



数字逻辑设计与微处理器系统 第四讲

Sequential Logic Design

马上 博士




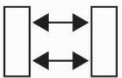
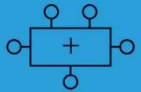

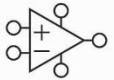

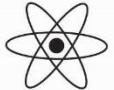
通信抗干扰技术国家级重点实验室
电子科技大学

2021年4月



Chapter Outline

- Introduction
- Latches and Flip-Flops
- Synchronous Logic Design
- Finite State Machines
- Timing of Sequential Logic
- Parallelism

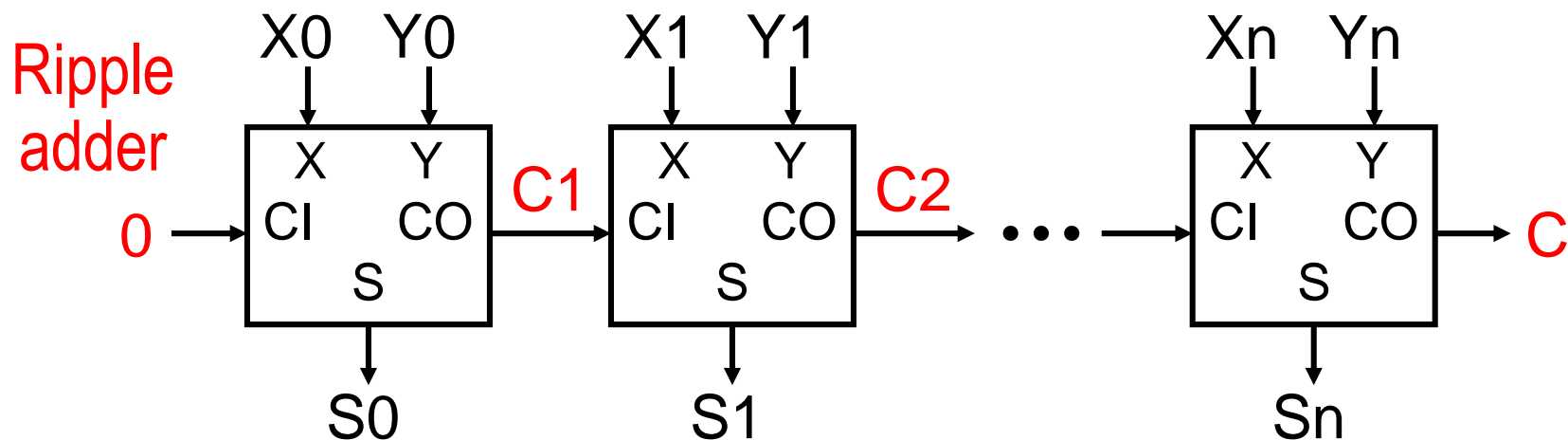
Application Software	
Operating Systems	
Architecture	
Micro-architecture	
Logic	
Digital Circuits	
Analog Circuits	
Devices	
Physics	

Introduction

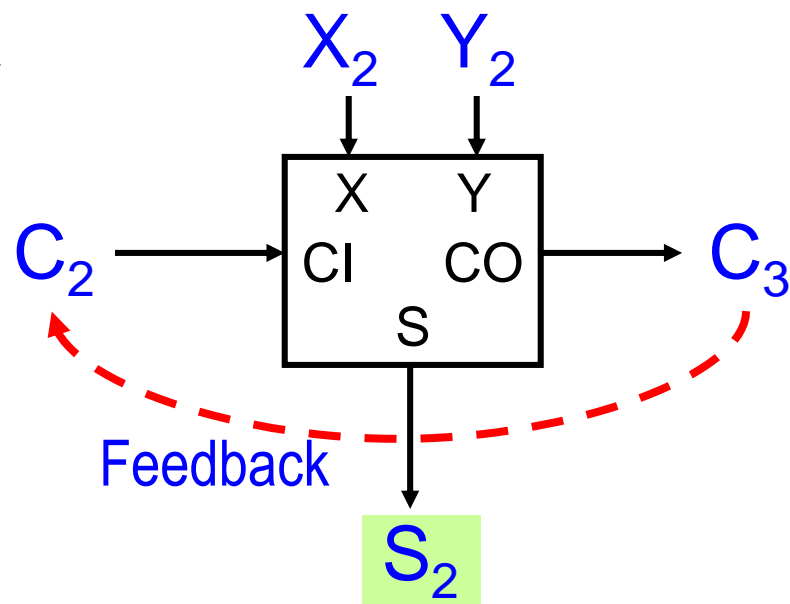


Types of Logic Circuits

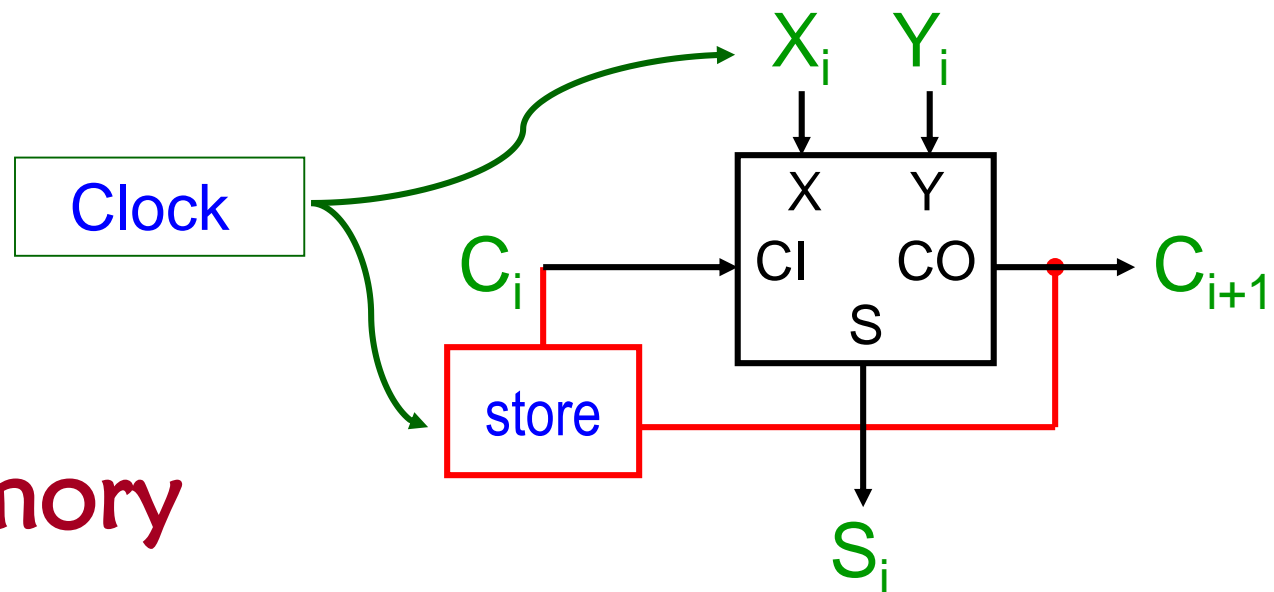
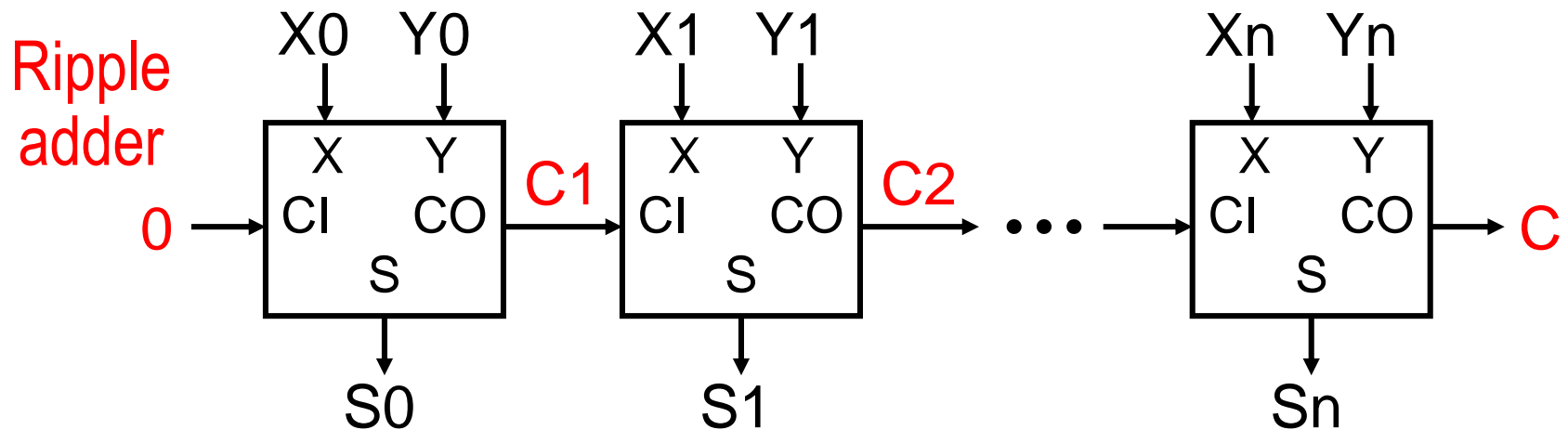
- **Combinational logic circuit**
 - **Outputs depend only on the current values of the inputs**
 - **No feedback loop, No Memory Device**
- **Sequential logic circuit**
 - **Outputs depend on**
 - **the current values of the inputs**
 - **the prior values of the inputs**
 - **Feedback loop, Memory Device**



Consider: Can we use one full adder to implement multi-bit ripple adder?



- Feedback
- clock control

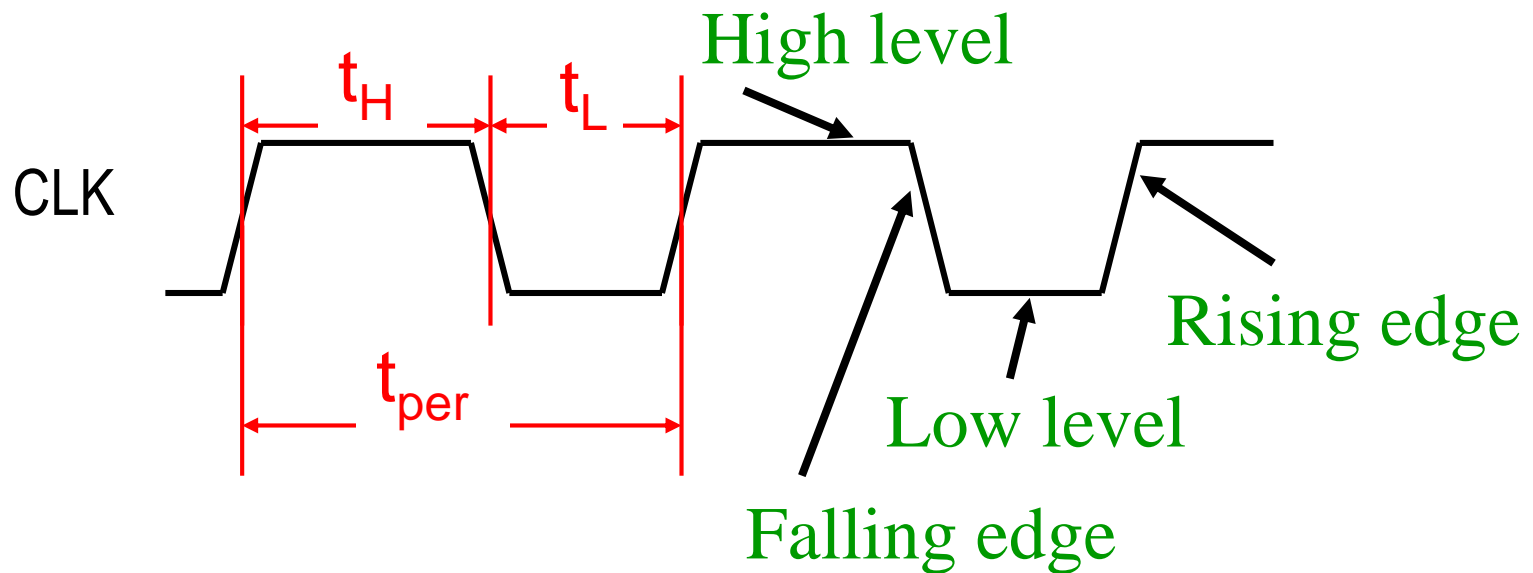


Need Memory



Need Clock

- The state changes of most sequential circuits occur at times specified by a free-running clock signal.



Period = t_{per}

Frequency = $1/t_{per}$

Duty cycle = t_H/t_{per} (or t_L/t_{per})

Some definitions

- **State:** all the information about a circuit necessary to explain its future behavior
 - The state is a collection of state variables
 - A circuit with a set of n bits state variables has 2^n possible states
- **Latches and flip-flops:** state elements that store one bit of state
- **Synchronous sequential circuits:** combinational logic followed by a bank of flip-flops

Latches and Flip-Flops

Book: C3-3.2

References: C7-7.1, 7.2

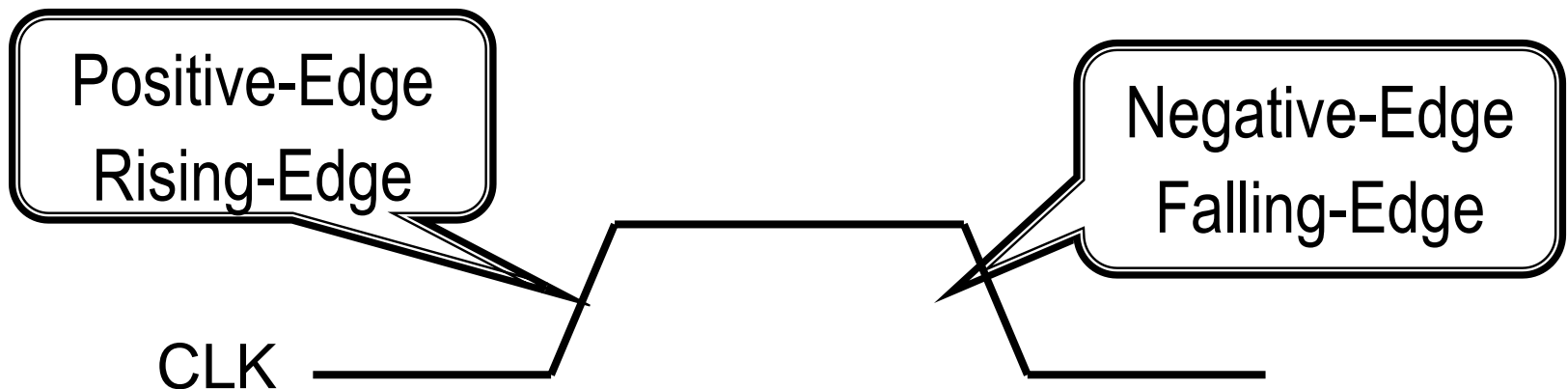
- Bistable circuit
- SR Latch
- D Latch
- JK Flip-flop
- D Flip-flop
- T Flip-flop

双稳态电路(Bistable circuit)

锁存器(Latch)

触发器(Flip-flop)

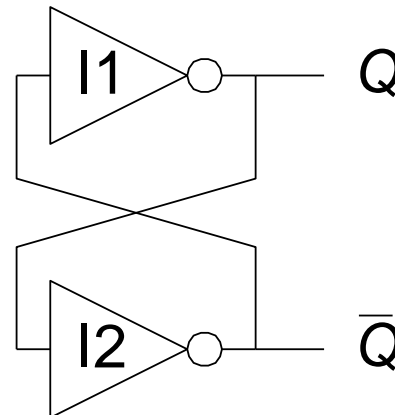
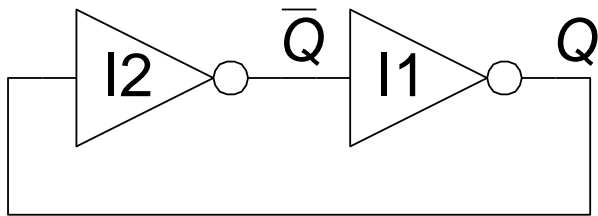
---Change its outputs only at the Rising or Falling Edge of a controlling CLK signal.



Bistable Metastable

Bistable circuit

- Fundamental building block of other state elements
- Two outputs: Q , \bar{Q}
- No inputs

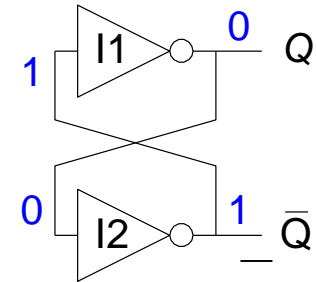


Bistable circuit

- Consider the two possible cases:

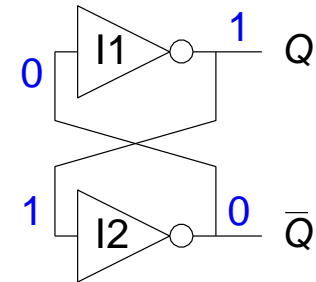
– $Q = 0$:

then $\bar{Q} = 1$, $Q = 0$ (consistent)

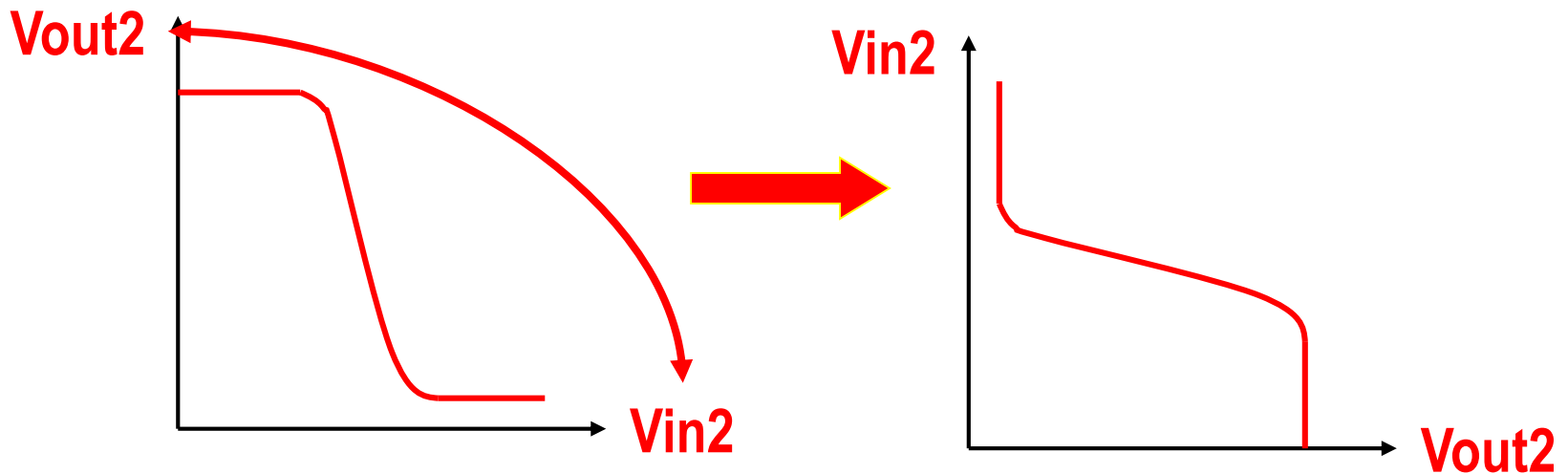
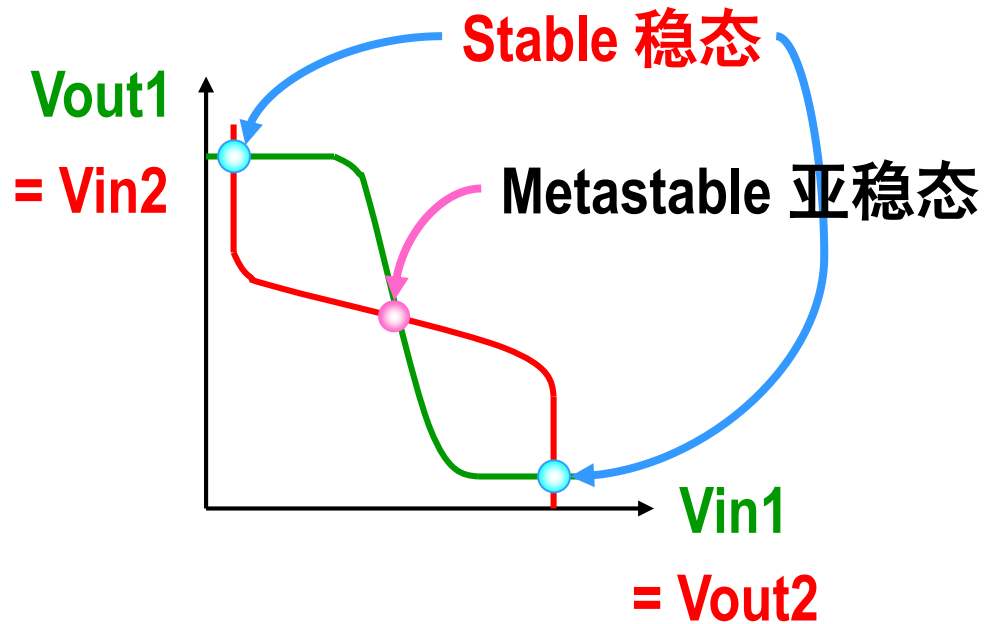
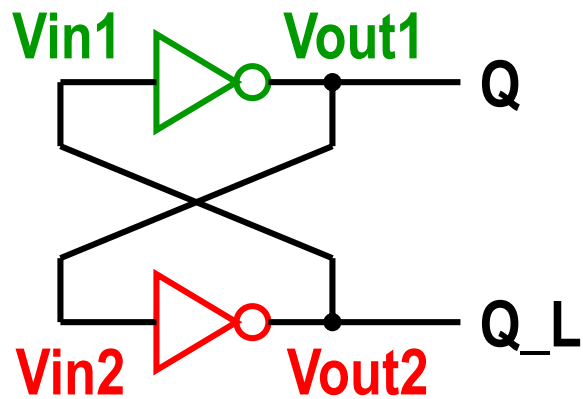


– $Q = 1$:

then $\bar{Q} = 0$, $Q = 1$ (consistent)



- Stores 1 bit of state in the state variable, Q (or \bar{Q})
- But there are **no inputs to control the state**

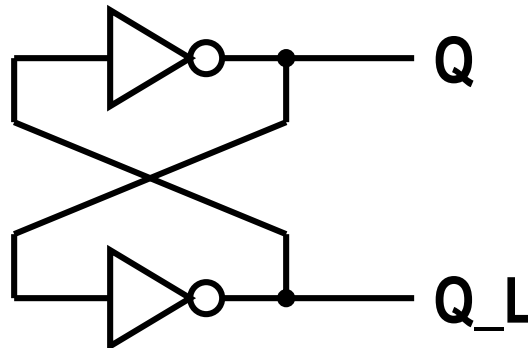


Metastable Behavior

(亚稳态特性)

- Metastability is the situation where the inputs cause an indeterminate output in a feedback circuit
- Random Noise will tend to Drive a circuit that is Operating at the Metastable Point to one of the Stable operating point.

(随机噪声会驱动工作于亚稳态点的电路转移到一个稳态的工作点上去)



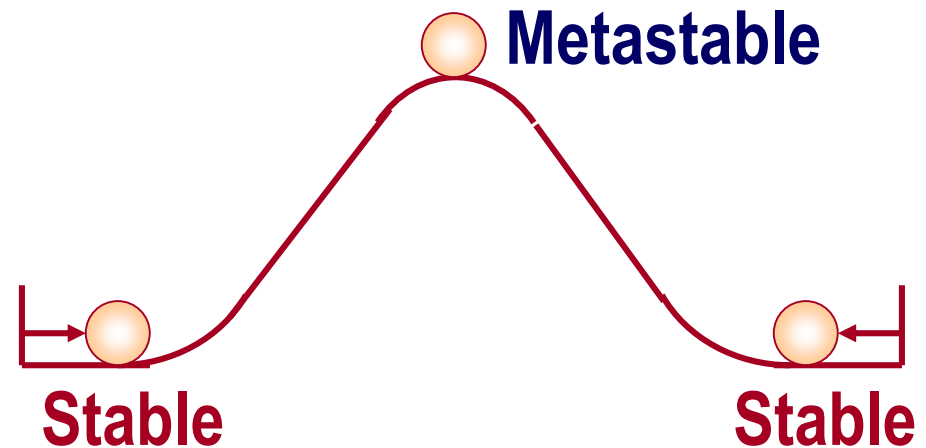
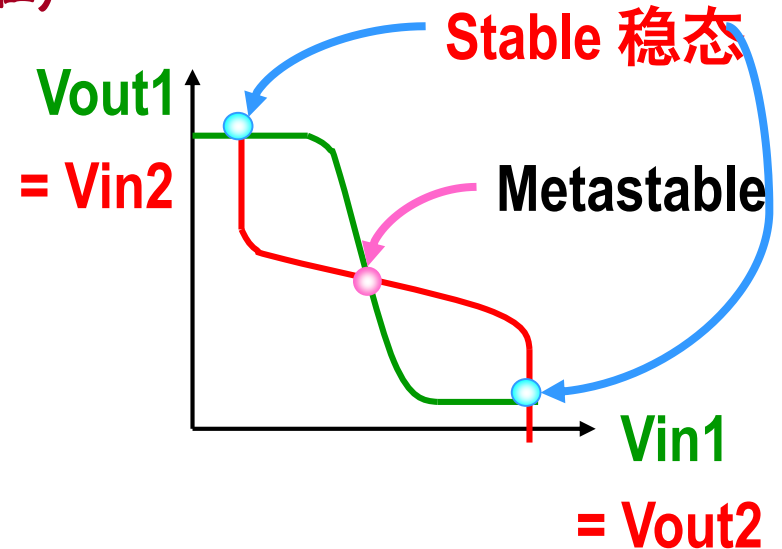
Metastable Behavior

(亚稳态特性)

➤ Apply a definite Pulse Width from a Stable state to the Other.

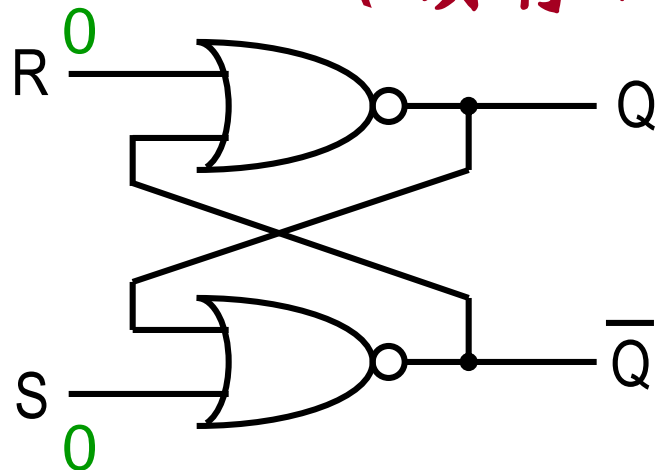
(从一个“稳态”转换到另一个“稳态”需加一定宽度的脉冲 (足够的驱动))

➤ All sequential circuits are susceptible to metastable behavior.

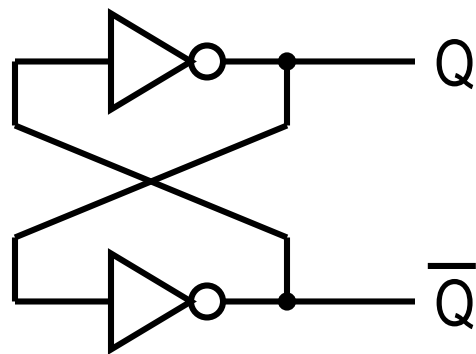


Latches

SR Latches (锁存器)



NOR → NOT



Memory!

Function Behaviors
Considering the four possible cases:

- $S = 1, R = 0$
- $S = 0, R = 1$
- $S = 0, R = 0$
- $S = 1, R = 1$

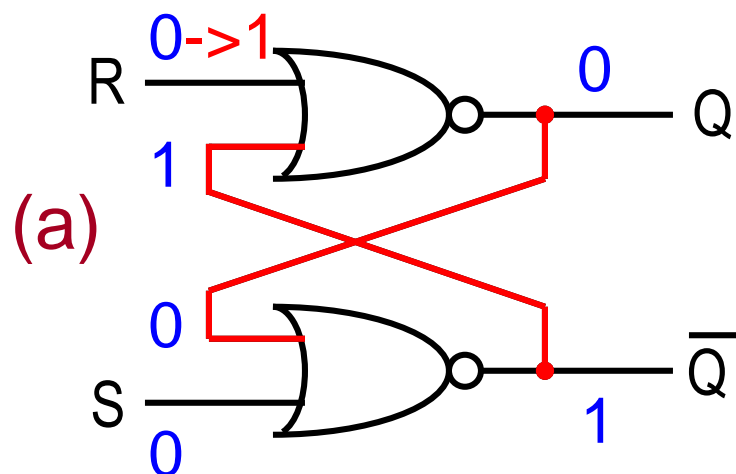
(1) $S = R = 0$

Keep the last state $Q^* = Q$
电路维持原态 $\bar{Q}^* = \bar{Q}$

新 态	$Q^{n+1} = Q^n$	原 态
	$\bar{Q}^{n+1} = \bar{Q}^n$	



SR Latches



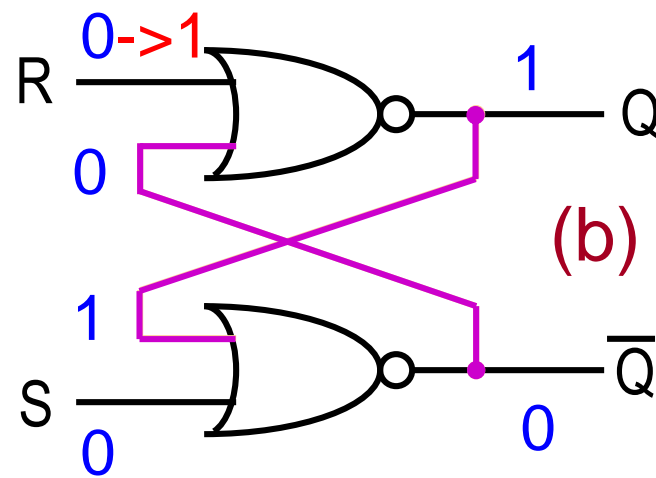
(2) $S = 0, R = 1$

Reset

锁存器清0: $Q^{n+1}=0$ $\bar{Q}^{n+1}=1$

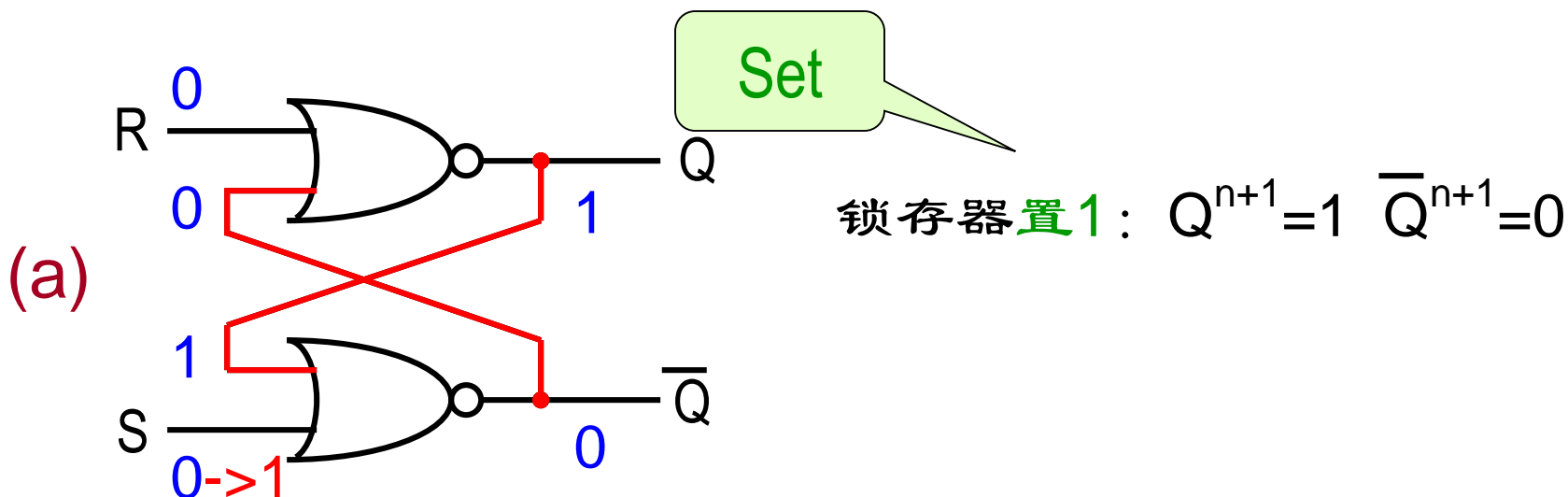
a. 原态: $Q^n=0, \bar{Q}^n=1$
新态: $Q^{n+1}=0, \bar{Q}^{n+1}=1$

b. 原态: $Q^n=1, \bar{Q}^n=0$
新态: $Q^{n+1}=0, \bar{Q}^{n+1}=1$



SR Latches

(3) $S = 1, R = 0$

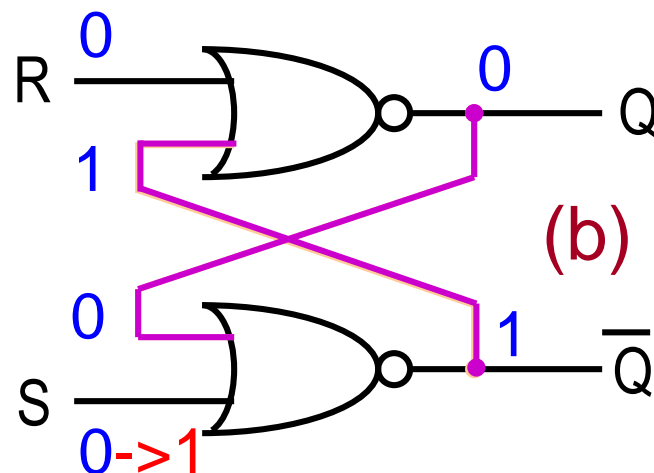


a. 原态: $Q^n=1, \bar{Q}^n=0$

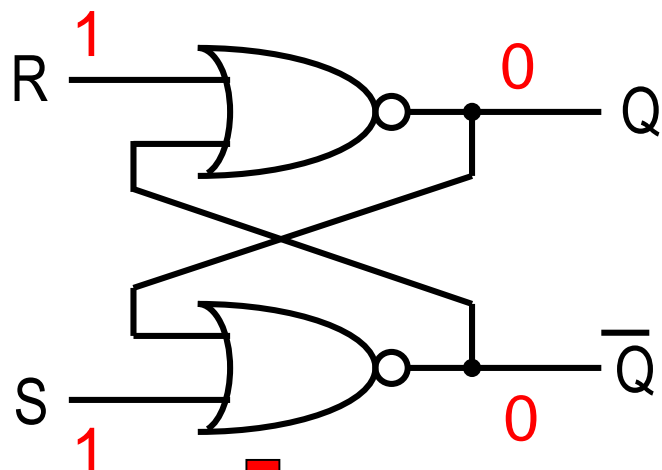
新态: $Q^{n+1}=1, \bar{Q}^{n+1}=0$

b. 原态: $Q^n=0, \bar{Q}^n=1$

新态: $Q^{n+1}=1, \bar{Q}^{n+1}=0$



SR Latches

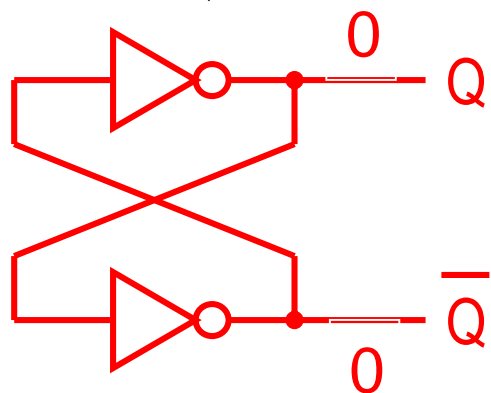


(4) $S = R = 1$

$$Q^{n+1} = \overline{Q}^{n+1} = 0$$

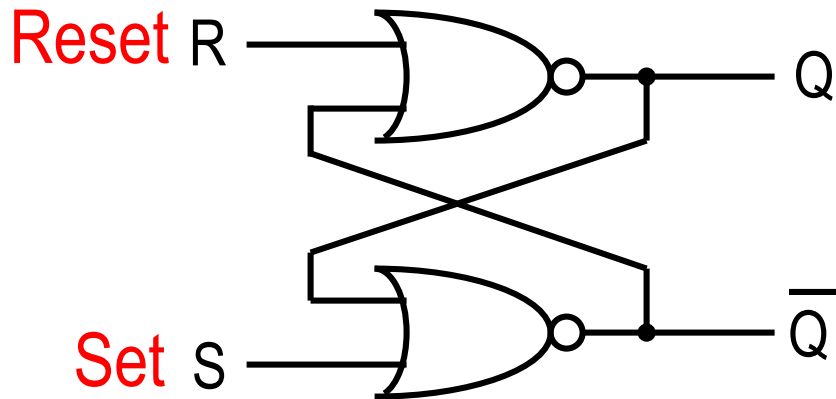
“禁止”

Invalid State



亚稳态，对噪声敏感
状态不确定

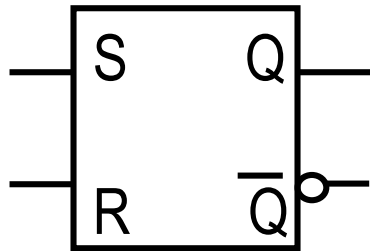
SR Latches



Function Table

S	R	Q	QL
0	0	维持原态	
0	1	0	1
1	0	1	0
1	1	0*	0*

Logic Symbol



State transition table

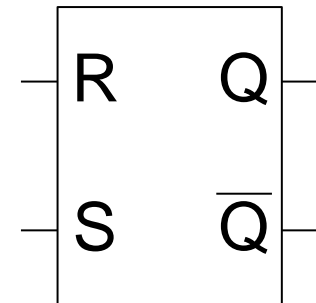
S	R	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0*
1	1	1	0*



SR Latches

- SR stands for Set/Reset Latch
 - Stores one bit of state (Q) (**when $S = R = 0$**)
- Control what value is being stored with S , R inputs
 - **Set:** Make the output 1
($S = 1, R = 0, Q = \mathbf{1}$)
 - **Reset:** Make the output 0
($S = 0, R = 1, Q = \mathbf{0}$)
- **Must do something to avoid invalid state (when $S = R = 1$)**

SR Latch
Symbol



SR Latches

SR

$Q^{n+1} \backslash Q^n$	00	01	11	10
0	0	0	×	1
1	1	0	×	1

特征方程

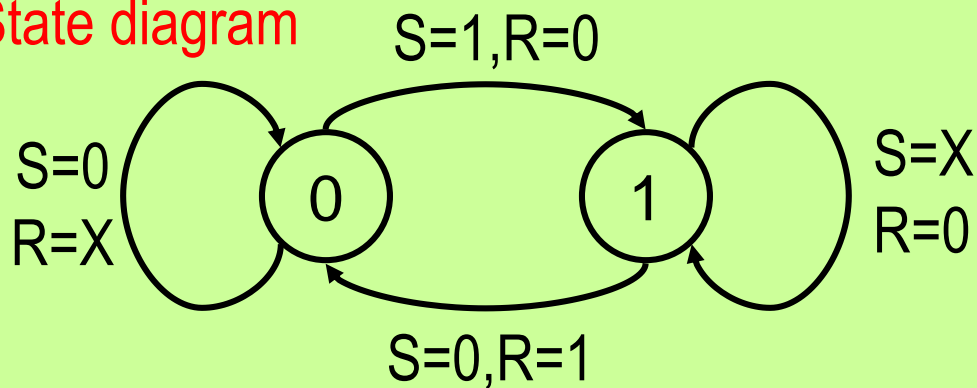
$$\begin{cases} Q^{n+1} = S + R' \cdot Q^n \\ S \cdot R = 0 \end{cases}$$

约束条件

State transition table

S	R	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0*
1	1	1	0*

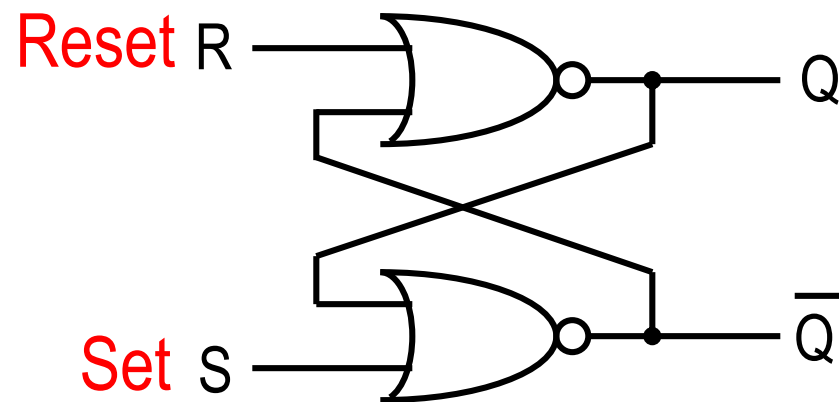
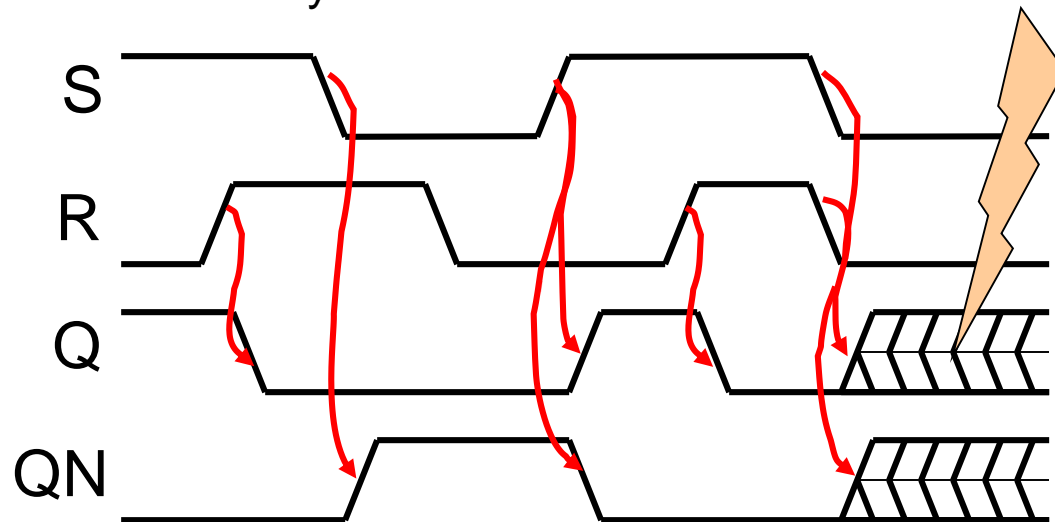
State diagram



