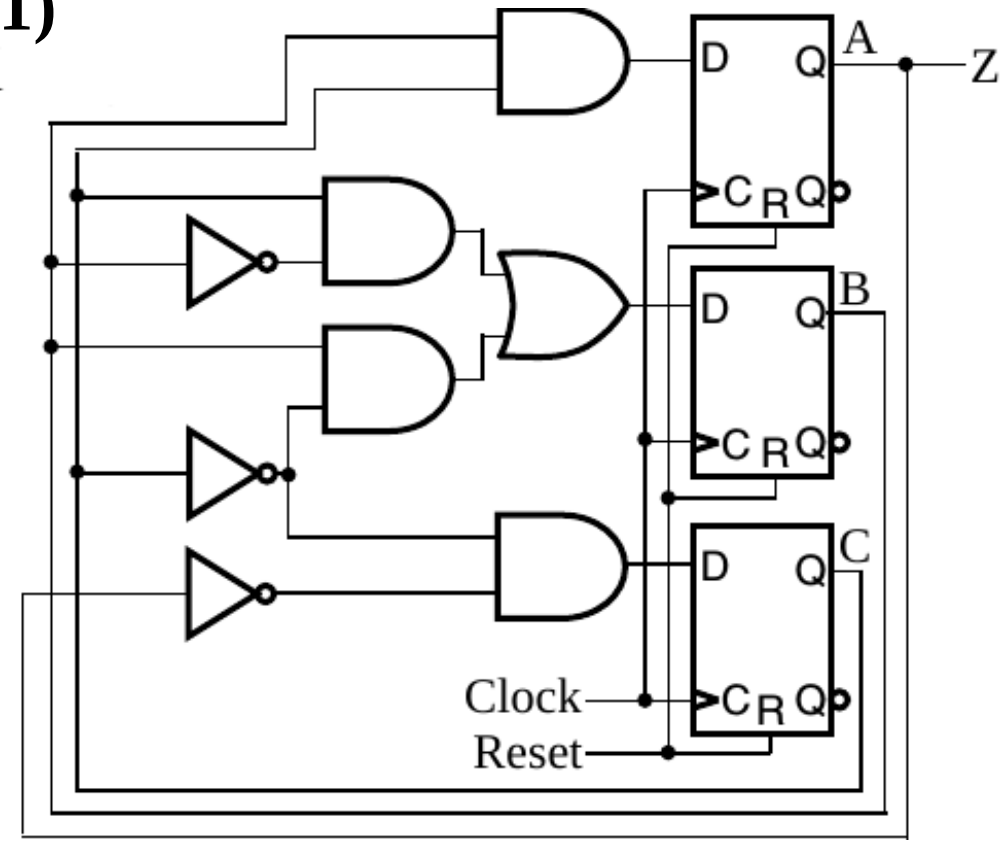
$$D_A = BC$$
$$D_B = \bar{B}C + B\bar{C}$$
$$D_C = \bar{A}\bar{C}$$
$$Z = A$$



3.4 时序电路分析

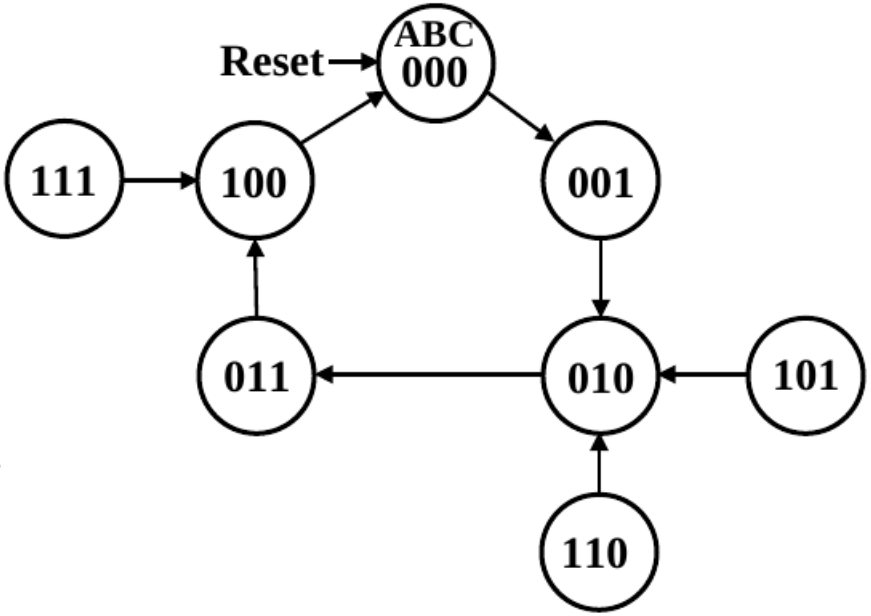
- 例子
  - 状态表 **X’ = X(t+1)**

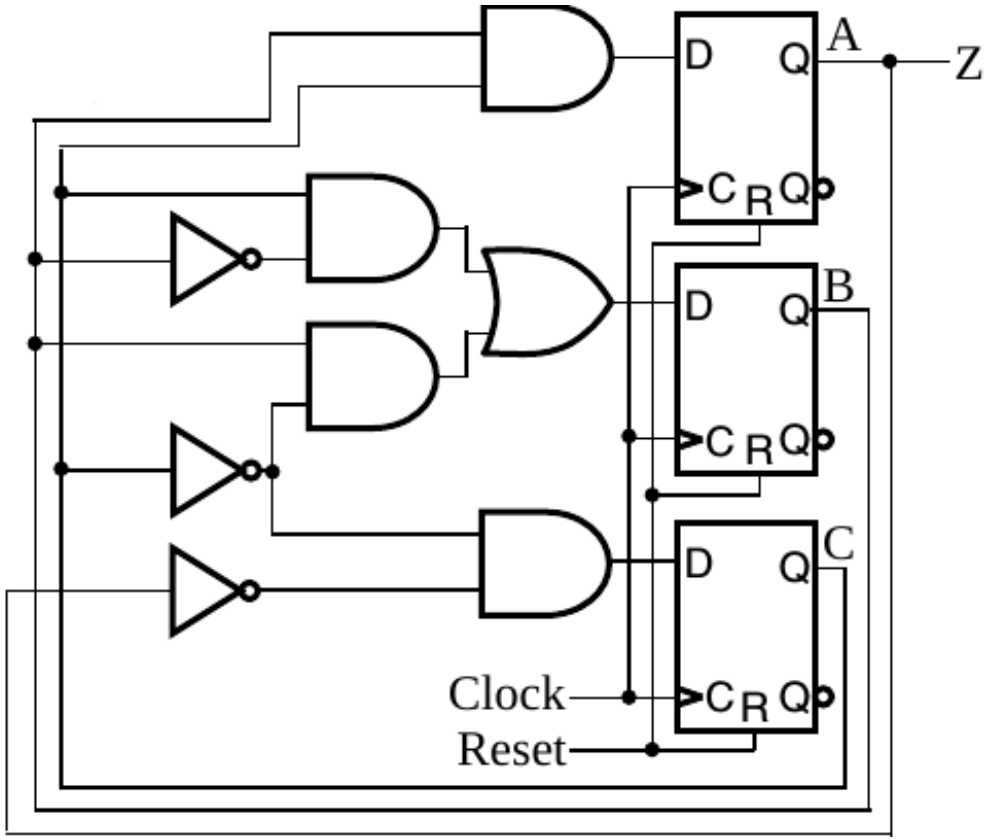
A B C	A’B’C’	Z
0 0 0	0 0 1	0
0 0 1	0 1 0	0
0 1 0	0 1 1	0
0 1 1	1 0 0	0
1 0 0	0 0 0	1
1 0 1	0 1 0	1
1 1 0	0 1 0	1
1 1 1	1 0 0	1



3.4 时序电路分析

- 例子
  - 状态图





  - 哪些状态用到了?
  - 电路布尔函数?

提纲

1. 时序电路简介
2. 基本存储单元
3. 时序电路分析
4. **时序电路设计**
5. 状态机设计

4.1 设计准则

- 组合电路 vs. 时序电路

组合电路

- 规范化
- 形式化
- 优化
- 工艺映射
- 验证

时序电路

- 规范化
- 形式化
- 状态分配
- 优化
- 工艺映射
- 验证

4.2 设计过程

- 具体步骤
  - ① **规范化：**规格说明
  - ② **形式化：**得到状态表或状态图
  - ③ **状态分配：**给存储单元的状态分配二进制码
  - ④ **确定触发器的输入方程：**
    - 选择触发器类型，并且根据状态表中的下一状态推导出触发器输入方程，也就是状态方程
  - ⑤ **确定输出方程：**
    - 根据状态表中的输出推导出输出方程
  - ⑥ **优化：**对方程进行优化
  - ⑦ **工艺映射：**
    - 从方程中得到电路并且映射到触发器和门工艺
  - ⑧ **验证：**验证最终设计的正确性

优化

4.2 设计过程

4.2.1 规范化

- 描述电路的时序行为（规格）
- 规格说明的形式
  - 文字描述
  - 数学描述
  - 硬件描述语言
  - 表格描述
  - 方程描述
  - 图表述

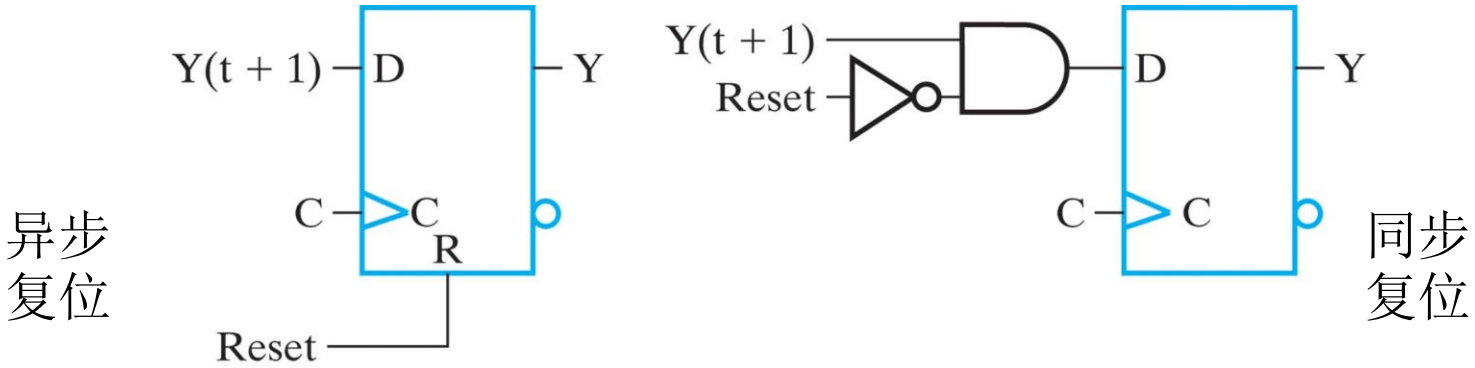
例：序列识别器　　001101011101……

➢ 一个时序电路, 其能在输入序列中发现目标序列时产生特定的输出, 如识别一个输入序列1101的出现

4.2 设计过程

4.2.2 形式化

- 状态：一系列加载到电路的历史输入的抽象
- 初始状态：提供一个硬件机制使电路从任何未知状态进入初始状态
  - 复位（Reset）信号



4.2 设计过程

4.2.2 形式化

- 状态：一系列加载到电路的历史输入的抽象
- 初始状态
- 获得状态图或状态表

例：序列识别器 001101011101……

➢ 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

  - 针对序列识别器开发一个过程, 这个过程能将问题描述转化成状态图。
  - 下一步, 状态图会被转换成状态表, 根据状态表来进行电路设计。

4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【分析】

- 初始状态, 没有任何的序列发生。
- 增加一个状态来识别第一个符号。
- 当有后续符号发生时, 增加新的状态来识别。
- 最终状态表示输入序列发生。
- 加入状态转移弧, 来指定当一个字符没有按照目标顺序出现时的动作。
- 增加其他的在非序列输入上的状态转移弧, 来转移到指定输入子序列发生的状态。

4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【分析】

- 1111101包含1101，“11” 是目标序列的子序列。因此，时序机在接受下一个字符时必须记住其已经遇到了两个1。
- 序列1101101也包含1101，其中初始子序列和最后子序列有重叠，即：1101101 或1101101
- 序列1101每次都需要被识别出来.