SR Latches

Try to hold an unstable state : metastable

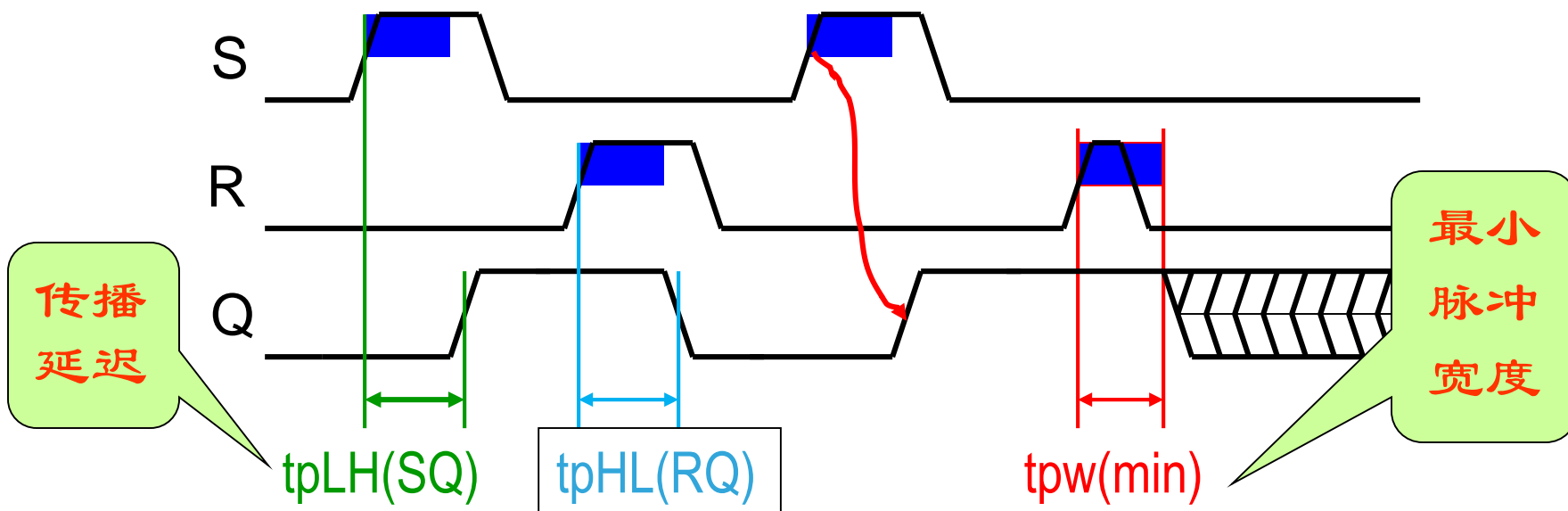
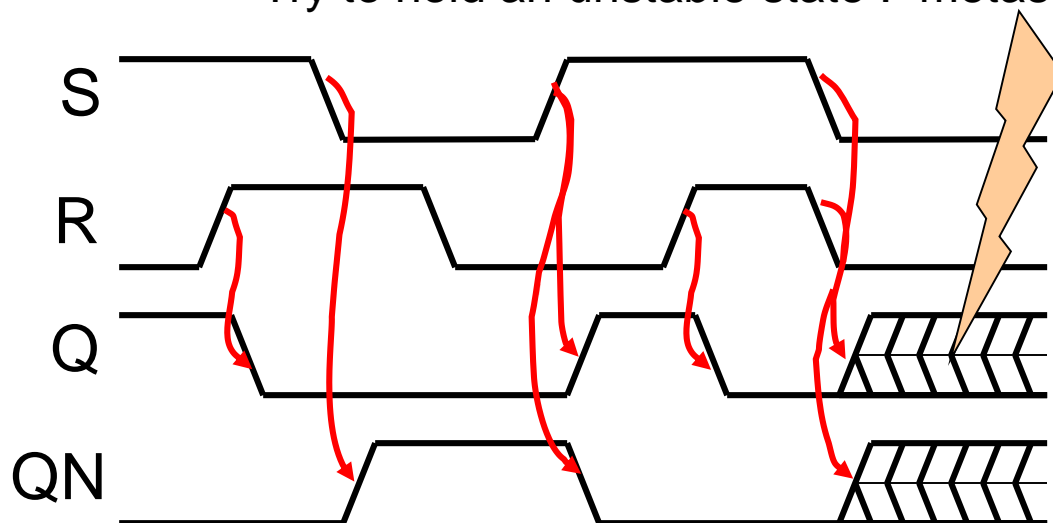
S	R	Q	QN
0	0	维持原态	
0	1	0	1
1	0	1	0
1	1	0*	0*



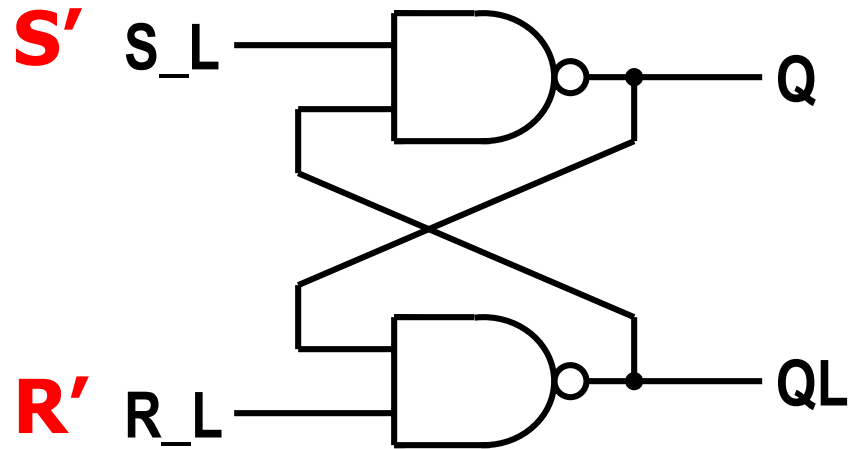
SR Latches

Try to hold an unstable state : metastable

S	R	Q	QN
0	0	维持原态	
0	1	0	1
1	0	1	0
1	1	0*	0*



$S'-R'$ latch



S-R latch function table

S_L	R_L	Q	Q_L	
1	1	维持原态		
1	0	0	1	清0
0	1	1	0	置1
0	0	1*	1*	不定

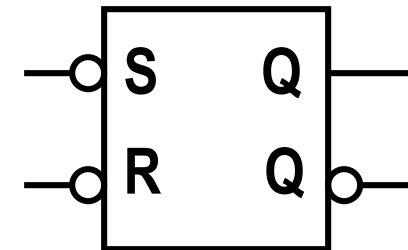
$S_L = R_L = 1$ 电路维持原态

$S_L = 1, R_L = 0$ $Q = 0, Q_L = 1$

$S_L = 0, R_L = 1$ $Q = 1, Q_L = 0$

$S_L = R_L = 0$ $Q = Q_L = 1$, 不定状态

Logical symbol

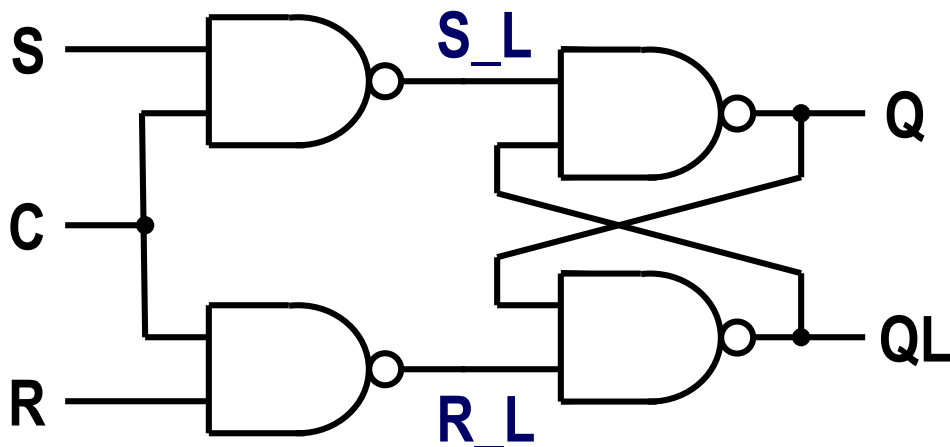




S-R Latch with Enable

—— 又称“时钟S-R锁存器”

功能表



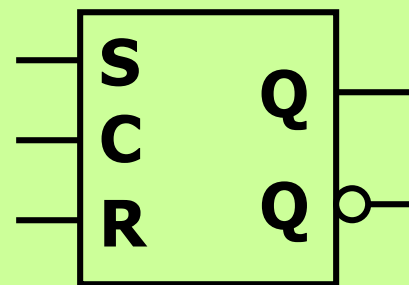
C	S	R	Q	Q _L
0	X	X	维持原态	
1	0	0	维持原态	
1	0	1	0	1
1	1	0	1	0
1	1	1	1*	1*

(1). $C = 0$ 时: 维持原态

(2). $C = 1$ 时: 与S-R锁存器相似

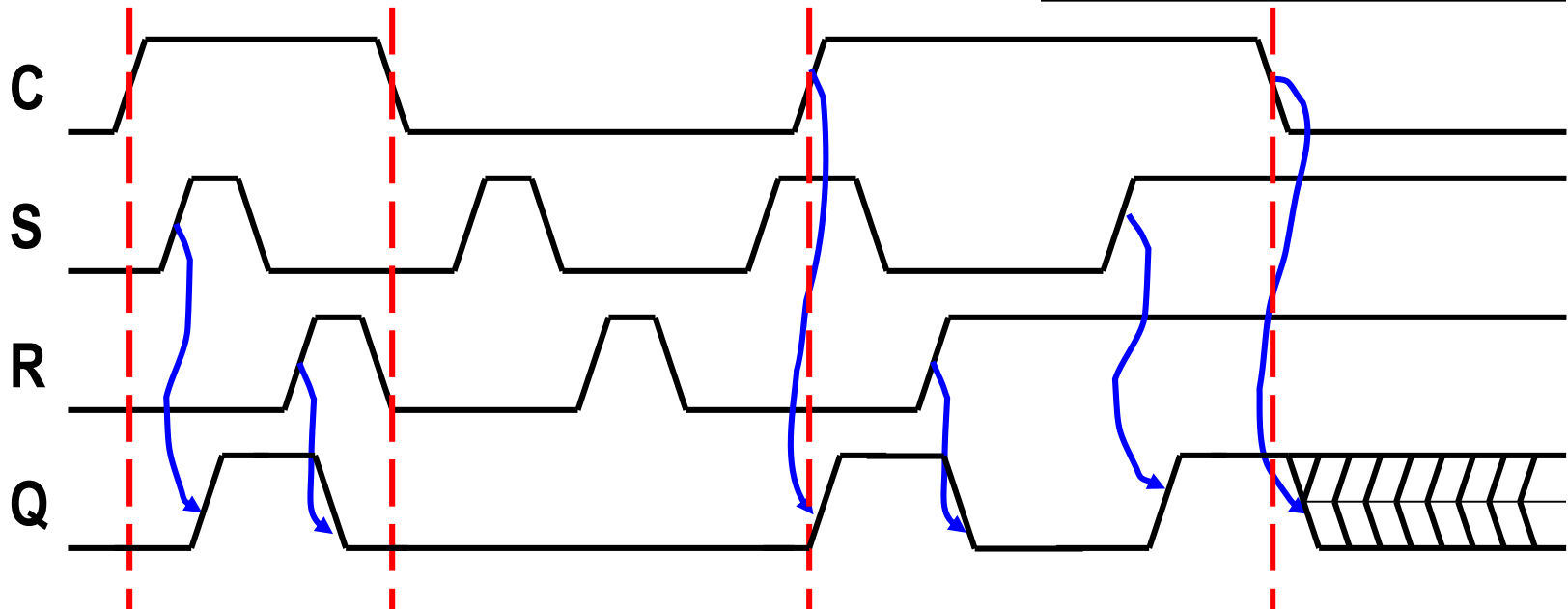
注意: 当 $S=R=1$ 时, 若 C 由 $1 \rightarrow 0$,
则下一状态不可预测。

逻辑符号

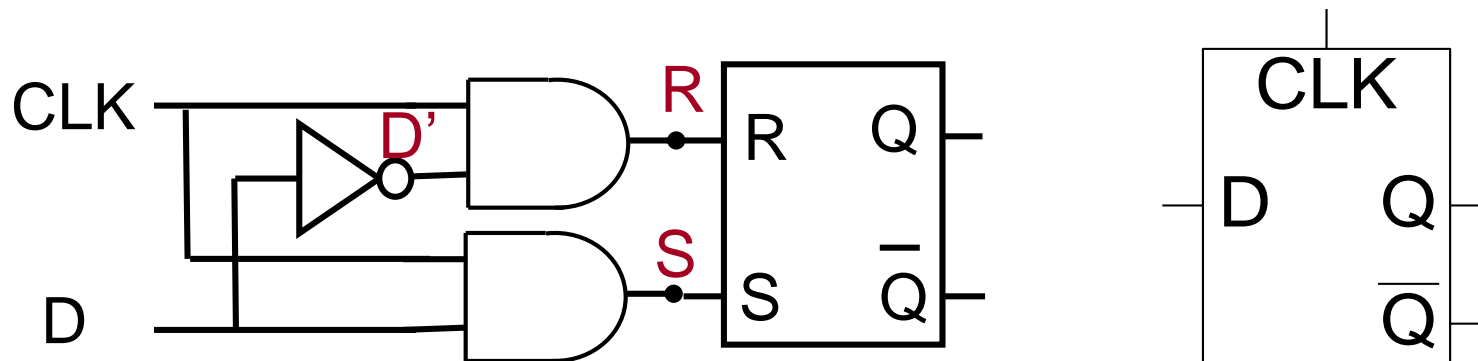


Typical operation of an S-R latch with enable

C	S	R	Q	QL
0	X	X	维持原态	
1	0	0	维持原态	
1	0	1	0	1
1	1	0	1	0
1	1	1	1*	1*



D Latches Transparent Latch (透明锁存器)



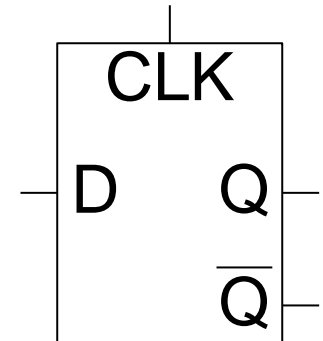
CLK	D	D'	S	R	Q	Q'
0	X	X'	0	0	维持	维持
1	0	1	0	1	0	1
1	1	0	1	0	1	0

The state is decided by the input **D**irectly !

D Latches

- Two inputs: CLK , D
 - CLK : controls *when* the output changes
 - D (the data input): controls *what* the output changes to
- Function
 - When $CLK = 1$,
 D passes through to Q (*transparent*)
 - When $CLK = 0$,
 Q holds its previous value (*opaque*)
- Avoids invalid case when
 $Q \neq \text{NOT } \bar{Q}$

D Latch
Symbol



D Latches

Function Description of a D Latch

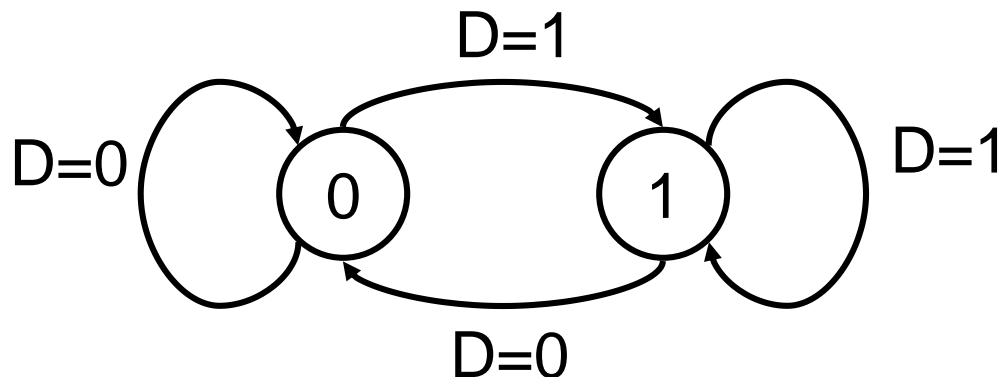
State transition table

D	Q^{n+1}
0	0
1	1

Characteristic equation :

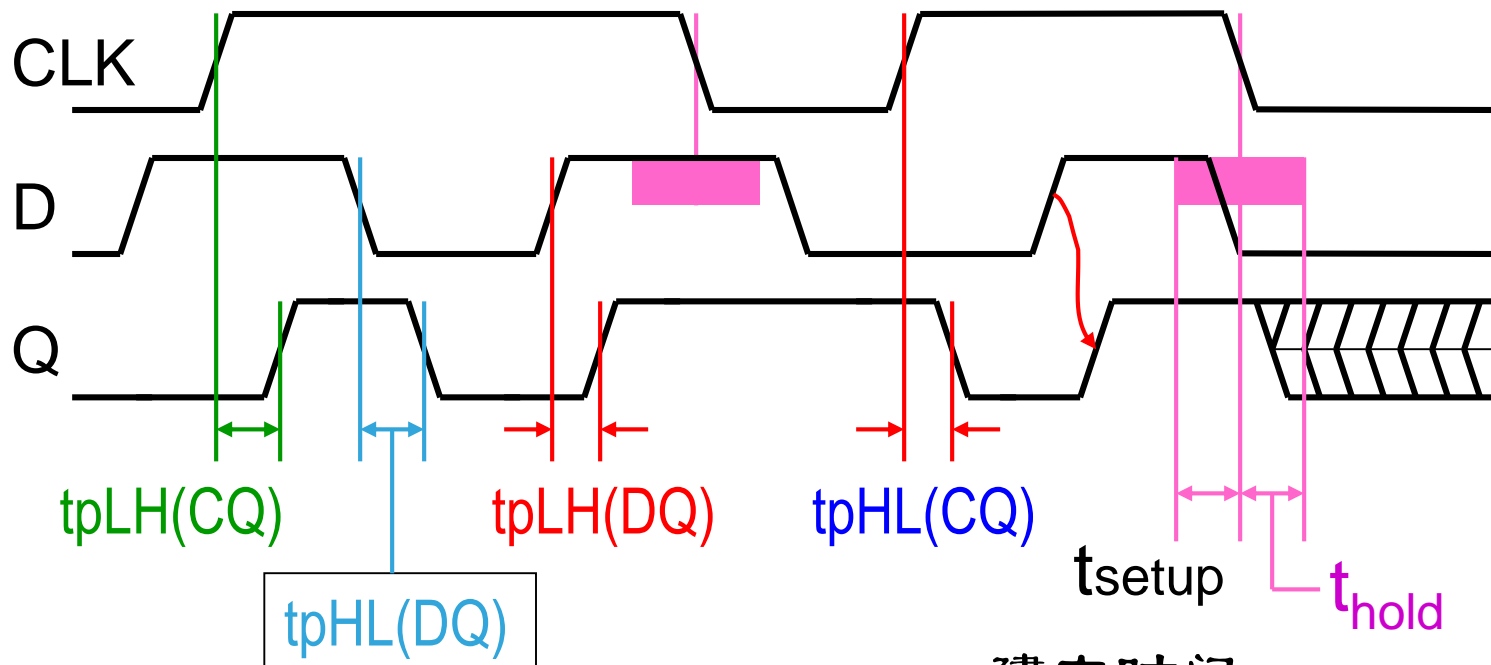
$$Q^{n+1} = D \quad (\text{CLK}=1)$$

State diagram



Timing Parameters for a D Latch

- ✱ **Setup time** (建立时间, 输入信号先于时钟到达的时间)
- ✱ **Hold time** (保持时间, 有效时钟沿后输入信号保持的时间)



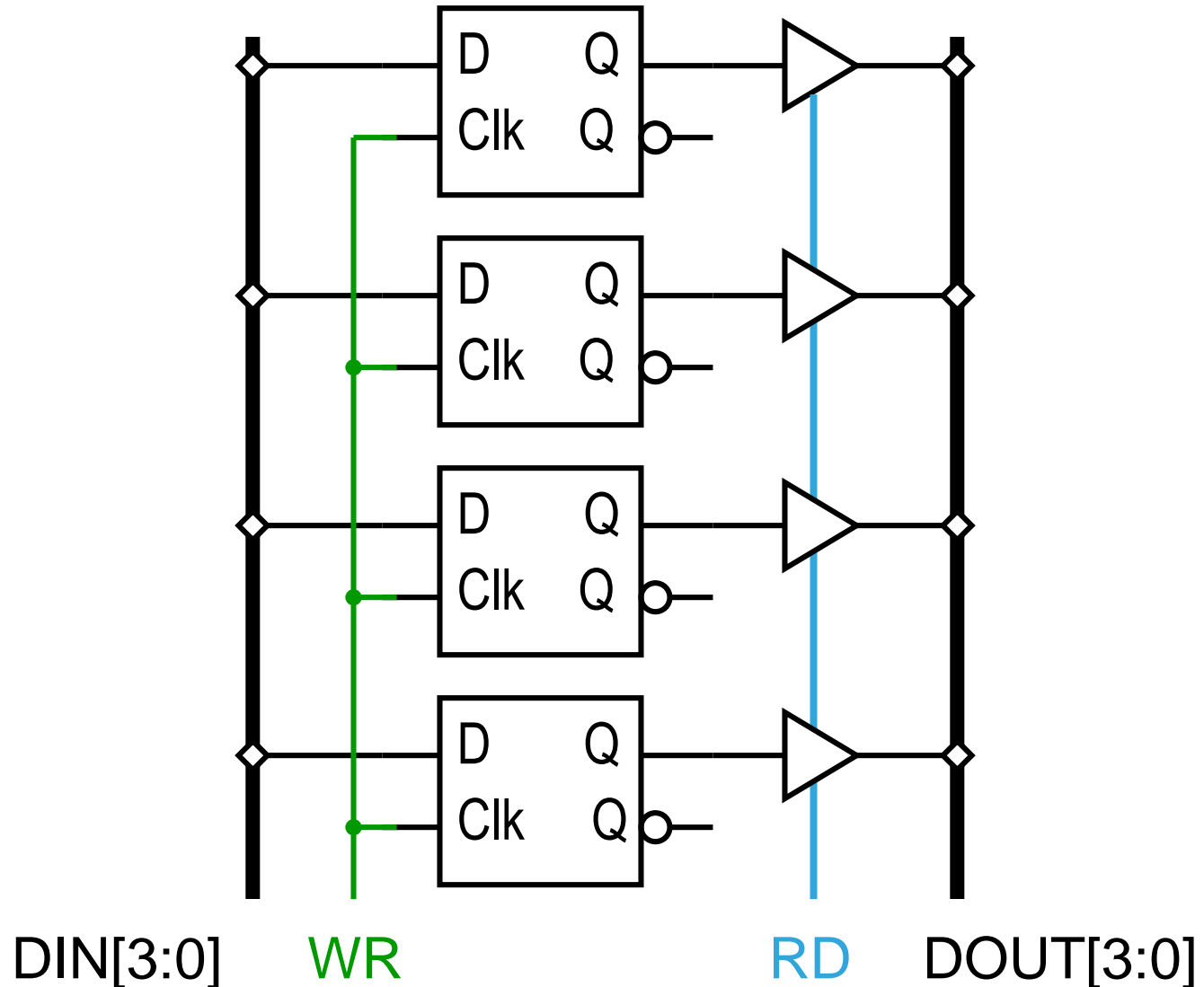
在CLK的下降沿附近有一个时间窗
这段时间内D输入一定不能变化

建立时间
setup time

保持时间
hold time

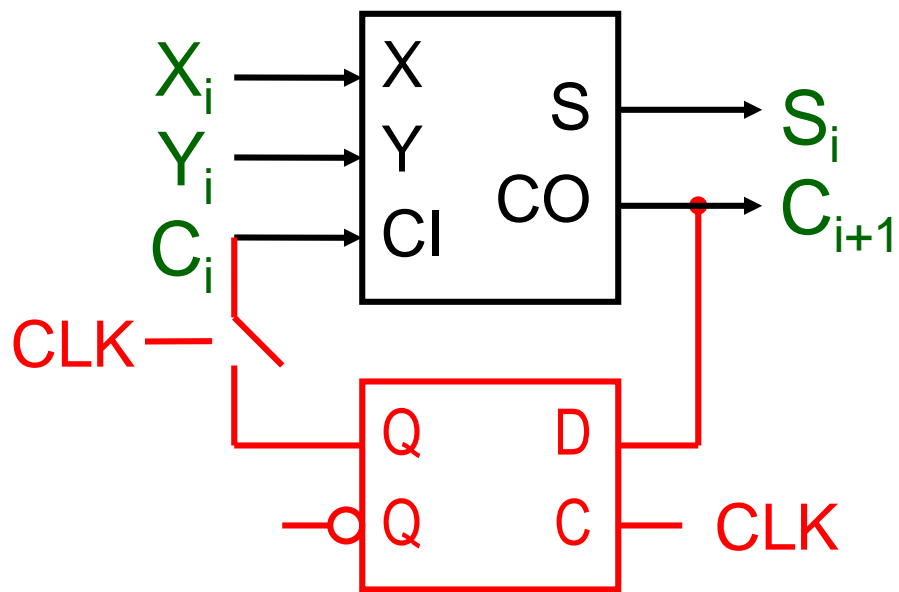
D Latches

Applications of Latches

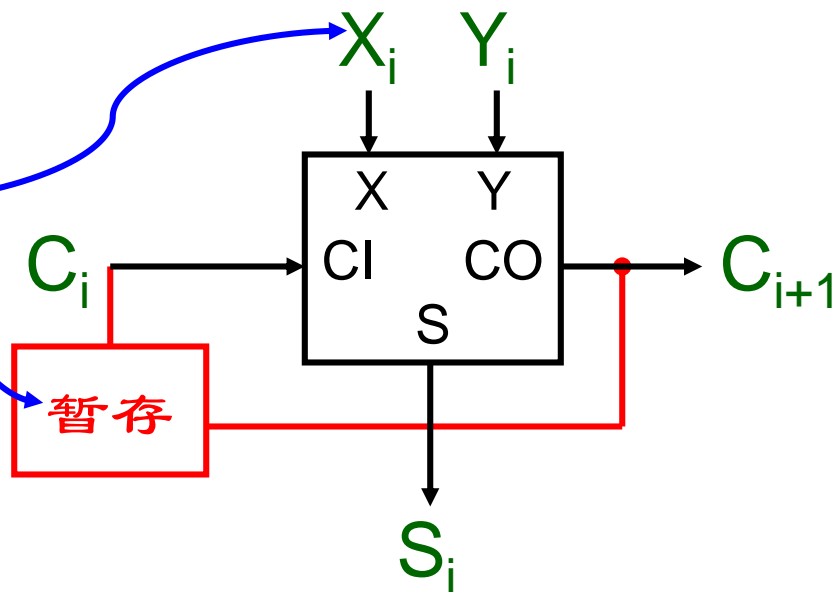


D Latches

Applications of Latches



时钟控制



Flip-Flop



Flip-Flop

- **Classified by Function**

- **D Flip-Flop**
- **S-R Flip-Flop**
- **J-K Flip-Flop**
- **T Flip-Flop**

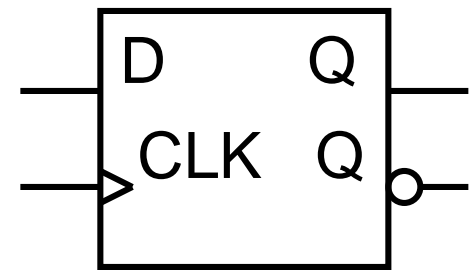
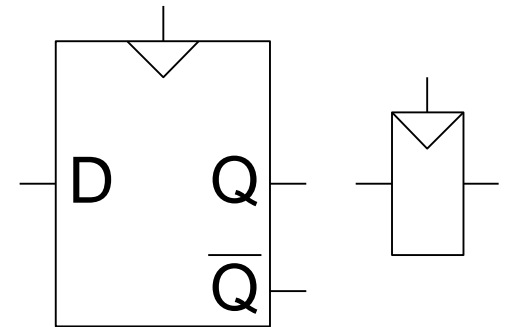
- **Other Types Flip-Flop**

- **Flip-Flop with enable、 Scan Flip-Flop...**

D Flip-Flop

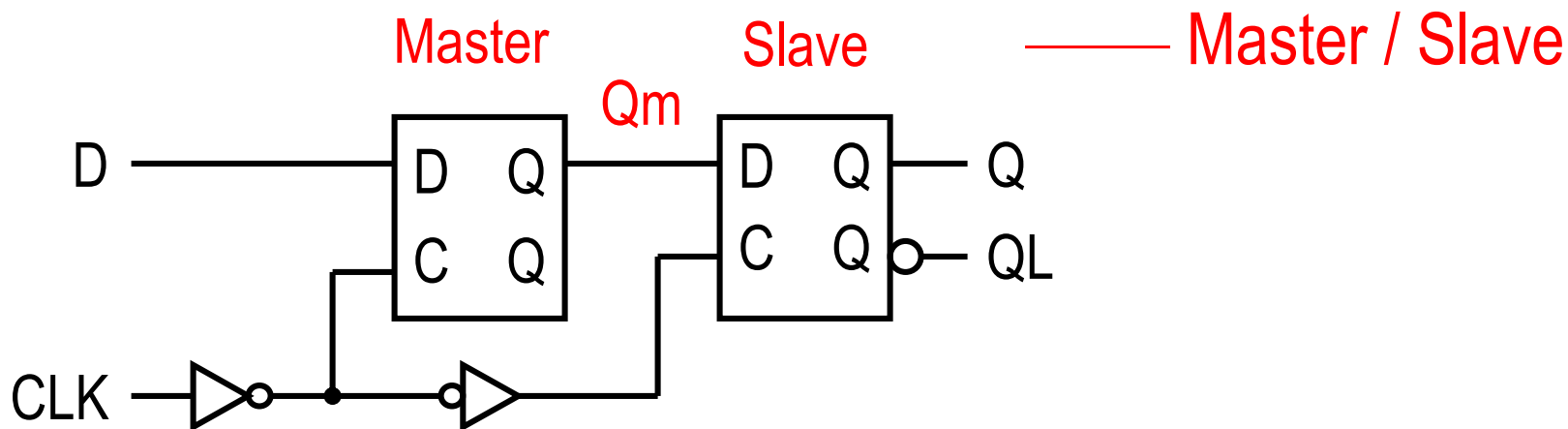
- **Inputs:** *CLK*, *D*
- **Function**
 - Samples *D* on rising edge of *CLK*
 - When *CLK* rises from 0 to 1, *D* passes through to *Q*
 - Otherwise, *Q* holds its previous value
 - *Q* changes only on rising edge of *CLK*
- Called *edge-triggered*
- Activated on the clock edge

D Flip-Flop
Symbols



D Flip-FLOP

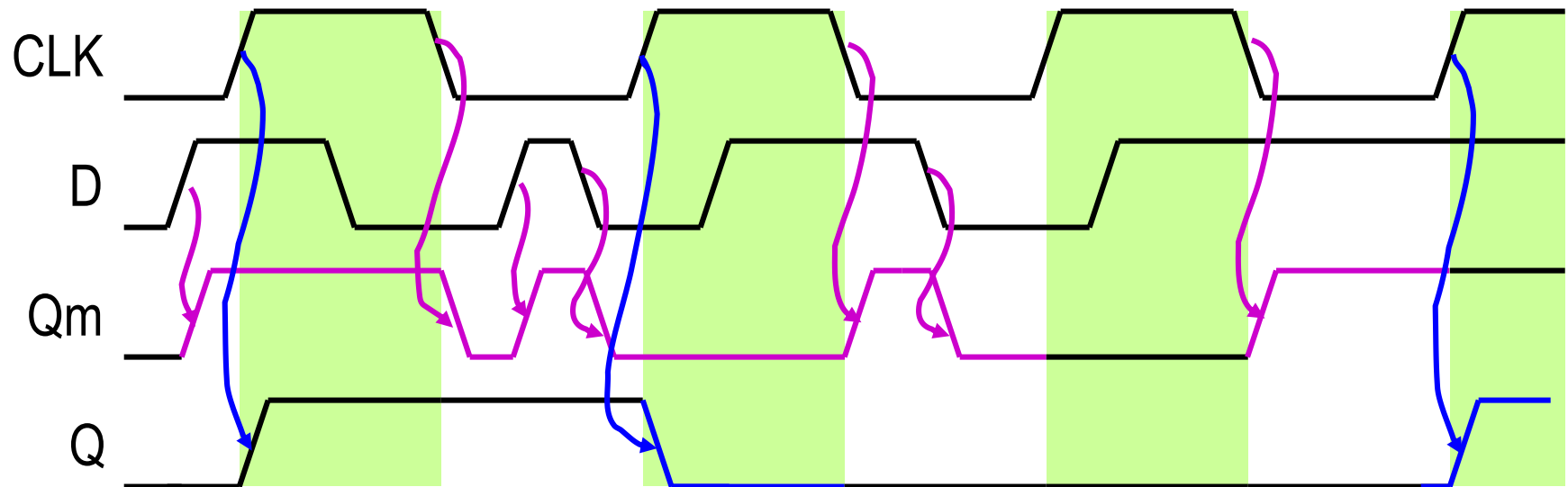
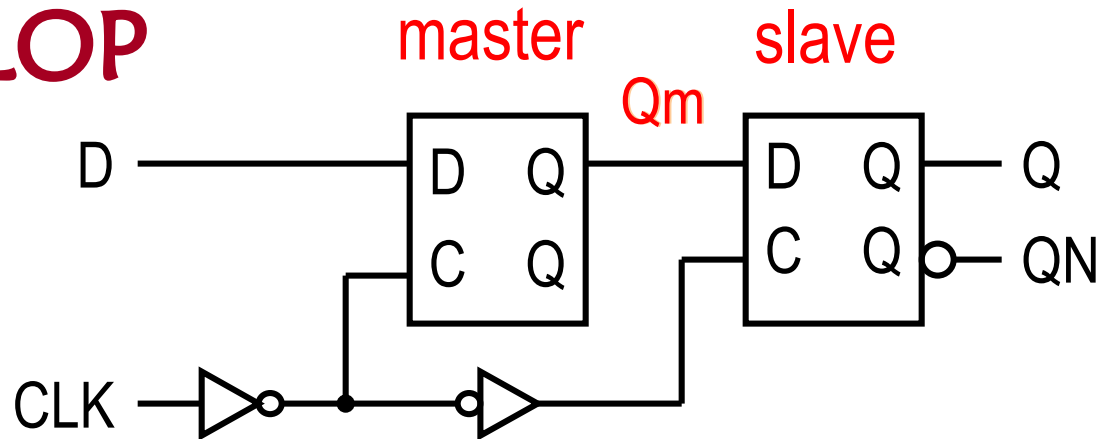
Edge-Triggered D Flip-Flops



CLK=0时, 主锁存器工作, 接收输入信号 $Q_m = D$
从锁存器不工作, 输出 Q 保持不变

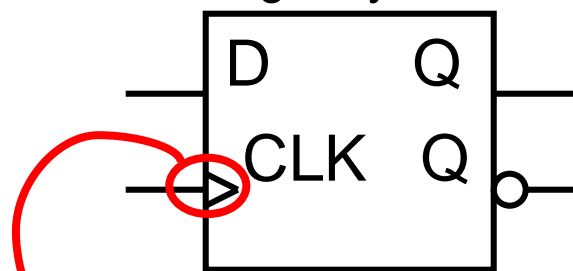
CLK=1时, 主锁存器不工作, Q_m 保持不变
从锁存器工作, 将 Q_m 传送到输出端

D Flip-FLOP



D Flip-FLOP

Logic symbol

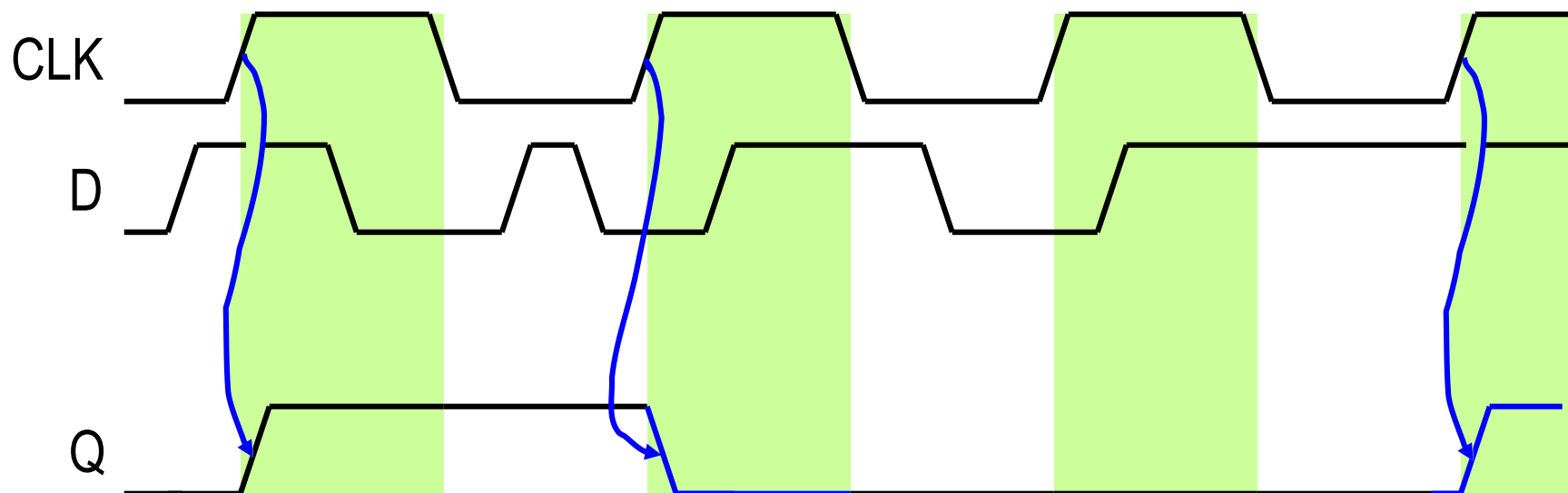


Dynamic-input indicator

表示边沿触发特性

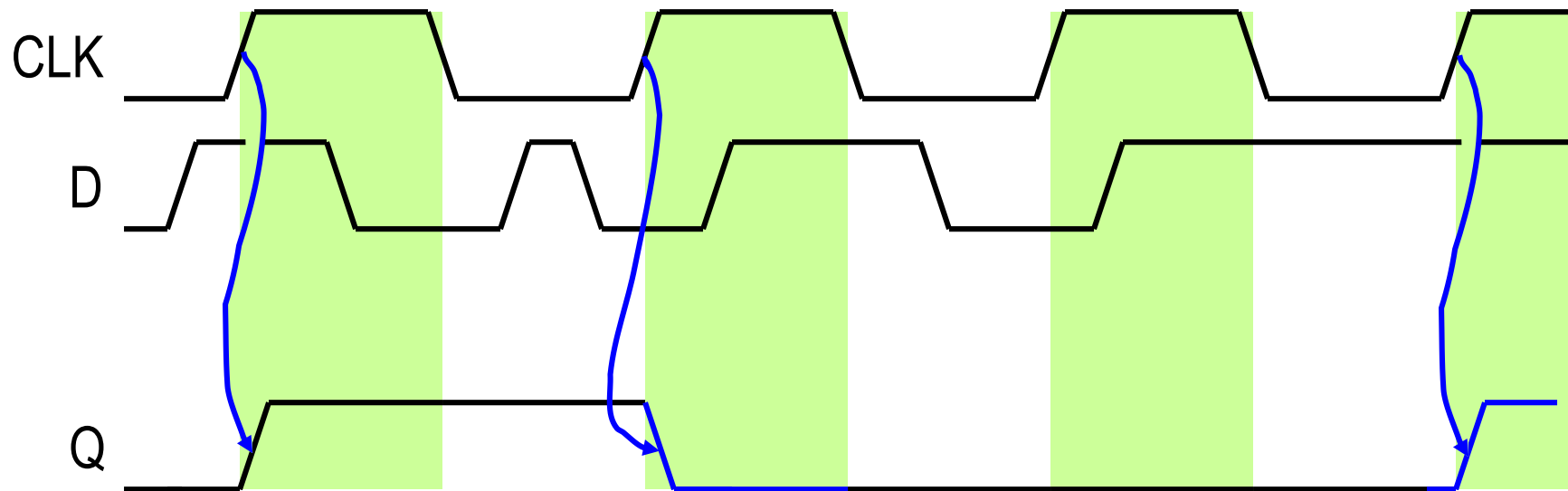
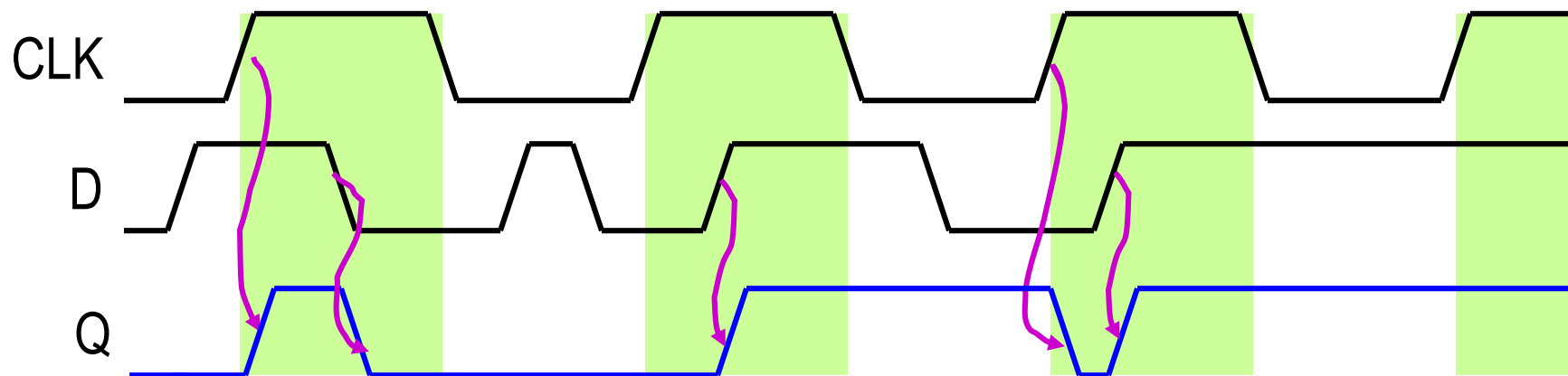
D	CLK	Q	QN
0		0	1
1		1	0
X	0	Hold	
X	0	Hold	

Function table



The state changed only at the trigger time !

D latch — 电平有效



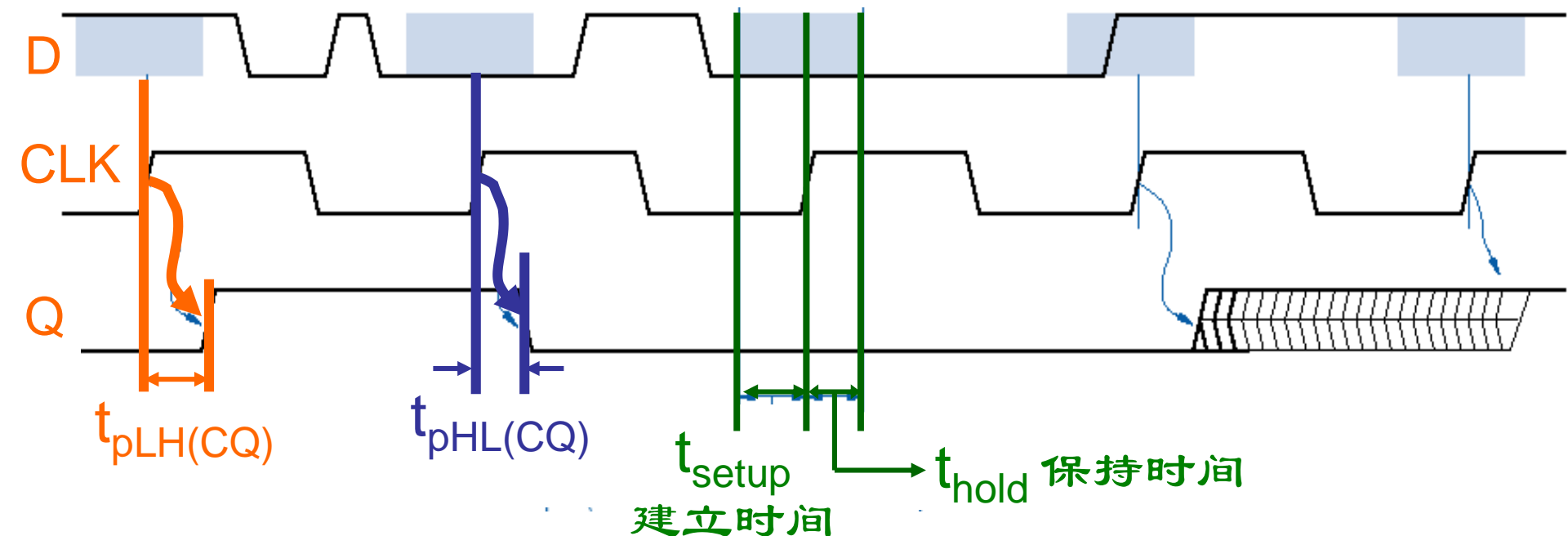
D flip-flop — 边沿有效

Time behavior of D Flip-Flop

- **Propagation deay ($\text{CLK} \rightarrow \text{Q}$)**

- ✱ **Setup time** (建立时间, 输入信号先于时钟到达的时间)

- ✱ **Hold time** (保持时间, 有效时钟沿后输入信号保持的时间)



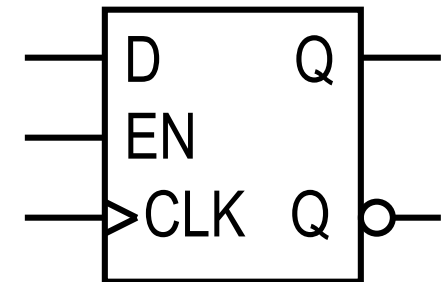
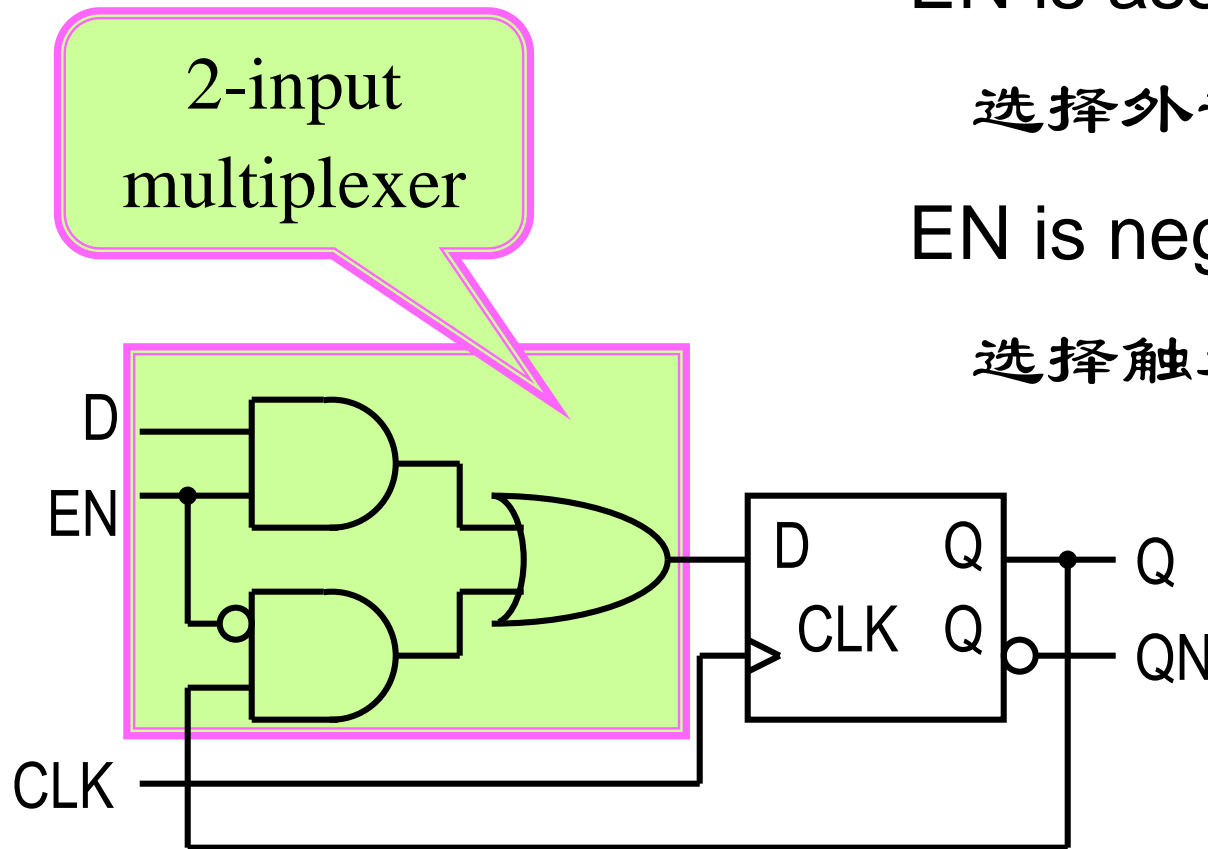
D Flip-Flop with Enable

EN is asserted (=1)

选择外部D输入

EN is negated (=0)

选择触发器当前的输出



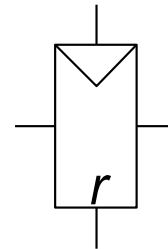
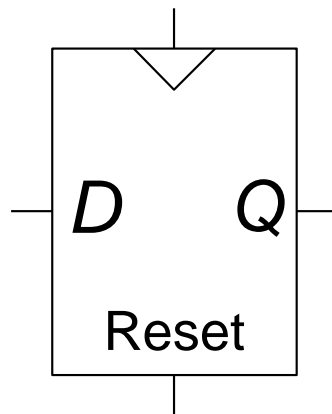
逻辑符号

$$Q^* = en \cdot D + en' \cdot Q$$

D Flip-Flop with Reset

- **Inputs:** CLK , D , $Reset$
- **Function:**
 - **$Reset = 1$:** Q is forced to 0
 - **$Reset = 0$:** flip-flop behaves as ordinary D flip-flop

Symbols



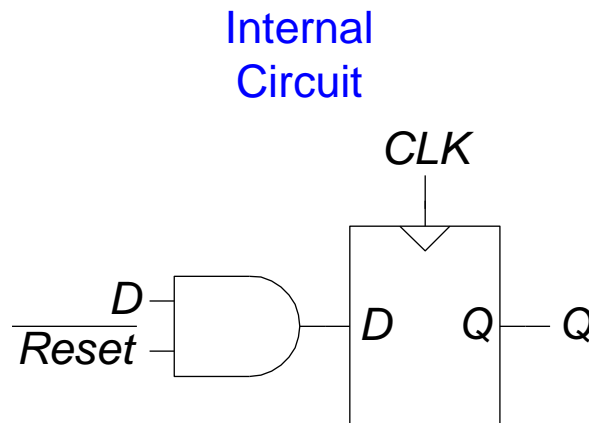


D Flip-Flop with Reset

- Two types:
 - **Synchronous:** resets at the clock edge only
 - **Asynchronous:** resets immediately when $Reset = 1$
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop
- Synchronously resettable flip-flop?

D Flip-Flop with Reset

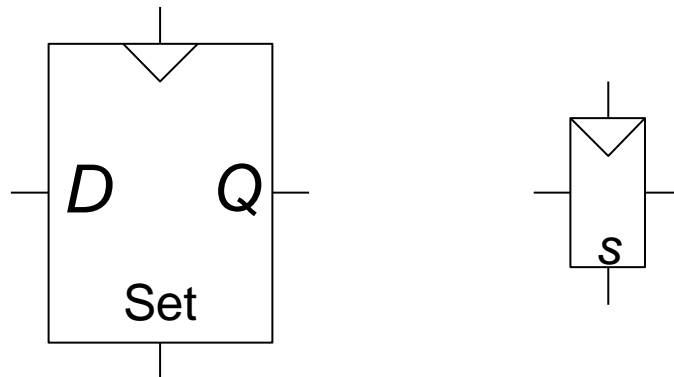
- Two types:
 - **Synchronous:** resets at the clock edge only
 - **Asynchronous:** resets immediately when $Reset = 1$
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop
- Synchronously resettable flip-flop?



D Flip-Flop with Set

- **Inputs:** CLK , D , Set
- **Function:**
 - **$Set = 1$:** Q is set to 1
 - **$Set = 0$:** the flip-flop behaves as ordinary D flip-flop

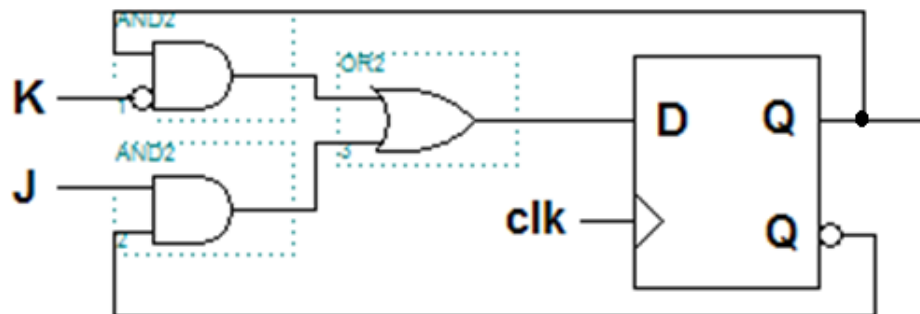
Symbols



J-K Flip-Flop

状态转移真值表

J	K	Q^n	Q^{n+1}	
0	0	0	0	} 维持
0	0	1	1	
0	1	0	0	} 清0
0	1	1	0	
1	0	0	1	} 置1
1	0	1	1	
1	1	0	1	} 翻转
1	1	1	0	



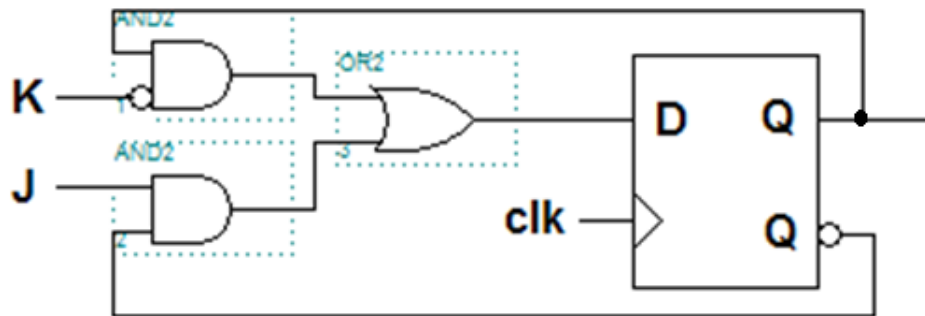
Q^{n+1}

		JK			
		00	01	11	10
Q^n	0	0	0	1	1
	1	1	0	0	1

特征方程

$$Q^* = J \cdot Q' + K' \cdot Q$$

J-K Flip-Flop

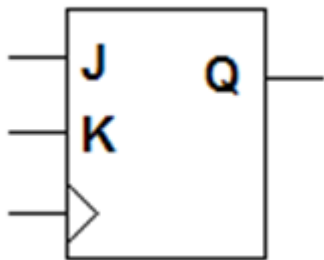






Characteristic Equation :

$$Q^* = J \cdot Q' + K' \cdot Q$$

Function Table

Logic Symbol

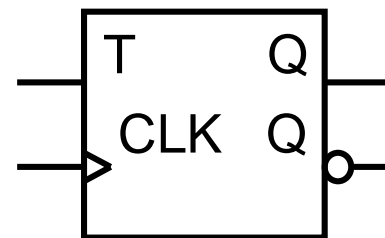
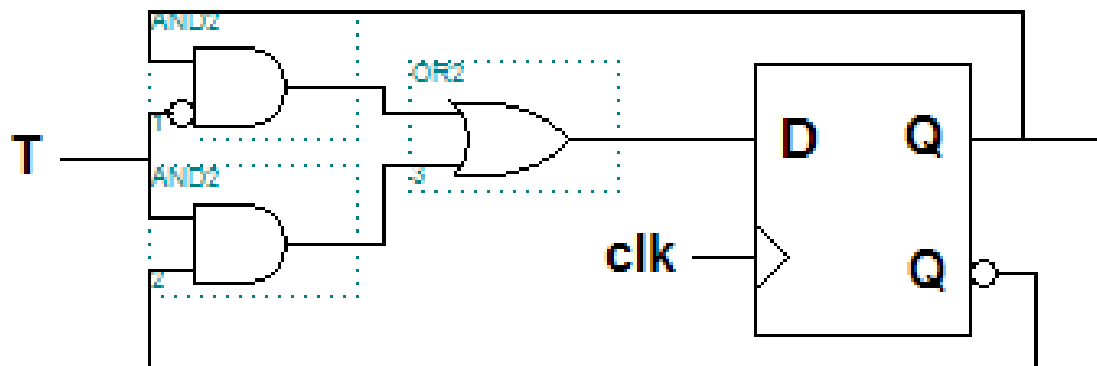


J	K	CLK	Q	QN
X	X	0	HOLD	HOLD
X	X	1	HOLD	HOLD
0	0		HOLD	HOLD
0	1		0	1
1	0		1	0
1	1		Last QN	Last Q

HOLD:保持

T Flip-flop

T (toggle)



$$Q^* = T \cdot Q' + T' \cdot Q$$

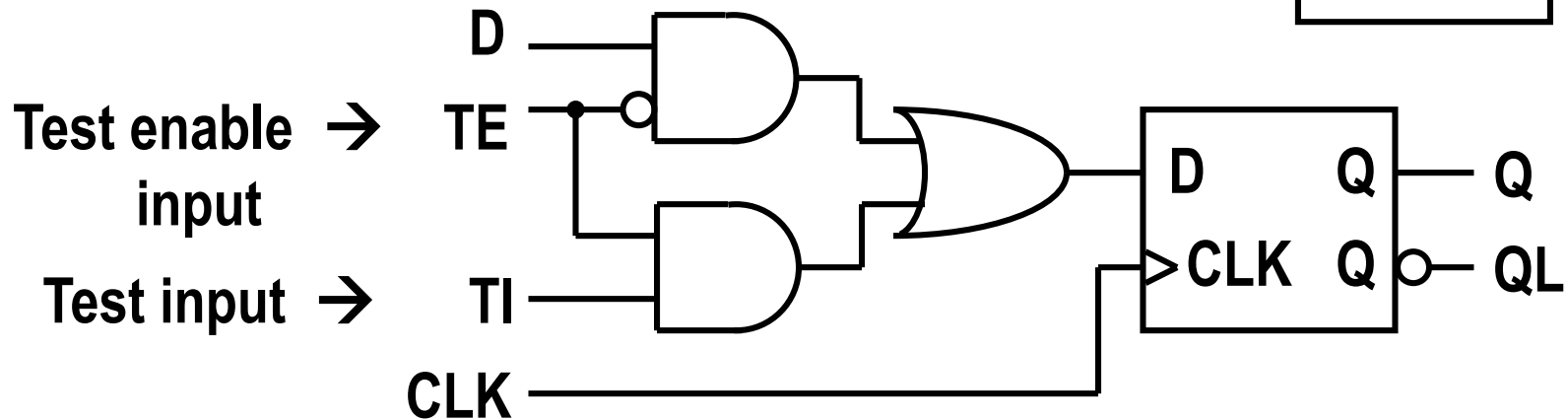
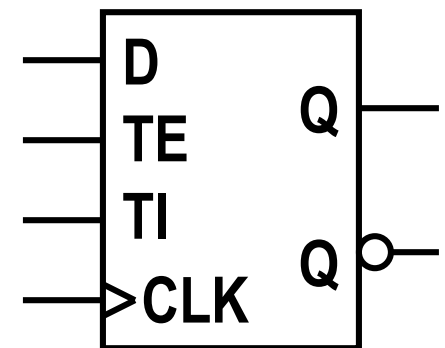
当 $T = 1$ ，触发器的特征方程为： $Q^* = Q'$ (也称为 **T'触发器**)

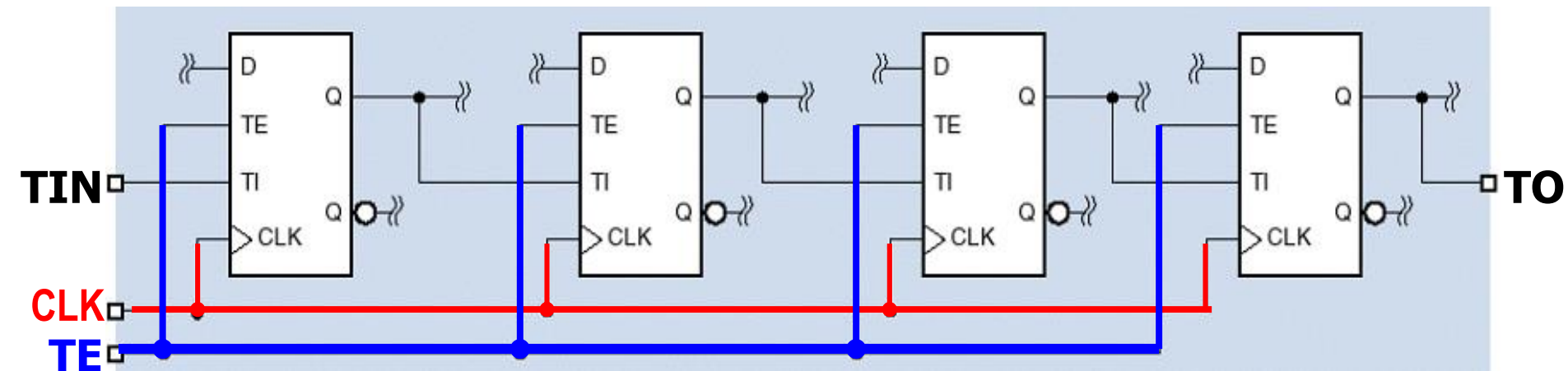
Scan Flip-Flop (扫描触发器)

Function table: P536 Fig.7-22

- ★ $TE = 0 \rightarrow$ normal mode $Q=D$
- ★ $TE = 1 \rightarrow$ test mode

Logic Symbol





★ **TE = 0 → normal mode**

★ **TE = 1 → test mode**

- ★ 每个触发器的输出端Q都与后一个触发器的TI端连接
- ★ TIN 端扫入一组测试向量（需若干个时钟触发沿）
- ★ 再经过若干个时钟的正常操作（TE=0）
- ★ 可以在TO端观察（扫出）电路的新状态



Summary of Latches and Flip-flops

- **锁存器和触发器的区别**

—— **电平有效和边沿有效的区别**

- Bistable elements without an edge-triggered clock are commonly called latches
- a flip-flop is edge-triggered/it is a bistable element with a clock input.



Summary of Latches and Flip-flops

- 按照逻辑功能的不同特点，通常可分为
 - S-R锁存器
 - D触发器，D锁存器
(most commonly used in practice.)
 - J-K触发器
 - T触发器

Characteristic equations

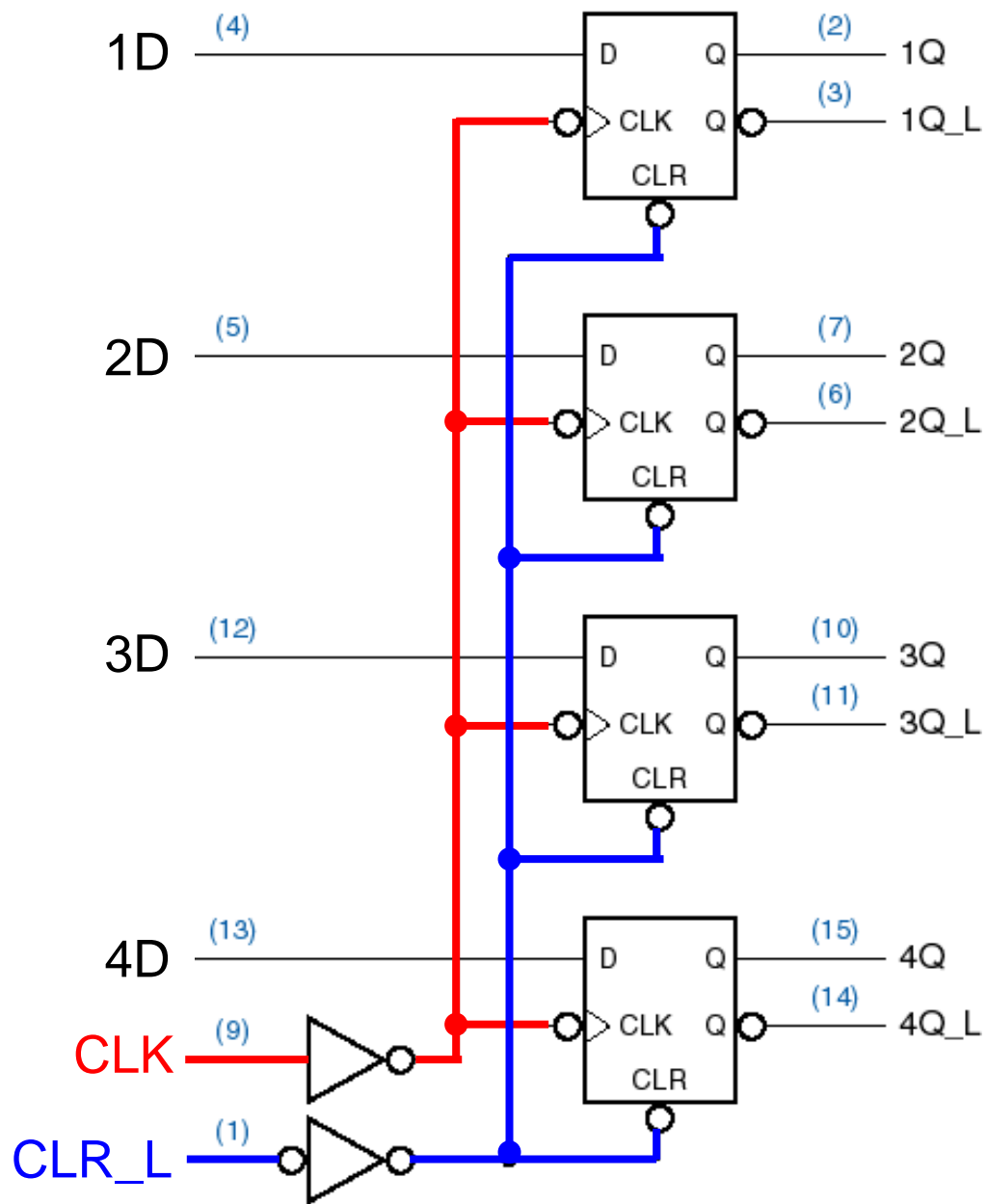
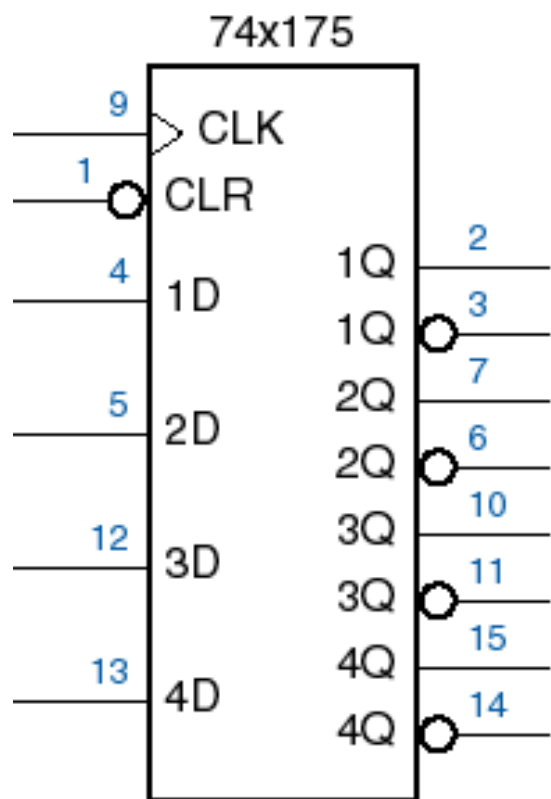
Device type	Characteristic Equation
S-R latch	$Q^* = S + R'.Q$
D latch	$Q^* = D$
D flip-flop	$Q^* = D$
D flip-flop with enable	$Q^* = EN.D + EN'.Q$
J-K flip-flop	$Q^* = J.Q' + K'.Q$
T' flip-flop	$Q^* = Q'$

Register

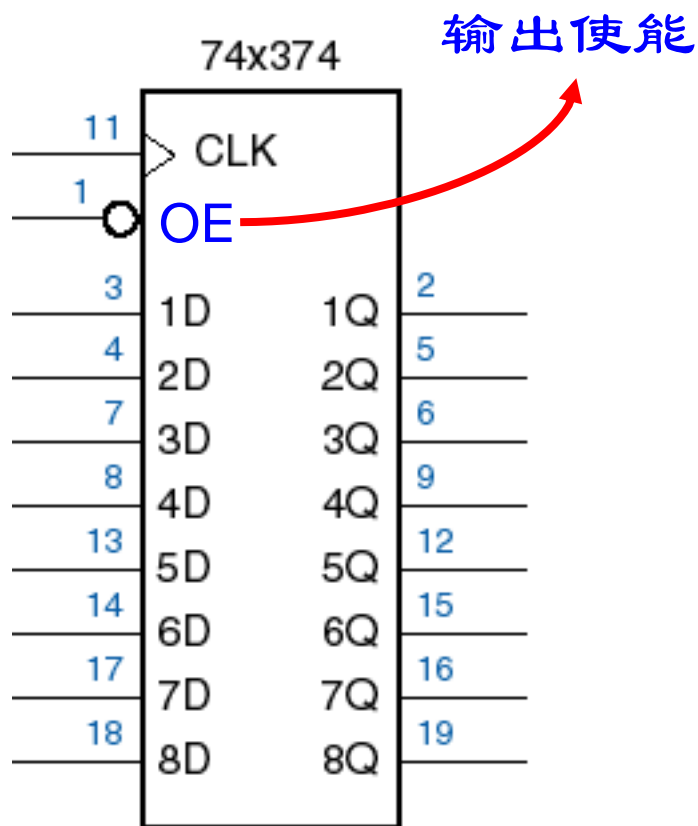
Register

- An N-bit register is a bank of **N** flip-flops (commonly D flip-flops) that share a common CLK input, so that all bits of the register are updated at the same time
- Registers are often used to store a collection of related bits, such as a byte of data in a computer.

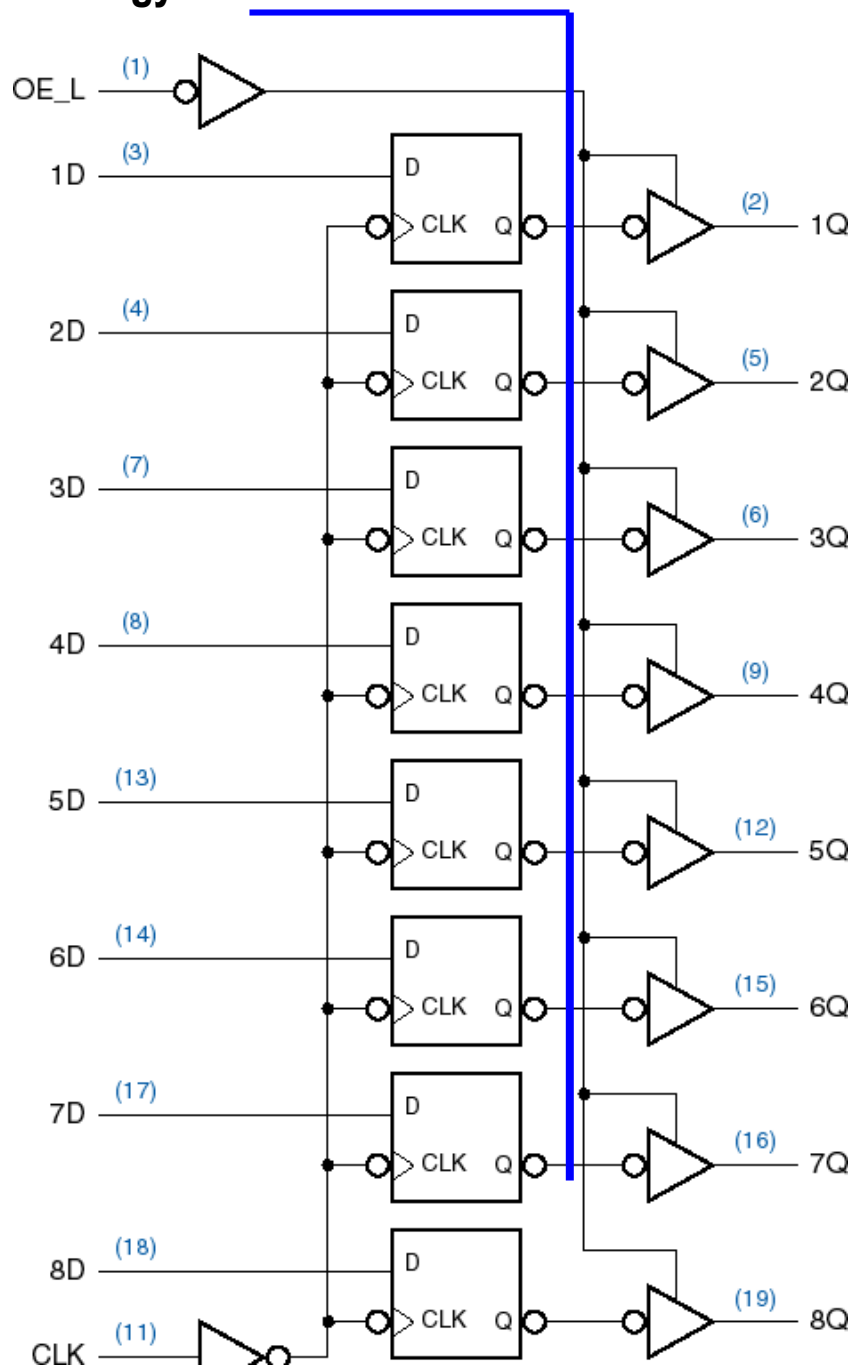
4-bit Register 74x175



8-bit Register



74x374 (三态输出)



Design Flip-Flop

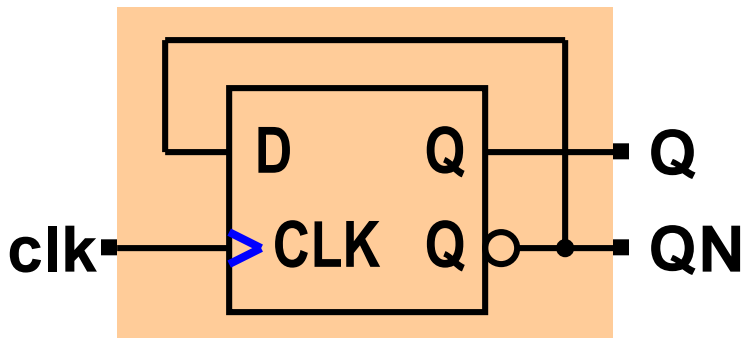
Use a D/J-k Flip-Flop to Design a T Flip-Flop

- Using a D flip-flop

$$D: Q^* = D$$

$$T: Q^* = Q'$$

$$D = Q'$$

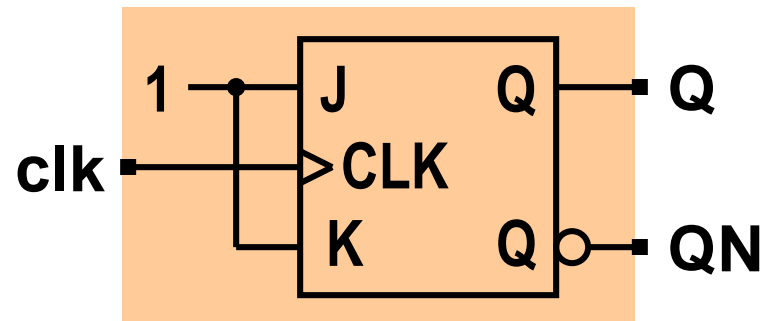


- Using a J-K flip-flop

$$JK: Q^* = J \cdot Q' + K' \cdot Q$$

$$T: Q^* = Q'$$

$$J = K = 1$$



Quiz

JK→D

Quiz

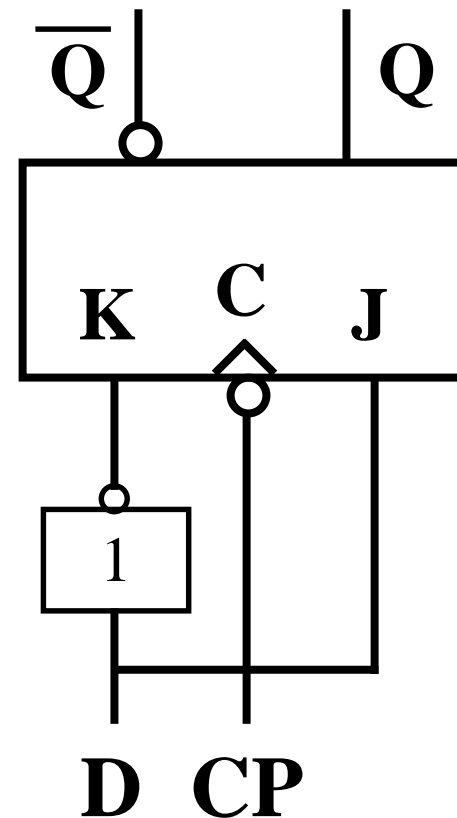
JK→D

For example: JK→D

JK - flip - flop: $Q^{n+1} = J\overline{Q}^n + \overline{K}Q^n$

D - flip - flop: $Q^{n+1} = D = D\overline{Q}^n + DQ^n$

Get : $J = D, K = \overline{D}$



Synchronous Logic Analysis and Design

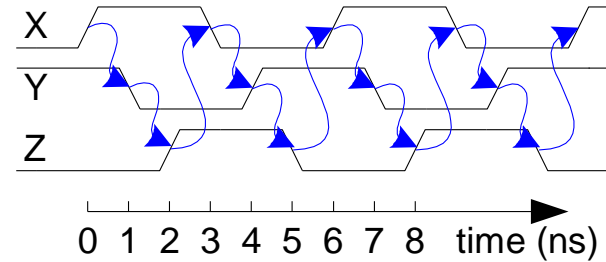
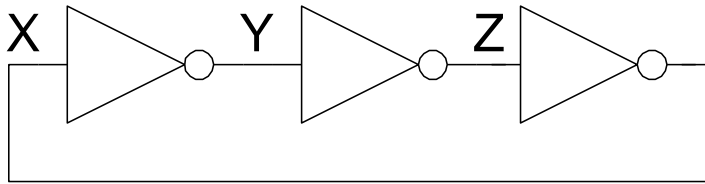
Book: C3-3.3

References: C7-7.3

Introduction

Q: 如何消除组合逻辑的竞争冒险？

the ring oscillator



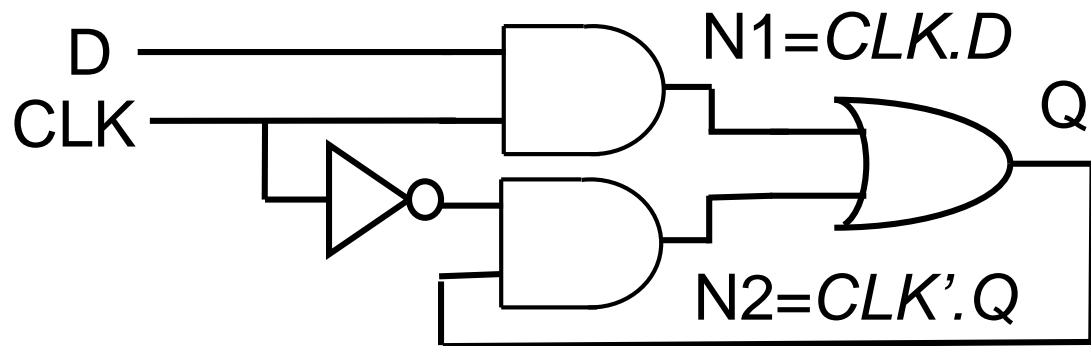
- No inputs and 1-3 outputs
- Period depends on inverter delay
- It has a *cyclic path*: output fed back to input
- **It's period is difficult to accurately predict.**

Breaks cyclic paths by **inserting registers**

---Registers contain **state** of the system

---State changes at clock edge: system **synchronized** to the clock

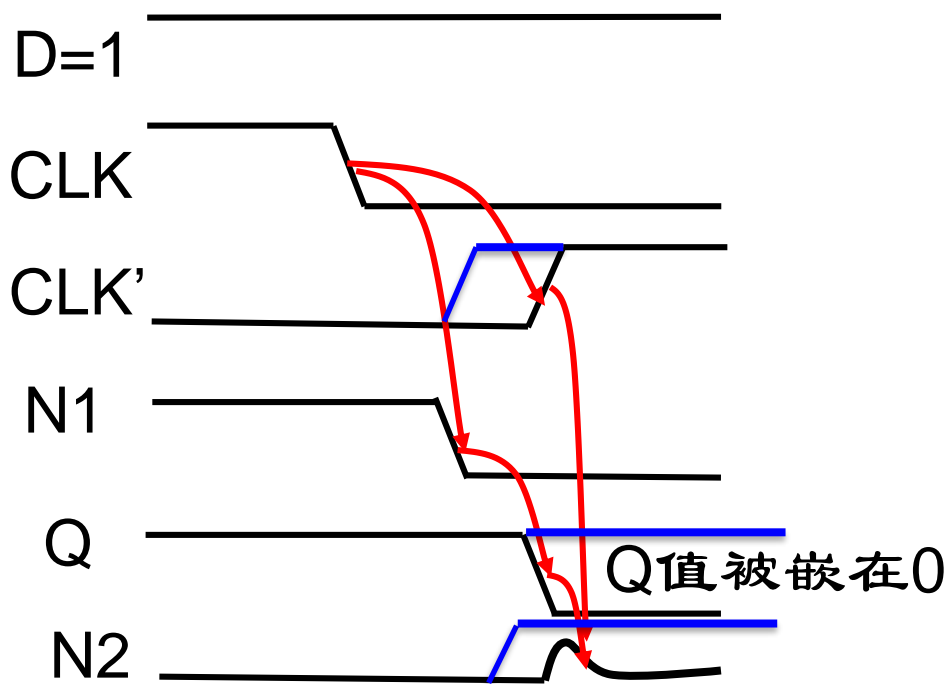
如果时钟足够慢，在下一个时钟沿到达之前，所有寄存器的输入都能够达到稳定，竞争可以被消除。



$$= CLK.D + CLK'.Q$$

$$= CLK.D + CLK'.Q_{pre}$$

CLK	D	Qprev	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



异步电路，出现了电路竞争

- **Rules of synchronous sequential circuit composition:**
 - Every circuit element is either a register or a combinational circuit
 - At least one circuit element is a register
 - All registers receive the same clock signal
 - Every cyclic path contains at least one register
- **Two common synchronous sequential circuits**
 - Finite State Machines (FSMs)
(State Machines : a generic name given to sequential circuit design)
 - Pipelines



电路的specification：输入、输出、功能和时序

同步时序电路：有限个离散的状态；含一个时钟输入，系统的状态在时钟的上升沿或者下降沿发生变化

功能说明：在**当前状态**和输入的各种组合下，此电路的**下一状态**和输出值

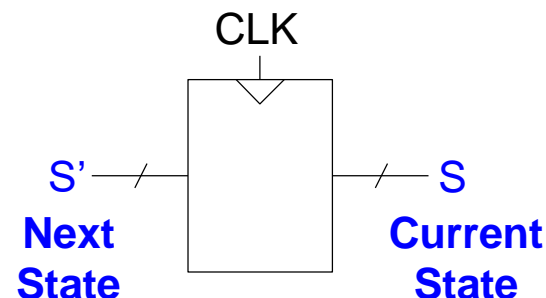
---Current state：当前系统的状态

---Next state：下一个时钟沿系统要进入的状态

时序说明：从时钟上升沿（或者下降沿）直到输出发生变化的时间，输入相对时钟上升沿（或者下降沿）稳定的时间...