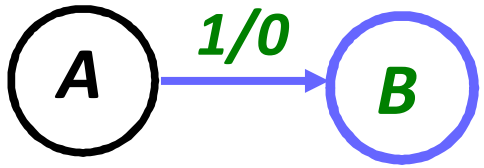
4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

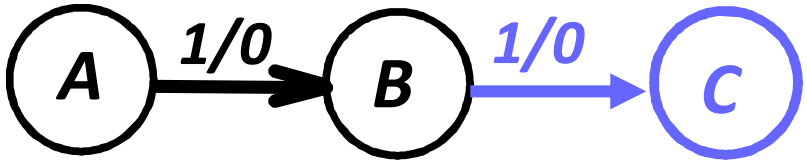
- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【形式化为状态图】



假设为Mealy型状态图

- 初始状态A。
- 序列中出现第一个“1”的状态记为B，即识别出序列中第一个“ 1 ”的状态。输出“0”表示还没有识别出整个序列。
- 序列中出现第二个“1”个状态记为C，即识别出序列中连续的两个“1”。输出“0”表示还没有识别出整个序列。



4.2 设计过程

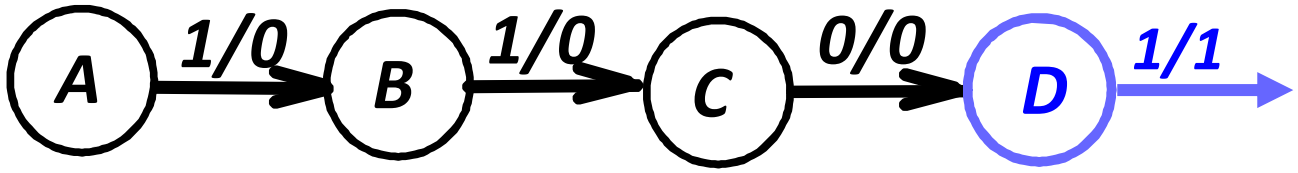
4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【形式化为状态图】

- 以此类推，后续出现“0”和“1”之后，识别出序列1101，因而从D出发的转移弧上输出为1。



问题：D出发的转移弧转向何处？ 1101101

4.2 设计过程

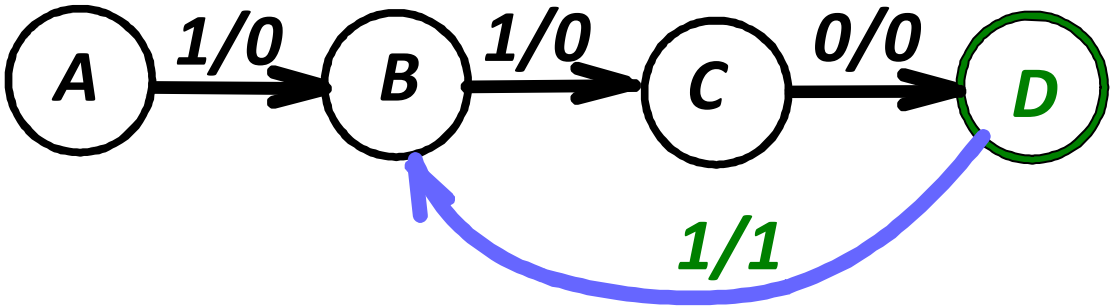
4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【形式化为状态图】

- 识别序列中的最后 1 个“1”也是也可以被看做是识别出来了 1 个“1”。因此 D 状态指向只识别出 1 个“1”的 A 后面的状态一致。



4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

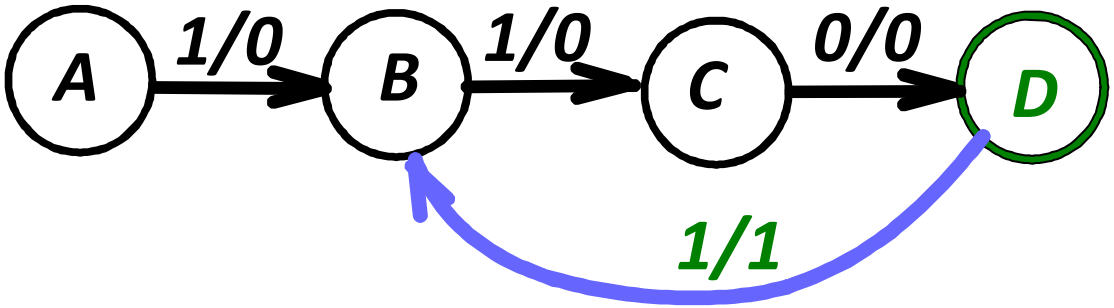
- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【形式化为状态图】

各状态的意义：

A：无子序列被识别；**B**：观察到子序列1

C：观察到子序列11；**D**：观察到子序列110



4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【形式化为状态图】

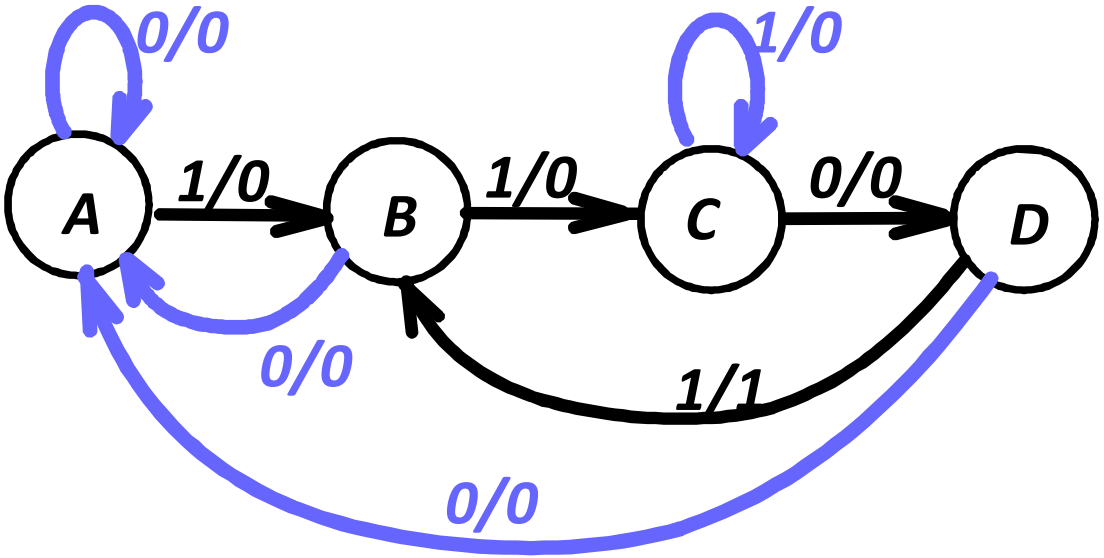
- 增加输入没有被列出的转移弧．哪些转移弧缺失？

从A出发的0?

从B出发的0?

从C出发的1?

从D出发的0?



4.2 设计过程

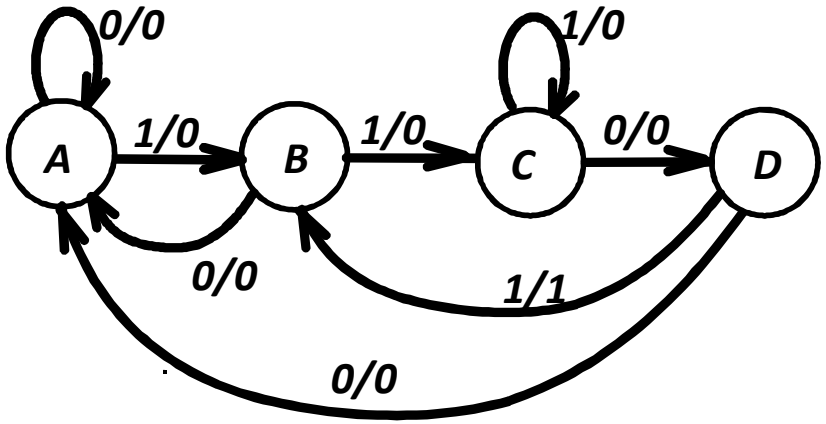
4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【由状态图得到状态表】

- 一个输入：序列的值
- 四个状态：A、B、C、D
- 一个输出：识别出1101



Present State	Next State x=0 x=1	Output x=0 x=1
A		
B		
C		
D		

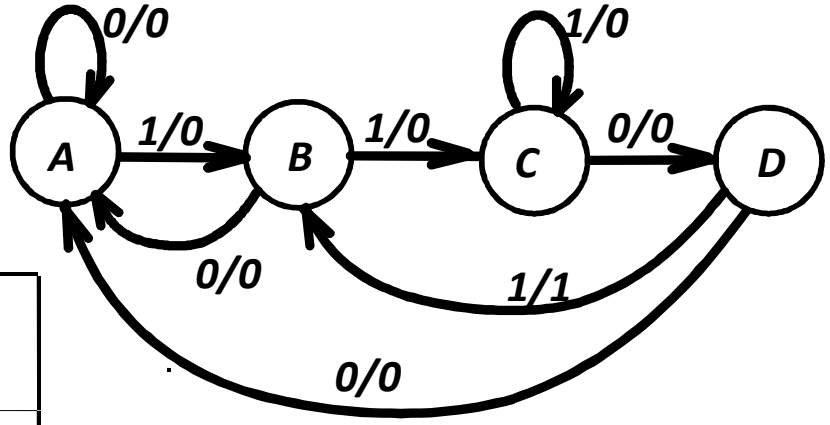
4.2 设计过程

4.2.2 形式化

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【由状态图得到状态表】

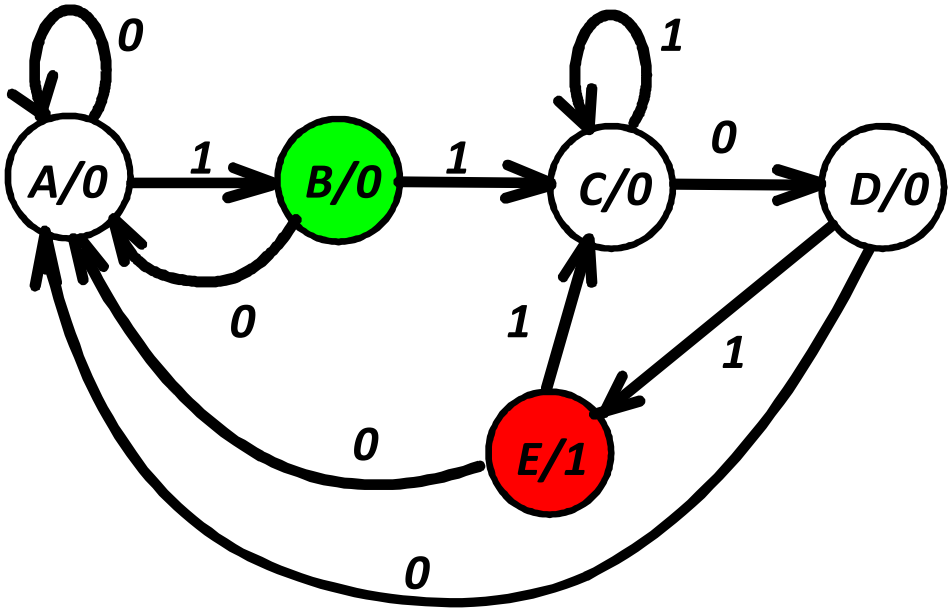


Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

4.2 设计过程

4.2.2 形式化

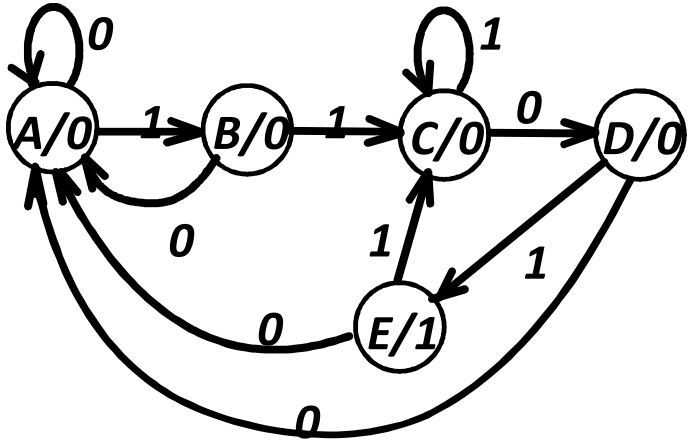
- Mealy型和Moore型的转化
  - Moore型的输出和状态是对应的，需要增加状态“E”表示对最后一个输入“1”的输出是“1”
  - 序列识别器的Moore模型比Mealy模型具有更多的状态



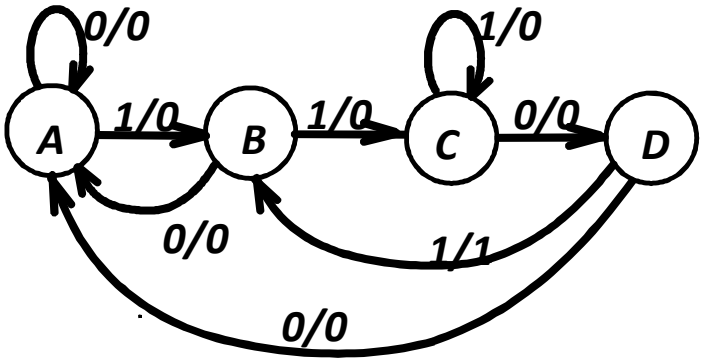
4.2 设计过程

4.2.2 形式化

- Mealy型和Moore型的转化
  - Moore型具有更多的状态：Moore is more



Present State	Next State x=0	Next State x=1	Output y
A	A	B	0
B	A	C	0
C	D	C	0
D	A	E	0
E	A	C	1



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

4.2 设计过程

4.2.3 状态分配（状态赋值）

- 通过编码将形式化得到的所有状态分配**唯一**的代码
  - m个状态至少需要n位二进制：$n \geq \lceil \log_2 m \rceil$
  - 至多存在$2^n - m$个未使用状态
- 常用的分配方式
  - **计数赋值**：A(00)、B(01)、C(10)、D(11)
  - **格雷码赋值**： A(00)、B(01)、C(11)、D(10)
  - **单热点（one-hot）赋值**：针对m个状态中的每个状态使用一个触发器，因此产生m位长的代码。其中一位为1，剩余为0

4.2 设计过程

4.2.3 状态分配（状态赋值）

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【计数赋值】

- A(00)、B(01)、C(10)、D(11)
- 赋值后的状态表

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1

4.2 设计过程

4.2.3 状态分配（状态赋值）

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【格雷码赋值】

- A(00)、B(01)、C(11)、D(10)
- 赋值后的状态表

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1

4.2 设计过程

4.2.3 状态分配（状态赋值）

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【单热点赋值】

- A(0001)、B(0010)、C(0100)、D(1000)
- 赋值后的状态表

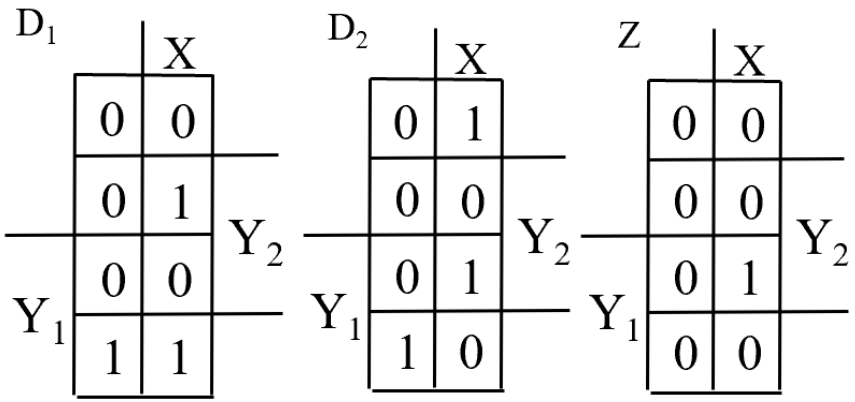
Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0001	0001	0010	0	0
0010	0001	0100	0	0
0100	1000	0100	0	0
1000	0001	0010	0	1

4.2 设计过程

- 4.2.4 确定触发器的状态方程
- 4.2.5 确定输出方程
- 4.2.6 优化

【两个D触发器（D_1/D_2）、计数赋值、输入Y_1/Y_2、输出Z】

Present State Y_2 Y_1	Next State x = 0 D_2D_1	Next State x = 1 D_2D_1	Output x = 0 Z	Output x = 1 Z
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1



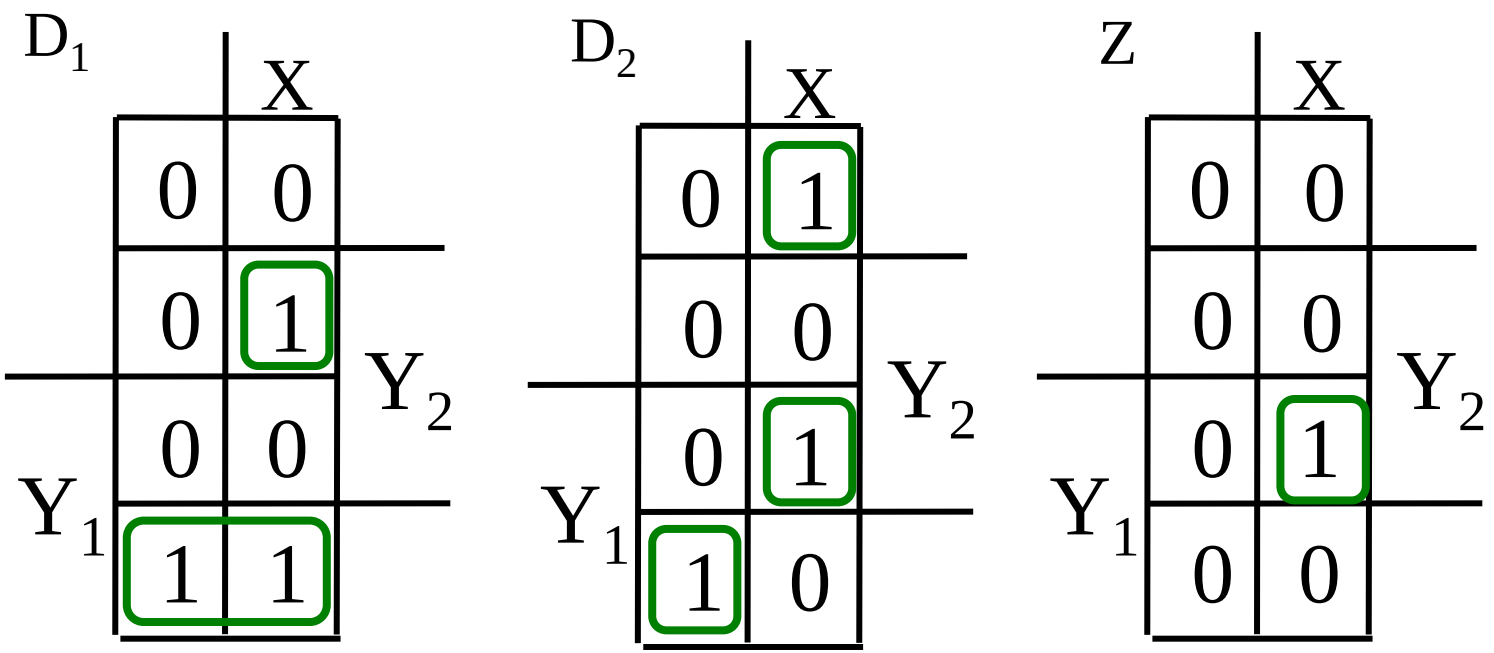
4.2 设计过程

- 4.2.4 确定触发器的状态方程
- 4.2.5 确定输出方程
- 4.2.6 优化

$$D_1 = Y_1\bar{Y}_2 + X\bar{Y}_1Y_2$$
$$D_2 = \bar{X}Y_1\bar{Y}_2 + X\bar{Y}_1\bar{Y}_2 + XY_1Y_2$$
$$Z = XY_1Y_2$$