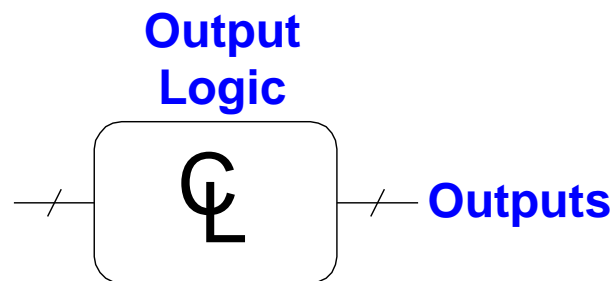
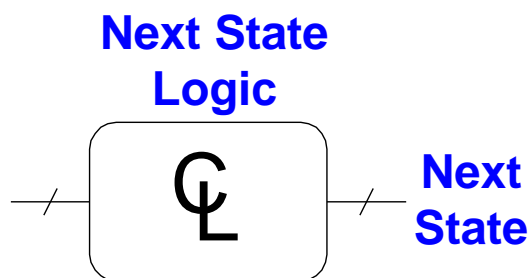
Synchronous Logic

- Consists of:
 - **State register**
 - Stores current state
 - Loads next state at clock edge
 - **Combinational logic**
 - Computes the next state
 - Computes the outputs



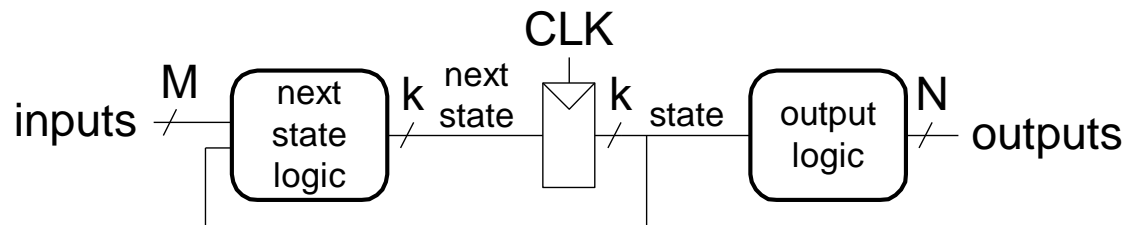
current state: 目前系统的状态

Next state: 下一个时钟沿系统要进入的状态

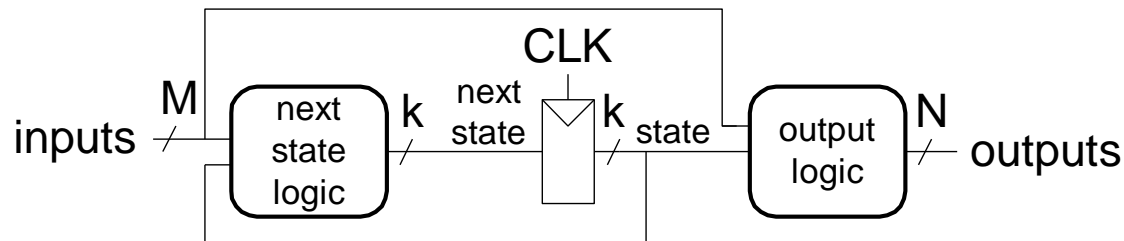


- Next state determined by current state and inputs
- Two types of finite state machines differ in output logic:
 - **Moore FSM:** outputs depend only on current state
 - **Mealy FSM:** outputs depend on current state *and* inputs

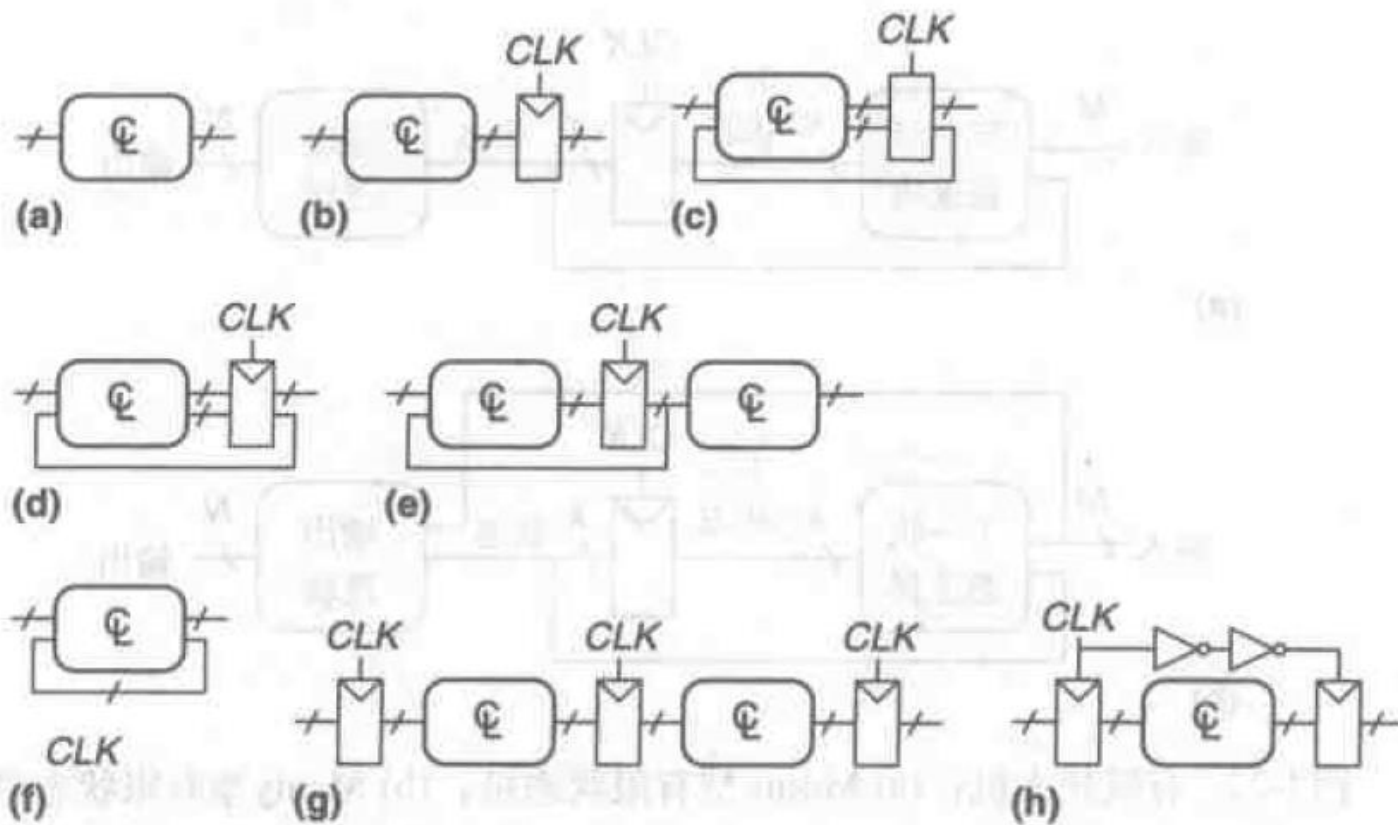
Moore FSM



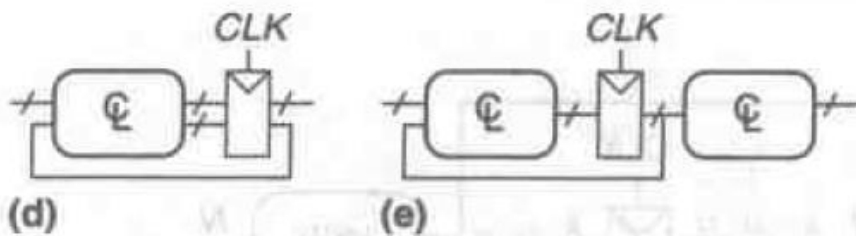
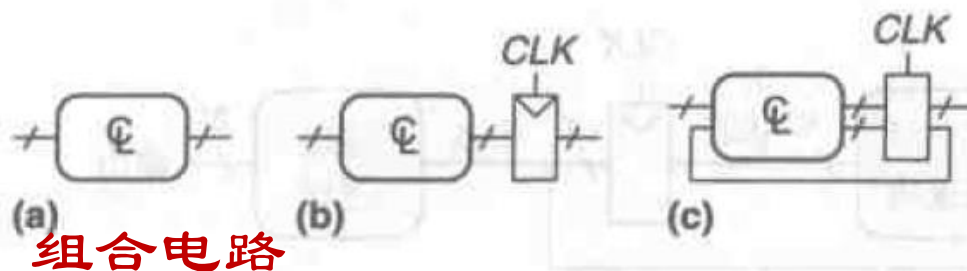
Mealy FSM



下列电路中哪些电路是同步时序电路？



下列电路中哪些电路是同步时序电路？



流水线同步时序电路

异步时序电路

Finite State Machines (FSM)

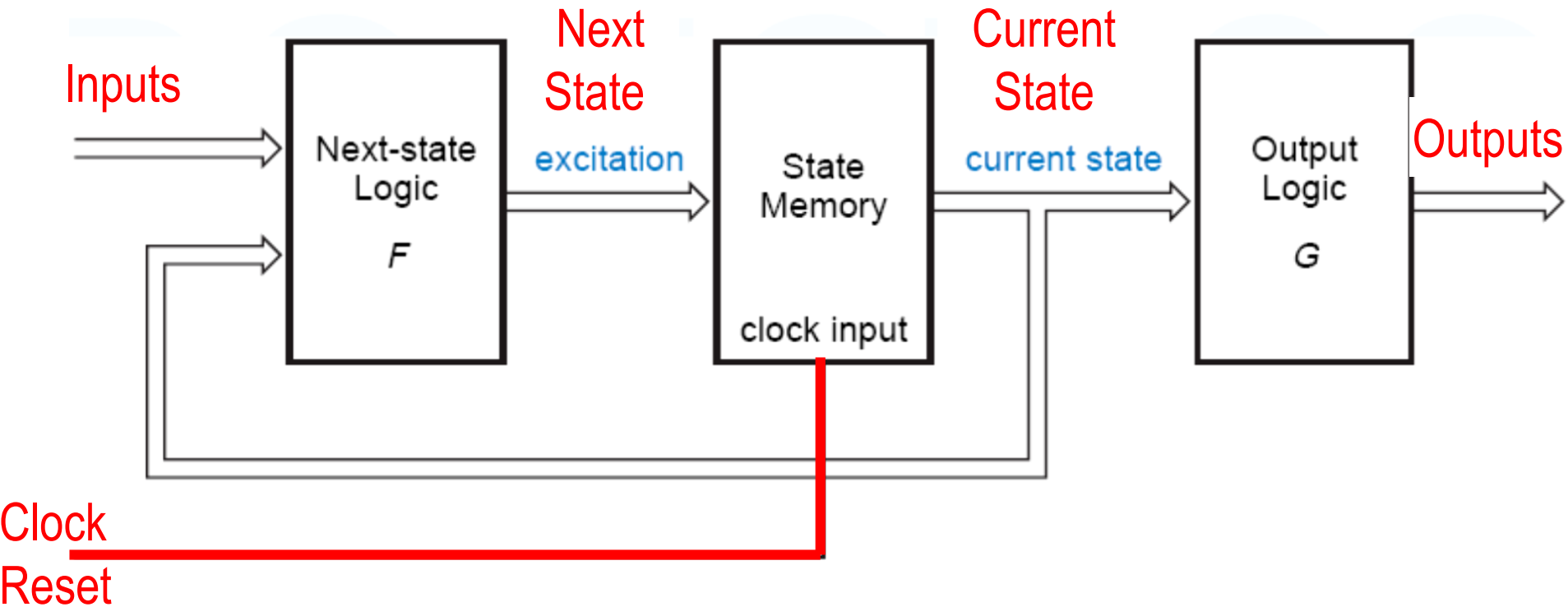
Book: C3-3.4

References: C7-7.4,7.5

Finite State Machines (FSM)

- **State-Machine Structure**
- **Clocked Synchronous State Machine Analysis**
- **Clocked Synchronous State-Machine Design**

State-Machine Structure(Moore)



- ❖ **State Memory** (i.e., register, flip-flops)
- ❖ **Next state = F (current state, input)** (**Excitation equation**)
- ❖ **Output = G (current state)** (**Output equation**)



● State Memory(State Register)

- store the *Current State* of the machine
- if there are "**n**" registers, there can be **2^n** states
- outputs: Current State
- inputs : Next State
- State Transition
 - Upon a clock edge, the machine changes from the "Current State" to the "Next State"
 - After the clock edge, we reassign back the names (i.e., $Q = \text{Current State}$, $Q^* = \text{Next State}$)



- **Excitation equation** (激励方程或驱动方程)

- Express the excitation signals as functions of the current state and input.

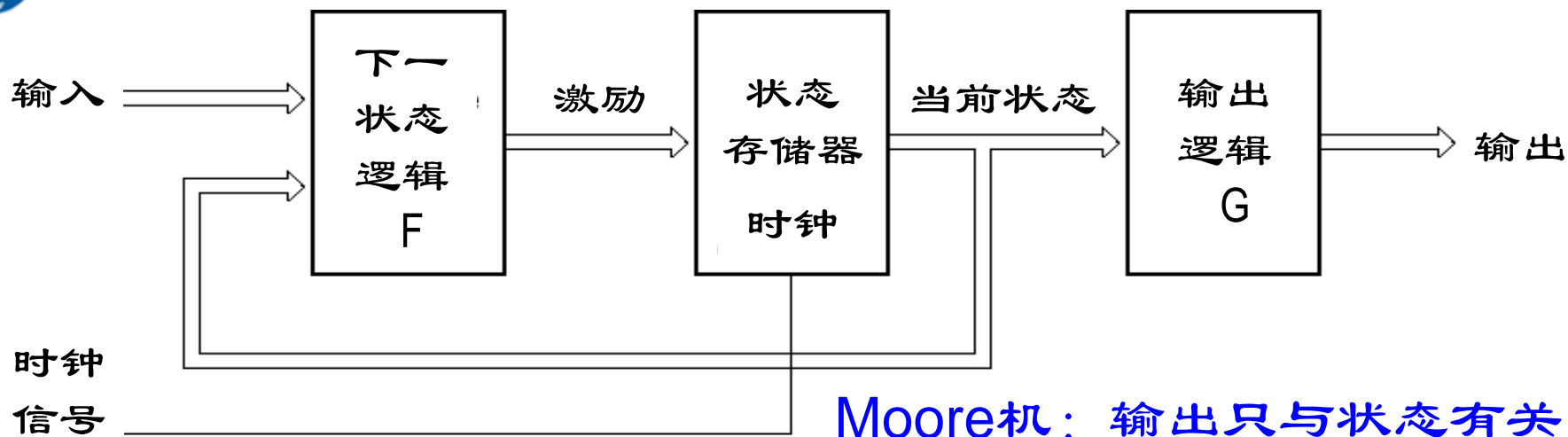
- Next state = F (current state, input)

- **Output equation** (输出方程)

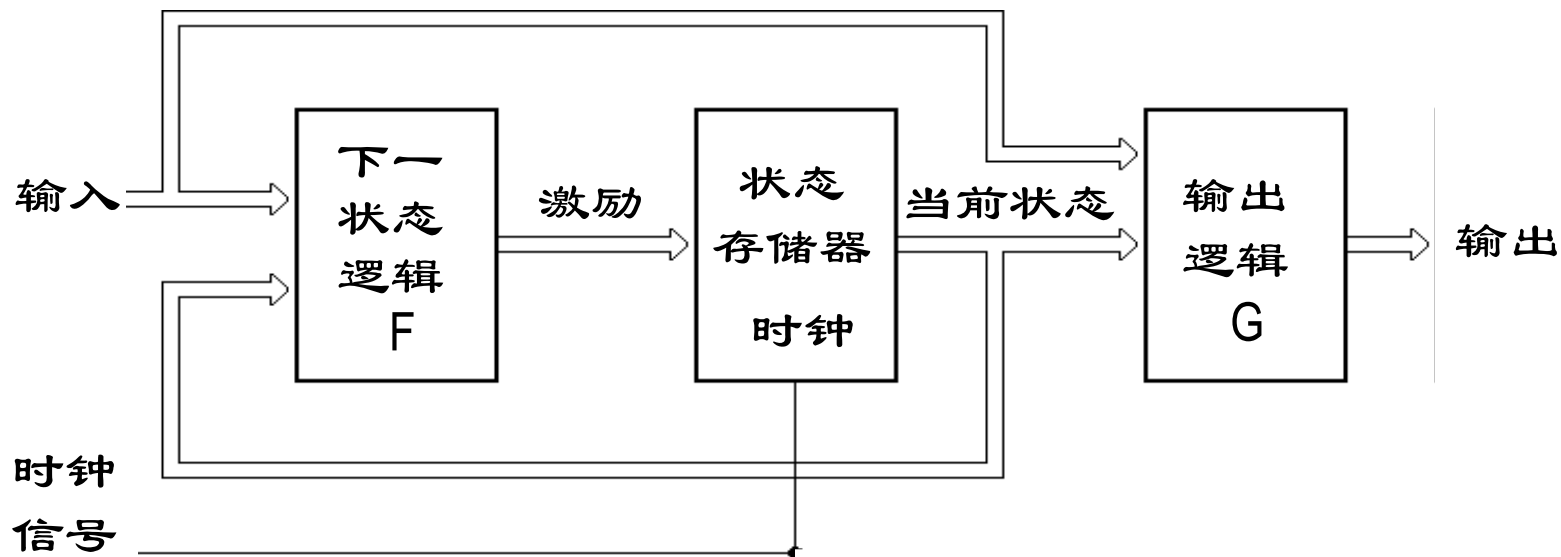
- Output = G (current state, input)

- **Transition equation** (转移方程)

- turn a "Characteristic Equation" into an "Excitation Equation"



Mealy机：输出取决于状态和输入



Clocked Synchronous State Machine Analysis



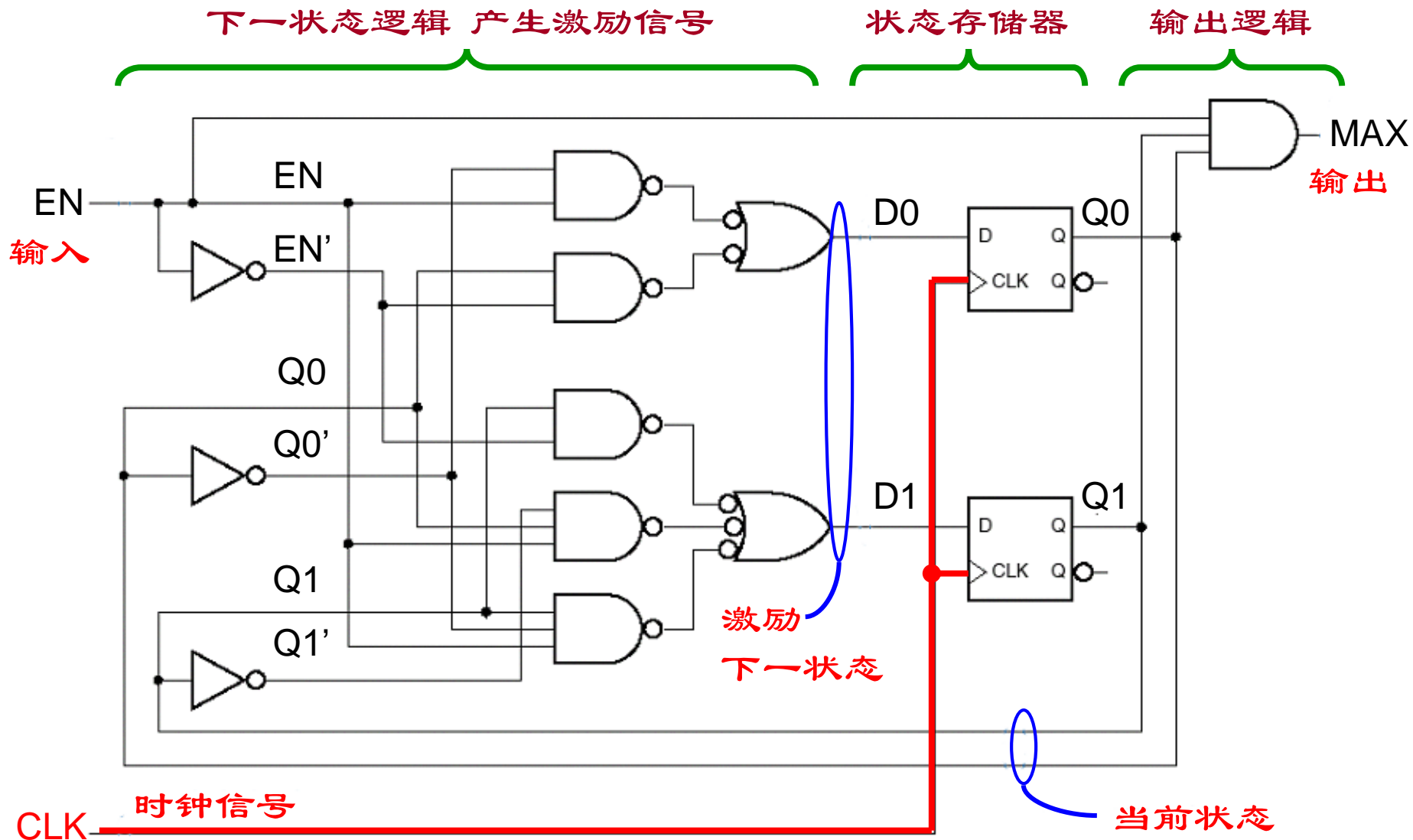
Clocked Synchronous State Machine Analysis

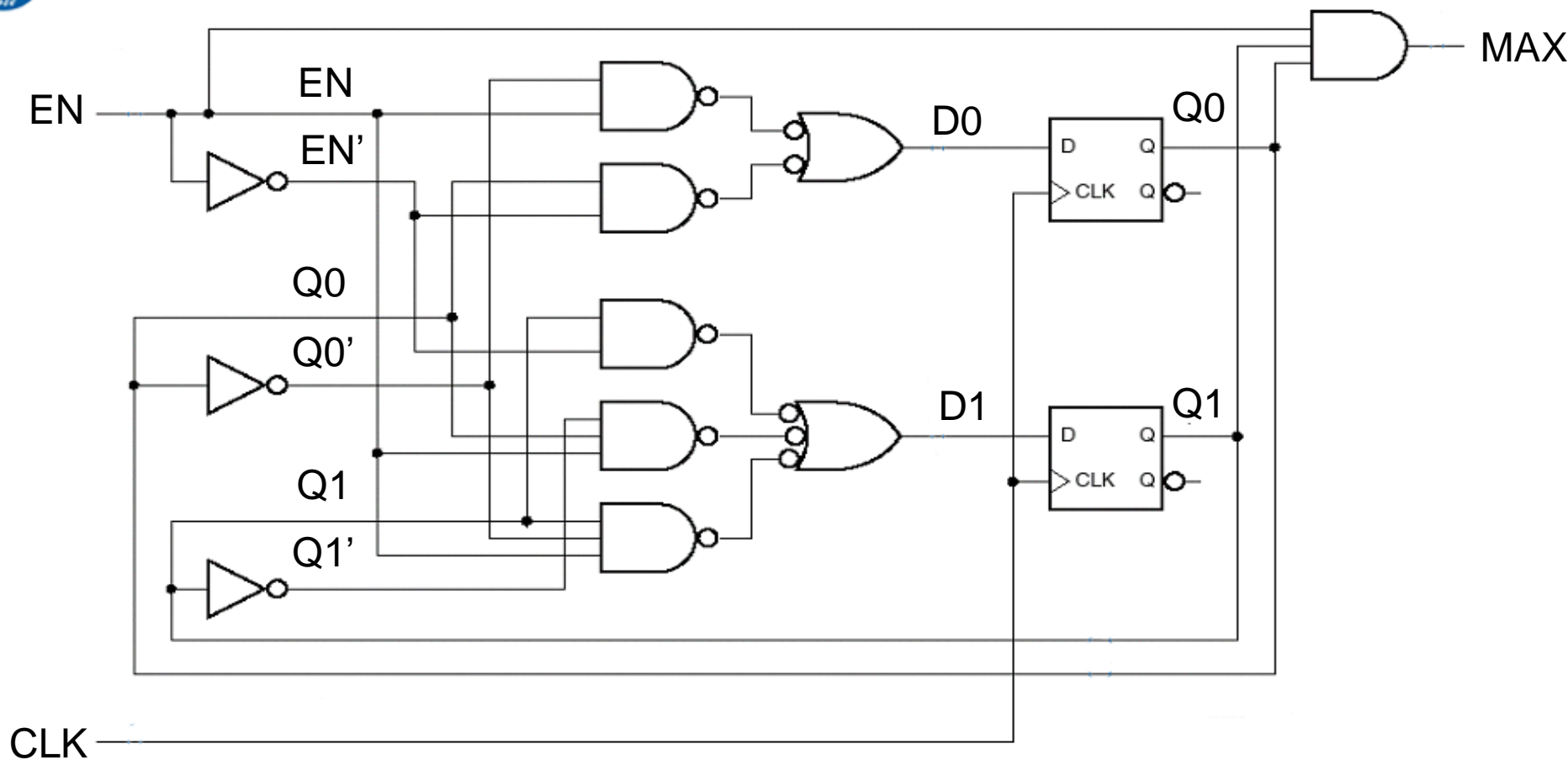
● 基本步骤：

- 确定下一状态函数 F 和输出函数 G
- 将 F 代入触发器的特征方程得到下一状态 Q^*
- 利用 Q^* 、 G 构造状态/输出表
- 画出状态图、波形图（可选）
- 检查电路是否可以自启动
- 描述电路功能



Example: Clocked Synchronous State Machine Analysis (D Flip-Flop)

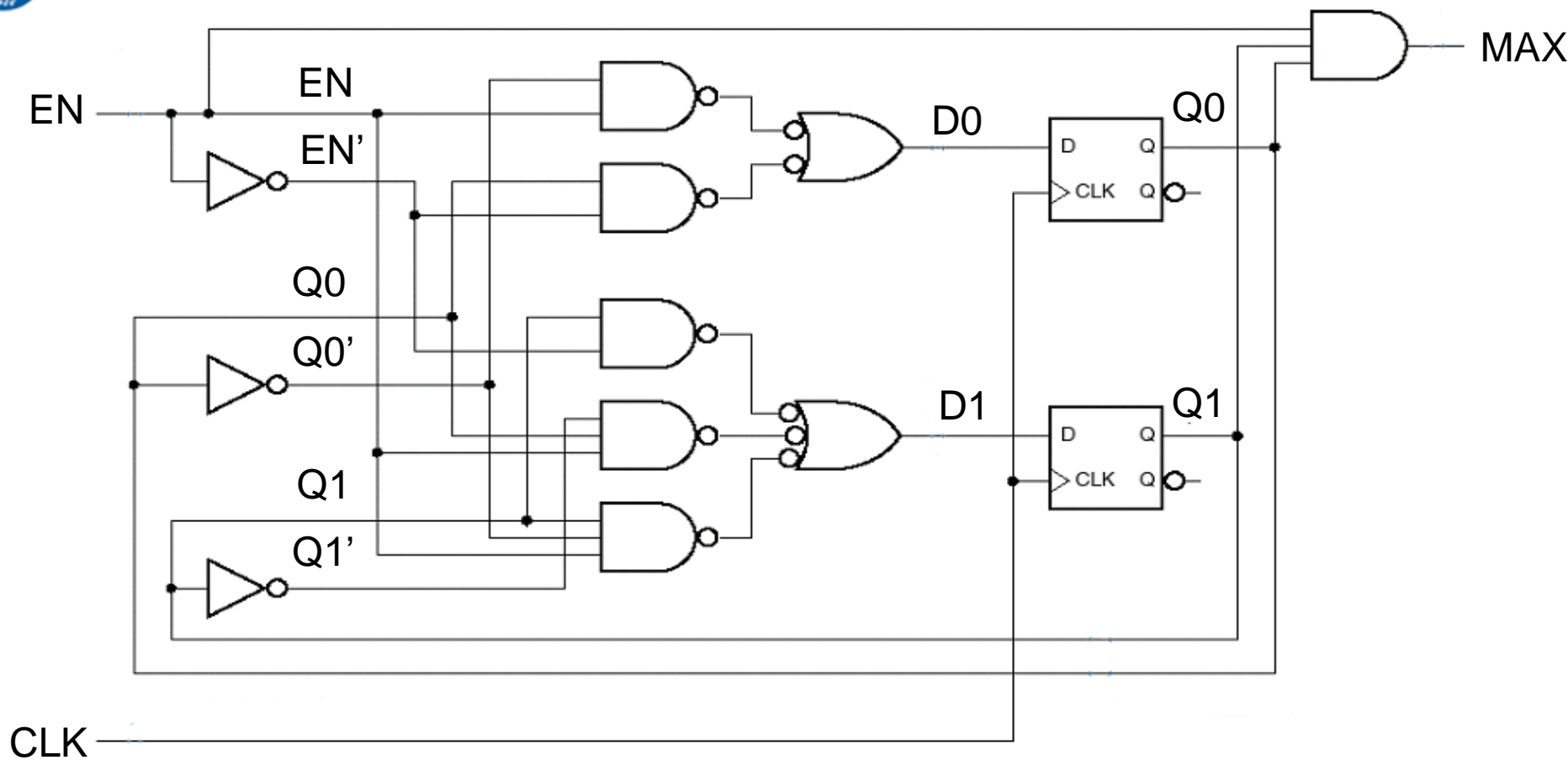




1、由电路得到激励方程 (F)

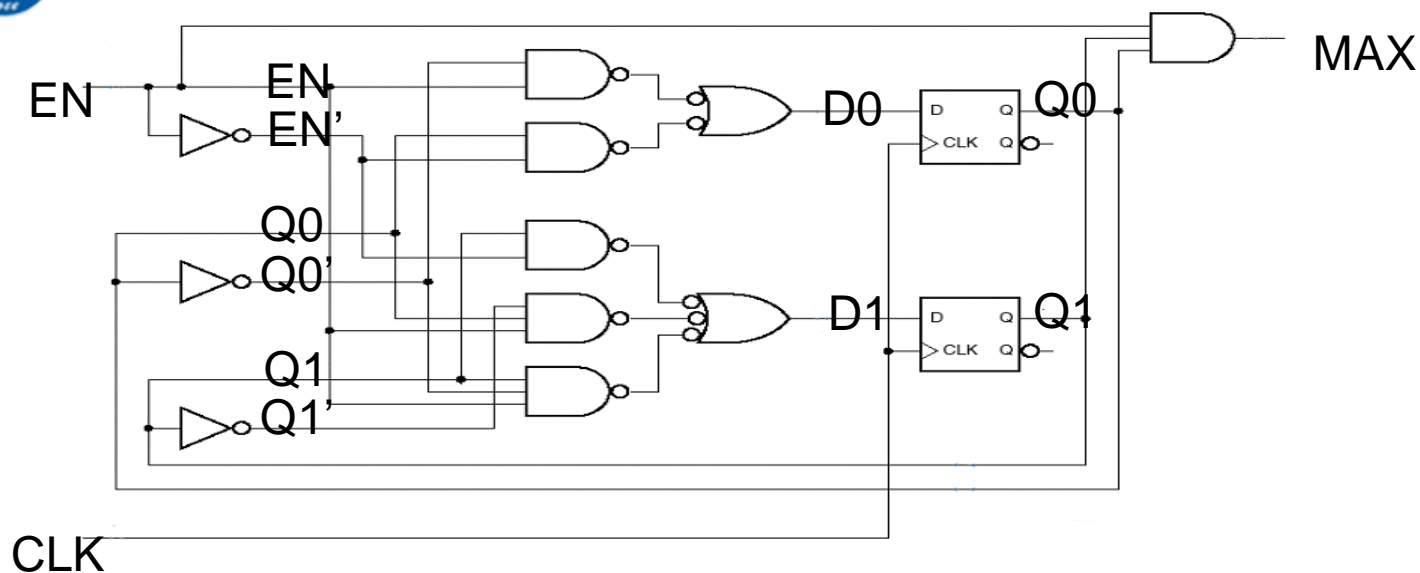
$$D0 = Q0 \cdot EN' + Q0' \cdot EN$$

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



2、由电路得到输出方程 (G)

$$MAX = Q1 \cdot Q0 \cdot EN$$



3、由激励方程和触发器特征方程

得到转移方程 (状态方程)

D触发器特征方程: $Q^* = D$

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

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

1、由电路得到激励方程 (F)

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

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

4、由转移方程和输出方程得到状态/输出表

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

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

$$MAX = Q1 \cdot Q0 \cdot EN$$

状态输出表

S	EN	
	0	1
0 0	00, 0	01, 0
0 1	01, 0	10, 0
1 0	10, 0	11, 0
1 1	11, 0	00, 1
Q1Q0	Q1*Q0*, MAX	

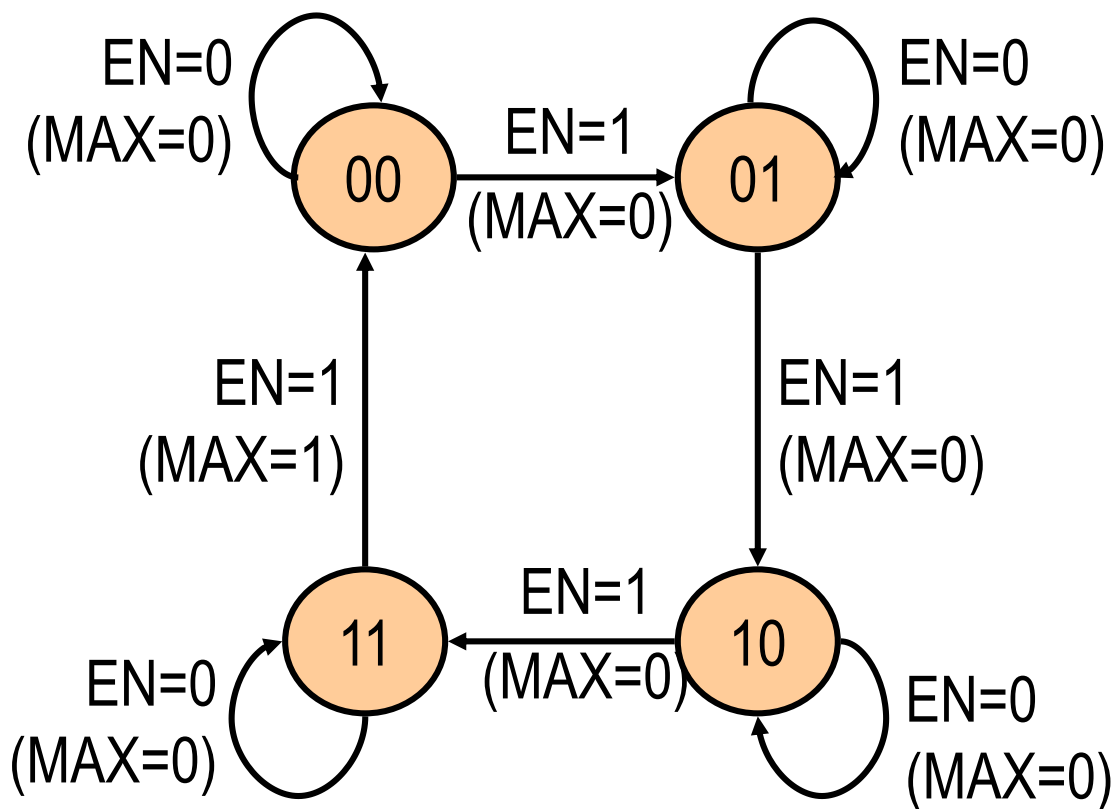
状态转换表

EN	Q1	Q0	Q1*	Q0*	MAX
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	1

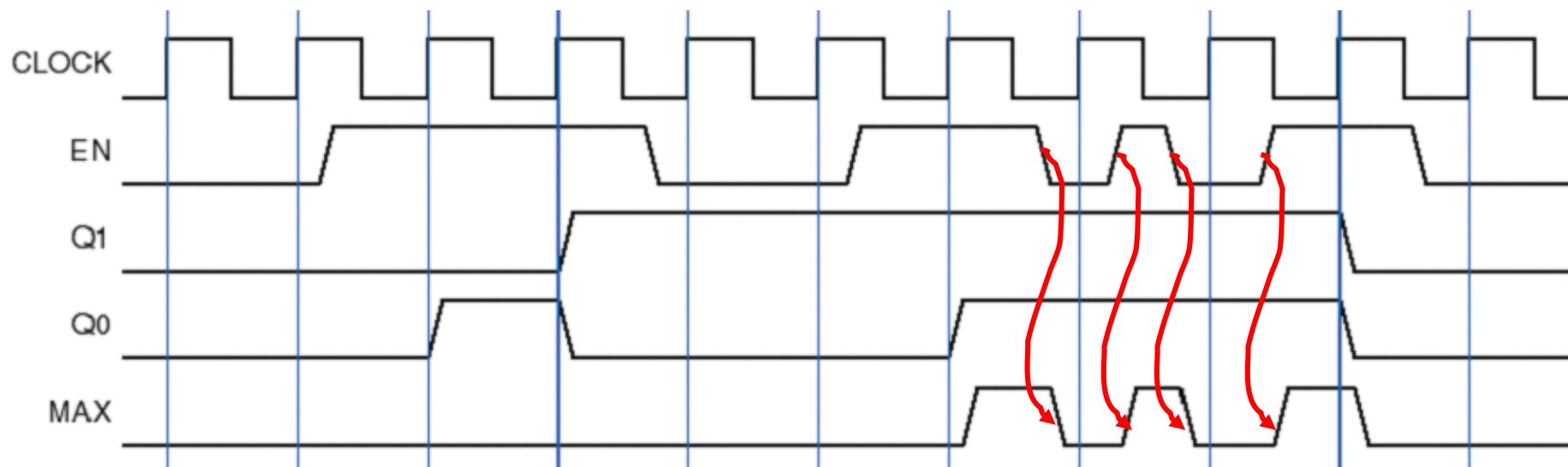
5、State Diagram (画状态图)

逻辑功能描述：具有使能端EN的2位二进制模4加法计数器
电路输出与输入有关 —— Mealy机

S	EN	
	0	1
0 0	00,0	01,0
0 1	01,0	10,0
1 0	10,0	11,0
1 1	11,0	00,1
Q1Q0	Q1*Q0*, MAX	



6、Timing Diagram (画时序图)



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

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

$$MAX = Q1 \cdot Q0 \cdot EN$$

- 可以给每个状态命名

- 通常用 S 表示当前状态, S^* 表示下一状态

$Q1\ Q0$	EN	
	0	1
00	00	01
01	01	10
10	10	11
11	11	00
$Q1^*\ Q0^*$		

Transition table

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

state table

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

state/output
table

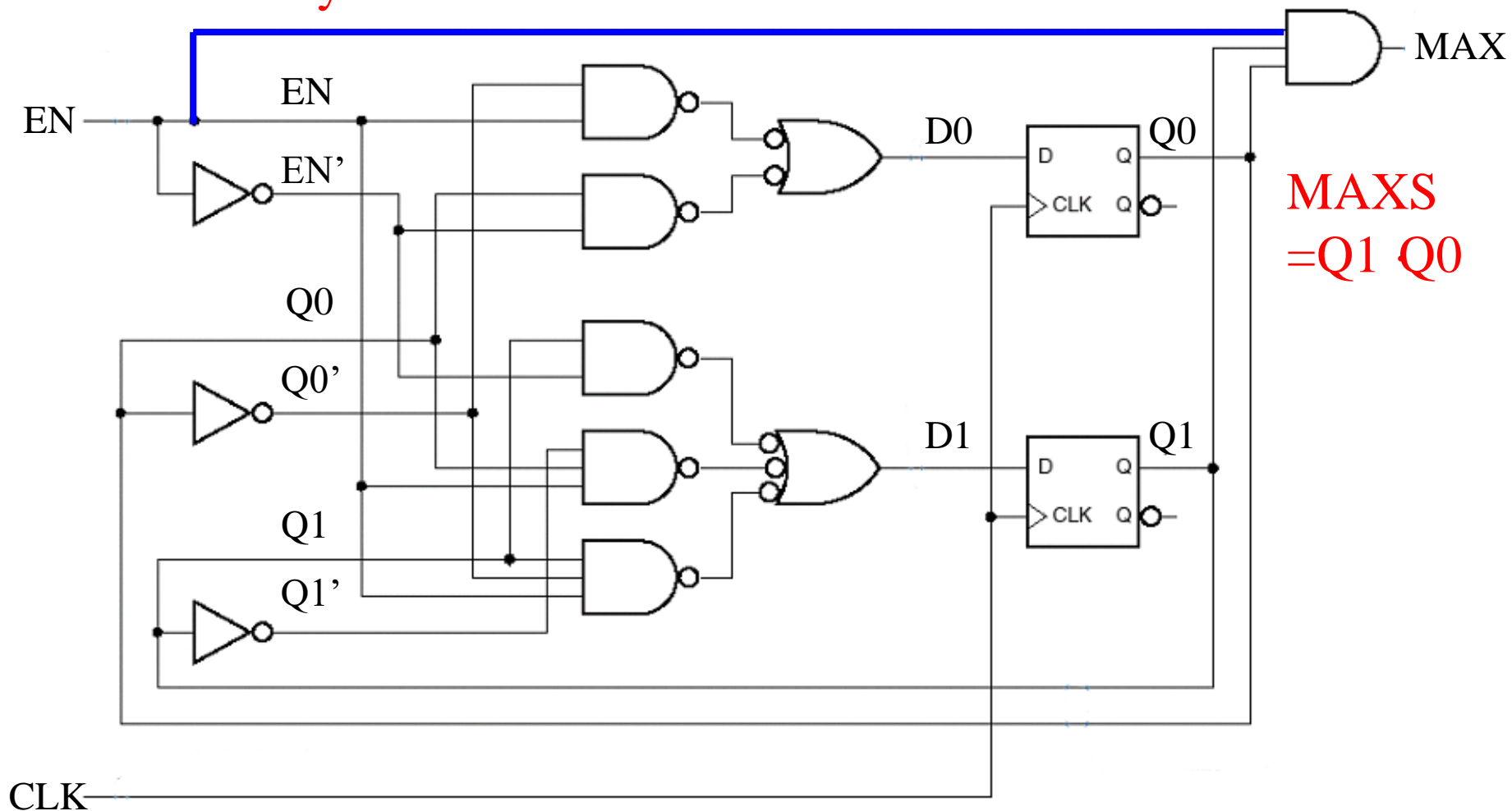


State Machine Tables

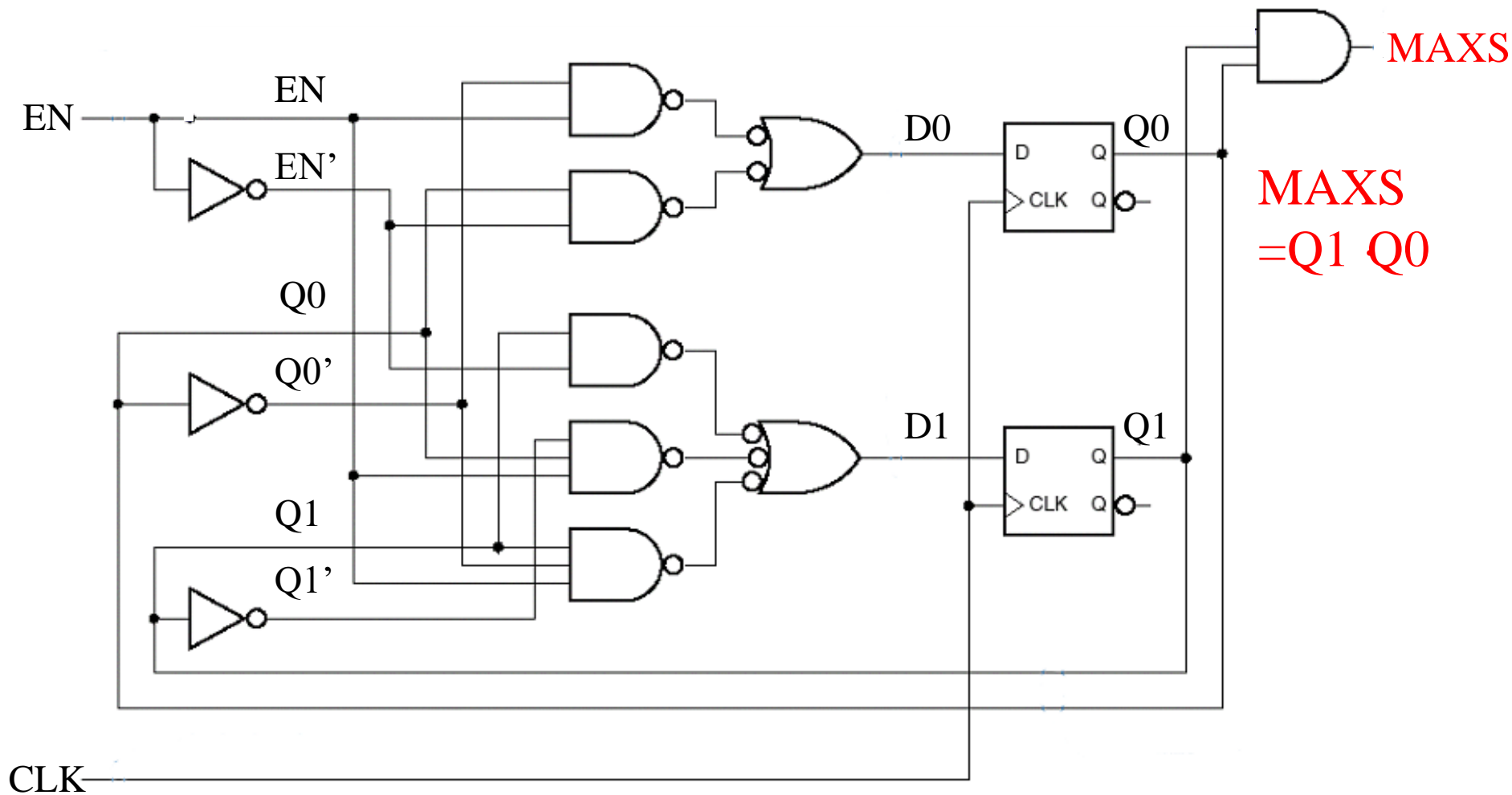
Officially, we use the following terms:

- **State Table** - list of the descriptive state names and how they transition
- **Transition Table** - using the explicitly state encoded variables and how they transition
- **Output Table** - listing of the outputs for all possible combinations of Current States and Inputs
- **State/Output Table** - combined table listing Current/Next states and corresponding outputs

Mealy机

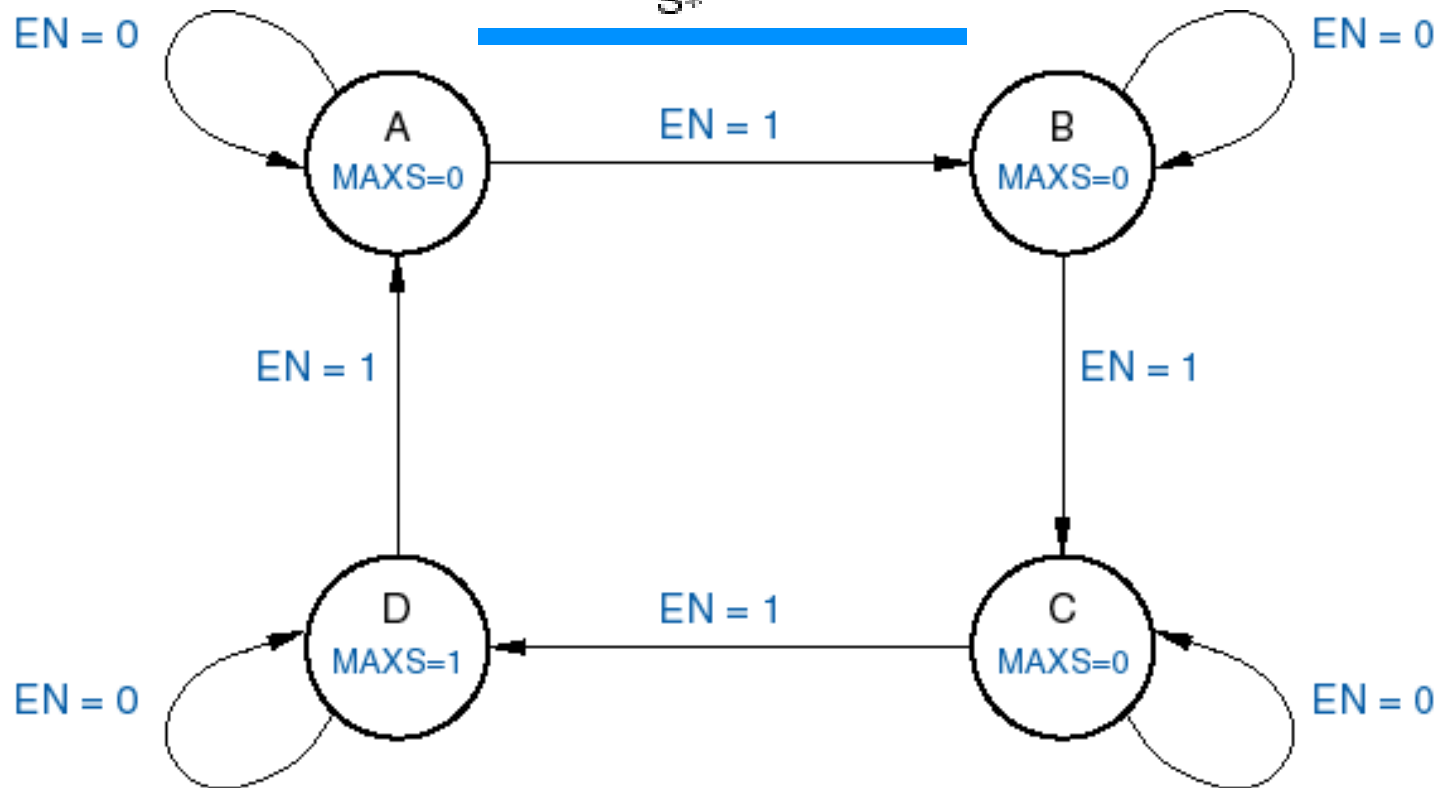


Moore机



对应的Moore机的
状态输出表和状
态图

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

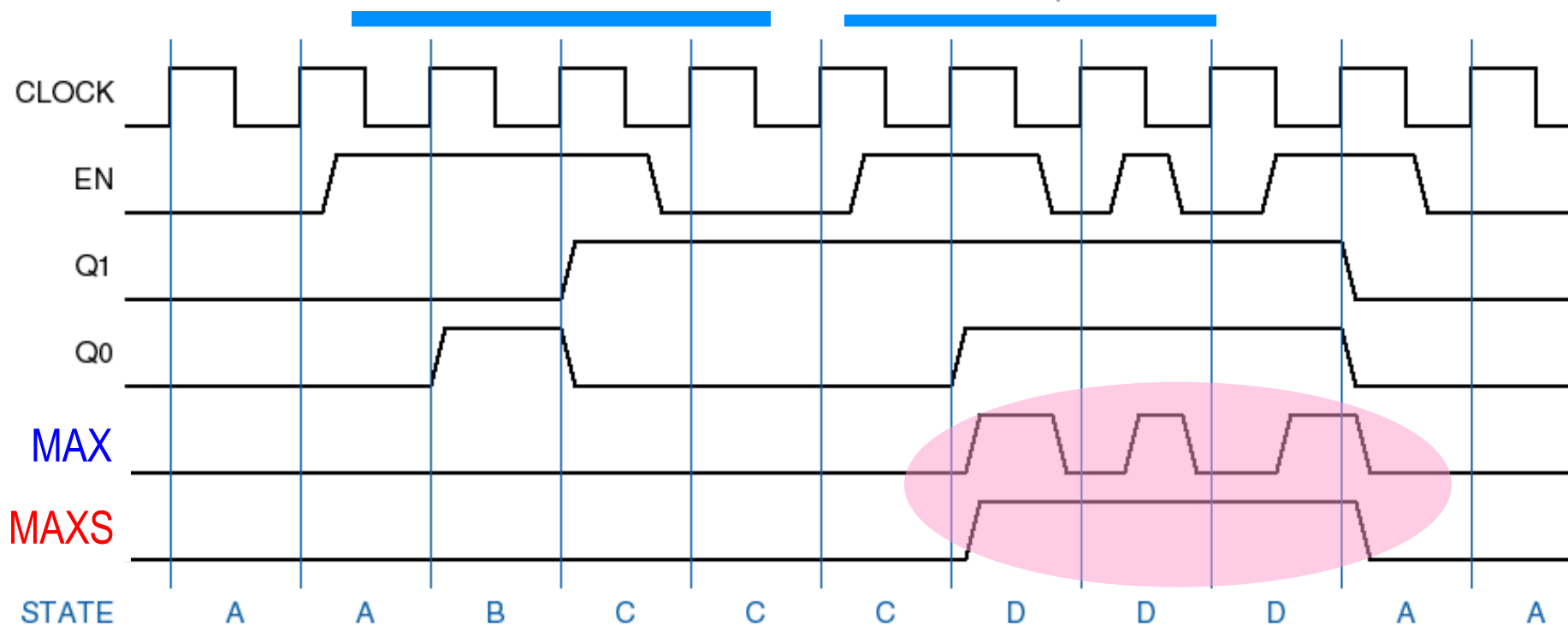


Moore机

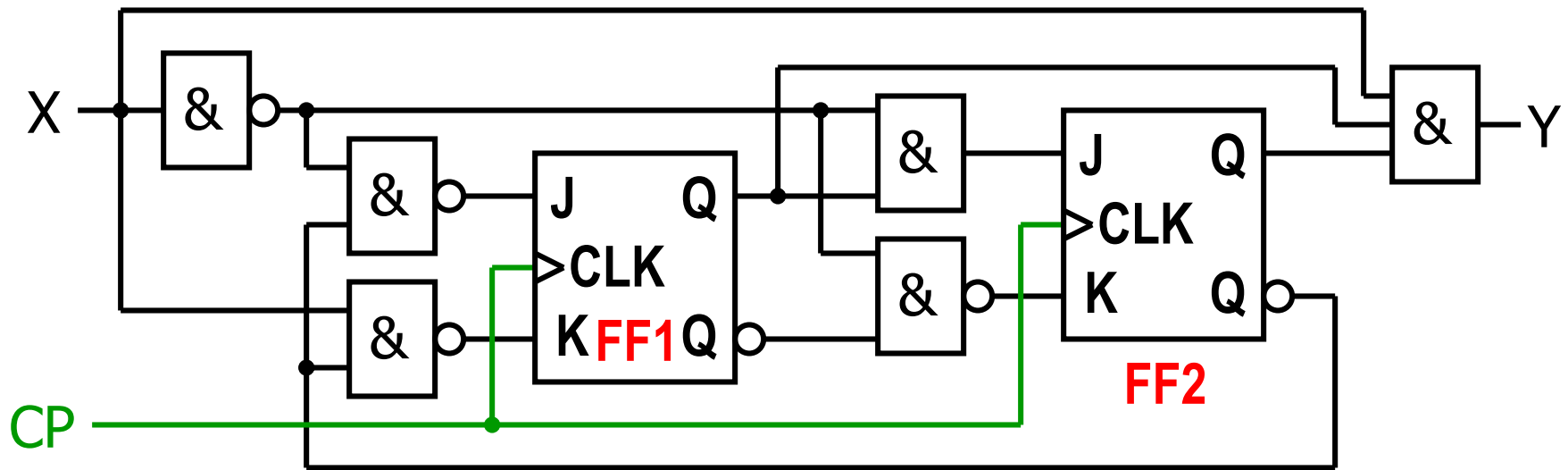
<i>EN</i>			
<i>S</i>	<i>0</i>	<i>1</i>	<i>MAXS</i>
A	A	B	0
B	B	C	0
C	C	D	0
D	D	A	1

Mealy机

<i>EN</i>		
<i>S</i>	<i>0</i>	<i>1</i>
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	D, 0
D	D, 0	A, 1



Example: Clocked Synchronous State Machine Analysis (J_K Flip-Flop)



1、由电路得到激励方程(输入方程)

$$\begin{cases} J_1 = (X' \cdot Q_2')' = X + Q_2 \\ K_1 = (X \cdot Q_2')' \end{cases} \quad \begin{cases} J_2 = X' \cdot Q_1 \\ K_2 = (X' \cdot Q_1')' \end{cases}$$

2、由电路得到输出方程 $Y = X \cdot Q_2 \cdot Q_1$



3、得到状态转换方程

J-K触发器特征方程为： $Q^* = J \cdot Q' + K' \cdot Q$, $CP \uparrow$

$$\begin{aligned} Q_1^* &= J_1 \cdot Q_1' + K_1' \cdot Q_1 = (X + Q_2) \cdot Q_1' + X' \cdot Q_2' \cdot Q_1 \\ &= Q_2 \cdot Q_1' + X \cdot Q_1' + X' \cdot Q_2' \cdot Q_1, \quad CP \uparrow \end{aligned}$$

$$Q_2^* = J_2 \cdot Q_2' + K_2' \cdot Q_2 = X' \cdot Q_1 \cdot Q_2' + X' \cdot Q_1' \cdot Q_2, \quad CP \uparrow$$

1、由电路得到激励方程（输入方程）

$$\begin{cases} J_1 = (X' \cdot Q_2')' = X + Q_2 \\ K_1 = (X \cdot Q_2')' \end{cases} \quad \begin{cases} J_2 = X' \cdot Q_1 \\ K_2 = (X' \cdot Q_1')' \end{cases}$$

2、由电路得到输出方程 $Y = X \cdot Q_2 \cdot Q_1$



4、由状态方程和输出方程列状态转换表

$$Q_1^* = Q_2 \cdot Q_1' + X \cdot Q_1' + X \cdot Q_2' \cdot Q_1, \text{ CP}\uparrow$$

$$Q_2^* = X' \cdot Q_2' \cdot Q_1 + X' \cdot Q_2 \cdot Q_1', \text{ CP}\uparrow$$

$$Y = X \cdot Q_2 \cdot Q_1$$

状态转换表

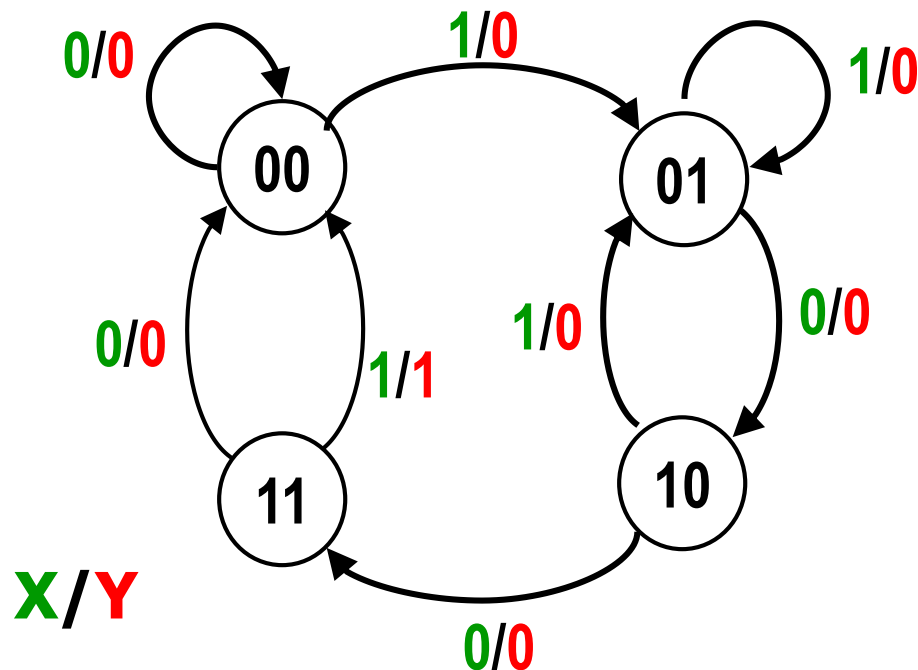
S	X	
	0	1
0 0	00, 0	01, 0
0 1	10, 0	01, 0
1 0	11, 0	01, 0
1 1	00, 0	00, 1
Q2Q1	Q2*Q1*, Y	

X	Q2 Q1	Q2* Q1*	Y
0	0 0	0 0	0
0	0 1	1 0	0
0	1 0	1 1	0
0	1 1	0 0	0
1	0 0	0 1	0
1	0 1	0 1	0
1	1 0	0 1	0
1	1 1	0 0	1



5、画状态图

S	X	
	0	1
0 0	00, 0	01, 0
0 1	10, 0	01, 0
1 0	11, 0	01, 0
1 1	00, 0	00, 1
Q2Q1	Q2*Q1*, Y	



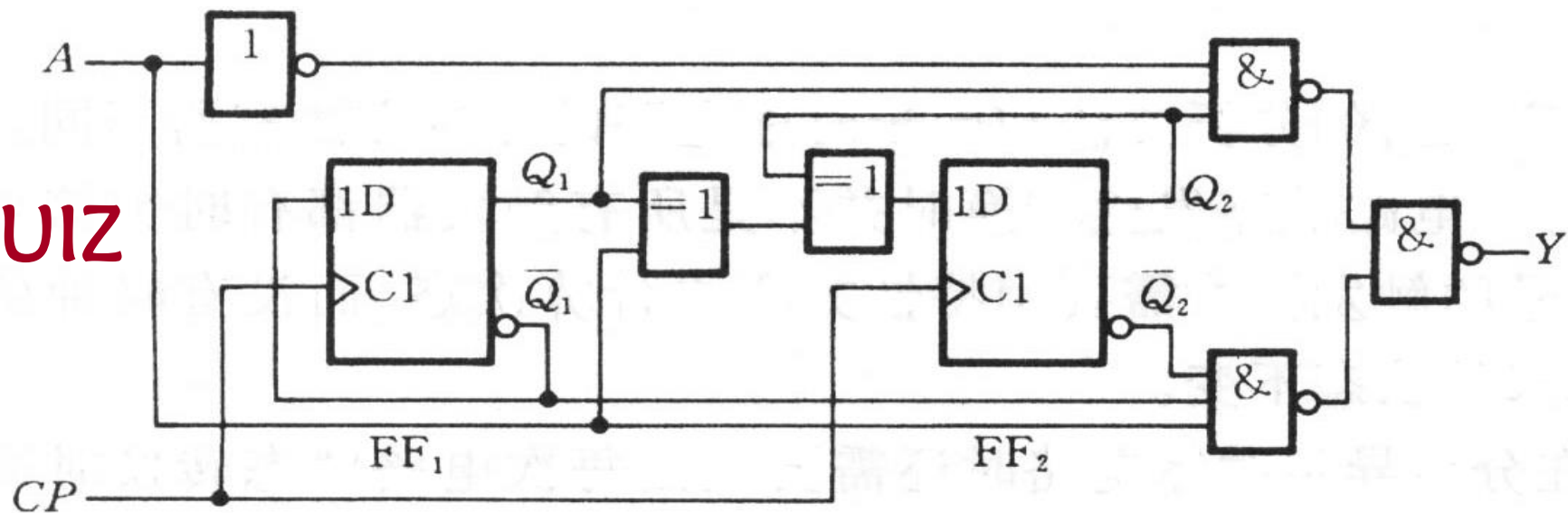
状态00：开始判断
状态01：输入一个1
状态10：连续输入10
状态11：连续输入100

逻辑功能：1001序列检测器
输入端连续输入1001时，输出1

X: 0100100111001100100...

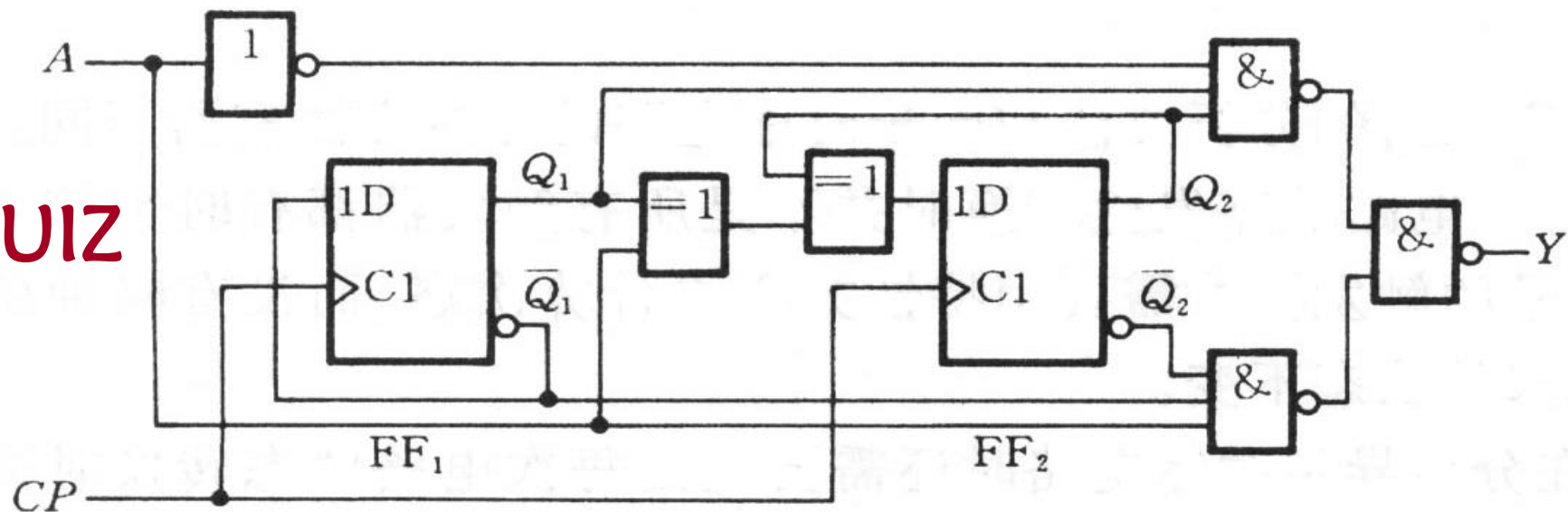
Y: 0000100000001000100...

QUIZ





QUIZ



1、列驱动方程、状态方程、输出方程

$$\begin{cases} D1 = Q_1' \\ D2 = A \oplus Q_1 \oplus Q_2 \end{cases} \quad Y = A' \cdot Q_2 \cdot Q_1 + A \cdot Q_2' \cdot Q_1'$$

$$\begin{cases} Q_1^* = D1 = Q_1', \text{ CP}\uparrow \\ Q_2^* = D2 = A \oplus Q_1 \oplus Q_2, \text{ CP}\uparrow \end{cases}$$

2、列状态转换表

$$Q_1^* = Q_1', \text{ CP}\uparrow$$

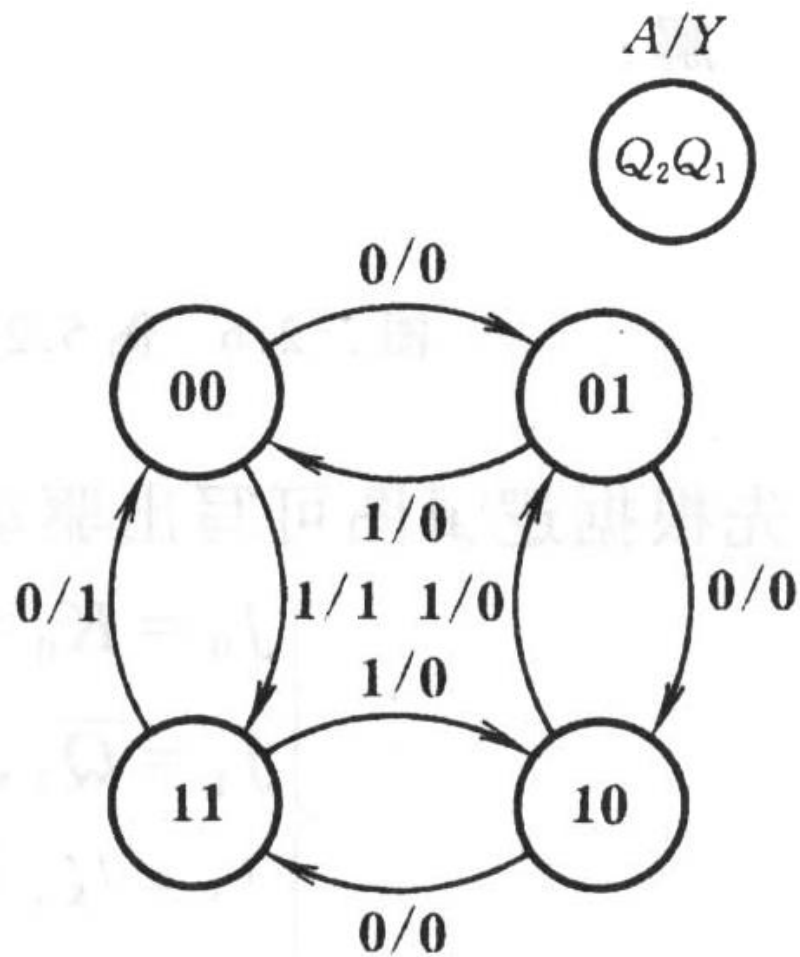
$$Q_2^* = A \oplus Q_1 \oplus Q_2, \text{ CP}\uparrow$$

$$Y = A' \cdot Q_2 \cdot Q_1 + A \cdot Q_2' \cdot Q_1'$$

A	Q ₂	Q ₁	Q ₂ *	Q ₁ *	Y
0	0	0	0	1	0
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	0	1	0
1	1	1	1	0	0

S	A	
	0	1
0 0	01, 0	11, 1
0 1	10, 0	00, 0
1 0	11, 0	01, 0
1 1	00, 1	10, 0
Q2Q1	Q2*Q1*, Y	

3、画状态转换图



功能描述：可逆计数器
A=0加法，**A=1**减法

S	A	
	0	1
0 0	01, 0	11, 1
0 1	10, 0	00, 0
1 0	11, 0	01, 0
1 1	00, 1	10, 0
Q2Q1	Q2*Q1*, Y	

*异步电路分析



*异步电路分析

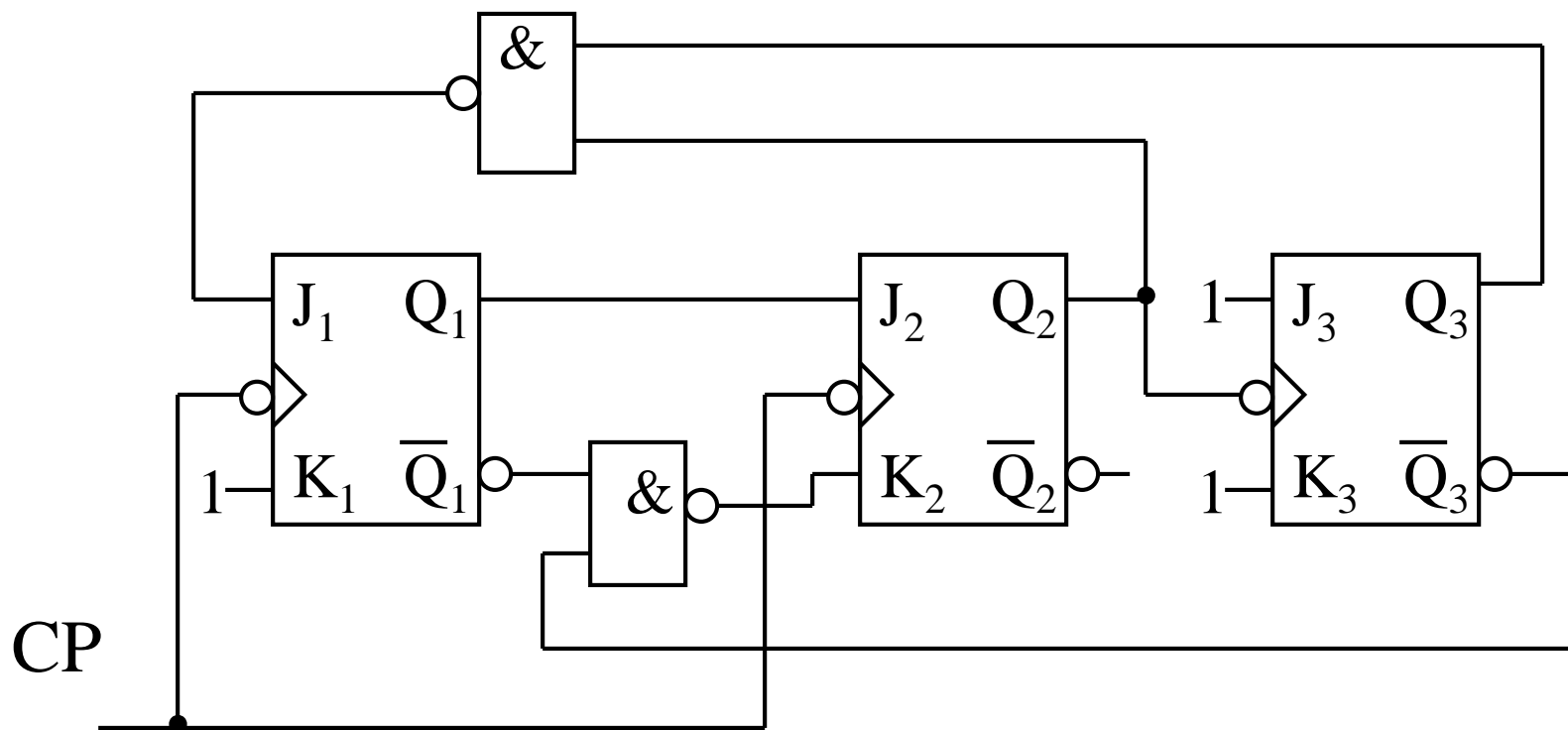
1) 时钟方程

$CP_2 = CP_1 = CP$, 外部时钟

$CP_3 = Q_2^n$, $Q_2 \downarrow$

该电路为异步时序电路

Moore型

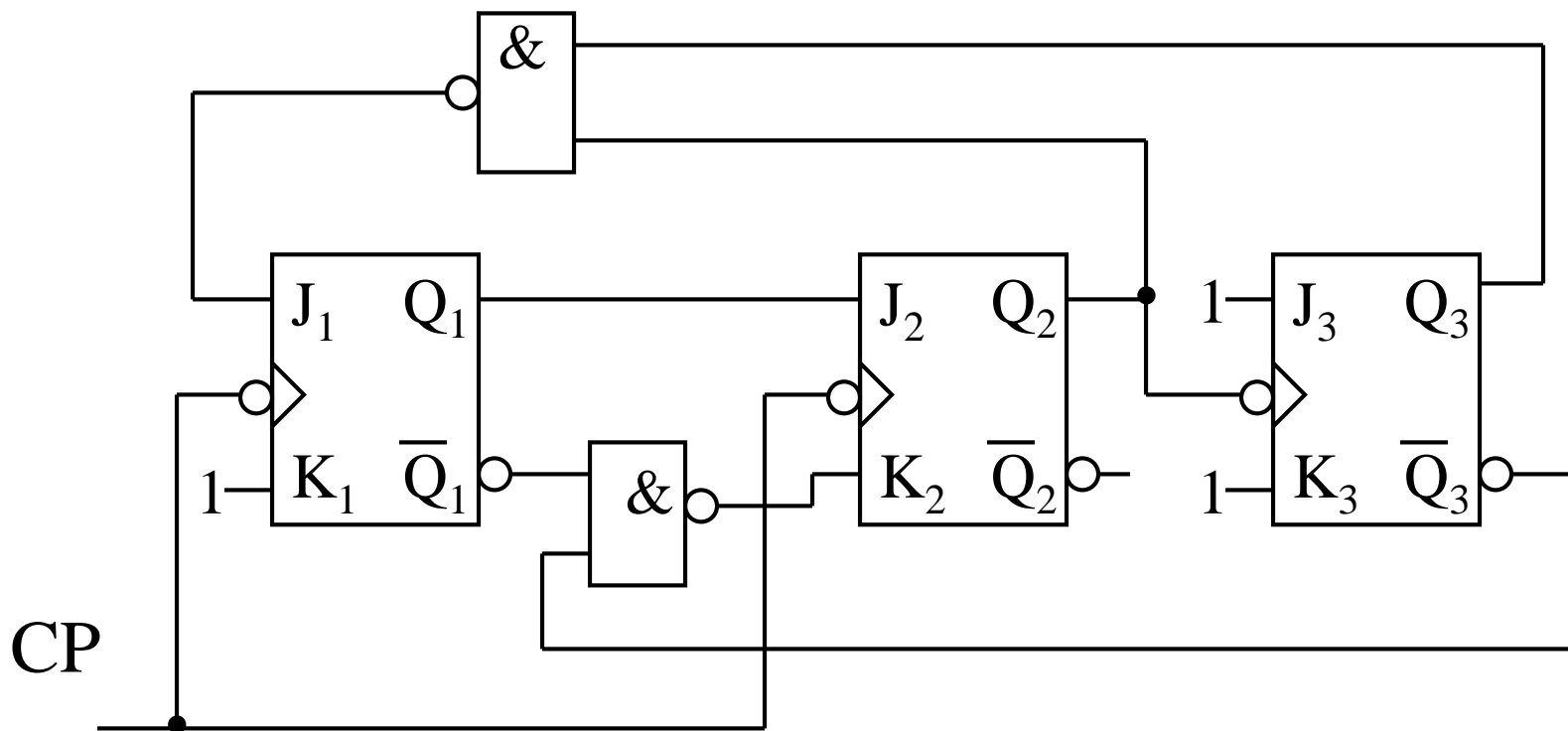


2) 驱动方程 (激励方程)

$$\begin{cases} \mathbf{J}_1 = \mathbf{Q}_2^n \mathbf{Q}_3^n \\ \mathbf{K}_1 = 1 \end{cases}$$

$$\begin{cases} \mathbf{J}_2 = \mathbf{Q}_1^n \\ \mathbf{K}_2 = \overline{\mathbf{Q}_1^n} \overline{\mathbf{Q}_3^n} \end{cases}$$

$$\begin{cases} \mathbf{J}_3 = \mathbf{1} \\ \mathbf{K}_3 = \mathbf{1} \end{cases}$$



3) 状态方程

$$\text{由 } Q^{n+1} = J\overline{Q}^n + \overline{K}Q^n, CP \downarrow$$

$$\text{得: } Q_1^{n+1} = \overline{Q_2^n Q_3^n} \overline{Q_1^n}, CP \downarrow$$

$$Q_2^{n+1} = Q_1^n \overline{Q_2^n} + \overline{Q_1^n} \overline{Q_3^n} Q_2^n, CP \downarrow$$

$$Q_3^{n+1} = \overline{Q_3^n}, Q_2 \downarrow$$

$$\left\{ \begin{array}{l} J_1 = \overline{Q_2^n Q_3^n} \\ K_1 = 1 \end{array} \right. \quad \left\{ \begin{array}{l} J_2 = Q_1^n \\ K_2 = \overline{Q_1^n} \overline{Q_3^n} \end{array} \right. \quad \left\{ \begin{array}{l} J_3 = 1 \\ K_3 = 1 \end{array} \right.$$



4) 状态表

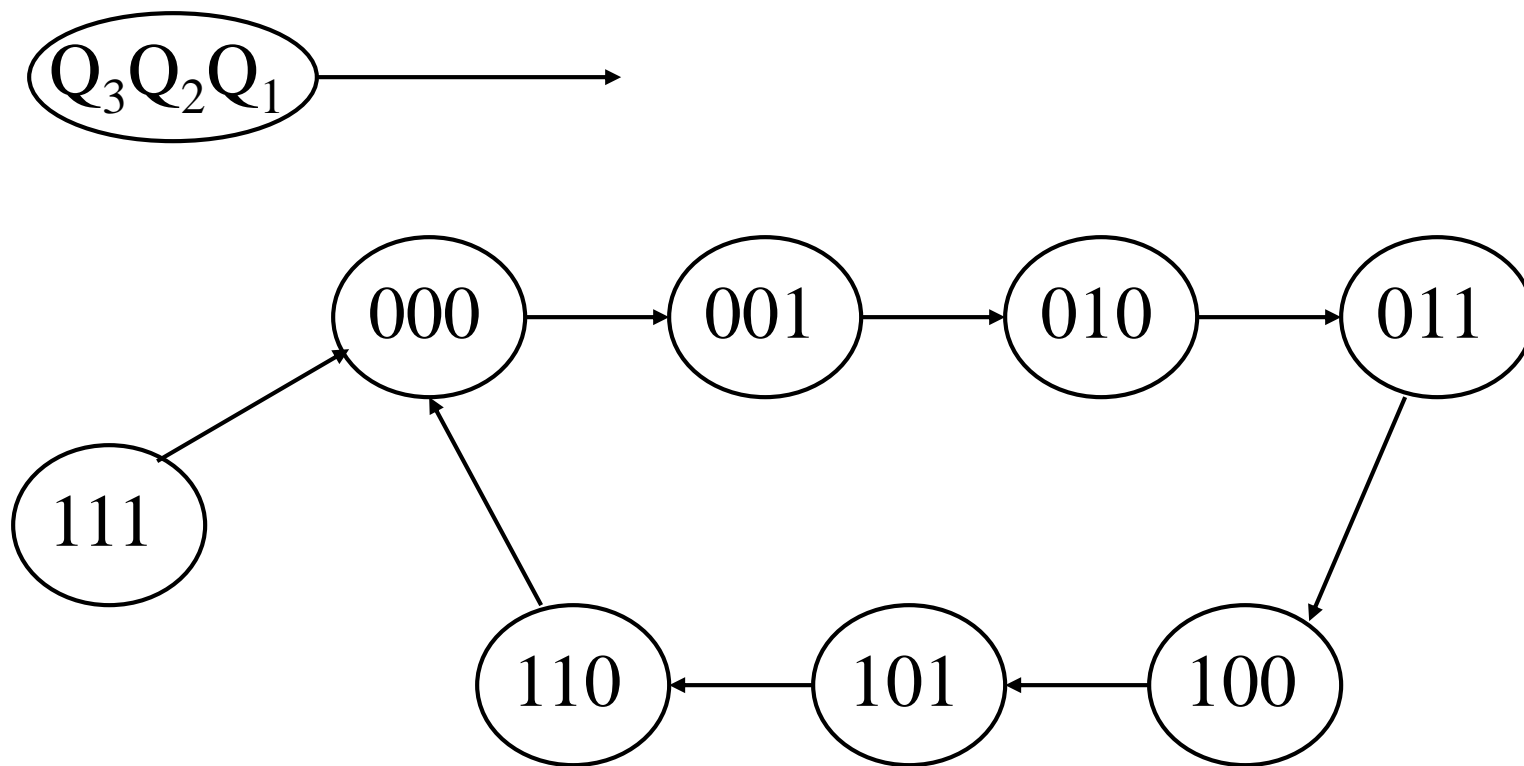
$$Q_3^{n+1} = \overline{Q_3}^n, Q_2 \downarrow$$

Q_3^n	Q_2^n	Q_1^n	$CP_3 = Q_2(\neg \downarrow)$	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}
0	0	0	0 $\xrightarrow{\text{X}}$ 0	0	0	1
0	0	1	0 $\xrightarrow{\text{X}}$ 1	0	1	0
0	1	0	1 $\xrightarrow{\text{X}}$ 1	0	1	1
0	1	1	1 $\xrightarrow{\checkmark}$ 0	1	0	0
1	0	0	0 $\xrightarrow{\text{X}}$ 0	1	0	1
1	0	1	0 $\xrightarrow{\text{X}}$ 1	1	1	0
1	1	0	1 $\xrightarrow{\checkmark}$ 0	0	0	0
1	1	1	1 $\xrightarrow{\checkmark}$ 0	0	0	0

$$Q_1^{n+1} = \overline{Q_2^n} Q_3^n \overline{Q_1^n}$$

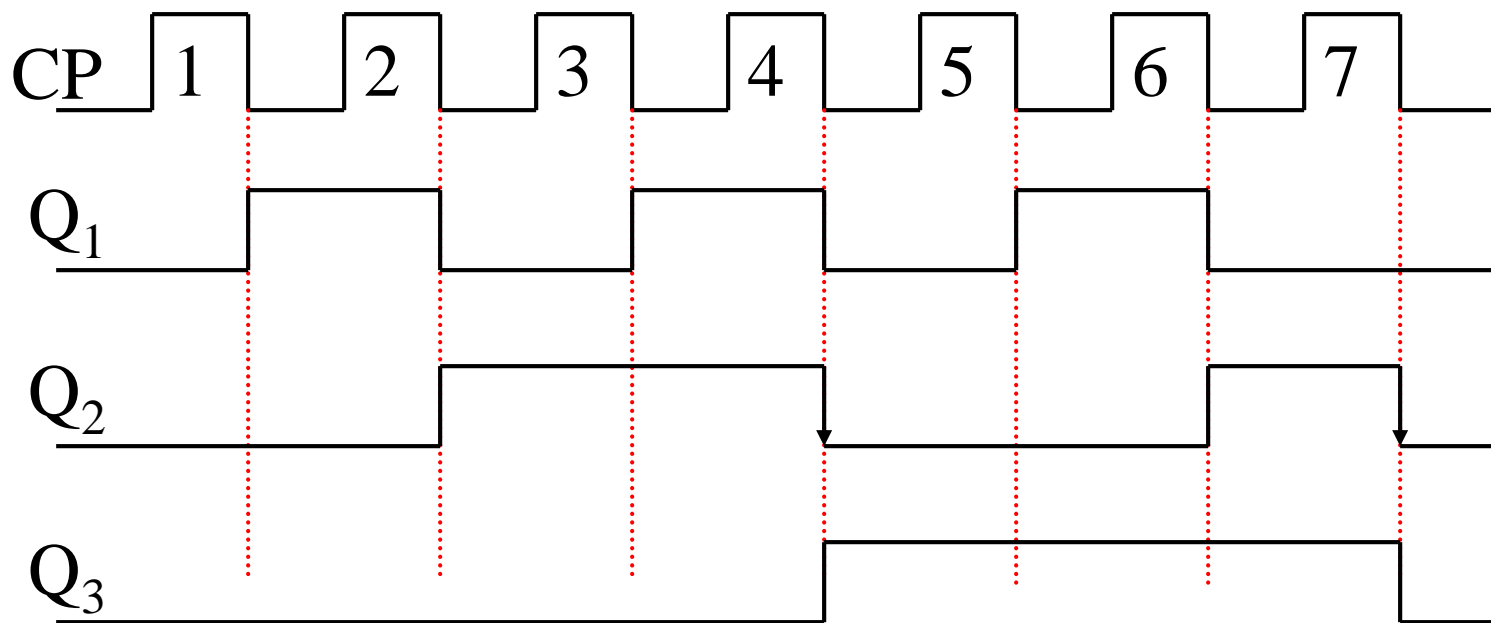
$$Q_2^{n+1} = Q_1^n \overline{Q_2^n} + \overline{Q_1^n} \overline{Q_3^n} Q_2^n$$