门输入成本G=22

【两个D触发器（D_1/D_2）、计数赋值、输入Y_1/Y_2、输出Z】



4.2 设计过程

- 4.2.4 确定触发器的状态方程
- 4.2.5 确定输出方程
- 4.2.6 优化

【两个D触发器（D_1/D_2）、格雷码赋值、输入Y_1/Y_2、输出Z】

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1

D_1

Y_1 \ Y_2	X=0	X=1
0	0	0
1	1	1
0	0	0

D_2

Y_1 \ Y_2	X=0	X=1
0	1	1
1	1	1
0	1	1

Z

Y_1 \ Y_2	X=0	X=1
0	0	0
1	0	0
0	0	0

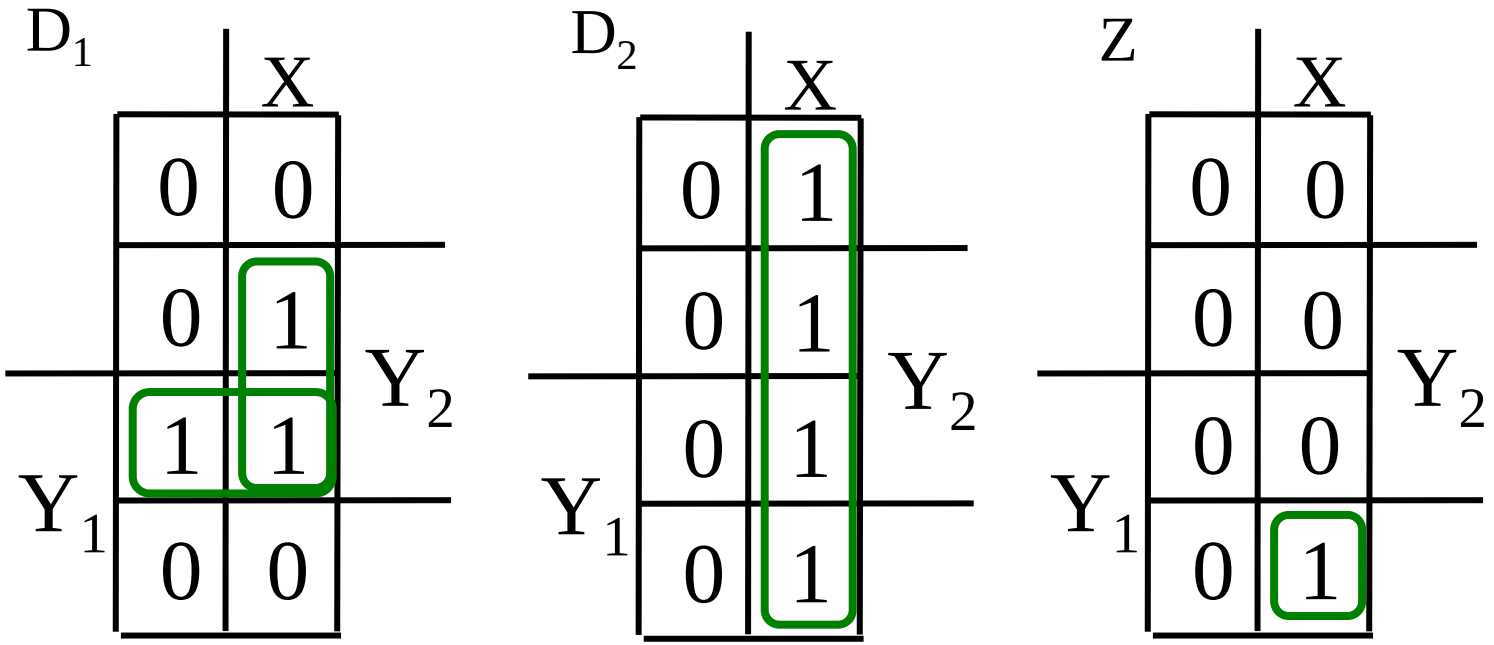
4.2 设计过程

- 4.2.4 确定触发器的状态方程
- 4.2.5 确定输出方程
- 4.2.6 优化

$$D_1 = Y_1Y_2 + XY_2$$
$$D_2 = X$$
$$Z = XY_1\overline{Y}_2$$

门输入成本G=9

【两个D触发器（D_1/D_2）、格雷码赋值、输入Y_1/Y_2、输出Z】



4.2 设计过程

- 4.2.4 确定触发器的状态方程
- 4.2.5 确定输出方程
- 4.2.6 优化

【四个D触发器（D_{0123}）、单热点赋值、输入Y_{0123}、输出Z】

- 四个状态的编码：(Y_0,Y_1,Y_2,Y_3) = 0001, 0010, 0100和1000。
- 在方程中，仅仅需要包含是 1 的变量。如编码位0001的状态，可以被表示为Y_0，而不是$Y_3\ Y_2\ Y_1\ Y_0$。
- 提供了简化的分析和设计。
- 组合逻辑可能会简单，但是触发代价会更高，可能不是代价低的方案。

4.2 设计过程

- 4.2.4 确定触发器的状态方程
- 4.2.5 确定输出方程
- 4.2.6 优化

【四个D触发器（D_{0123}）、单热点赋值、输入Y_{0123}、输出Z】

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0001	0001	0010	0	0
0010	0001	0100	0	0
0100	1000	0100	0	0
1000	0001	0010	0	1

$$D_0 = \overline{X}(Y_0 + Y_1 + Y_3) \text{ or } \overline{X}\,\overline{Y}_2$$

$$D_1 = X(Y_0 + Y_3)$$

$$D_2 = X(Y_1 + Y_2) \text{ or } X(\overline{Y_0 + Y_3})$$

$$D_3 = \overline{X}\,Y_2$$

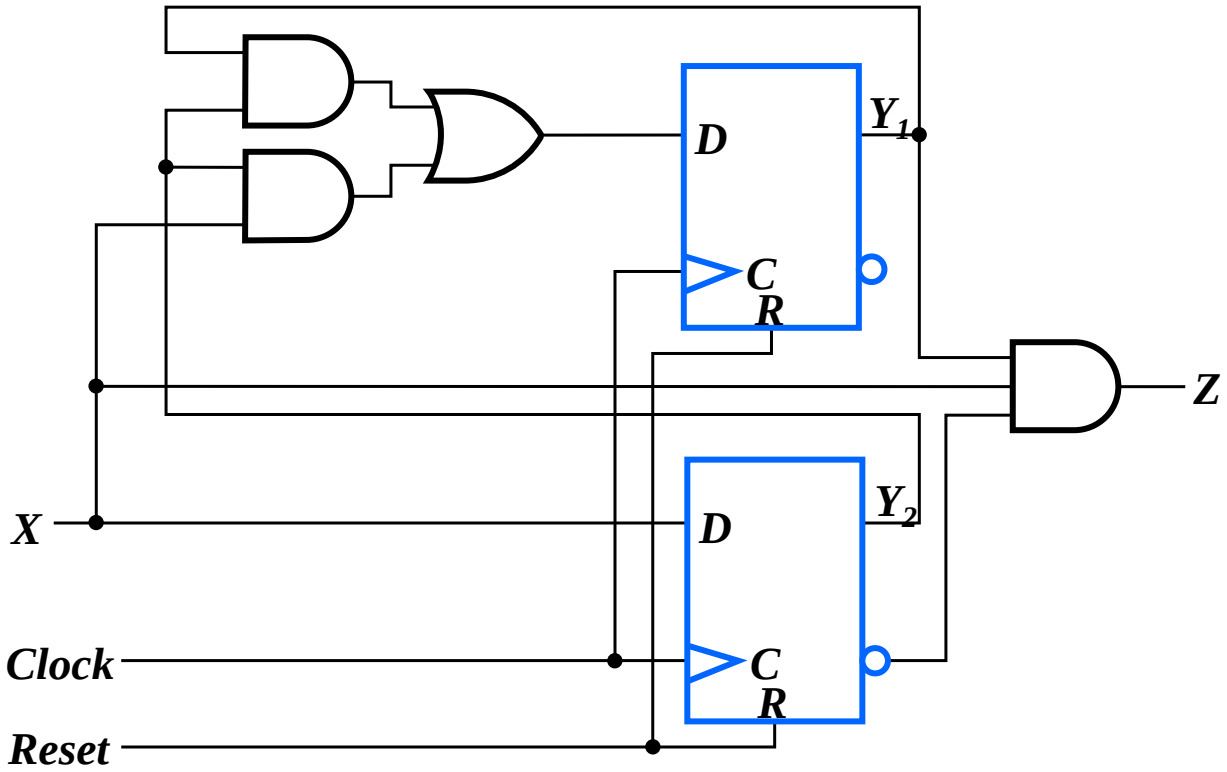
$$Z = XY_3$$

4.2 设计过程

$$D_1 = Y_1Y_2 + XY_2$$
$$D_2 = X$$
$$Z = XY_1\overline{Y}_2$$

- 4.2.7 工艺映射
  - 2个具有Reset的 D 触发器
  - 3个与门和1个或门

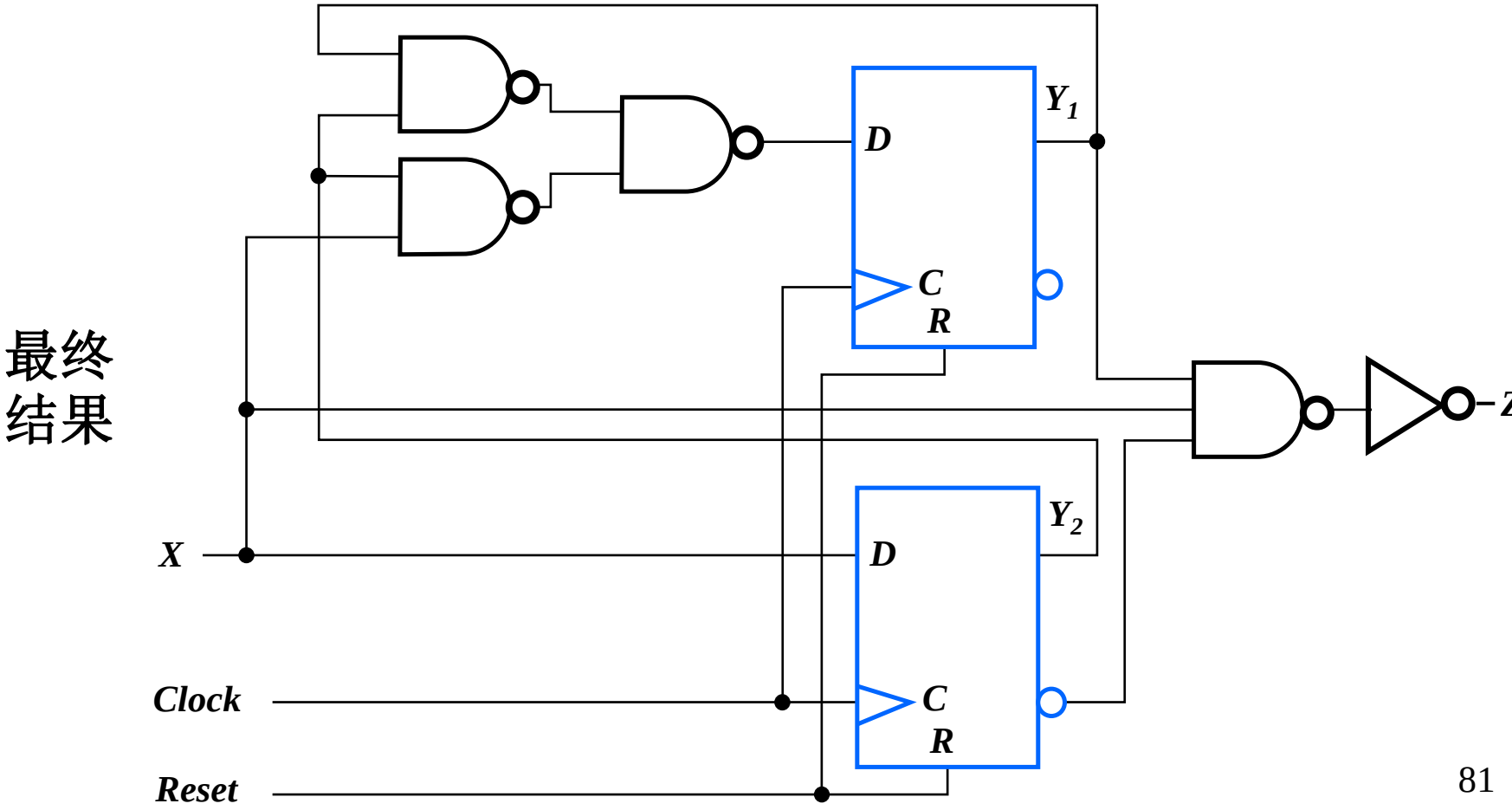
初始结果



4.2 设计过程

4.2.7 工艺映射

- 与非门、非门、D触发器

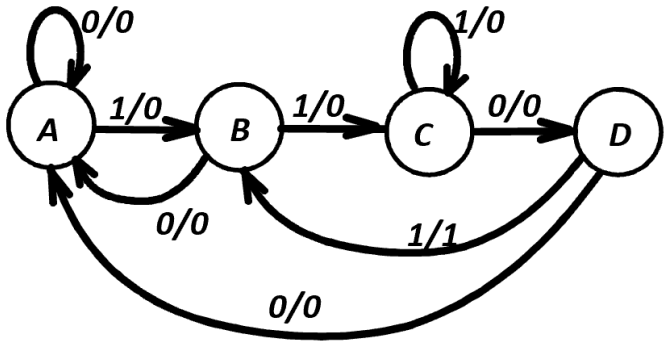


4.2 设计过程

4.2.8 验证

- 时序电路可以通过呈现电路产生的原始状态图或状态表进行验证
  - **手工验证**：对于较小的电路，可直接加载各种状态与输入的组合，并验证输出和下一状态是否正确。
  - **模拟验证**：通过一个输入组合序列和时钟信号，并在时钟上升沿后验证输出和下一个状态值。

4.2 设计过程



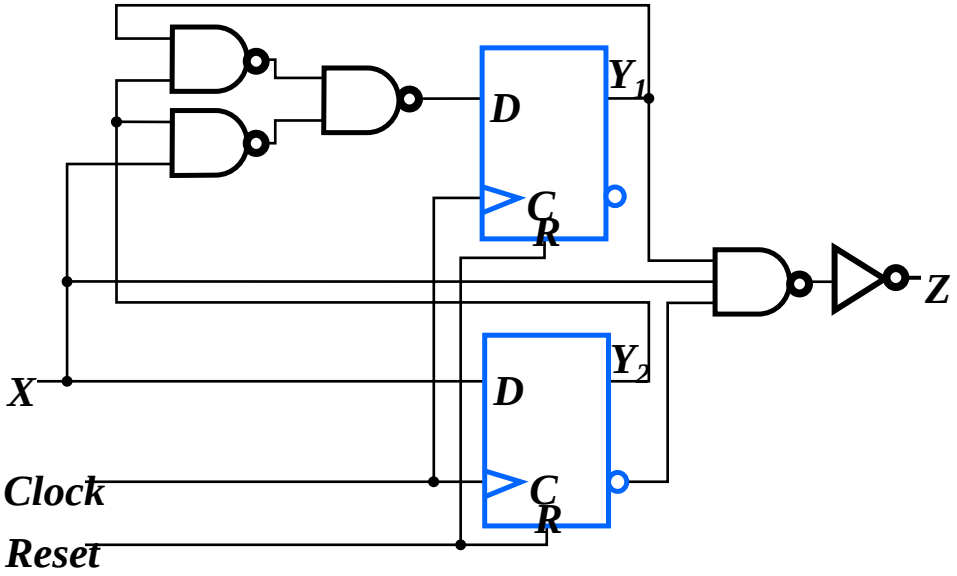
4.2.8 验证

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【手工验证】

- 4个状态和2个输入需要验证8种状态与输入组合
  - 复位状态：加载1，初始状态A(0,0)
  - X输入0，输出为0，下一状态A(0,0)
  - 在状态A下，X输入1，输出为0，下一状态B(0,1)
  - ……



4.2 设计过程

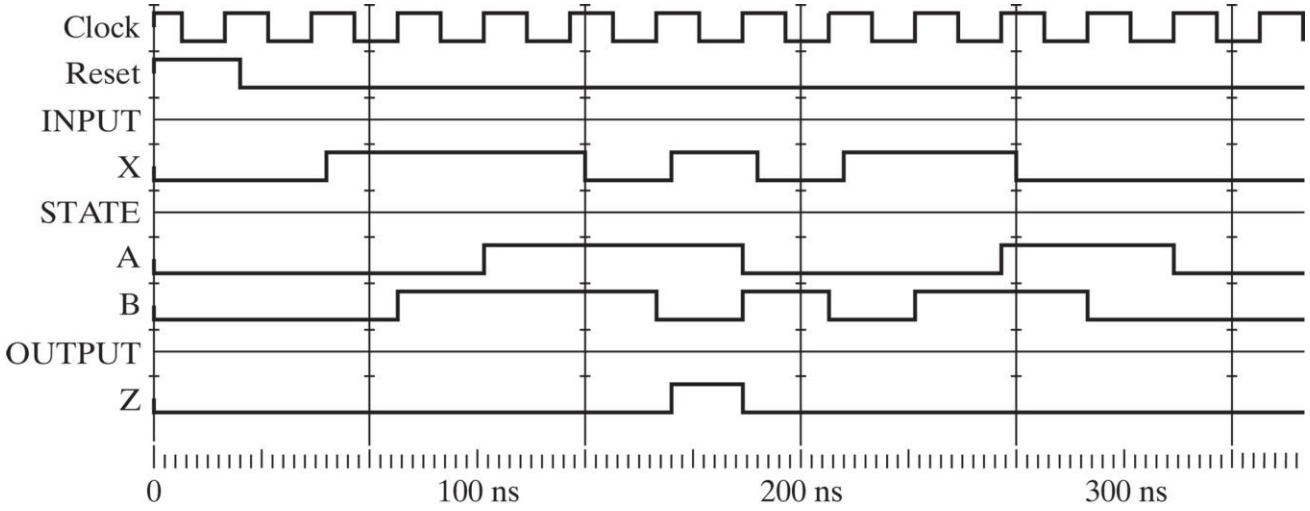
4.2.8 验证

例：序列识别器 001101011101……

- 一个时序电路, 能在输入序列中发现特定的输出, 如识别1101的出现

【模拟验证】

- 加载所有状态与输入组合对的输入序列
- 生成相应的输入序列和下一状态序列



提纲

1. 时序电路简介
2. 基本存储单元
3. 时序电路分析
4. 时序电路设计
5. **状态机设计**

本章内容

三. 状态机设计

- 3.1 传统的状态图和状态表
- 3.2 状态机图模型
- 3.3 约束检查
- 3.4 状态机图应用和设计

有限状态机

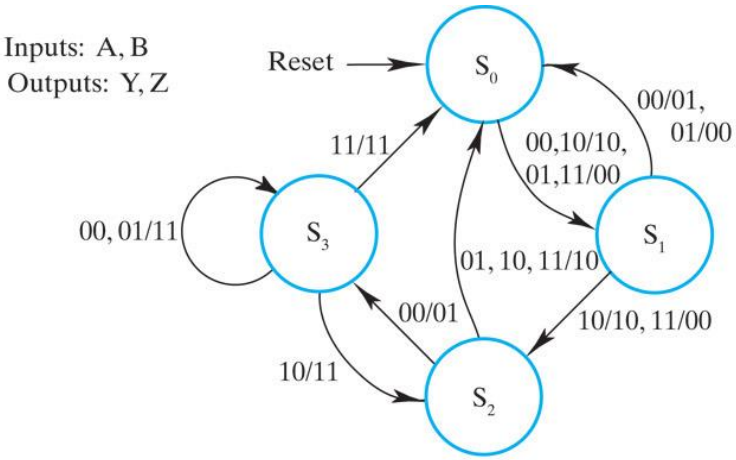
- 有限状态机(Finite State Machine, FSM)包含三个集合I, O和S, 两个函数f和 g：
  - I是输入的组合集合,
  - O是输出的组合集合,
  - S 是状态集合,
  - f是下一个状态函数 f(I, S),
  - g是输出函数f(S) [Moore 模型] 或者f(I, S) [Mealy模型]
- FSM是时序电路分析的基本数学模型.
- 传统的状态图和状态表是两种表示FSM的方式.

状态图和状态表

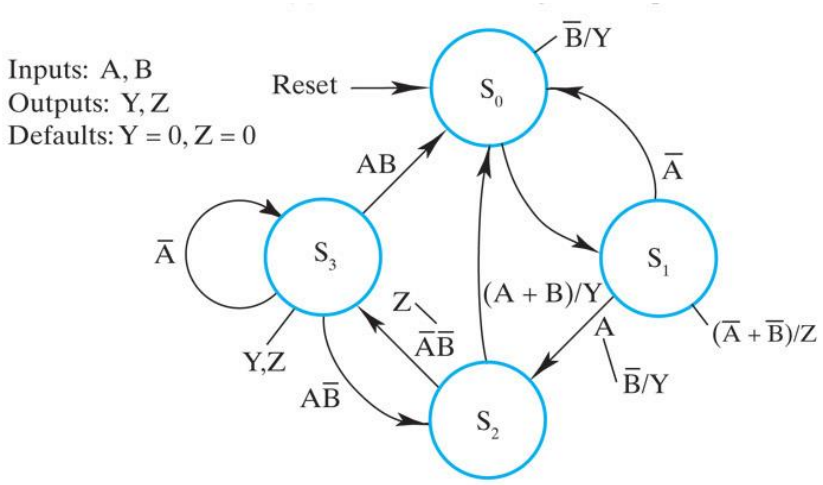
- 两种方式都需要:
  - 在定义下一状态时，都需要针对每个状态枚举所有的输入组合
  - 在定义Mealy输出时，都需要针对每个状态枚举所有的输入组合
  - 针对每个状态(Moore) 或输入－状态对(Mealy), 枚举所有对应的输出
- 对于状态图，所有的Mealy输出都需要在有向线段上指定.