5) 状态图



功能：具有自启动能力的异步七进制计数器

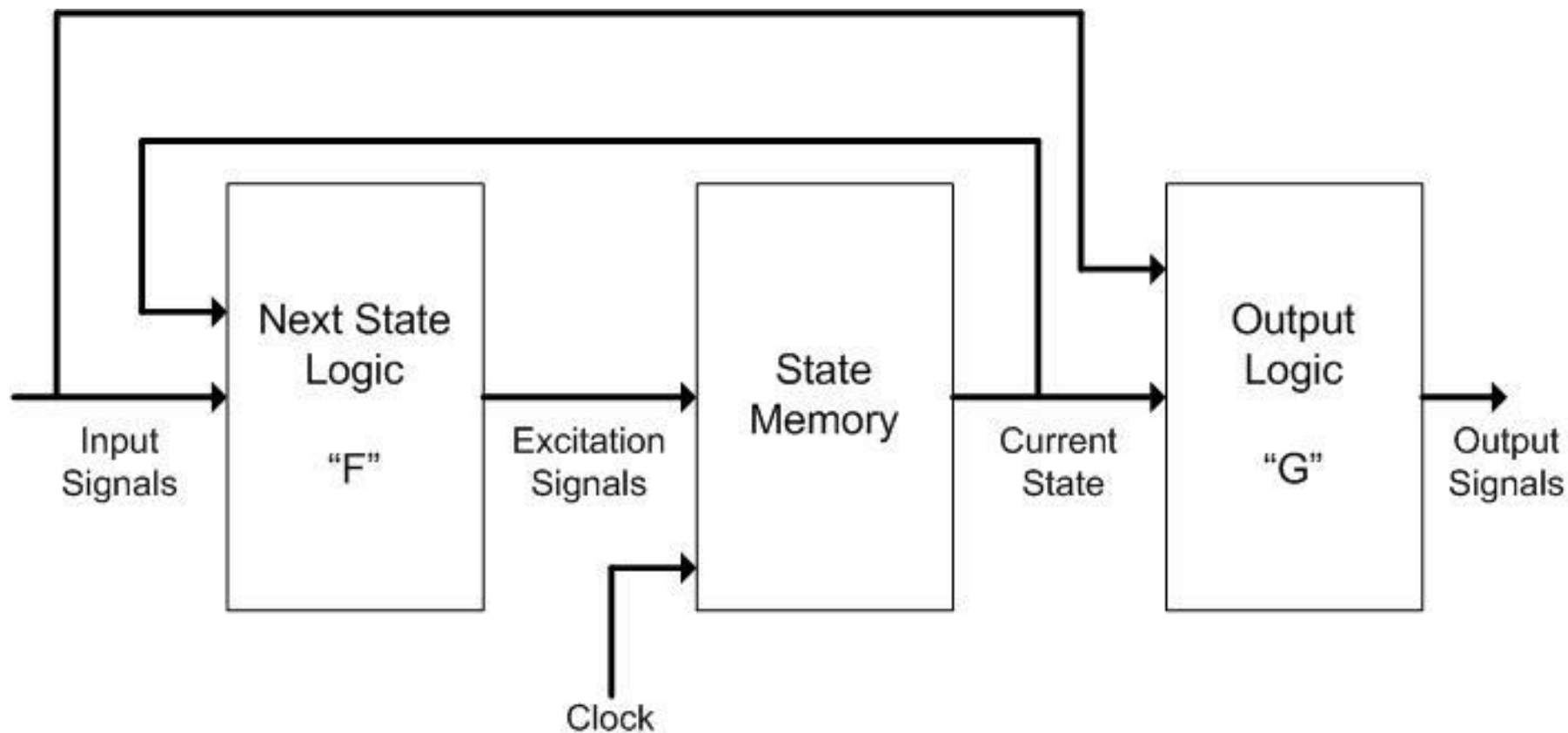
6) 波形图



功能：具有自启动能力的异步七进制计数器

Clocked Synchronous State-Machine Design

Clocked Synchronous State-Machine Design





基本步骤:

- State Diagram or State/output table
(根据命题构造状态图或状态/输出表)
- State minimization (状态化简 (状态最小化))
- State assignment (状态编码 (选择状态变量))
- Transition/output table
(建立转移/输出表, 得到状态转移方程和输出方程)
- Chose a flip-flop type (选择触发器作为状态存储器)
- Excitation equations (构造激励表, 得到激励方程)
- Draw a Logic Diagram (画逻辑电路图)



- **设计入门：两个简单的例子**
 - 设计一个3位二进制模8计数器
 - 设计一个110序列检测器
- **状态表设计举例**
 - 例一 (Reference 7.4) ;
 - 例二 (Book:P419) ;
 - 例三 (Book:P421)
- **状态图设计 (雷鸟车尾灯)**

Example 1 : 3-bit binary counter



Design a Modulo-8 3-bit Binary Counter

(设计一个模8的3位二进制计数器)

1、State diagram (逻辑抽象, 得到状态图 (表))

对时钟信号计数, 可不用输入

—— Moore机

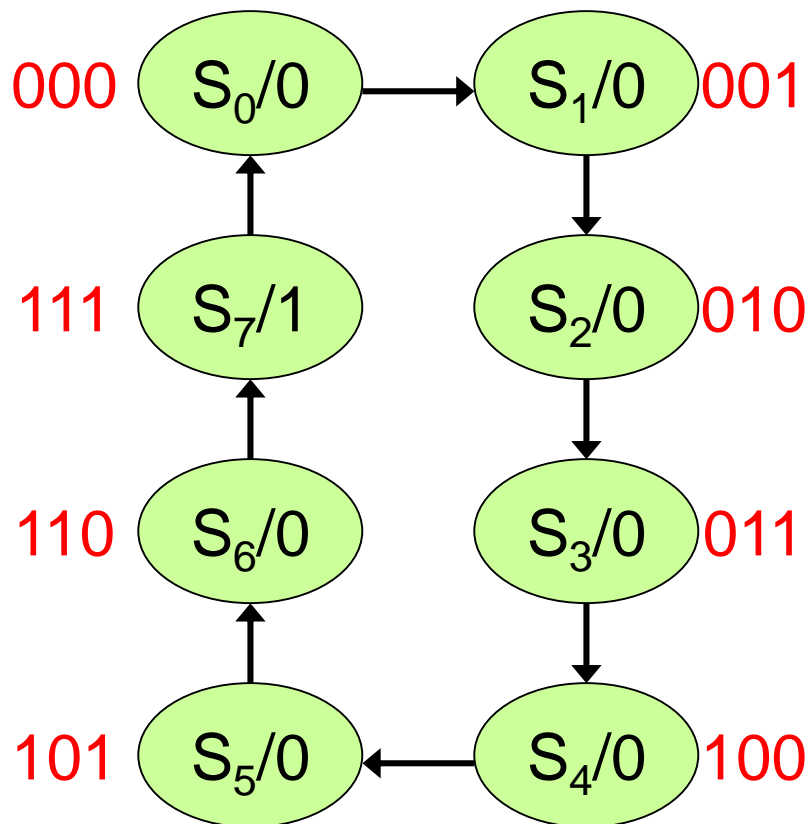
取进位信号为输出变量

需要8个有效状态

2、State Assignment (状态编码)

取自然二进制数 000~111作

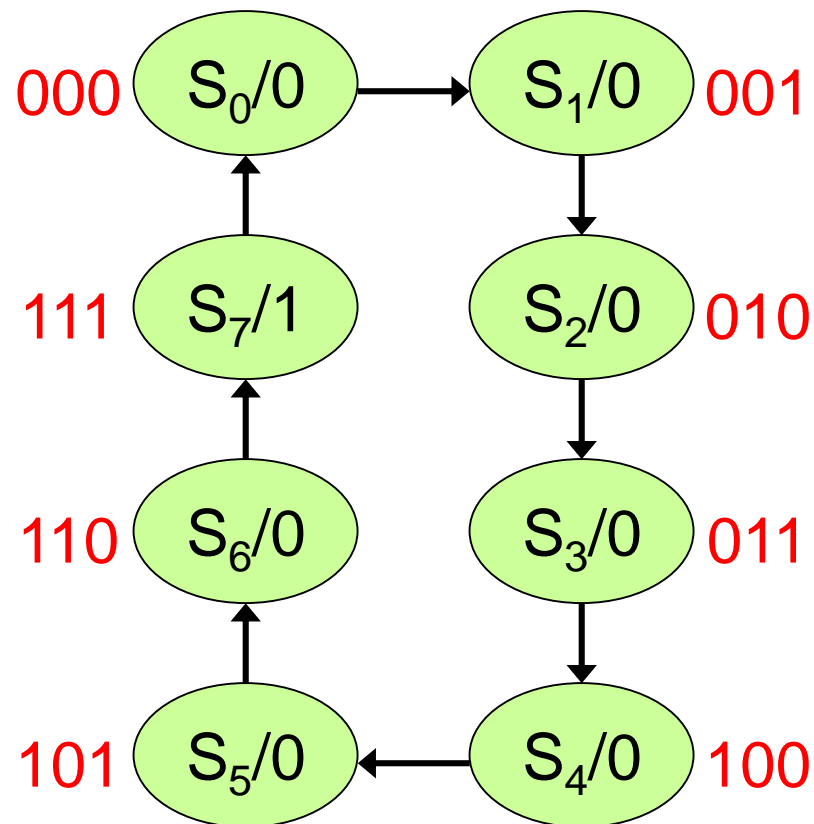
为 $S_0 \sim S_7$ 的编码



3、Create a Transition/Output table to obtain the State transition and output equations

(构造转移/输出表，求取 状态转移方程和 输出方程)

	Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*	C
S_0	0	0	0	0	0	1	0
S_1	0	0	1	0	1	0	0
S_2	0	1	0	0	1	1	0
S_3	0	1	1	1	0	0	0
S_4	1	0	0	1	0	1	0
S_5	1	0	1	1	1	0	0
S_6	1	1	0	1	1	1	0
S_7	1	1	1	0	0	0	1



3、Create a Transition/Output table to obtain the State transition and output equations

(构造转移/输出表，求取状态转移方程和输出方程)

Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*	C
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	1	0	1	0
1	0	1	1	1	0	0
1	1	0	1	1	1	0
1	1	1	0	0	0	1

$$Q_0^* = Q_0'$$

Q_0^*

$Q_1 Q_0$	00	01	11	10
0	1	0	0	1
1	1	0	0	1

3、 Create a Transition/Output table to obtain the State transition and output equations

(构造转移/输出表，求取状态转移方程和输出方程)

Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*	C
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	1	0	1	0
1	0	1	1	1	0	0
1	1	0	1	1	1	0
1	1	1	0	0	0	1

$$Q_0^* = Q_0'$$

$$Q_1^* = Q_1' \cdot Q_0 + Q_1 \cdot Q_0'$$

Q_1^*

$Q_1 Q_0$	00	01	11	10
0	0	1	0	1
1	0	1	0	1

3、 Create a Transition/Output table to obtain the State transition and output equations

(构造转移/输出表，求取状态转移方程和输出方程)

Q_2	Q_1	Q_0	$Q_2^* Q_1^* Q_0^*$	C
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	1 0 1	0
1	0	1	1 1 0	0
1	1	0	1 1 1	0
1	1	1	0 0 0	1

输出方程： $C = Q_2 \cdot Q_1 \cdot Q_0$

$$Q_0^* = Q_0'$$

$$Q_1^* = Q_1' \cdot Q_0 + Q_1 \cdot Q_0'$$

$$Q_2^* = Q_2' \cdot Q_1 \cdot Q_0 + Q_2 \cdot Q_1' + Q_2 \cdot Q_0'$$

Q_2^*

$Q_1 Q_0$	00	01	11	10
0	0	0	1	0
1	1	1	0	1



4、Choose a Flip-Flop Type, Obtain the Excitation Equations

(触发器选型，得到激励方程)

$$Q_0^* = Q_0' \xrightarrow{\text{翻转}} Q^* = T \cdot Q' + T' \cdot Q \quad (\text{T触发器})$$
$$T_0 = 1$$

$$Q_1^* = Q_1' \cdot Q_0 + Q_1 \cdot Q_0'$$
$$T_1 = Q_0$$

$$\begin{aligned} Q_2^* &= Q_2' \cdot Q_1 \cdot Q_0 + Q_2 \cdot Q_1' + Q_2 \cdot Q_0' \\ &= Q_2' \cdot Q_1 \cdot Q_0 + Q_2 \cdot (Q_1' + Q_0') \\ &= Q_2' \cdot Q_1 \cdot Q_0 + Q_2 \cdot (Q_1 \cdot Q_0)' \\ T_2 &= Q_1 \cdot Q_0 \end{aligned}$$

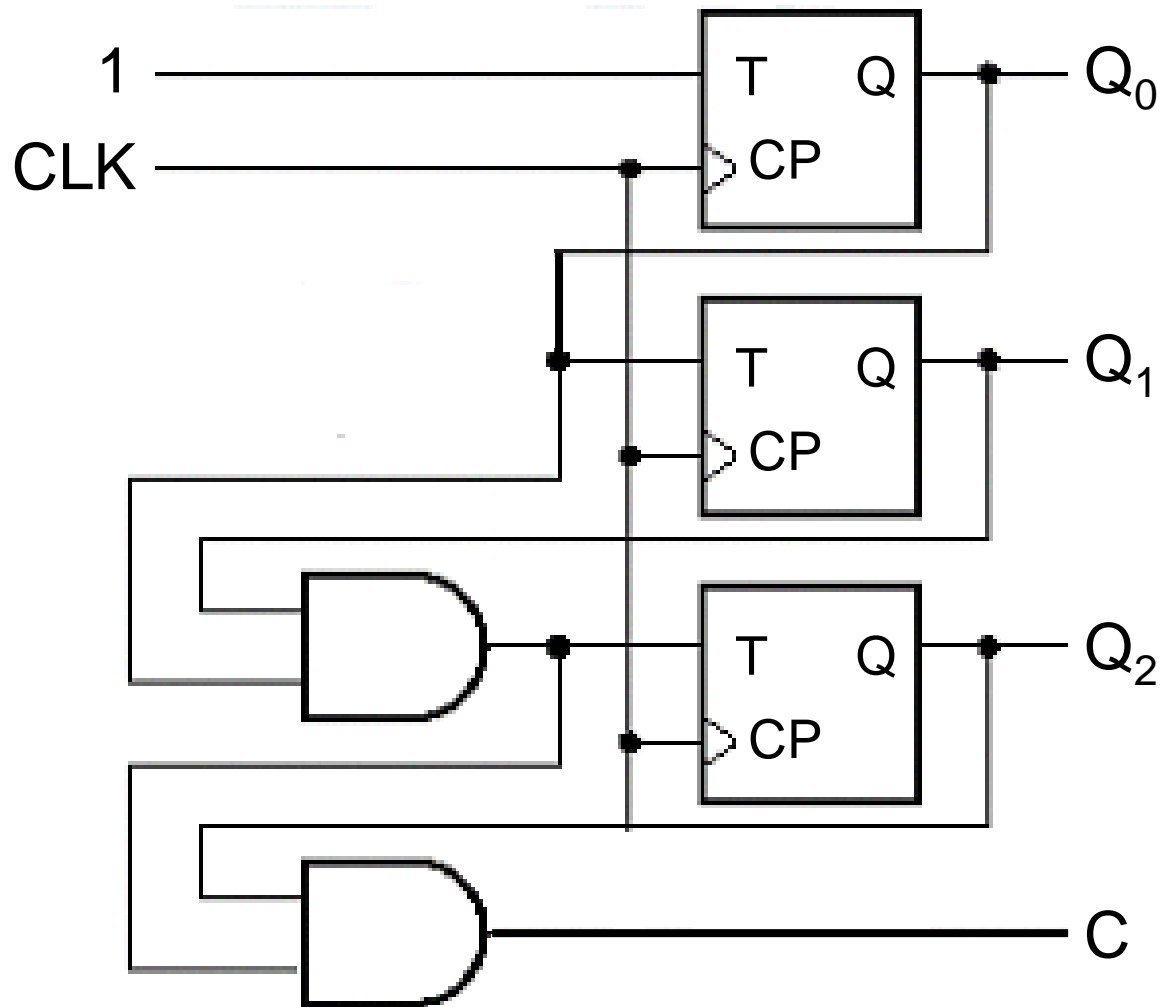
5、Draw a Logic Diagram (画逻辑电路图)

$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_1 \cdot Q_0$$

$$C = Q_2 \cdot Q_1 \cdot Q_0$$



Example 2 : Serial Data Detector of 110

Design a Serial Data Detector of 110

(设计一个110串行数据检测器)

电路检测到输入连续出现110时，输出为1

用A表示输入数据；用Z表示检测结果。

1、State/Output table

状态S	A	
	0	1
开始，等待第一个1 → STA	STA/0	A1/0
A上捕获一个1 → A1	STA/0	A11/0
A上连续捕获11 → A11	OK/1	A11/0
A上连续捕获110 → OK	STA/0	A1/0
	S*/Z	

Mealy机



识别等效状态 如果两个状态

- 对于所有输入组合产生相同的输出
- 对于每种输入组合具有相同或等效的下一状态

2、State Minimization

3、State Assignment

状态S		A	
		0	1
00	STA	STA/0	A1/0
01	A1	STA/0	A11/0
10	A11	STA/1	A11/0
		S*/Z	

4、State Equations and Output Equations

状态S		A	
		0	1
00	STA	STA/0	A1/0
01	A1	STA/0	A11/0
10	A11	STA/1	A11/0

S^*/Z

$Q_1^*Q_0^*/Z \ (S^*/Z)$

Q_1Q_0 A		00	01	11	10
		0	1	1	0
0		00/0	00/0	dd/d	00/1
1		01/0	10/0	dd/d	10/0

Q_1Q_0		Z			
A		00	01	11	10
	0	0	0	d	1
	1	0	0	d	0

$$Z = A' \cdot Q_1$$

4、State Equations and Output Equations

Q_1Q_0		A			
		00	01	11	10
	0	00/0	00/0	dd/d	00/1
	1	01/0	10/0	dd/d	10/0

Q_1Q_0		Q_1^*			
A		00	01	11	10
	0	0	0	d	0
	1	0	1	d	1

$$Q_1^* = A \cdot Q_1 + A \cdot Q_0$$

Q_1Q_0		Q_0^*			
A		00	01	11	10
	0	0	0	d	0
	1	1	0	d	0

$$Q_0^* = A \cdot Q_1' \cdot Q_0'$$



5、Excitation Equations

$$Q_0^* = A \cdot Q_1' \cdot Q_0' = (A \cdot Q_1') \cdot Q_0' + 1' \cdot Q_0$$

$$\begin{aligned} Q_1^* &= A \cdot Q_1 + A \cdot Q_0 = A \cdot Q_1 + A \cdot Q_0 \cdot (Q_1 + Q_1') \\ &= A \cdot Q_1 + \underline{A \cdot Q_0 \cdot Q_1} + A \cdot Q_0 \cdot Q_1' \\ &= A \cdot Q_1 + A \cdot Q_0 \cdot Q_1' \end{aligned}$$

选择D触发器 $Q^* = D$

$$D_0 = A \cdot Q_1' \cdot Q_0'$$

$$D_1 = A \cdot Q_1 + A \cdot Q_0$$

选择J-K触发器

$$Q^* = J \cdot Q' + K' \cdot Q$$

$$\begin{cases} J_0 = A \cdot Q_1' \\ K_0 = 1 \end{cases} \quad \begin{cases} J_1 = A \cdot Q_0 \\ K_1 = A' \end{cases}$$



6、Check the Circuit Self-Startup (检查电路的自启动性)

$$Q_0^* = A \cdot Q_1' \cdot Q_0'$$

$$Q_1^* = A \cdot Q_1 + A \cdot Q_0$$

$$Z = A' \cdot Q_1$$

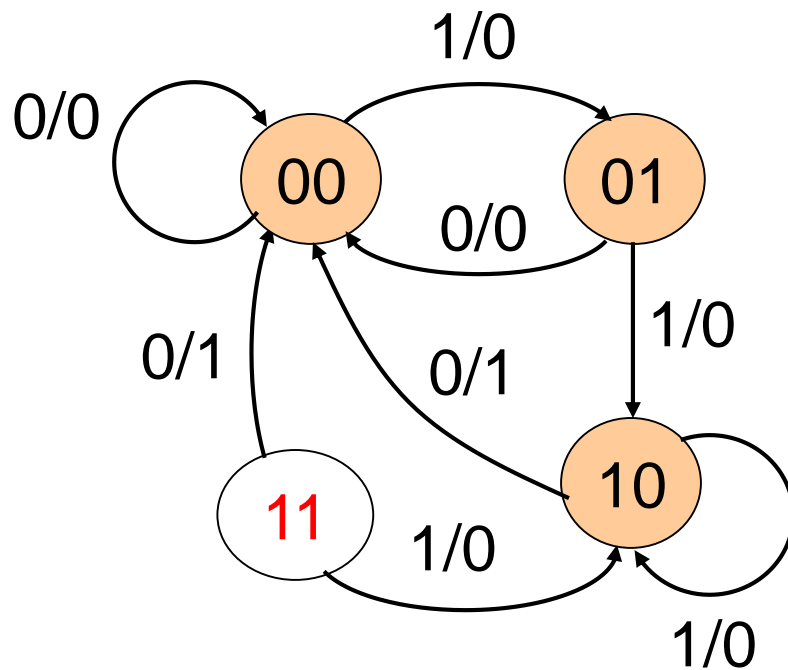
当电路进入无效状态11后,

A=0时, 下一状态为 00

A=1时, 下一状态为 10

该电路是自启动的

7、Draw a Logic Diagram (画逻辑电路图) (Quiz)



全状态图

如果设计成Moore型？

电路检测到输入连续出现110时，输出为1

用A表示输入数据；用Z表示检测结果。

Moore机

1、得到状态转换表

开始，等待第一个1 → STA

A上捕获一个1 → A1

A上连续捕获11 → A11

A上连续捕获110 → OK

状态S	A		Z
	0	1	
开始，等待第一个1 → STA	STA	A1	0
A上捕获一个1 → A1	STA	A11	0
A上连续捕获11 → A11	OK	A11	0
A上连续捕获110 → OK	STA	A1	1
	S*		

对比(Merely):

状态S	A	
	0	1
STA	STA/0	A1/0
A1	STA/0	A11/0
A11	STA/1	A11/0
	S*/Z	

➤思考:

同样的状态机电路设计, Merely型电路需要的触发器数可能多些, 还是 Moore型电路需要的触发器数可能多些?

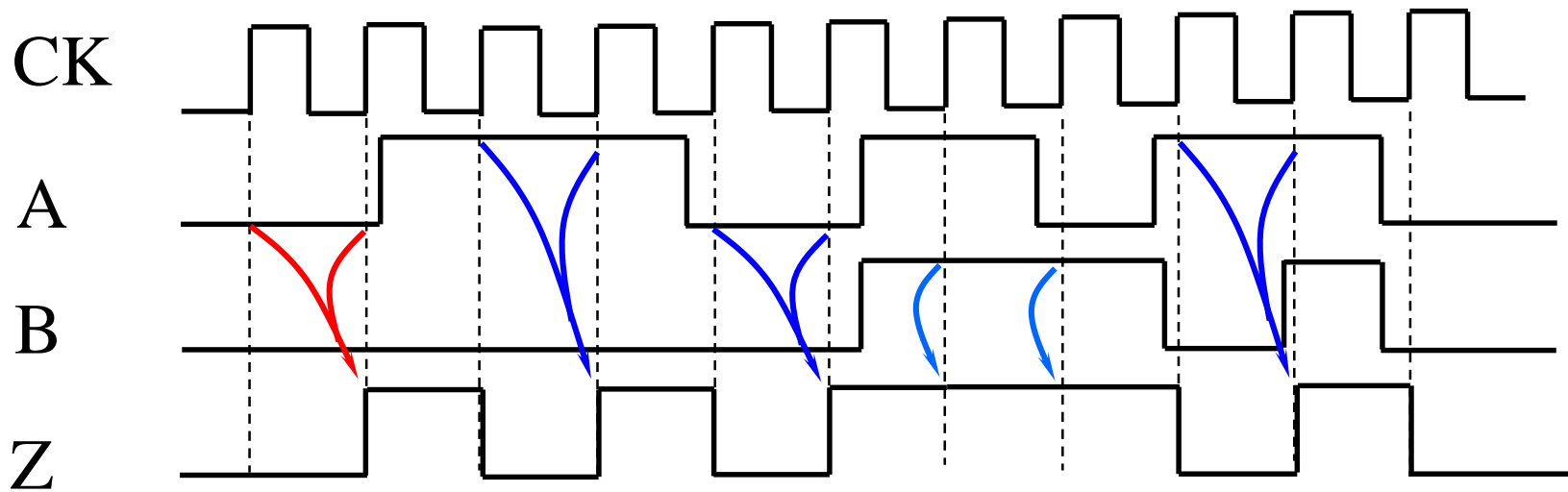
Example 3



State Table Design Example 1

- Design a clocked synchronous state machine with two inputs, A and B, and a single output Z that is 1 if:
 - A had the same value at each of the two previous clock ticks
(前两个时钟A保持相同的值) **or**
 - B has been 1 since the last time that the first condition was true.
(上一次当条件一满足时, B一直为1)

State Table Design Example 1



1、构造状态转换表

S	AB				Z
	00	01	11	10	
初始状态 → INIT	A0	A0	A1	A1	0
A上捕获一个0 → A0	OK0	OK0	A1	A1	0
A上捕获一个1 → A1	A0	A0	OK1	OK1	0
A上连续两个0 → OK0	OK0	OK0	OK1B	A1	1
A上连续两个1 → OK1	A0	OK0B	OK1	OK1	1
因B而OK，A为1 → OK1B	A0	OK0B	OK1	OK1	1
因B而OK，A为0 → OK0B	OK0	OK0	OK1B	A1	1
状态含义	S*				

1、构造状态转换表

2、状态最小化

2、状态最小化

S	AB				Z	
	00	01	11	10		
初始状态 → INIT	A0	A0	A1	A1	0	
A上捕获一个0 → A0	OK0	OK0	A1	A1	0	
A上捕获一个1 → A1	A0	A0	OK1	OK1	0	
OK, A值为0 → OK0	OK0	OK0	OK1	A1	1	
OK, A值为1 → OK1	OK1	A0	OK0	OK1	OK1	1

状态含义

S*

1、构造状态转换表

2、状态最小化

3、状态编码

Assignment				
State Name	最简单的 Q1-Q3	分解的 Q1-Q3	单热点的 Q1-Q5	准单热点的 Q1-Q4
INIT	000	000	00001	0000
A0	001	100	00010	0001
A1	010	101	00100	0010
OK0	011	110	01000	0100
OK1	100	111	10000	1000

真的需要一一尝试吗？ 合理的状态赋值



*合理的状态赋值

- 选择复位时容易进入的状态作为初始状态
- 使每次转移时要发生改变的状态变量数最小化
- 使一组相关状态中不变化的状态变量数最大化
- 发现和利用问题描述中的**对称性**
- 相对于状态机的输入效果或者输出特性，将状态变量组**分解**为有明确含义的位或字段
- 可以使用多余最小值的状态变量数（便于分解）
- 未用状态的考虑

4、根据状态表和状态编码构造转移/输出表

5个输入变量：

A,B,Q1,Q2,Q3

4个输出变量：

Z,D1,D2,D3

使用D触发器

转移/激励表

D1 D2 D3

Q1Q2Q3	AB				Z
	00	01	11	10	
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	D1 D2 D3				

5、触发器选型，得到激励方程和输出方程

D2

AB

Q2Q3 \ AB	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

Q1=0

最小冒险，未用状态→初始状态

Q2Q3 \ AB	00	01	11	10
00	1	1	0	0
01	0	0	1	1
11	0	1	1	1
10	1	1	1	0

Q1=1

Q1Q2Q3	AB				Z
	00	01	11	10	
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	D1 D2 D3				

输出方程： $Z = Q1 \cdot Q2$

D2

AB

Q2Q3

00 01 11 10

00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

Q1=0

最小冒险，未用状态→初始状态

$$D2 = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B$$

D2

AB

Q2Q3

00 01 11 10

00	0	0	0	0
01	d	d	d	d
11	d	d	d	d
10	d	d	d	d

Q1=0

最小成本，未用状态作为无关项

$$D2 = Q1 \cdot Q3' \cdot A' + Q3 \cdot A + Q2 \cdot B$$

AB

Q2Q3

00 01 11 10

00	1	1	0	0
01	0	0	1	1
11	0	1	1	1
10	1	1	1	0

Q1=1

$$D1 = Q2' \cdot Q3' + Q1$$

$$D2 = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B$$

D1

		A B		A	
		00	01	11	10
Q2 Q3	00	1	1	1	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0
		B			
		Q3			

Q1=0

		A B		A	
		00	01	11	10
Q2 Q3	00	1	1	1	1
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	1	1
		B			
		Q3			

Q1=1

思考：最小成本法 $D1 = ?$

激励方程

$$\begin{cases} D1 = Q2' \cdot Q3' + Q1 \\ D2 = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B \\ D3 = Q2' \cdot Q3' \cdot A + Q1 \cdot A \end{cases}$$

D3

		A B		A	
		00	01	11	10
Q2	Q3	00	01	11	10
	00	0	0	1	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

Q1=0

		A B		A	
		00	01	11	10
Q2	Q3	00	01	11	10
	00	0	0	1	1
	01	0	0	1	1
	11	0	0	1	1
	10	0	0	1	1

Q1=1

思考：最小成本法 $D3 = ?$



$$\text{激励方程} \quad \begin{cases} D1 = Q2' \cdot Q3' + Q1 \\ D2 = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B \\ D3 = Q2' \cdot Q3' \cdot A + Q1 \cdot A \end{cases}$$

$$\text{输出方程: } Z = Q1 \cdot Q2$$

6、画逻辑电路图 (略)

说明:

- 最小冒险法

所有未用状态 \rightarrow “安全”状态

- 最小成本法

所有未用状态的下一状态作为无关项

电路的激励方程简单, 不够安全

Minimal cost design

The unused states go to the “don’t cares”

$$Q1^* = 1$$

$$Q2^* = Q3.A + Q2.B + Q1.Q3'.A'$$

$$Q3^* = A$$

$$Z = Q2$$

$$D1 = 1$$

$$D2 = Q3.A + Q2.B + Q1.Q3'.A'$$

$$D3 = A$$

Q1Q2Q3 \ AB			AB			
			00	01	11	10
000	1 0 0	1 0 0	1 0 1	1 0 1		
001	ddd	ddd	ddd	ddd		
011	ddd	ddd	ddd	ddd		
010	ddd	ddd	ddd	ddd		
110	1 1 0	1 1 0	1 1 1	1 0 1		
111	1 0 0	1 1 0	1 1 1	1 1 1		
101	1 0 0	1 0 0	1 1 1	1 1 1		
100	1 1 0	1 1 0	1 0 1	1 0 1		
			Q1*Q2*Q3*			



Design with J-K Flip-Flops

(用J-K触发器设计)

● 方法一

利用**状态方程**和**触发器特征方程**得到**激励方程**

● 方法二

利用**状态转移表**和**激励表**得到**激励方程**

J K		Q
0 0		保持
0 1		清0
1 0		置1
1 1		翻转

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



Design with J-K Flip-Flops

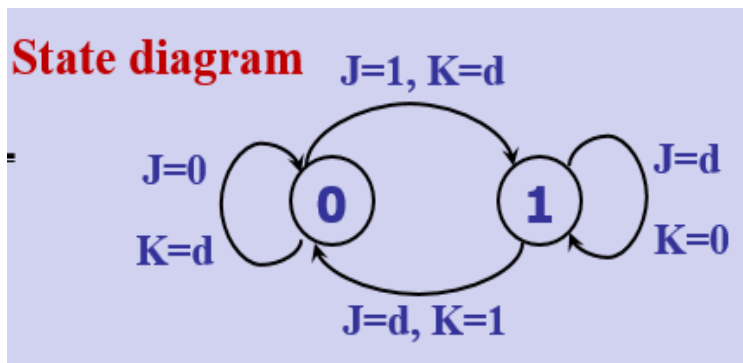
(用J-K触发器设计)

● 方法一

利用状态方程和触发器特征方程得到激励方程

● 方法二

利用状态转移表和激励表得到激励方程



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

Design with State Equations and Characteristic Equations

J-K触发器特征方程： $Q^* = J \cdot Q' + K' \cdot Q$

$$\text{状态方程} \begin{cases} Q1^* = Q2' \cdot Q3' + Q1 \\ Q2^* = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B \\ Q3^* = Q2' \cdot Q3' \cdot A + Q1 \cdot A \end{cases}$$

$$\begin{aligned} J1 &= Q2' \cdot Q3' \\ K1 &= 0 \end{aligned}$$

$$\begin{aligned} Q1^* &= Q2' \cdot Q3' + Q1 \\ &= Q2' \cdot Q3' \cdot (Q1' + Q1) + Q1 \\ &= Q2' \cdot Q3' \cdot Q1' + Q2' \cdot Q3' \cdot Q1 + Q1 \\ &= Q2' \cdot Q3' \cdot Q1' + Q1 \end{aligned}$$

Design with State Equations and Characteristic Equations

J-K触发器特征方程: $Q^* = J \cdot Q' + K' \cdot Q$

$$\text{状态方程} \begin{cases} Q1^* = Q2' \cdot Q3' + Q1 \\ Q2^* = Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B \\ Q3^* = Q2' \cdot Q3' \cdot A + Q1 \cdot A \end{cases}$$

$$\begin{aligned} J1 &= Q2' \cdot Q3' \\ K1 &= 0 \end{aligned}$$

$$\begin{aligned} Q3^* &= Q2' \cdot Q3' \cdot A + Q1 \cdot A \\ &= Q2' \cdot Q3' \cdot A + Q1 \cdot A \cdot (Q3' + Q3) \\ &= (Q2' \cdot A + Q1 \cdot A) \cdot Q3' + Q1 \cdot A \cdot Q3 \end{aligned}$$

$$\begin{aligned} J3 &= Q2' \cdot A + Q1 \cdot A \\ K3 &= Q1' + A' \end{aligned}$$

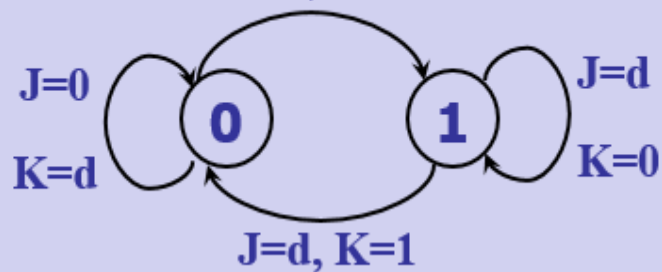
$$\begin{aligned}Q2^* &= Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot Q2 \cdot B \\&= (Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A) \cdot (Q2' + Q2) + Q1 \cdot Q2 \cdot B \\&= (Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A) \cdot Q2' \\&\quad + (Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot B) \cdot Q2\end{aligned}$$

$$\begin{aligned}K2 &= (Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A + Q1 \cdot B)' \\&= (Q1' + Q3 + A) \cdot (Q1' + Q3' + A') \cdot (Q1' + B') \\&= Q1' + Q3' \cdot A \cdot B' + Q3 \cdot A' \cdot B'\end{aligned}$$

$$\begin{aligned}J1 &= Q2' \cdot Q3' \\K1 &= 0\end{aligned}$$

$$\begin{aligned}J3 &= Q2' \cdot A + Q1 \cdot A \\K3 &= Q1 \cdot A\end{aligned}$$

$$\begin{aligned}J2 &= Q1 \cdot Q3' \cdot A' + Q1 \cdot Q3 \cdot A \\K2 &= Q1' + Q3' \cdot A \cdot B' + Q3 \cdot A' \cdot B'\end{aligned}$$



the Excitation table

Q1Q2Q3	AB				Z
	00	01	1	10	
000	100 1d,0d,0d	100 1d,0d,0d	101 1d,0d,1d	101 1d,0d,1d	0
100	110 d0,1d,0d	110 d0,1d,0d	101 d0,0d,1d	101 d0,0d,1d	0
101	100 d0,0d,d1	100 d0,0d,d1	111 d0,1d,d0	111 d0,1d,d0	0
110	110 d0,d0,0d	110 d0,d0,0d	111 d0,d0,1d	101 d0,d1,1d	1
111	100 d0,d1,d1	110 d0,d0,d1	111 d0,d0,d0	111 d0,d0,d0	1
J1K1 , J2K2 , J3K3					

利用卡诺图化简

Q1Q2Q3	AB				Z
	00	01	11	10	
000	1d,0d,0d	1d,0d,0d	1d,0d,1d	1d,0d,1d	0
100	d0,1d,0d	d0,1d,0d	d0,0d,1d	d0,0d,1d	0
101	d0,0d,d1	d0,0d,d1	d0,1d,d0	d0,1d,d0	0
110	d0,d0,0d	d0,d0,0d	d0,d0,1d	d0,d1,1d	1
111	d0,d1,d1	d0,d0,d1	d0,d0,d0	d0,d0,d0	1
	J1K1 , J2K2 , J3K3				

Example 4 : Tail Lights controller of Ford Thunderbird

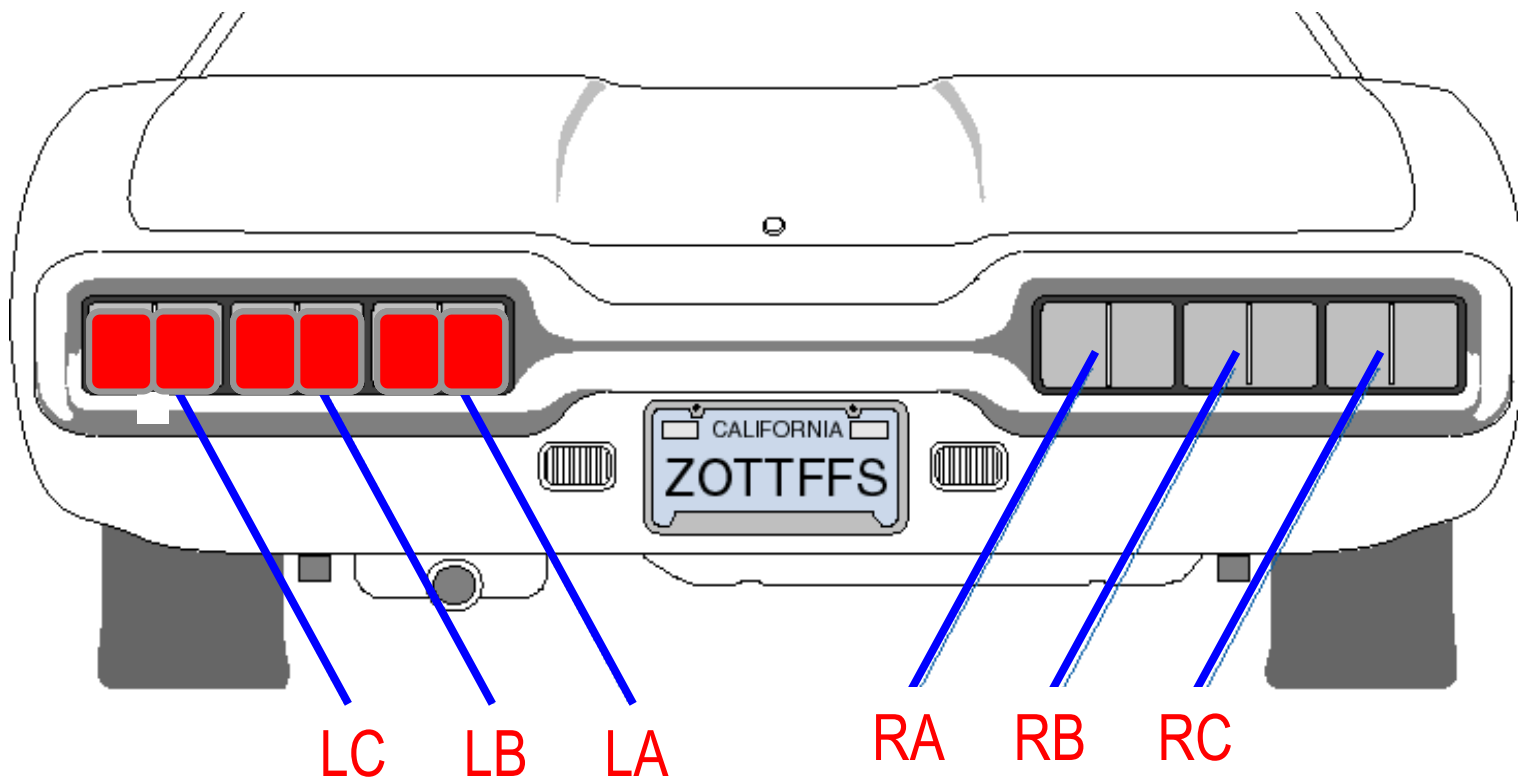
Control the Tail Lights of Ford Thunderbird

(福特雷鸟车尾灯控制)

输入：左转L、右转R、应急闪烁H (hazard)、时钟

输出：控制6个灯亮或灭

—— 可以完全由状态控制



直接利用状态控制输出

IDLE：全灭

L1：左边1个灯亮

L2：左边2个灯亮

L3：左边3个灯亮

R1：右边1个灯亮

R2：右边2个灯亮

R3：右边3个灯亮

LR3：全亮

状态	输出					
	LC	LB	LA	RA	RB	RC
IDLE	0	0	0	0	0	0
L1	0	0	1	0	0	0
L2	0	1	1	0	0	0
L3	1	1	1	0	0	0
R1	0	0	0	1	0	0
R2	0	0	0	1	1	0
R3	0	0	0	1	1	1
LR3	1	1	1	1	1	1