状态图和状态表

- 这些对于很少输入和输出的时序电路是可接受.
- 对具有大量输入和输出的电路，状态表和状态图两种表达方式都难以处理.
  - ⑩只允许在有向线段输出使得Mealy的输出变得复杂.
- 采用状态机图（state machine diagram, SMD）



状态图



状态机图

本章内容

三. 状态机设计

- 3.1 传统的状态图和状态表
- 3.2 状态机图模型
- 3.3 约束检查
- 3.4 状态机图应用和设计

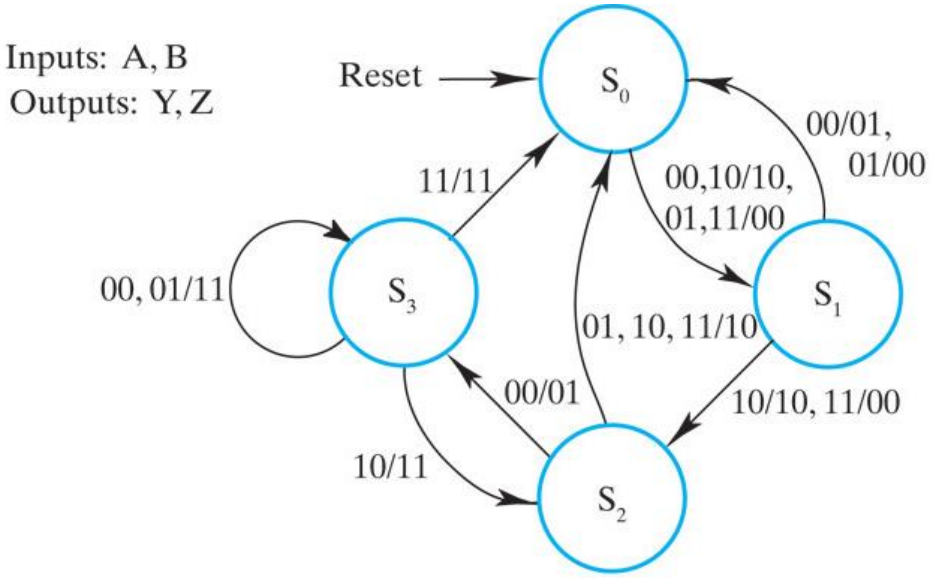
状态机图

- 和传统状态图类似，使用了状态结点和有向线段作为转移弧
- 增加了在状态上定义Mealy 输出的标识
- 基于输入条件，转移条件，输出条件产生输出行为：
  - 输入条件：采用输入变量的布尔表达式或方程来表示，其值为 0 或 1
  - 转移条件(Transition condition，TC)：在转移弧上的输入条件.
  - 输出条件(Output condition，OC): 如果其等于 1，则造成输出行为发生．否则，输出行为不发生.

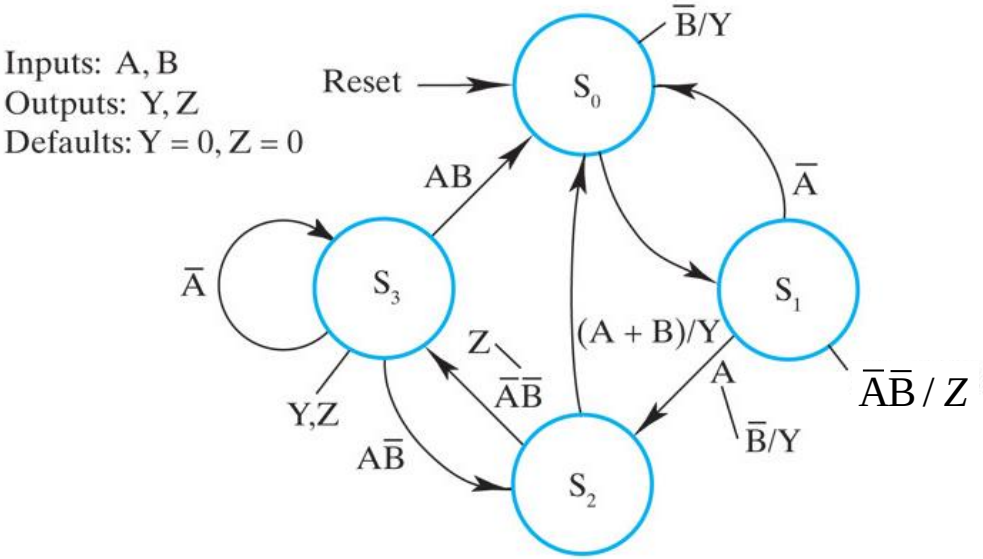
状态机图

- S_0->S_1，无条件转换
- S_1->S_0，$\overline{A}$=1
- S_2->S_0，A+B=1

状态图

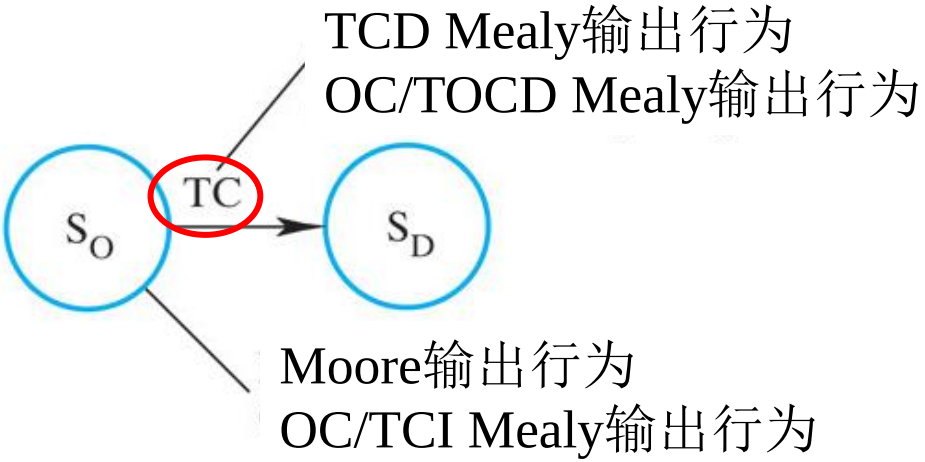


状态机图



状态机图

- 转移条件
  - 无条件转移：在弧上没有转移条件或者转移条件包含常量 1 .
  - 条件转移：在弧上有一个或者多个转移条件．如果任何一个转移条件为 1 ，则转移发生.

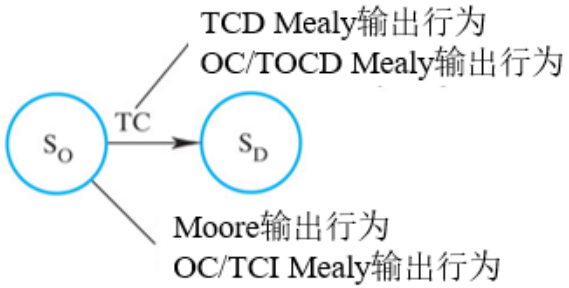


通用模板

状态机图

- 输出行为
  - Moore输出行为, 是无条件的，只依赖于状态，用一条线和相应的状态连接.
  - 非转移条件依赖(Transition condition-independent, TCI)：Mealy输出行为由输出条件和斜线驱动，和相应的状态连接．如果输出条件为１，则输出行为发生.
  - 转移条件依赖(Transition condition-dependent, TCD)：Mealy输出行为和相应的状态转移连接．如果输出条件为１,则输出行为发生.
  - 转移和输出条件依赖(Transition and output condition-dependent，TOCD)：Mealy输出行为由输出条件和斜线驱动，和相应的状态转移条件相连．如果转移条件和输出条件都为１，则输出行为发生.

通用模板



状态机图

- 输出行为
  - 单变量
    - 依附于状态的变量Z的出现意味着Z = 1，否则Z=0.
    - 依附于转移条件（可能是输出条件）的变量Z的出现意味着当条件满足时Z = 1 。除非Z是Moore 输出或者TCI标签的一部分，否则Z=0。
    - 单独的默认值语句可以用来显式地制定Z的默认值为0或1.

状态机图

- 输出行为
  - 向量变量
    - 依附于状态的方程‘Z = 向量值’的出现，指定了Z的值。
    - 依附于转移条件（可能是输出条件）的方程‘Z = 向量值’的出现，指定的Z的值。依附于转移的Z的值也可以被Moore输出指定或者作为TCI的一部分。否则，Z的值为默认值。

状态机图

- 总结来说，给定一个状态，一个输出行为在如下情况发生：

  a) 无条件(Moore);

  b) TCI并且输出条件(output condition, OC)为 1；

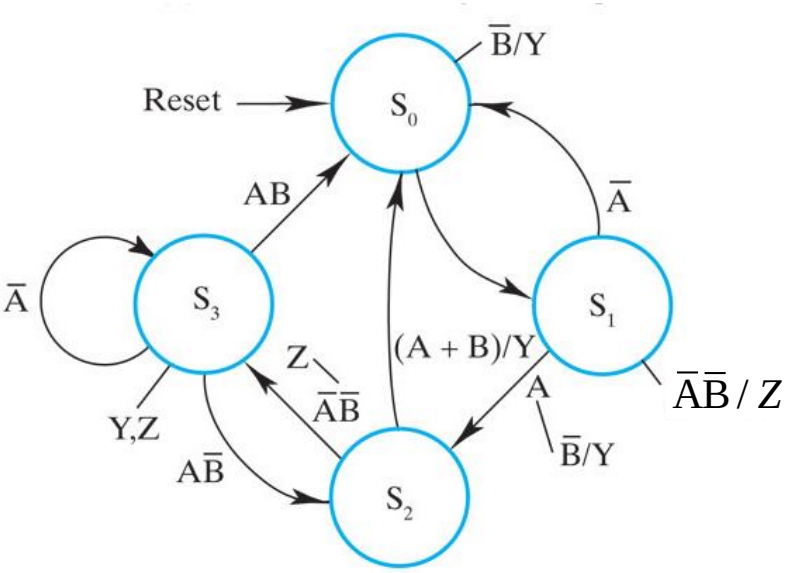
  c) TCD并且转移条件（transition condition， TC）为 1；

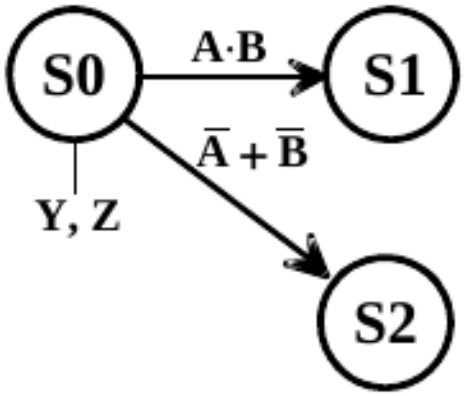
  d) TOCD并且TC和OC为 1 .

- Moore和TCI输出动作和一个状态相连，适用于从这个状态出发的状态转移.

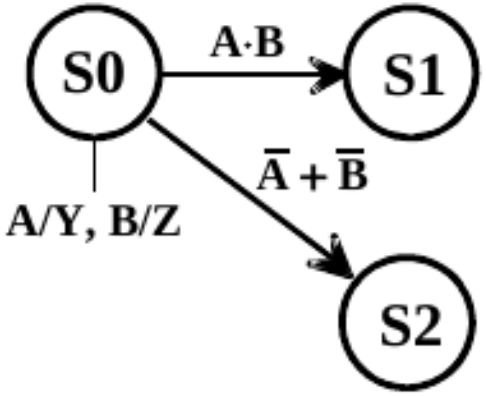
转移和输出条件例子

- 输入变量 A, B, C
- 输出变量 Y, Z
  默认: Y = 0, Z = 0

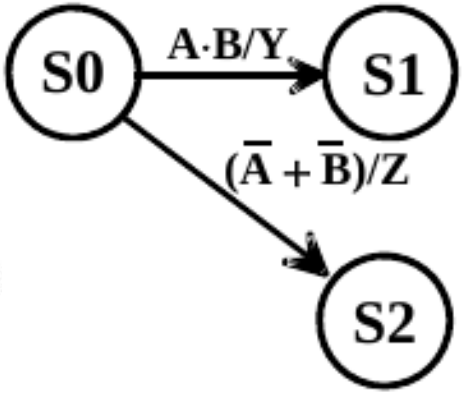




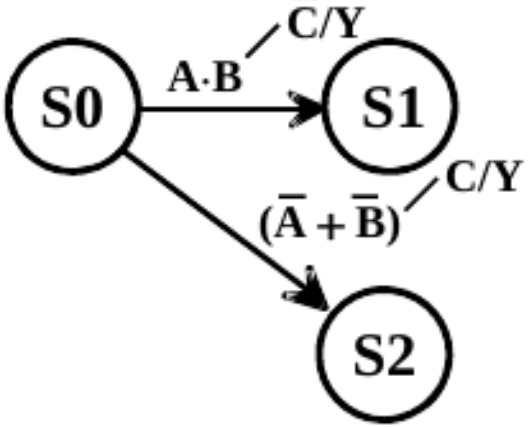
Ex. 1: Moore Outputs



Ex. 2: TCI Outputs



Ex. 3: TCD Outputs



Ex. 4: TCOD Outputs

状态机图使用

- 状态机图看起来复杂，但注意以下:
  - 只有无条件输出类型是适用于纯Moore型
  - TCD输出表示传统的Mealy模型，随着状态的增加，复杂度提高
  - 混合Moore和 Mealy ， 以及TCI 和 TCOD可以简化状态图和状态表

状态机表

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (and OCs)
State Name 1	State Code 1	Unused	Unconditional Next State 1	Next State Code 1	Moore or TCI Output (and OC)
		Transition Cond. 11	Next State 11	Next State Code 11	TCD or TOCD Output (and OC)
		Additional Transition Conditions and Entries for State Name 1			
State Name i	Entries for State Names i, i = 2, …n				

本章内容

三. 状态机设计

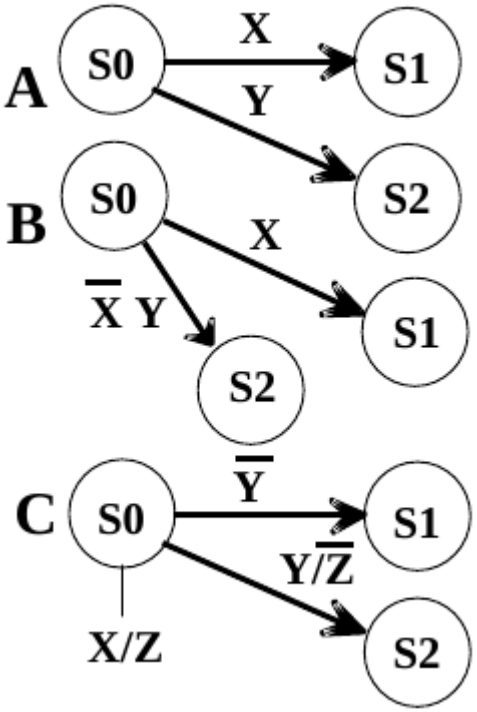
- 3.1 传统的状态图和状态表
- 3.2 状态机图模型
- 3.3 约束检查
- 3.4 状态机图应用和设计

约束检测

- TC 约束
  - 约束 1: 对状态 S_i, 从S_i出发的所有可能 TC 对 (T_{ij}, T_{ik})，
    $$T_{ij} \cdot T_{ik} = 0$$
  - 约束 2: 对状态S_i，对所有可能的TCs, T_{ij}
    $$\Sigma\, T_{ij} = 1$$
- OC 约束
  - 约束 1: 对状态S_i，在其上或者其状态转移上有一致的输出变量但不同值的输出行为，相应的输出条件对 (O_{ij}, O_{ik}) 是互斥的，即：
    $$O_{ij} \cdot O_{ik} = 0$$
  - 约束2： 对每个输出变量，在状态 S_i上或者在 S_i状态转移上的输出条件必须覆盖所有可能的输入变量组合，
    $$\Sigma\, O_{ij} = 1$$

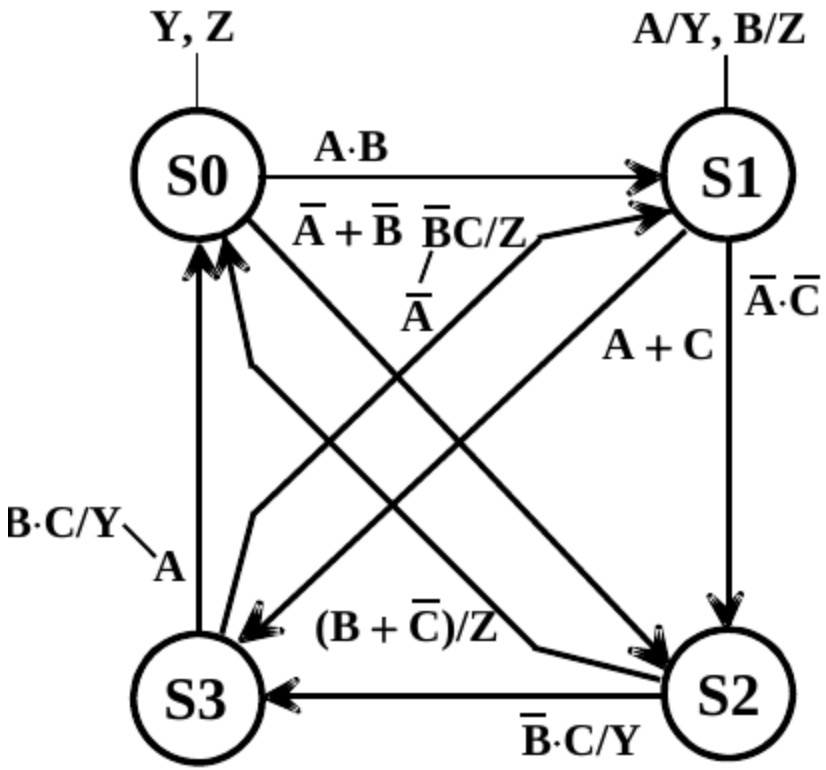
违反约束例子

- 转移约束
  - 例 A: $X \cdot Y \neq 0$ 且 $X + Y \neq 1$, 因此两个约束都违反
  - 例 B: $X \cdot \overline{X} Y = 0$, 但 $X + \overline{X} Y \neq 1$. 违反约束2
- 输出约束
  - 例 C: 对 Z = 1 和 Z = 0, $X \cdot Y \neq 0$, 因此违反约束1
  - 约束 $X + Y + \overline{Y} = 1$, 由于在 $\overline{Y}$ 上输出默认值Z， 因此约束2 满足。



约束检查例子

Defaults: Y = 0, Z = 0



- 转移约束
  - S0: $A\cdot B\cdot(\overline{A} + \overline{B}) = 0$;
    $A\cdot B + (\overline{A} + \overline{B}) = 1$
  - S1: $\overline{A}\cdot\overline{C}\cdot(A + C) = 0$;
    $\overline{A}\cdot\overline{C} + (A + C) = 1$
  - S2: $\overline{B}\cdot C\cdot(B + \overline{C}) = 0$;
    $\overline{B}\cdot C + (B + \overline{C}) = 1$
  - S3: $A \cdot \overline{A} = 0$;
    $A + \overline{A} = 1$
- 输出约束: ?

本章内容

三. 状态机设计

- 3.1 传统的状态图和状态表
- 3.2 状态机图模型
- 3.3 约束检查
- 3.4 状态机图应用和设计

基于状态机图设计过程

- 定义电路的输入输出变量，并且定义每个变量0和1的含义。
- 画出电路状态机图或者写出状态机表
  - 如果使用的是状态机图，将其转换成状态机表
- 从状态机表，推导出电路的下一状态和输出的优化方程。

例子 – 控制滑动门 – 规范化

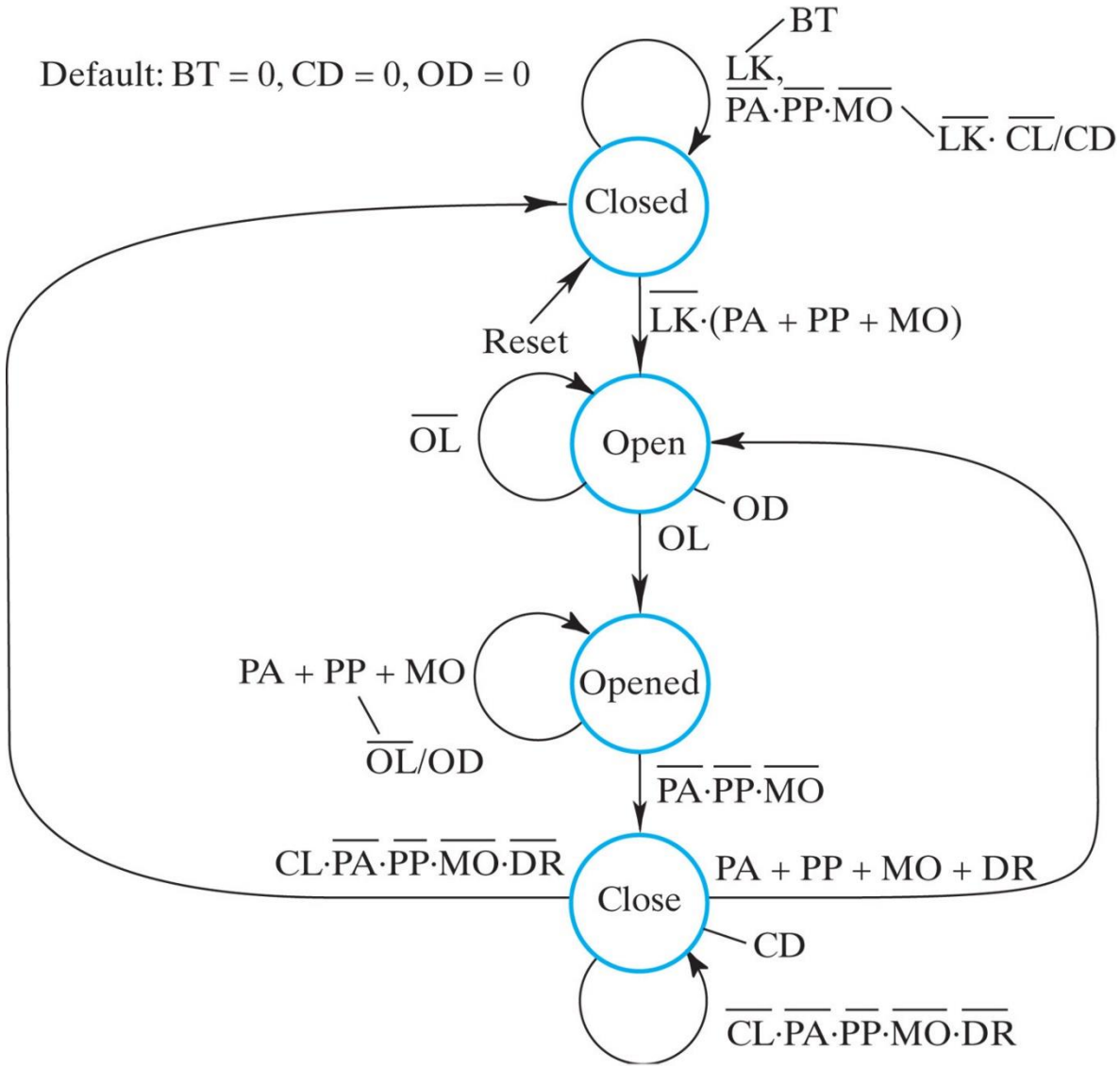
- 设计一个控制滑动门的时序电路
- 接近、存在、阻力时自动打开
- 手动按钮打开
- 控制盒里面的锁