1、构造状态图

无二义性的

IDLE：全灭

L1：左边1个灯亮

L2：左边2个灯亮

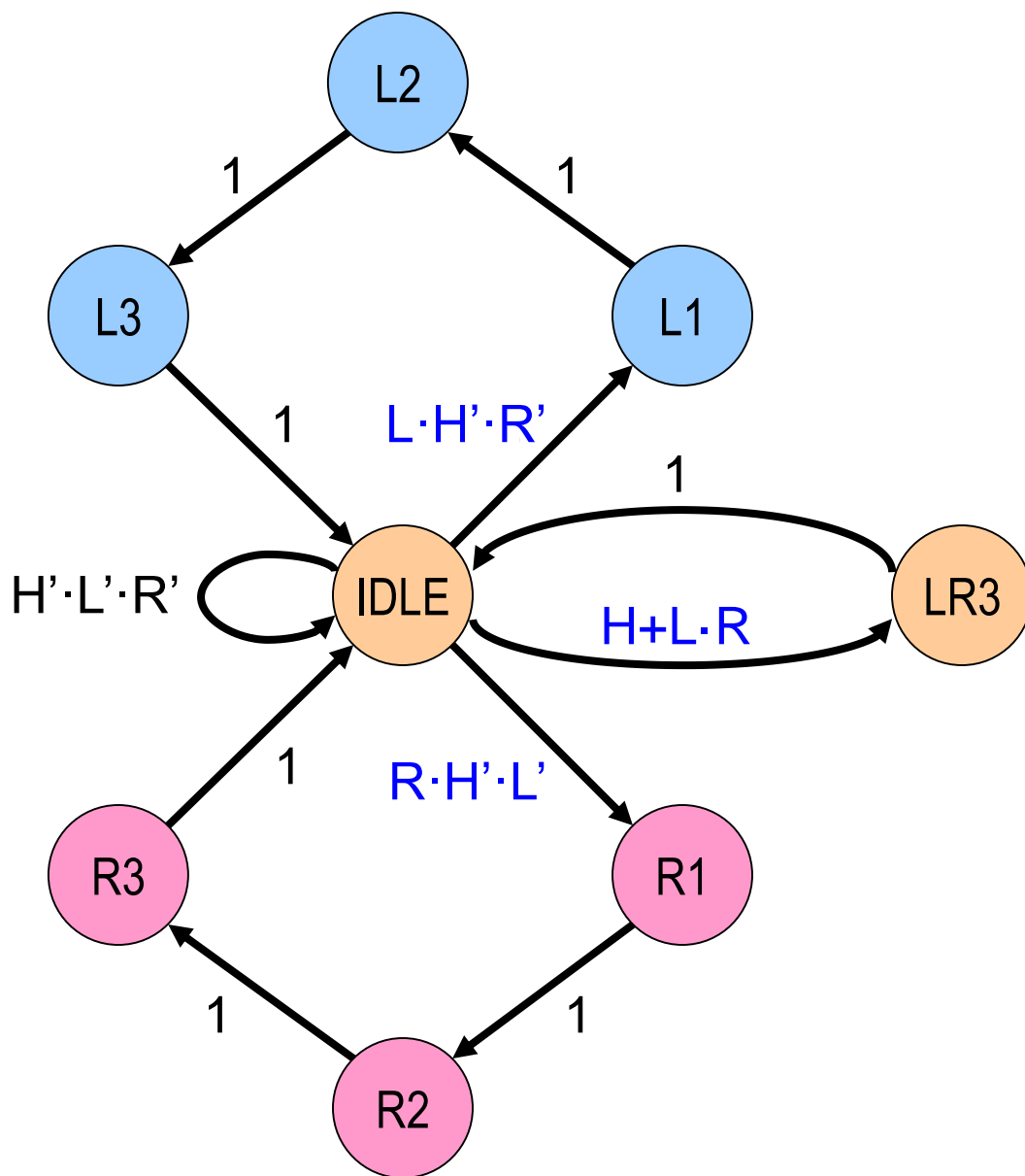
L3：左边3个灯亮

R1：右边1个灯亮

R2：右边2个灯亮

R3：右边3个灯亮

LR3：全亮



1、构造状态图

无二义性的

Mutual Exclusion

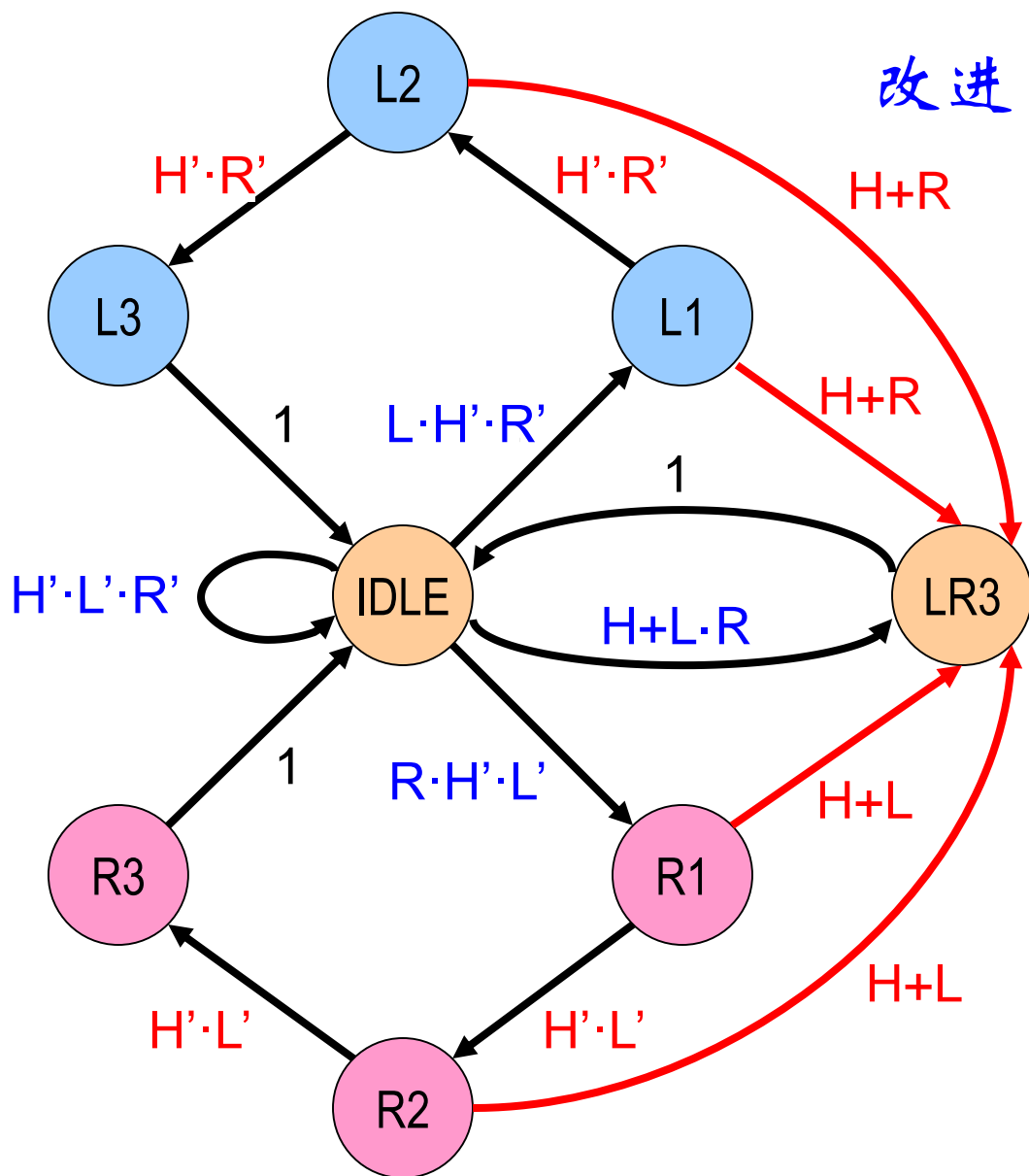
(互斥性)

离开某一状态的弧线上的任意一对转移表达式的逻辑积为0

All Inclusion

(完备性)

离开某一状态的弧线上的所有转移表达式的逻辑和为1。



1、构造状态图

2、状态编码

	Q2Q1Q0		
IDLE	0	0	0
L1	0	0	1
L2	0	1	1
L3	0	1	0
R1	1	0	1
R2	1	1	1
R3	1	1	0
LR3	1	0	0

1、构造状态图

2、状态编码

3、得到转移列表

Q2Q1Q0	S	转移表达式	S*	Q2*Q1*Q0*
0 0 0	IDLE	$H' \cdot L' \cdot R'$	IDLE	0 0 0
0 0 0		$L \cdot H' \cdot R'$	L1	0 0 1
0 0 0		$R \cdot H' \cdot L'$	R1	1 0 1
0 0 0		$H+L \cdot R$	LR3	1 0 0
⋮	⋮	⋮	⋮	⋮

用转移表综合

状态机

Q2Q1Q0	S	转移表达式	S*	Q2*Q1*Q0*
0 0 0	IDLE	$H' \cdot L' \cdot R'$	IDLE	0 0 0
0 0 0		$L \cdot H' \cdot R'$	L1	0 0 1
0 0 0		$R \cdot H' \cdot L'$	R1	1 0 1
0 0 0		$H+L \cdot R$	LR3	1 0 0
0 0 1	L1	$H' \cdot R'$	L2	0 1 1
0 0 1		$H+R$	LR3	1 0 0
0 1 1	L2	$H' \cdot R'$	L3	0 1 0
0 1 1		$H+R$	LR3	1 0 0
0 1 0	L3	1	IDLE	0 0 0
1 0 1	R1	$H' \cdot L'$	R2	1 1 1
1 0 1		$H+L$	LR3	1 0 0
1 1 1	R2	$H' \cdot L'$	R3	1 1 0
1 1 1		$H+L$	LR3	1 0 0
1 1 0	R3	1	IDLE	0 0 0
1 0 0	LR3	1	IDLE	0 0 0

用转移表综合

状态机

Q2Q1Q0	S	转移表达式	S*	Q2*Q1*Q0*
0 0 0		$H' \cdot L' \cdot R'$		0 0 0
0 0 0		$L \cdot H' \cdot R'$		0 0 1
0 0 0		$R \cdot H' \cdot L'$		1 0 1
0 0 0		$H+L \cdot R$		1 0 0
0 0 1		$H' \cdot R'$		0 1 1
0 0 1		$H+R$		1 0 0
0 1 1		$H' \cdot R'$		0 1 0
0 1 1		$H+R$		1 0 0
0 1 0		1		0 0 0
1 0 1		$H' \cdot L'$		1 1 1
1 0 1		$H+L$		1 0 0
1 1 1		$H' \cdot L'$		1 1 0
1 1 1		$H+L$		1 0 0
1 1 0		1		0 0 0
1 0 0		1		0 0 0

$Q0^* =$

$Q2' \cdot Q1' \cdot Q0' \cdot (L \cdot H' \cdot R')$

$+ Q2' \cdot Q1' \cdot Q0' \cdot (R \cdot H' \cdot L')$

$+ Q2' \cdot Q1' \cdot Q0 \cdot (H' \cdot R')$

$+ Q2 \cdot Q1' \cdot Q0 \cdot (H' \cdot L')$

$= Q2' \cdot Q1' \cdot Q0' \cdot H' \cdot (L \oplus R)$

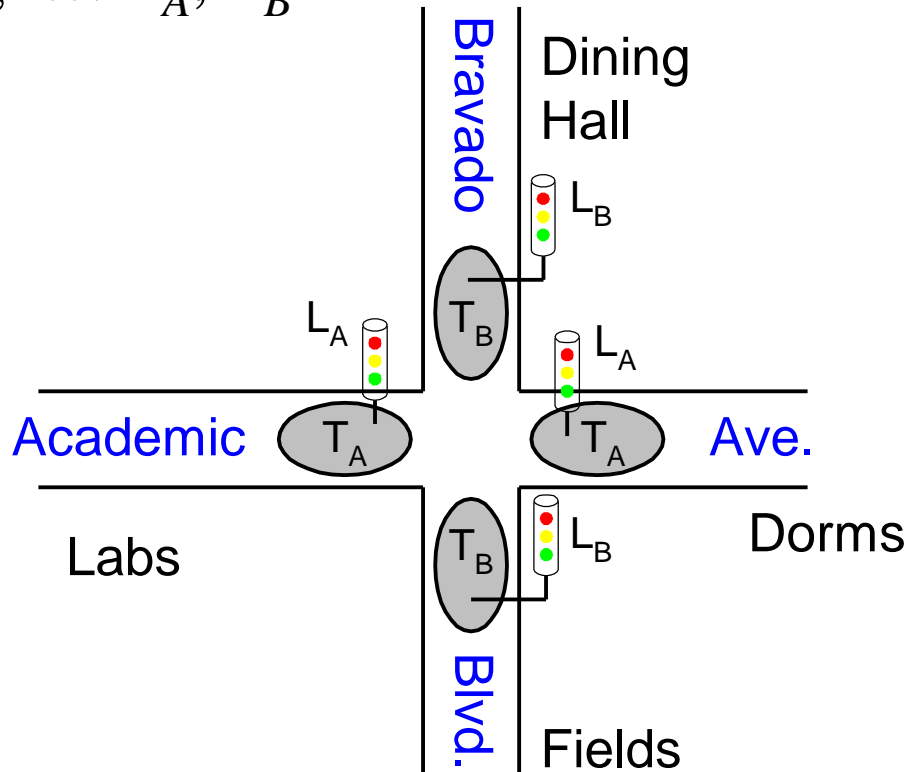
$+ Q2' \cdot Q1' \cdot Q0 \cdot (H' \cdot R')$

$+ Q2 \cdot Q1' \cdot Q0 \cdot (H' \cdot L')$

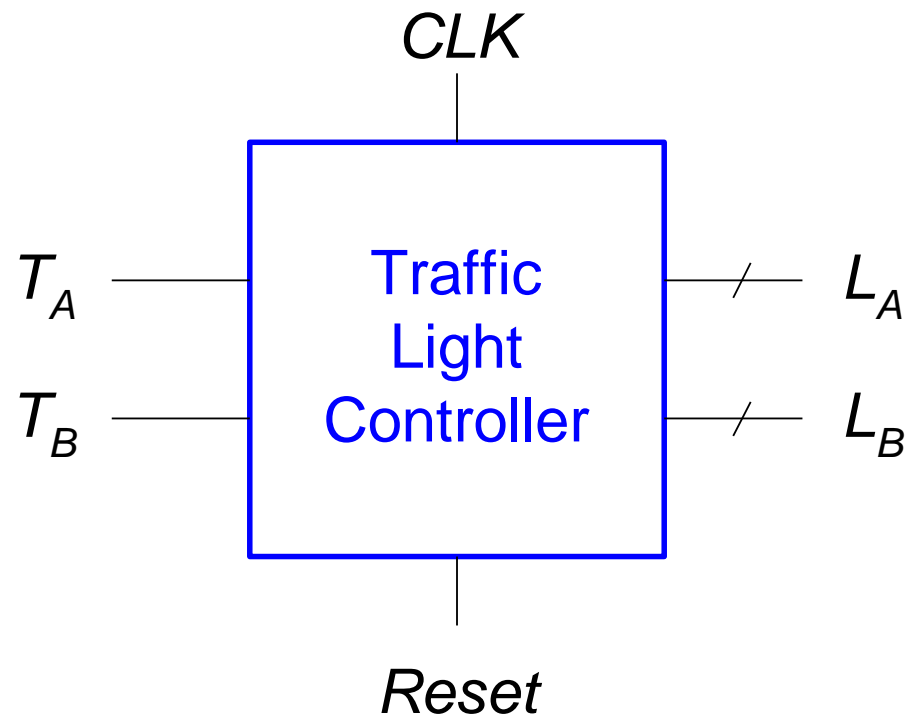
Example 5: traffic lights controller

Traffic Light Controller(交通灯控制器)设计

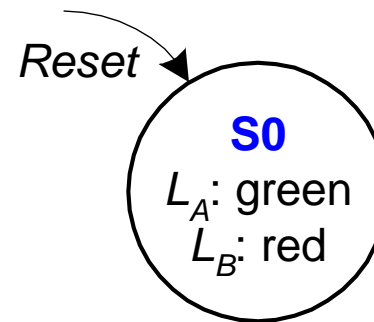
- Traffic light controller
 - Traffic sensors: T_A , T_B (TRUE when there's traffic)
 - Lights: L_A , L_B



- Inputs: CLK , $Reset$, T_A , T_B
- Outputs: L_A , L_B

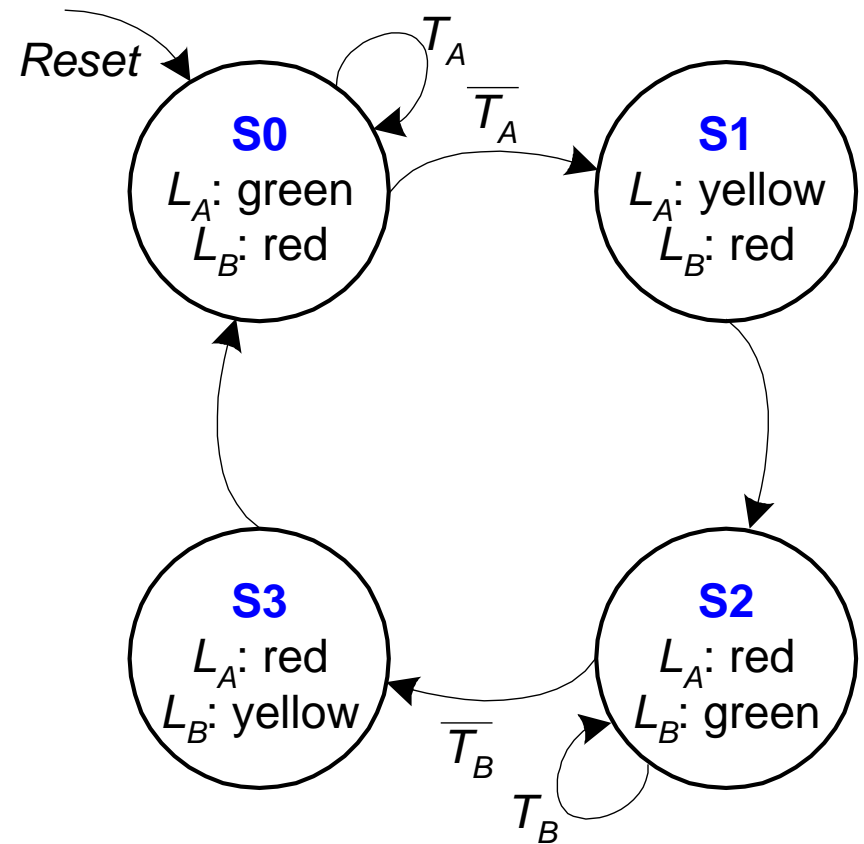


- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



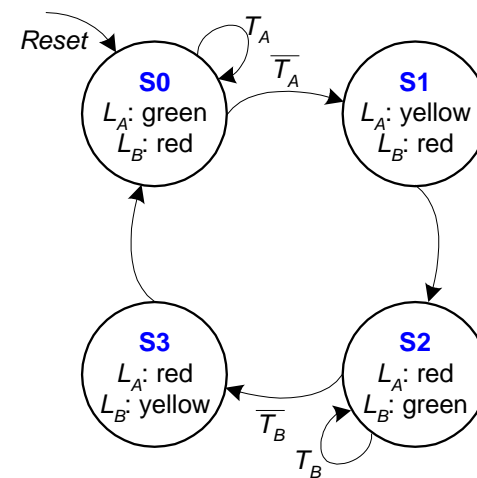
状态转移图

- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



状态转移表

Current State	Inputs		Next State
CS	T_A	T_B	NS
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0



状态转移表

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S_1'	S_0'
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

$$S_1' = S_1 \oplus S_0$$

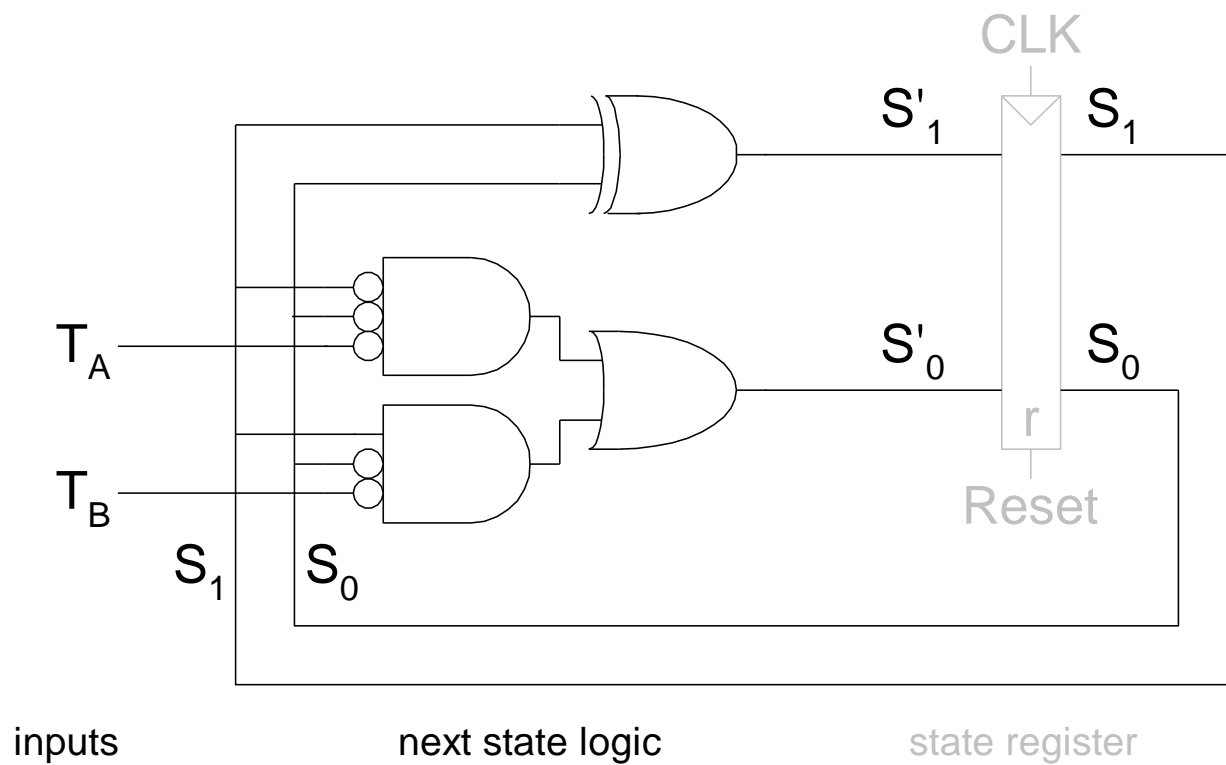
$$S_0' = \overline{S_1} \overline{S_0} \overline{T_A} + S_1 \overline{S_0} \overline{T_B}$$

$S : S_1 S_0$ (2 bits)

State	Encoding
S0	00
S1	01
S2	10
S3	11

状态编码

下一状态的逻辑实现



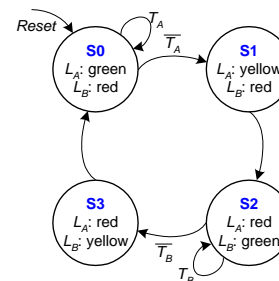
状态输出表

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

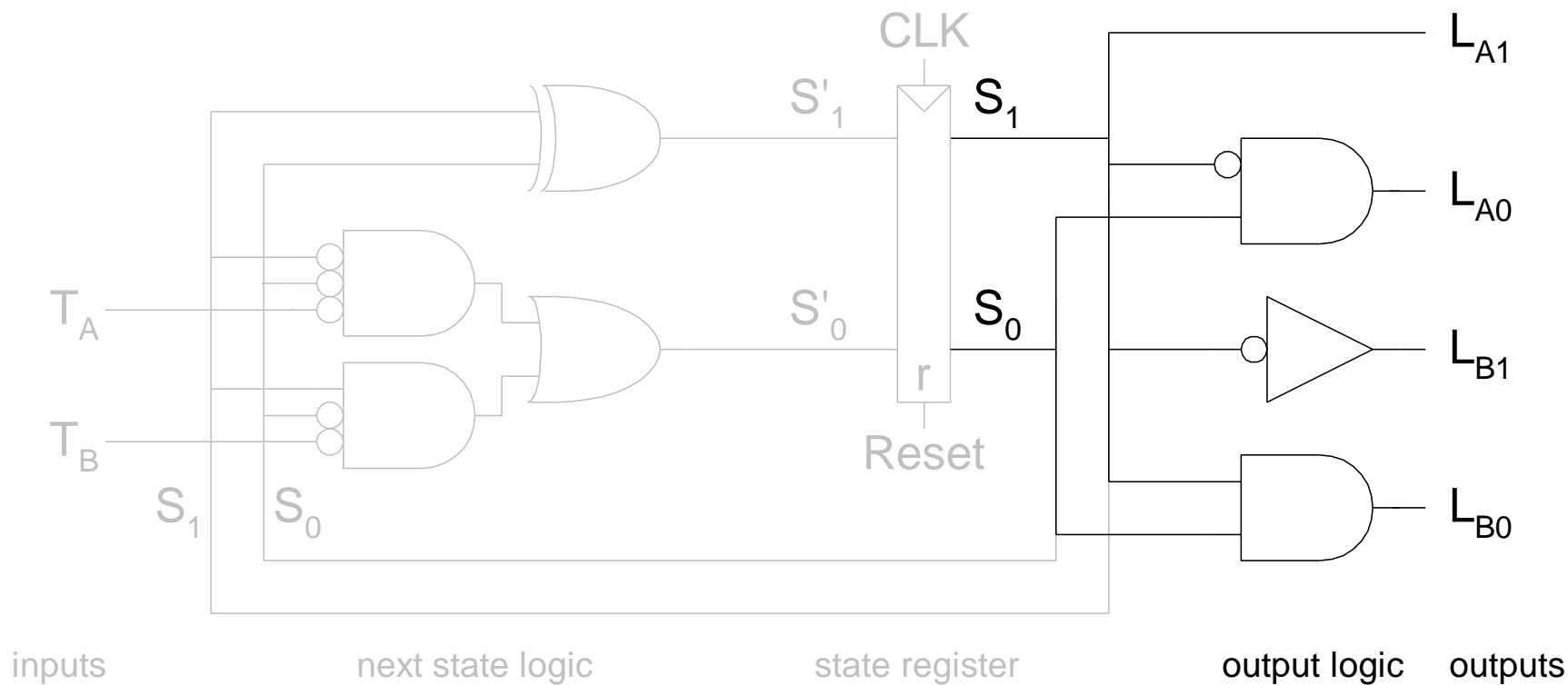
$$\begin{aligned}
 L_{A1} &= S_1 \\
 L_{A0} &= \overline{S_1} S_0 \\
 L_{B1} &= S_1 \\
 L_{B0} &= S_1 S_0
 \end{aligned}$$

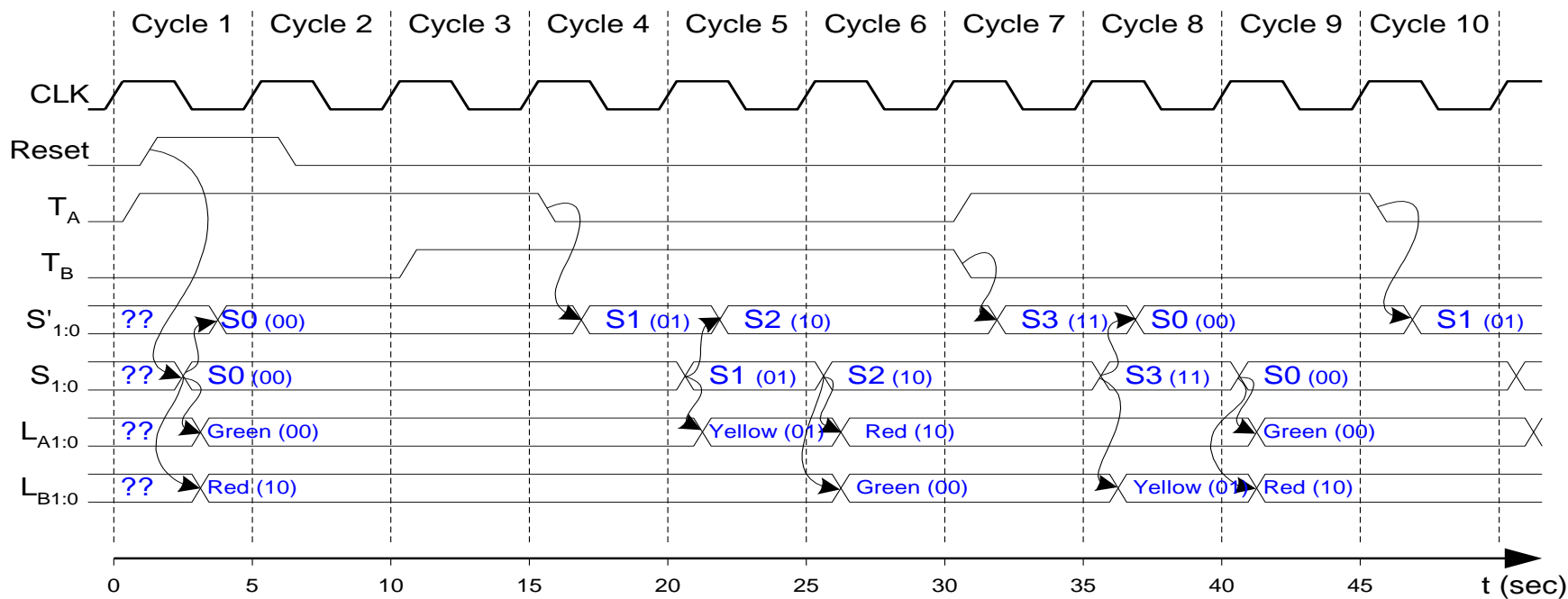
Output	Encoding
green	00
yellow	01
red	10

输出编码

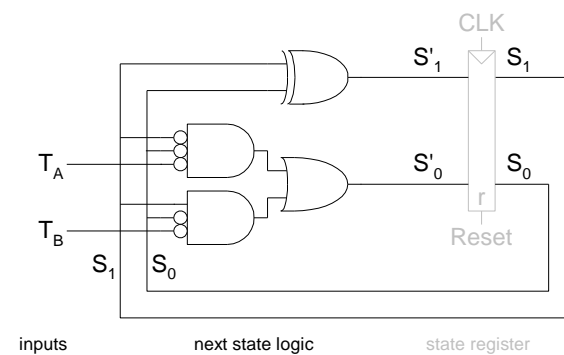
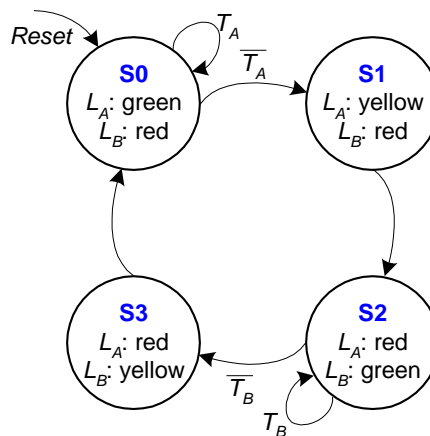


输出的逻辑实现





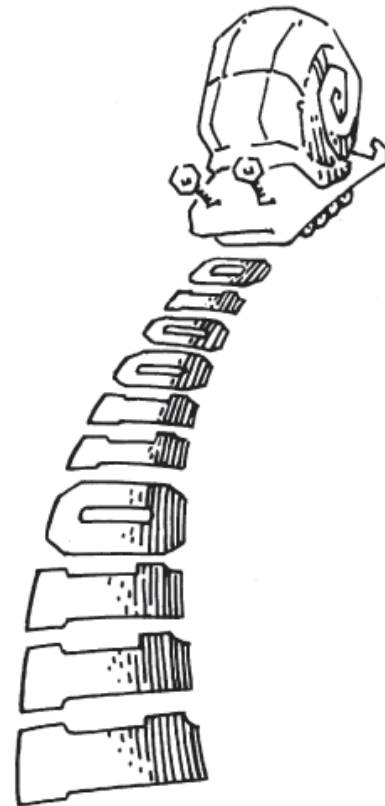
状态转移图



Example 6: '01' sequence detector

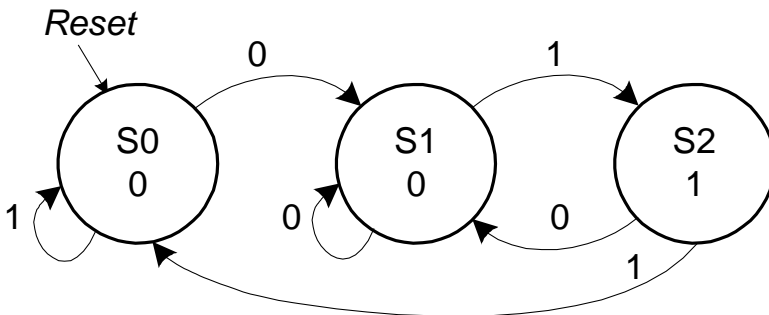
Alyssa P. Hacker has a snail that crawls down a paper tape with 1's and 0's on it. The snail smiles whenever the last two digits it has crawled over are 01. Design Moore and Mealy FSMs of the snail's brain.

序列检测器

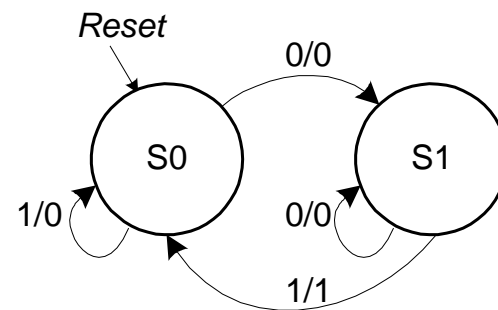


状态转移图

Moore FSM



Mealy FSM



Mealy FSM: arcs indicate input/output

Moore FSM

State transition table

Current State		Inputs	Next State	
S_1	S_0		S_1'	S_0'
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

$$S_1' = S_0 A$$

$$S_0' = \bar{A}$$

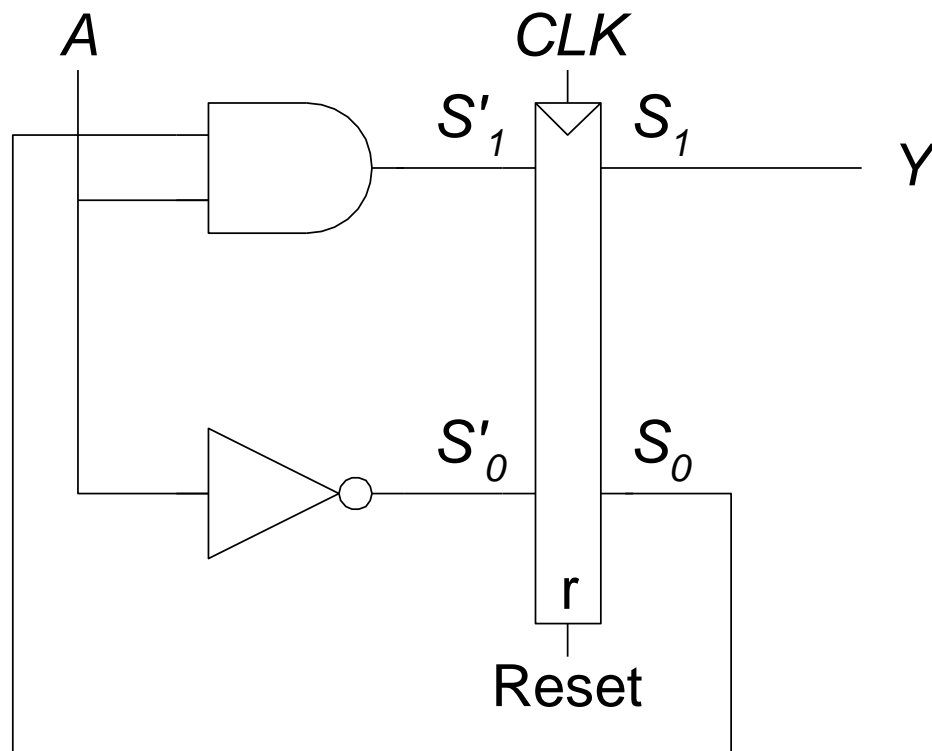
$$Y = S_1$$

State	Encoding
S0	00
S1	01
S2	10

State output table

Current State		Output
S_1	S_0	Y
0	0	0
0	1	0
1	0	1

Moore FSM



$$S_1' = S_0 A$$

$$S_0' = A^{\overline{}}$$

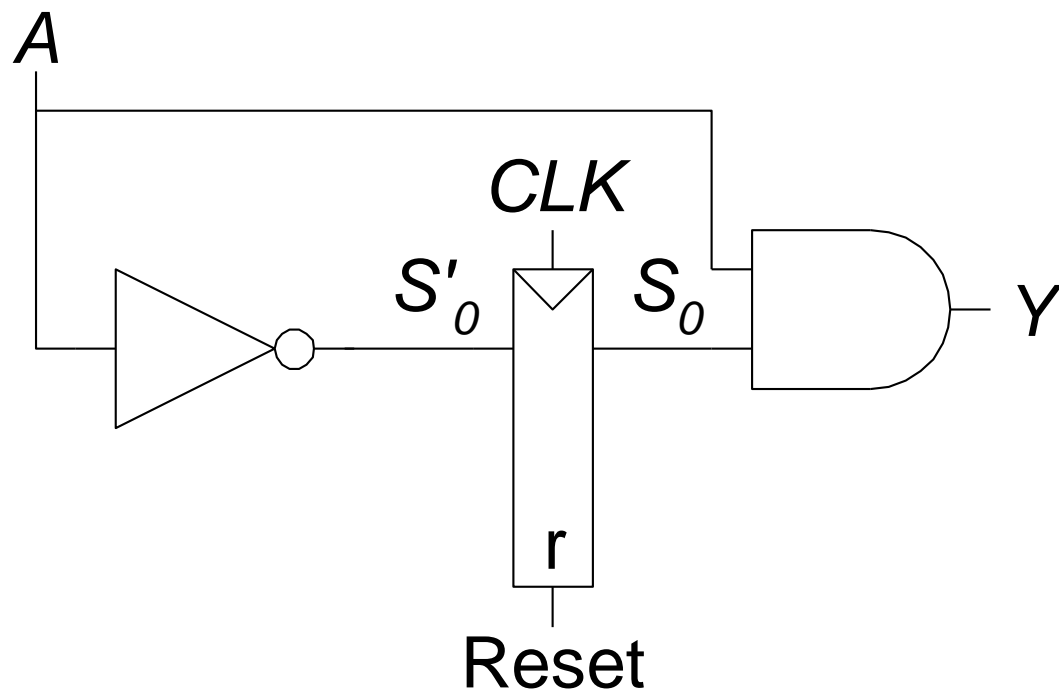
$$Y = S_1$$

Mealy FSM

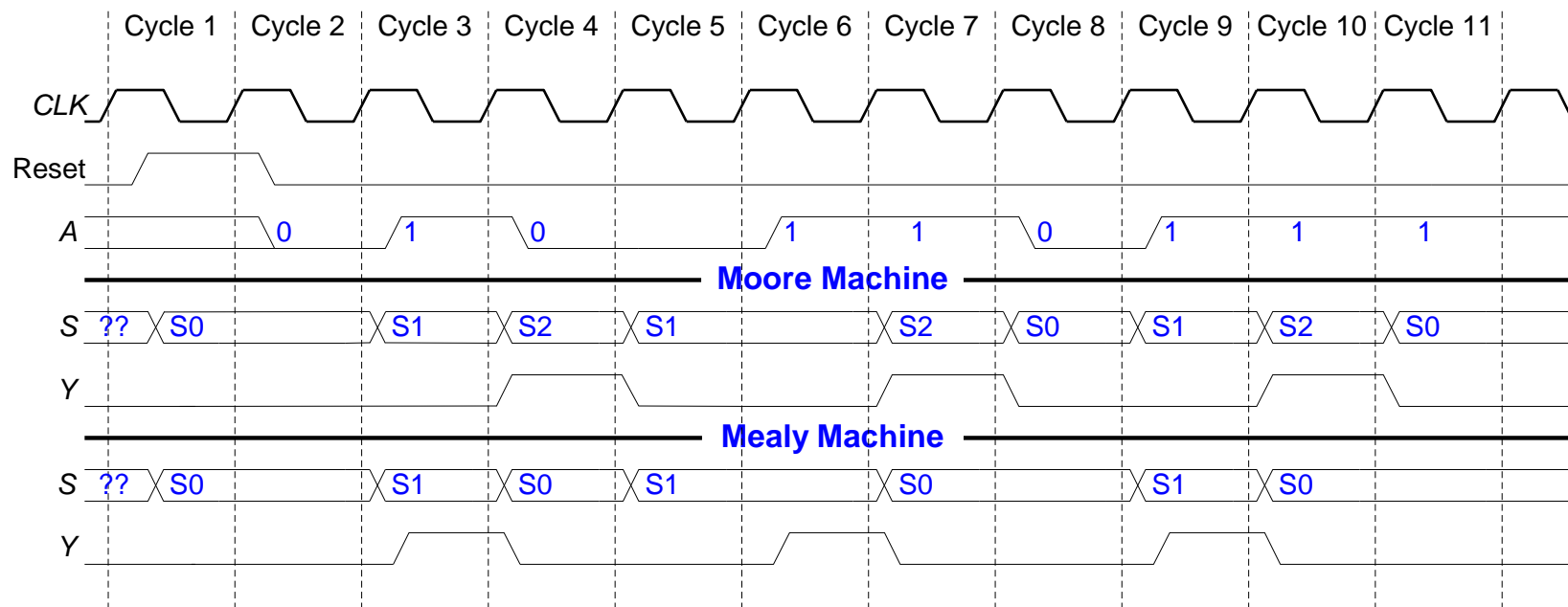
State transition output table

Current State	Input	Next State	Output
S_0	A	S^*_0	Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

State	Encoding
S0	0
S1	1



时序图



Quiz

若用One-hot编码，请画出
电路图

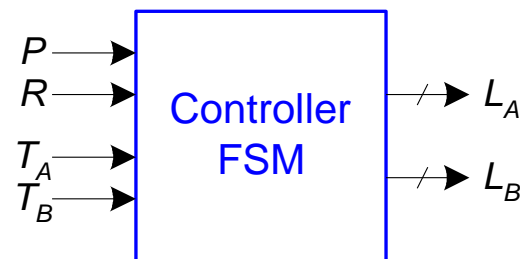
complex FSMs design exapmle



复杂的FSM如何设计？

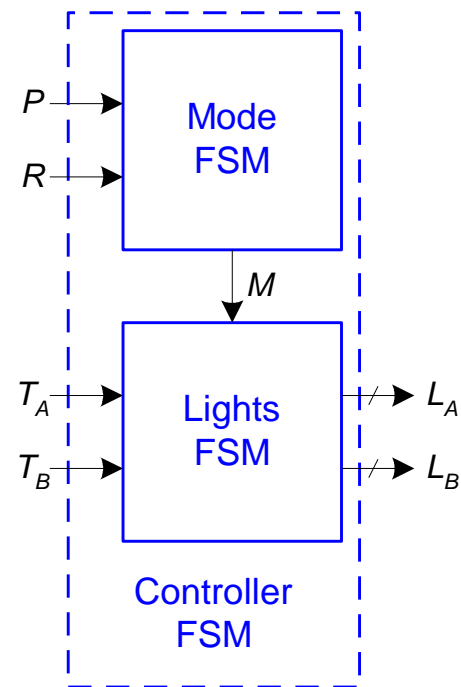
- Break complex FSMs into smaller interacting FSMs
- Example: Modify traffic light controller to have Parade Mode.
 - Two more inputs: P , R
 - When $P = 1$, enter Parade Mode & Bravado Blvd light stays green
 - When $R = 1$, leave Parade Mode