例子 – 控制滑动门 – 输入和输出

Input Symbol	Name	Meaning for Value 1	Meaning for Value 0
LK	锁	Locked	Unlocked
DR	门阻力传感器	Door resistance ≥ 15 lb	Door resistance < 15 lb
PA	接近传感器	Person/object approach	No person/object approach
PP	存在传感器	Person/object in door	No person/object in door
MO	手动打开按钮	Manual open	No manual open
CL	关限位	Door fully closed	Door not fully closed
OL	开限位	Door fully open	Door not fully open

Output Symbol	Name	Meaning for Value 1	Meaning for Value 0
BT	门闩	Bolt closed	Bolt open
CD	关闭门	Close door	Null action
OD	打开门	Close door	Null action

例子 – 控制滑动门 – 状态机图



例子 – 控制滑动门 – 状态机表、状态赋值

State	State Code	Input Condition	Next State	State Code	Non-Zero Outputs (Including TCD and TOCD Output Actions and Output Conditions*)
Closed	00	LK	Closed	00	BT*
	00	$\overline{PA}\cdot\overline{PP}\cdot\overline{MO}$	Closed	00	$\overline{LK}\cdot\overline{CL}$/CD*
	00	$\overline{LK}\cdot(PA + PP + MO)$	Open	01	
Open	01				OD
	01	$\overline{OL}$	Open	01	
	01	OL	Opened	11	
Opened	11	PA + PP + MO	Opened	11	$\overline{OL}$/OD*
	11	$\overline{PA}\cdot\overline{PP}\cdot\overline{MO}$	Close	10	
Close	10				CD
	10	$\overline{CL}\cdot\overline{PA}\cdot\overline{PP}\cdot\overline{MO}\cdot\overline{DR}$	Close	10	
	10	$CL\cdot\overline{PA}\cdot\overline{PP}\cdot\overline{MO}\cdot\overline{DR}$	Closed	00	
	10	PA + PP + MO + DR	Open	01	

例子 – 控制滑动门 – 输出方程、优化

$$\mathrm{BT} = \bar{Y}_1 \cdot \bar{Y}_2 \cdot \mathrm{LK}$$

$$\mathrm{CD} = Y_1 \cdot \bar{Y}_2 + \bar{Y}_1 \cdot \bar{Y}_2 \cdot \overline{\mathrm{LK}} \cdot \overline{\mathrm{CL}} \cdot \bar{X} = (Y_1 + \overline{\mathrm{LK}} \cdot \overline{\mathrm{CL}} \cdot \bar{X}) \cdot \bar{Y}_2$$

$$OD = \bar{Y}_1 \cdot Y_2 + Y_1 \cdot Y_2 \cdot \overline{\mathrm{OL}} \cdot X = (\bar{Y}_1 + \overline{\mathrm{OL}} \cdot X) \cdot Y_2$$

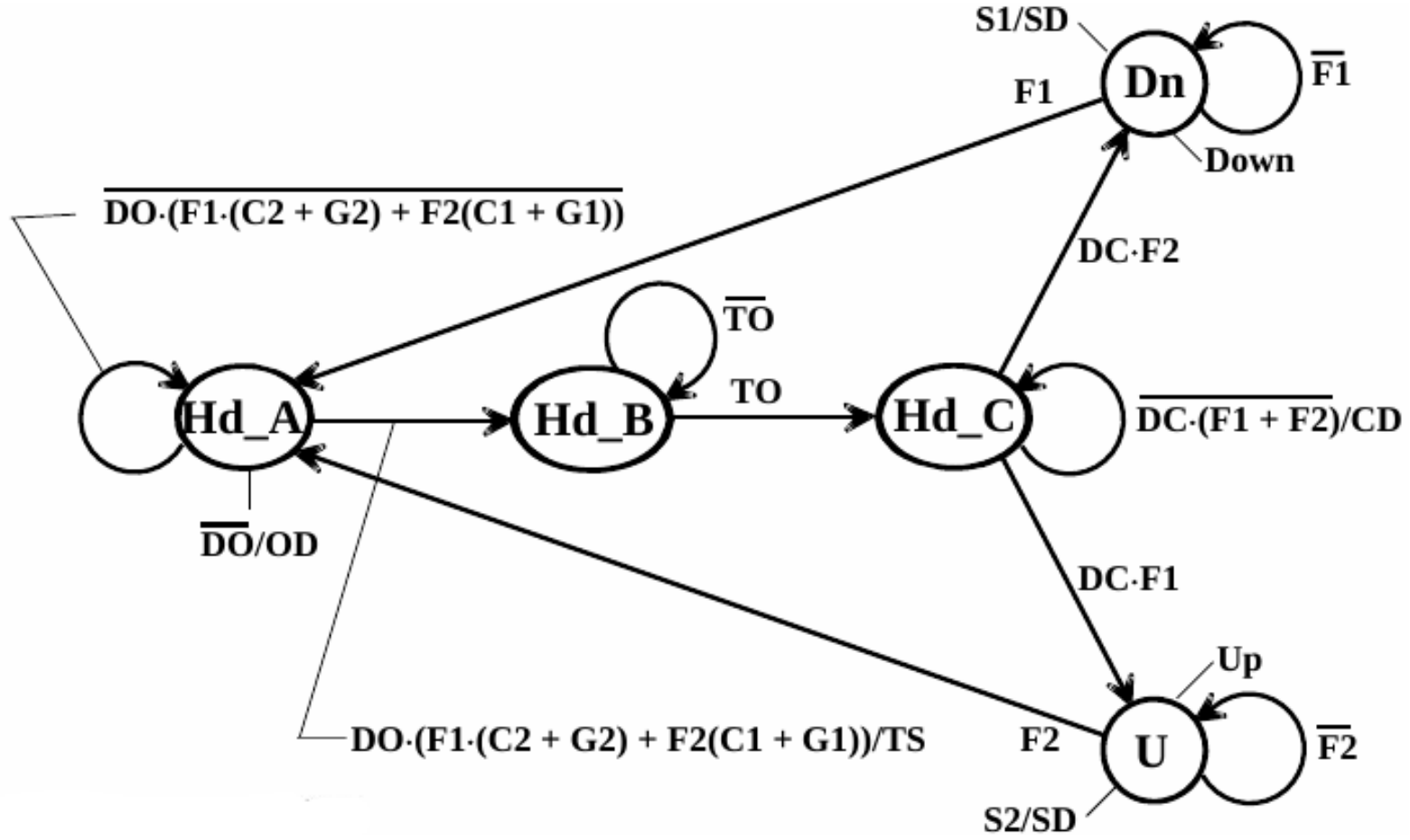
例子 – 电梯控制 – 规格化/输入输出

C1(C2)-到1层（2层）的call按钮（电梯外）：0 – 无动作; 1 – call
G1(G2) – 到1层（2层）的go按钮（电梯内）：0 – 无动作; 1 – go
F1(F2) –电梯在1层（2层）：0 – 电梯在；1 – 电梯不在
S1(S2)– 电梯在接近1层（2层）：0 –电梯不在接近；1 – 电梯在接近
DO – 门打开：0 – 门没有完全打开; 1 – 门完全打开
TO – 从按下到开始运动的时间间隔结束：0 –等待结束; 1 – 间隔结束
DC – 门关闭：0 – 门没有关闭; 1 –门关闭
TS – 计时器开始：0 – 无动作; 1 – 初始化和启动计时器
SD – 降速：0 – 电梯按正常速度运行; 1 – 电梯接近目标楼层降速
OD –开门：0 – 无动作; 1 – 开门
CD – 关门：0 –无动作; 1 –关门

例子 – 电梯控制 – 状态

- 初始状态:
  - U (Up)
  - Dn (Down)
  - Hd (Hold)
- 在Hd状态需要的系列动作
  - 开门
  - 使用定时器等待乘客
  - 关门
- 将Hd扩展成三个状态: Hd_A, Hd_B, Hd_C
- 状态向量:
  (U, Dn, Hd_C, Hd_B, Hd_A)

例子 - 电梯控制 - SMD


S1/SD
F1
Dn
$\overline{F1}$
Down
$\overline{DO \cdot (F1 \cdot (C2 + G2) + F2(C1 + G1))}$
DC·F2
$\overline{TO}$
TO
Hd_A
Hd_B
Hd_C
$\overline{DC \cdot (F1 + F2)}$/CD
$\overline{DO}$/OD
DC·F1
Up
DO·(F1·(C2 + G2) + F2(C1 + G1))/TS
F2
U
$\overline{F2}$
S2/SD

例子 – 电梯控制 – SMT

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (OCs)
Hd_A	00001				$\overline{DO}$/OD
		$\overline{(DO \cdot (F1 \cdot (C2 + G2) + F2 \cdot (C1 + G1))}$	Hd_A	00001	
		$DO \cdot (F1 \cdot (C2 + G2) + F2 \cdot (C1 + G1)$	Hd_B	00010	TS*
Hd_B	00010	$\overline{TO}$	Hd_B	00010	
		TO	Hd_C	00100	
Hd_C	00100	$\overline{DC \cdot (F1 + F2)}$	Hd_C	00100	CD*
		DC·F2	Dn	01000	
		DC·F1	Up	10000	

例子 – 电梯控制 – SMT

State	State Code	Transition Condition	Next State	Next State Code	Output Actions (OCs)
Dn	01000				Down, S1/SD
		$\overline{F1}$	Dn	01000	
		F1	Hd_A	00001	
U					Up, S2/SD
		$\overline{F2}$	Up	10000	
		F2	Hd_A	00001	

例子 – 电梯控制 – 方程

■ 触发输入

- $X = DO \cdot ((F1 \cdot (C2 + G2) + F2 \cdot (C1 + G1))$
- $Y = DC \cdot (F1 + F2)$
- $D_{Hd\_A} = Hd\_A \cdot \overline{X} + Dn \cdot F2 + U \cdot F1$
- $D_{Hd\_B} = Hd\_A \cdot X + Hd\_B \cdot \overline{TO}$
- $D_{Hd\_C} = Hd\_B \cdot TO + Hd\_C \cdot \overline{Y}$
- $D_{Dn} = Hd\_C \cdot DC \cdot F2 + Dn \cdot \overline{F1}$
- $D_{U} = Hd\_C \cdot DC \cdot F1 + U \cdot \overline{F2}$

⑩ 输出

- $Down = Dn$
- $Up = U$
- $SD = Dn \cdot S1 + U \cdot S2$
- $TS = Hd\_A \cdot X$
- $OD = Hd\_A \cdot \overline{DO}$
- $CD = Hd\_C \cdot \overline{Y}$

小结

- 基本存储单元
  - 锁存器
  - 触发器
- 状态表和状态图
  - Moore型
  - Mealy型
- 状态机图
  - 四种类型输出行为