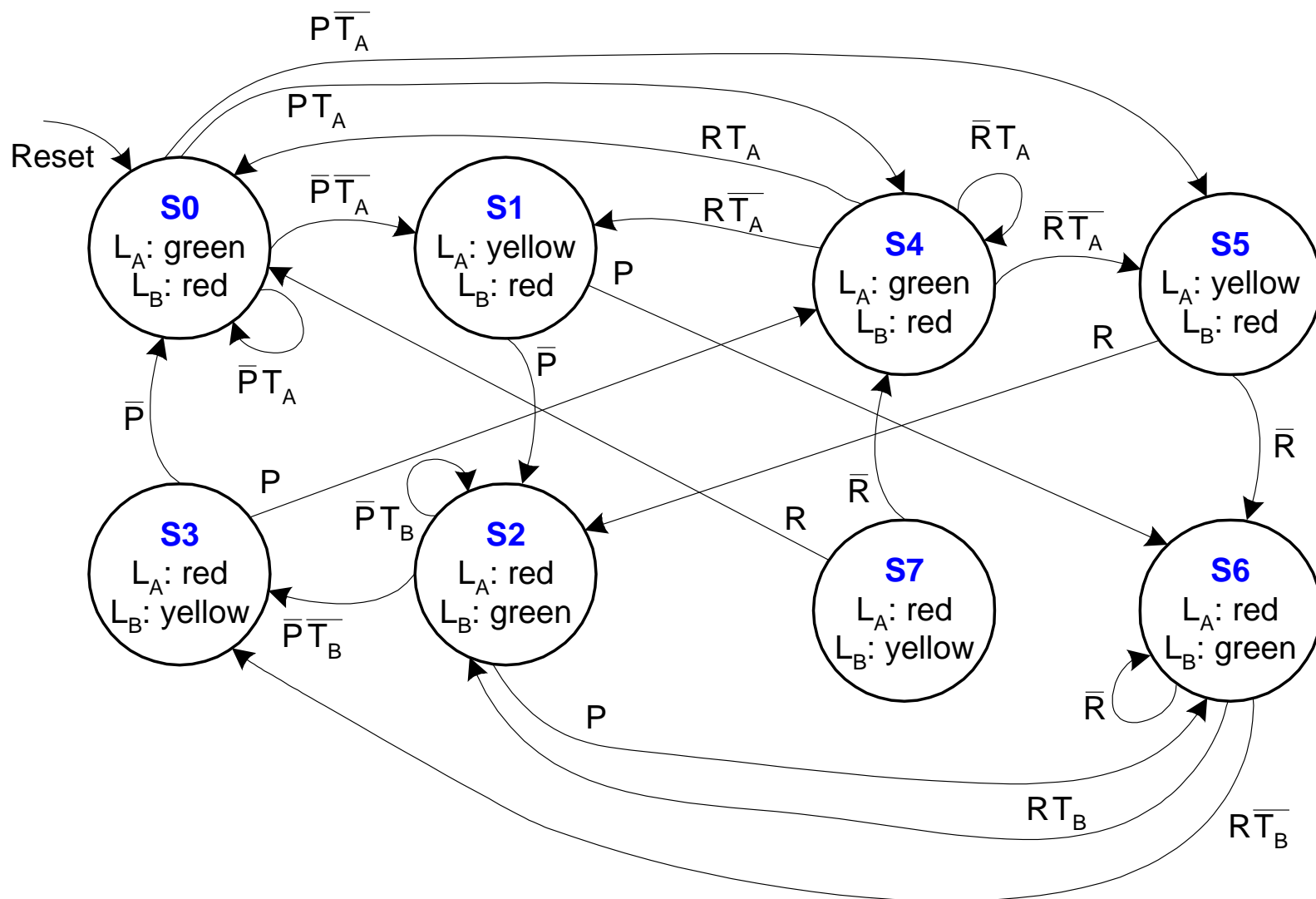
Unfactored FSM



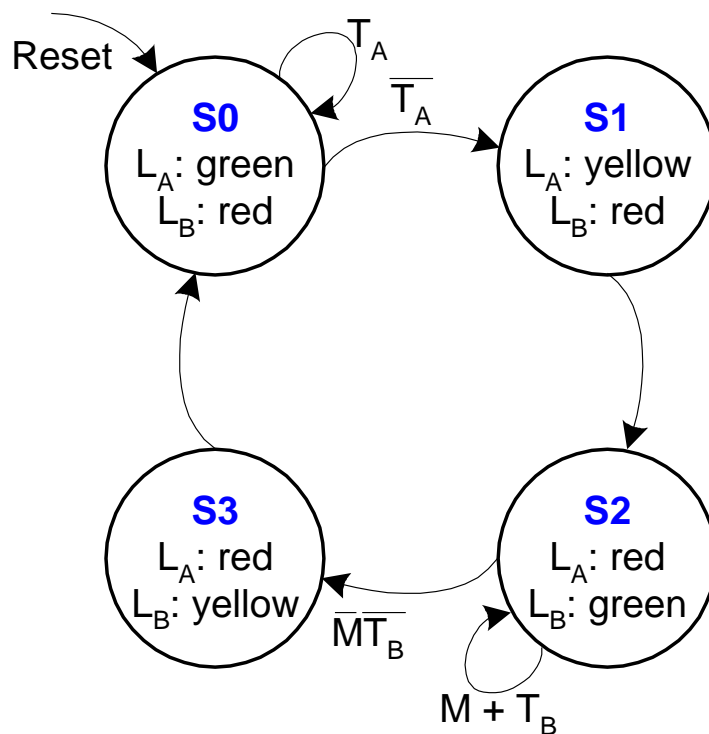
Factored FSM

模块化设计

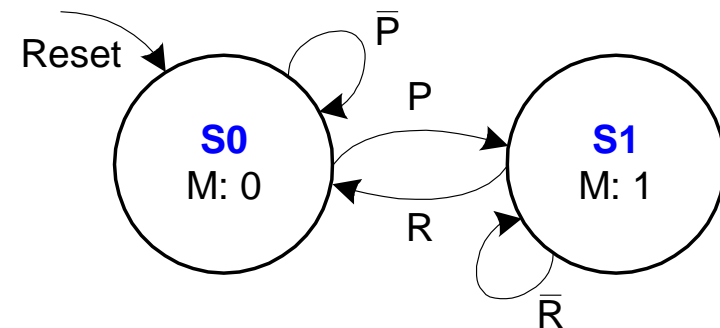




模块化设计



Lights FSM



Mode FSM

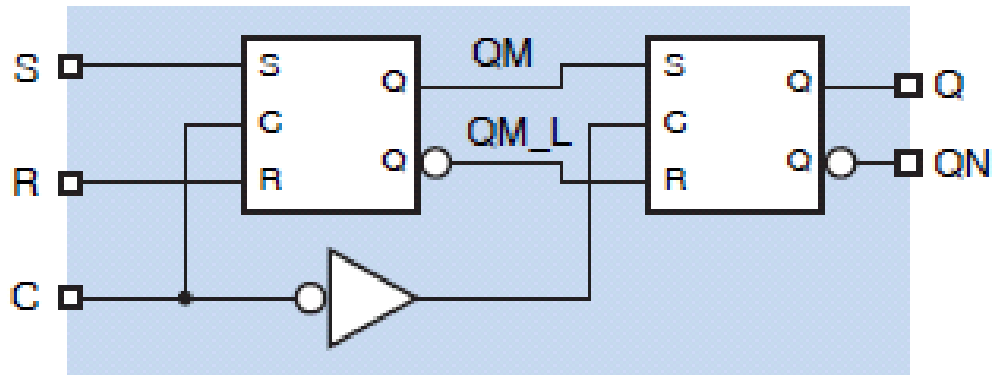


Summary: FSM设计步骤:





1. Identify inputs and outputs
2. Sketch state transition diagram
3. Write state transition table
4. Select state encodings
5. For Moore machine:
 - Rewrite state transition table with state encodings
 - Write output table
6. For a Mealy machine:
 - Rewrite combined state transition and output table with state encodings
7. Write Boolean equations for next state and output logic
8. Sketch the circuit schematic

Master/Slave RS flip-flop

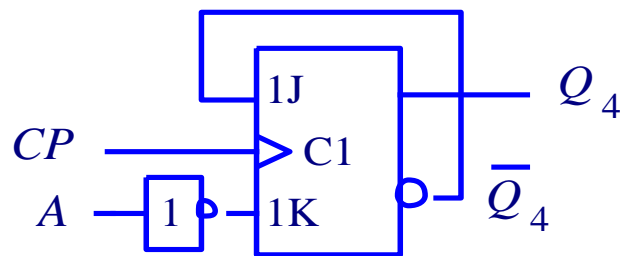
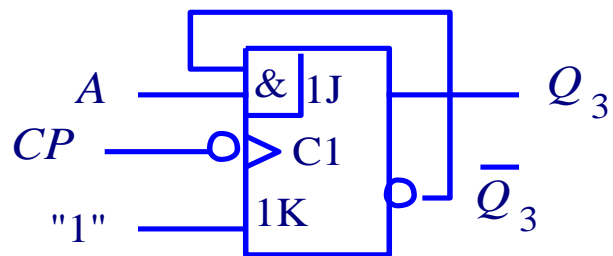
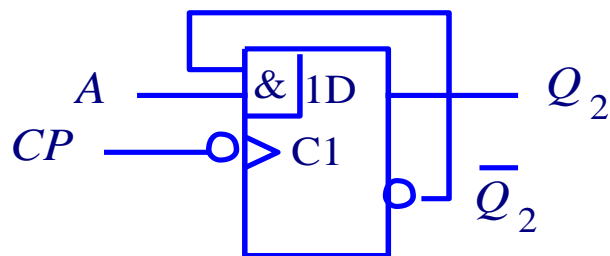
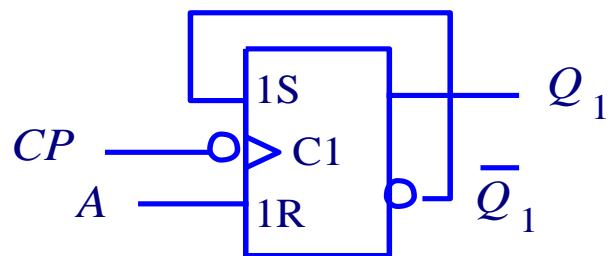
(a)



(b)

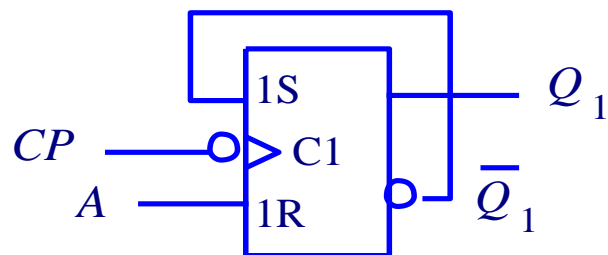
S	R	C	Q	QN
x	x	0	last Q	last QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		undef.	undef.

Quiz : 写出图示电路的状态方程

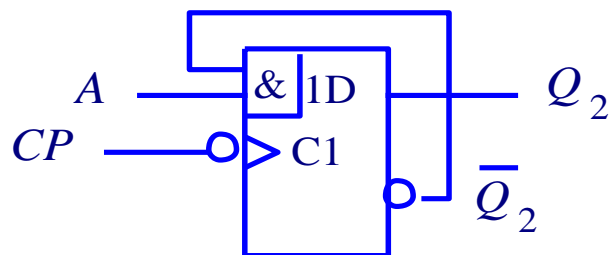




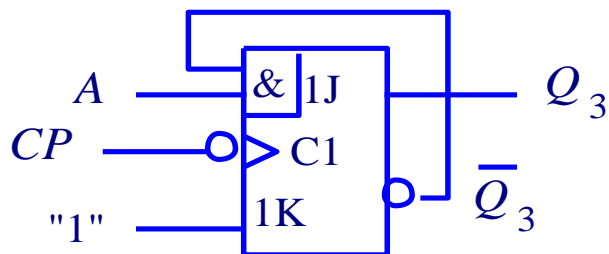
Quiz : 写出图示电路的状态方程



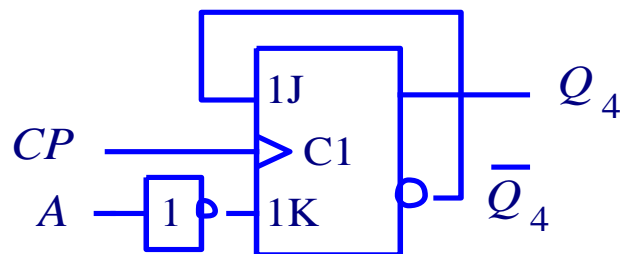
$$Q_1^{n+1} = \overline{Q_1^n} + \overline{A} \cdot Q_1^n$$



$$Q_2^{n+1} = A \cdot \overline{Q_2^n}$$



$$Q_3^{n+1} = A \cdot \overline{Q_3^n}$$



$$Q_4^{n+1} = A Q_4^n + \overline{Q_4^n}$$

讨论

(“1” 计数器 (P557))

设计一个电路，对两个输入X和Y同时计数，
当X和Y输入1的个数和为4的整数倍时输出为1

含义	S	XY				Z
		00	01	11	10	
00 起始状态 → S0	S0	S0	S1	S2	S1	1
01 收到一个1 → S1	S1	S1	S2	S3	S2	0
11 收到两个1 → S2	S2	S2	S3	S0	S3	0
10 收到三个1 → S3	S3	S3	S0	S1	S0	0
		S*				

原始状态转换表

含义	原始状态	输入XY				输出
		00	01	11	10	Z
没有1	S0	S0	S1	S2	S1	1
1个1	S1	S1	S2	S3	S2	0
2个1	S2	S2	S3	S0	S3	0
3个1	S3	S3	S0	S1	S0	0

状态编码 $S_0:00$; $S_1:01$; $S_2:10$; $S_3:11$

原始状态 $Q_1^n Q_2^n$	输入XY				输出
	00	01	11	10	Z
00	00	01	10	01	1
01	01	10	11	10	0
10	10	11	00	11	0
11	11	00	01	00	0

卡诺图

$Q_1^n Q_2^n$					
XY		00	01	11	10
00				1	1
01			1		1
11	1	1			
10		1			1

$Q_1^n Q_2^n$					
XY		00	01	11	10
00			1	1	
01	1				1
11			1	1	
10	1				1

$$Q_1^{n+1} = D_1 = \overline{X} \cdot \overline{Y} Q_1^n + \overline{X} Q_1^n \overline{Q_2^n} + \overline{Y} Q_1^n \overline{Q_2^n} + XY \overline{Q_1^n} + Y \overline{Q_1^n} Q_2^n + X \overline{Q_1^n} Q_2^n$$

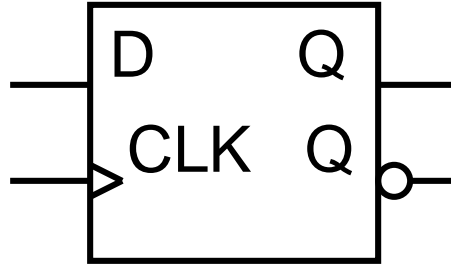
$$Q_2^{n+1} = D_2 = \overline{X} \cdot \overline{Y} Q_2^n + \overline{X} \cdot Y \overline{Q_2^n} + XY Q_2^n + X \overline{Y} \cdot \overline{Q_2^n}$$

$$Z = \overline{Q_1^n} \cdot \overline{Q_2^n}$$

Timing of Sequential Logic

Book: C3-3.5

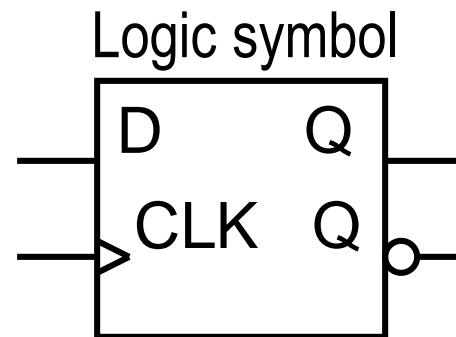
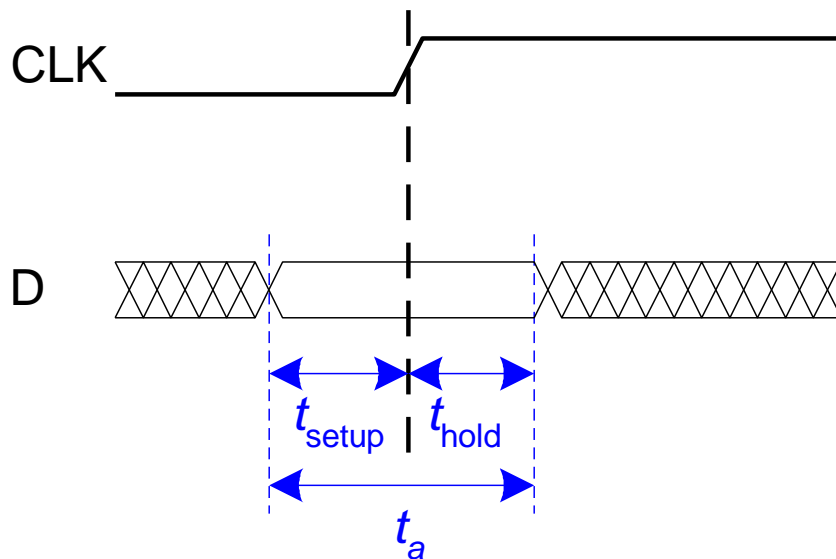
References: C8-8.14



- Flip-flop samples D at clock edge
- D must be stable when sampled
- Similar to a photograph, D must be stable around clock edge
- If not, metastability can occur

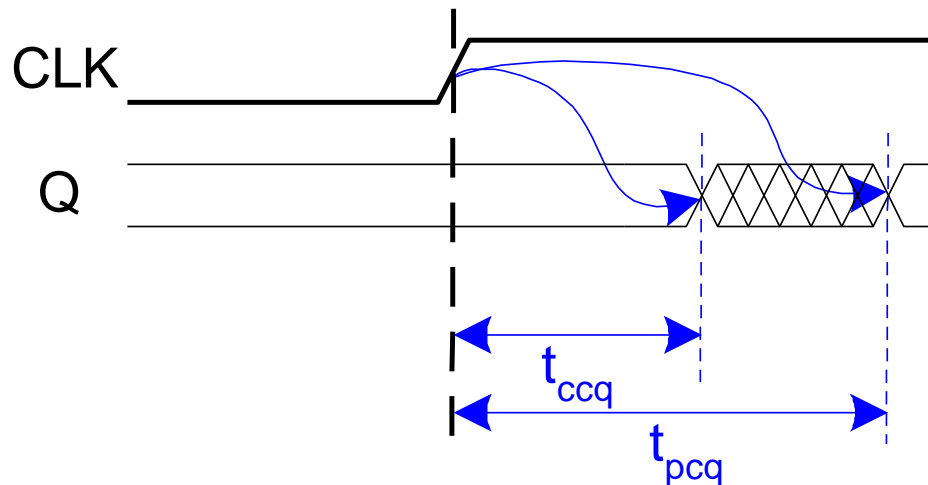
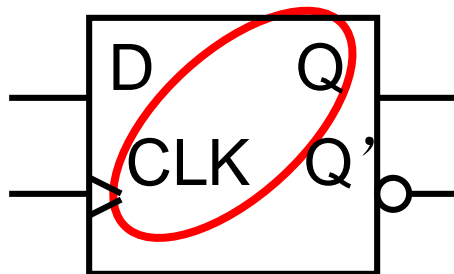
时序的概念

- **Setup time:** t_{setup} = time *before* clock edge data must be stable (i.e. not changing)
- **Hold time:** t_{hold} = time *after* clock edge data must be stable
- **Aperture time:** t_a = time *around* clock edge data must be stable ($t_a = t_{\text{setup}} + t_{\text{hold}}$)

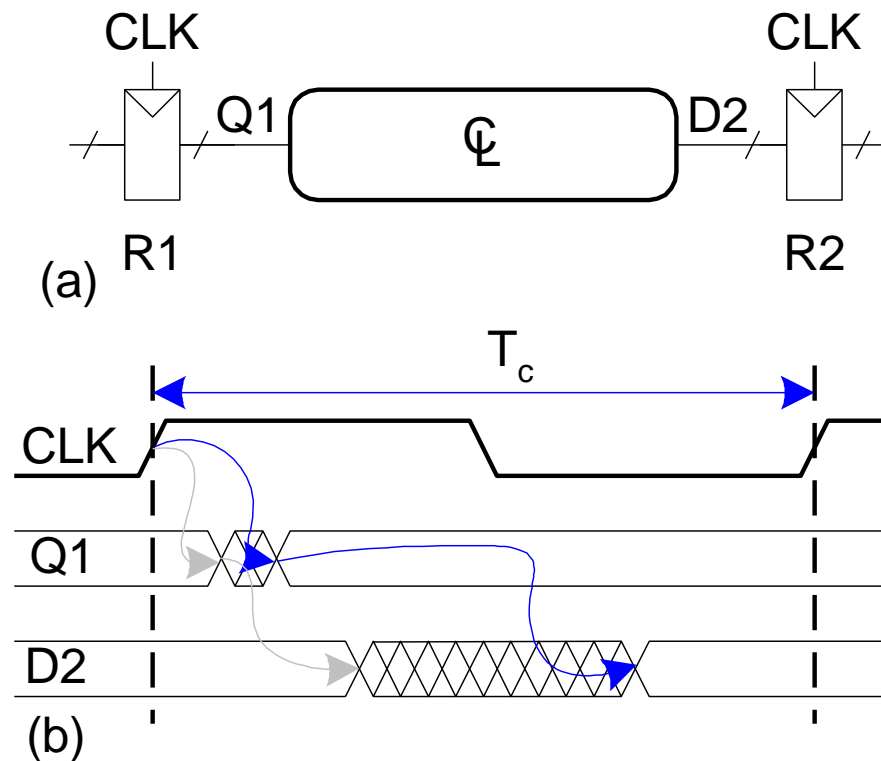


- Synchronous sequential circuit inputs must be stable during aperture (setup and hold) time around clock edge
- Specifically, inputs must be stable
 - at least t_{setup} before the clock edge
 - at least until t_{hold} after the clock edge

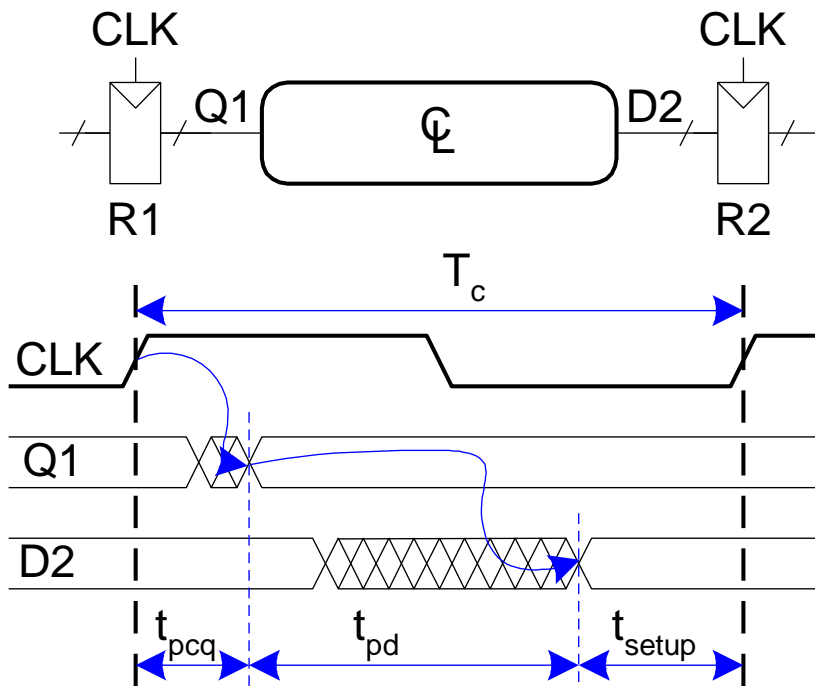
- **Propagation delay:** t_{pcq} = time after clock edge that the output Q is guaranteed to be stable (i.e., to stop changing)
- **Contamination delay:** t_{ccq} = time after clock edge that Q might be unstable (i.e., start changing)



- The delay between registers has a minimum and maximum delay, dependent on the delays of the circuit elements



- System frequency depends on the **maximum** delay from register R1 through combinational logic to R2
- The input to register R2 must be stable at least t_{setup} before clock edge



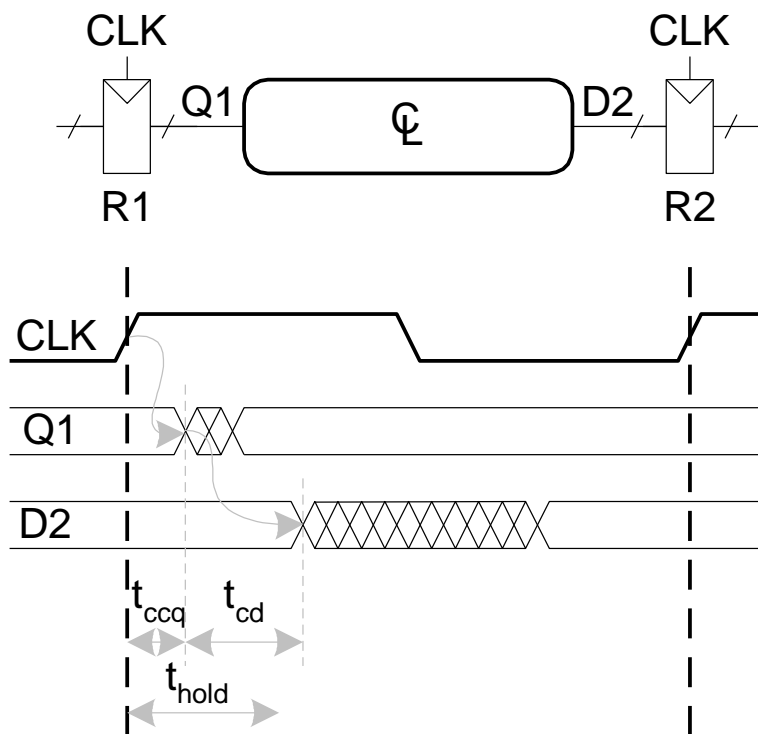
$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{\text{setup}})$$

$(t_{pcq} + t_{\text{setup}})$: **sequencing overhead**

t_{pd} 组合逻辑的最大延迟

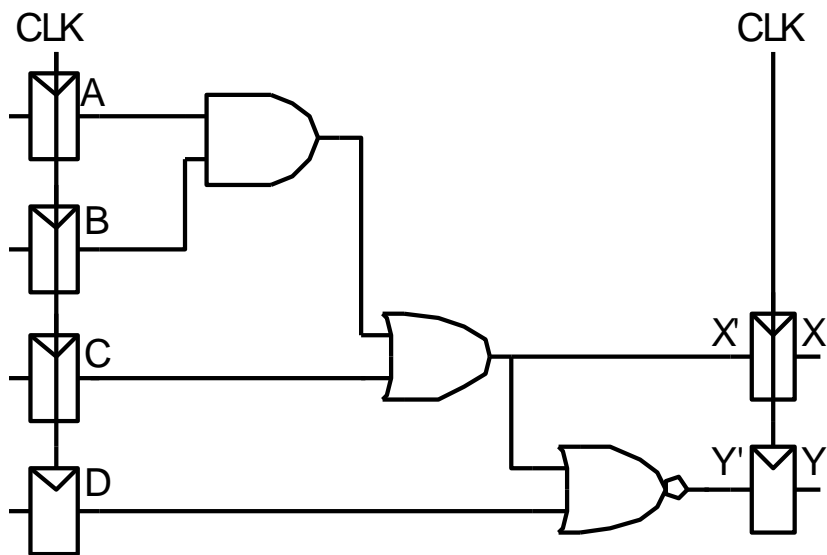
- Depends on the **minimum** delay from register R1 through the combinational logic to R2
- The input to register R2 must be stable for at least t_{hold} after the clock edge



$$t_{\text{hold}} < t_{\text{ccq}} + t_{\text{cd}}$$

$$t_{\text{cd}} > t_{\text{hold}} - t_{\text{ccq}}$$

t_{cd} 组合逻辑的最小延迟



Timing Characteristics

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 50 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 70 \text{ ps}$$

Per Gate :

$$t_{pd} = 35 \text{ ps}$$

$$t_{Cd} = 25 \text{ ps}$$

Setup time constraint: 与时钟和电路结构有关

$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}}$$

$$t_{\text{setup}} \leq T_c - (t_{pcq} + t_{pd})$$

$F_c \leq$ 与时钟和电路结构有关
Setup Time 决定系统工
作频率的范围

Hold time constraint:

$$t_{ccq} + t_{cd} > t_{hold}$$

与时钟无关，只与电路结构有关



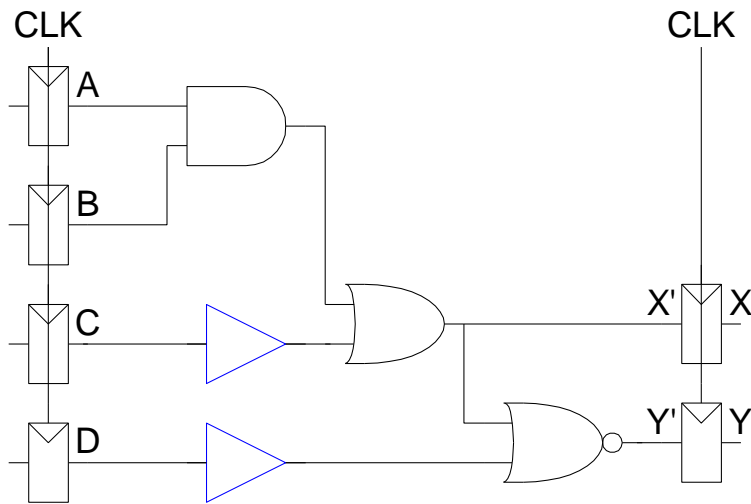
$$t_{\text{hold}} = 70 \text{ ps}$$

$$t_{cd} = 25 \text{ ps}$$
$$f_c \leq 1/T_c = 4.65 \text{ GHz}$$

$(30 + 25) \text{ ps} > 70 \text{ ps}$? **No!**

如何使得Hold Time 满足条件？

Add buffers to the short paths:



Setup time constraint:

$$t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps}$$

$$T_c \geq (50 + 105 + 60) \text{ ps} = 215 \text{ ps}$$

$$f_c \leq 1/T_c = 4.65 \text{ GHz}$$

Timing Characteristics

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 50 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 70 \text{ ps}$$

Per Gate:

$$t_{pd} = 35 \text{ ps}$$

$$t_{cd} = 25 \text{ ps}$$

Hold time constraint:

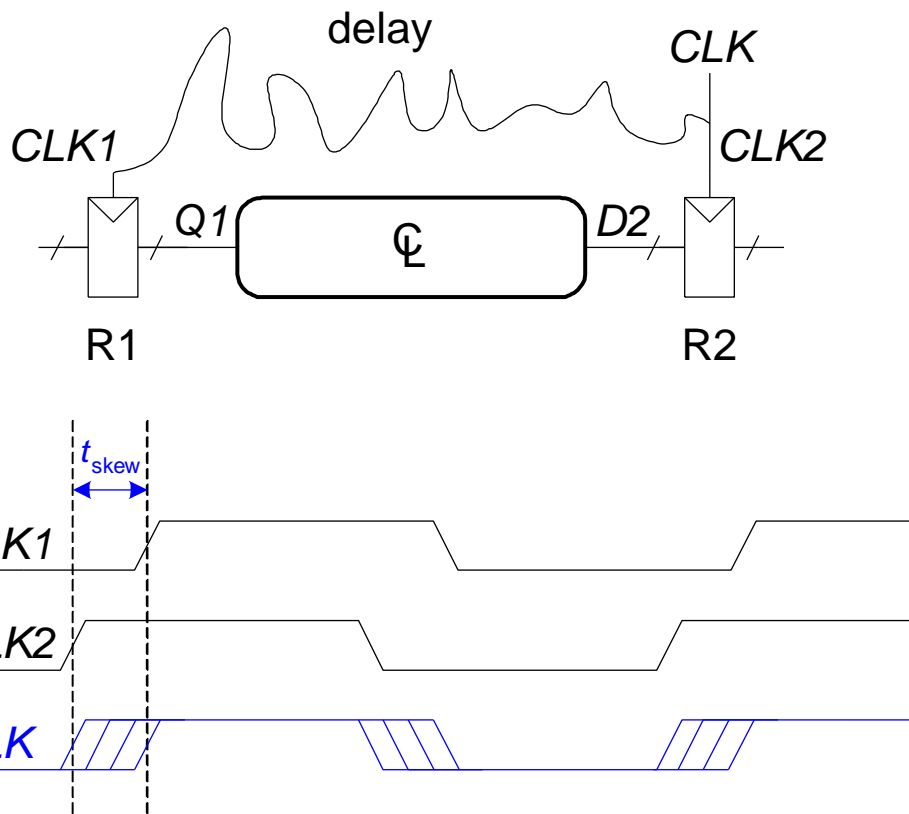
$$t_{cd} = 2 \times 25 \text{ ps} = 50 \text{ ps}$$

$$t_{ccq} + t_{cd} > t_{\text{hold}} ?$$

$$(30 + 50) \text{ ps} > 70 \text{ ps} ? \text{ Yes!}$$

Clock Skew:

- The clock doesn't arrive at all registers at same time
- Skew: difference between two clock edges
- Perform worst case analysis to guarantee dynamic discipline is not violated for any register – many registers in a system!

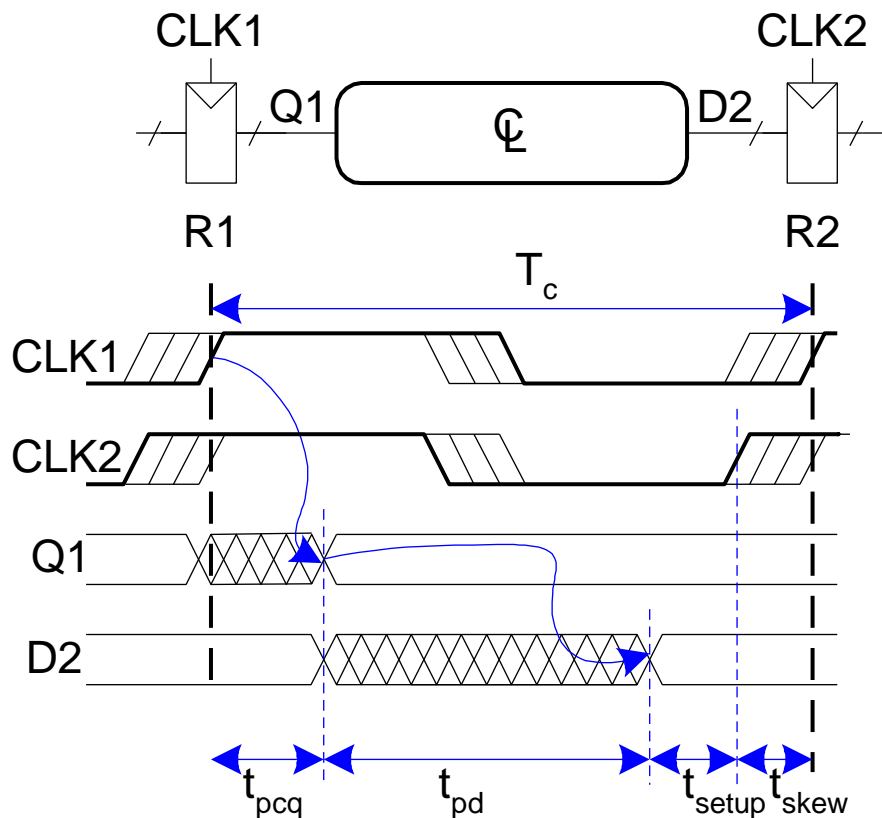




Clock Skew (时钟偏移)

- 一个时钟信号的扇出系数不足以驱动所有输入端，有必要提供多个完全相同的时钟，**使多个时钟信号的输出负载基本平衡**
- 注意时钟信号的通路**将CLOCK信号线布置为树形结构**

Worst case 1: CLK2 is earlier than CLK1



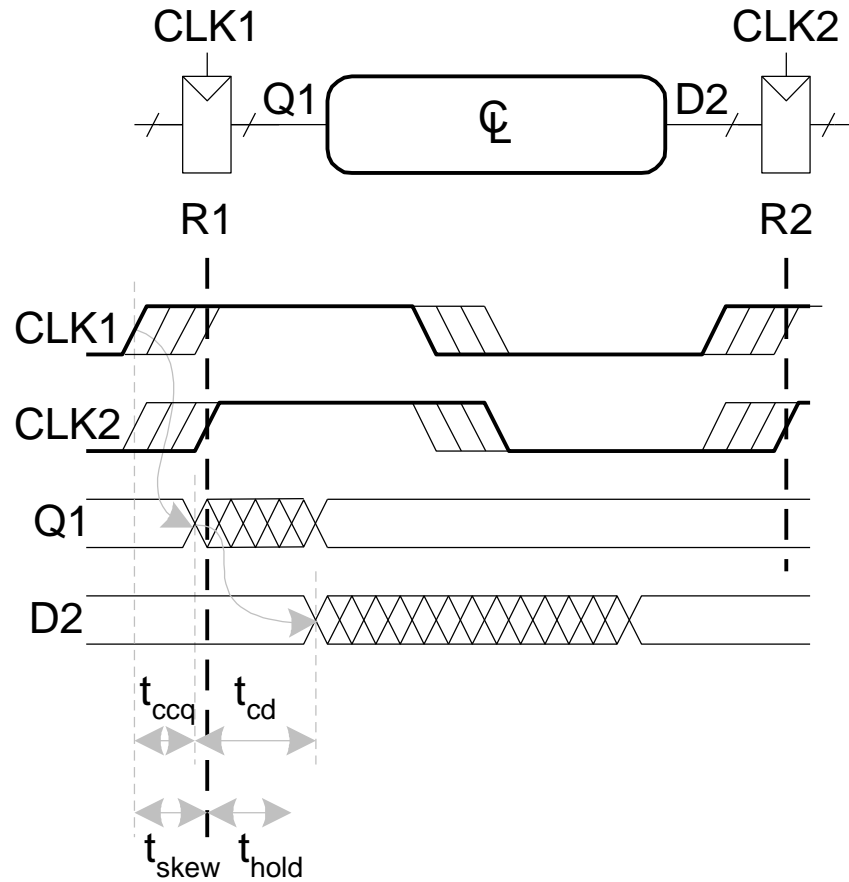
$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew})$$

$$t_{setup} \leq T_c - (t_{pcq} + t_{pd} + t_{skew})$$

t_{hold} ?

Worst case 2: CLK2 is later than CLK1



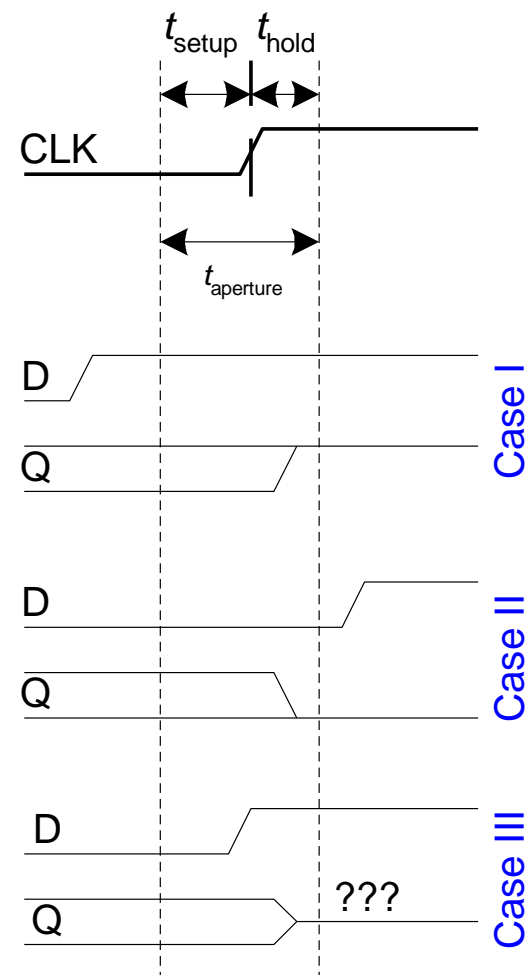
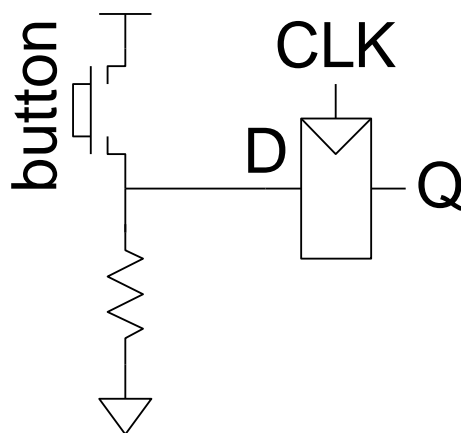
$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$

$$t_{cd} > t_{hold} + t_{skew} - t_{ccq}$$

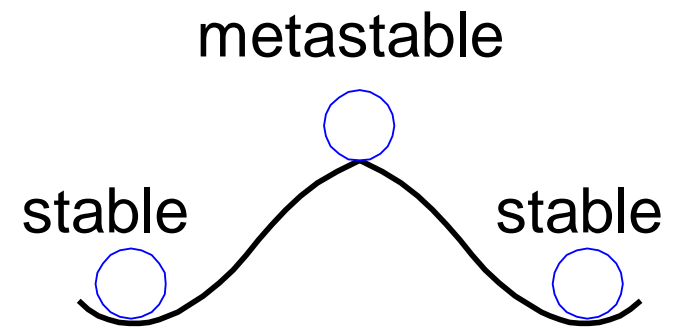
$$t_{hold} < t_{ccq} + t_{cd} - t_{skew}$$

$$t_{setup} \quad ?$$

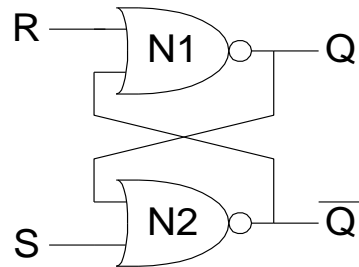
- Asynchronous (for example, user) inputs might violate the dynamic discipline



- **Bistable devices:** two stable states, and a metastable state between them
- **Flip-flop:** two stable states (1 and 0) and one metastable state
- If flip-flop lands in metastable state, could stay there for an undetermined amount of time



- Flip-flop has **feedback**: if Q is somewhere between 1 and 0, cross-coupled gates drive output to either rail (1 or 0)



- Metastable signal**: if it hasn't resolved to 1 or 0
- If flip-flop input changes at random time, **probability that output Q is metastable** after waiting some time, t :

$$P(t_{\text{res}} > t) = (T_0/T_c) e^{-t/\tau}$$

t_{res} : time to resolve to 1 or 0

T_0, τ : properties of the circuit

- **Intuitively:**

T_0/T_c : probability input changes at a bad time (during aperture)

$$P(t_{\text{res}} > t) = (T_0/T_c) e^{-t/\tau}$$

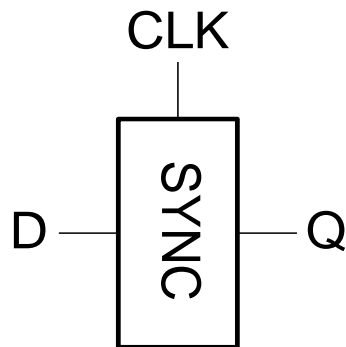
τ : time constant for how fast flip-flop moves away from metastability

$$P(t_{\text{res}} > t) = (T_0/T_c) e^{-t/\tau}$$

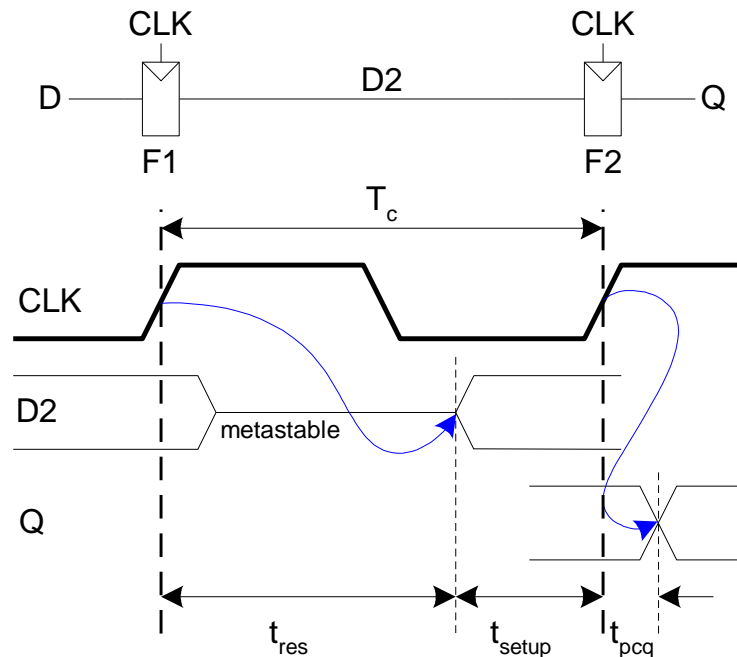
- In short, if flip-flop samples metastable input, if you wait long enough (t), the output will have resolved to 1 or 0 with high probability.

$$P(t_{\text{res}} > t) = (T_0/T_c) e^{-t/\tau}$$

- **Asynchronous inputs are inevitable** (user interfaces, systems with different clocks interacting, etc.)
- **Synchronizer goal:** make the probability of failure (the output Q still being metastable) low
- Synchronizer cannot make the probability of failure 0



- Synchronizer: built with two back-to-back flip-flops
- Suppose D is transitioning when sampled by F1
- Internal signal D2 has $(T_c - t_{\text{setup}})$ time to resolve to 1 or 0



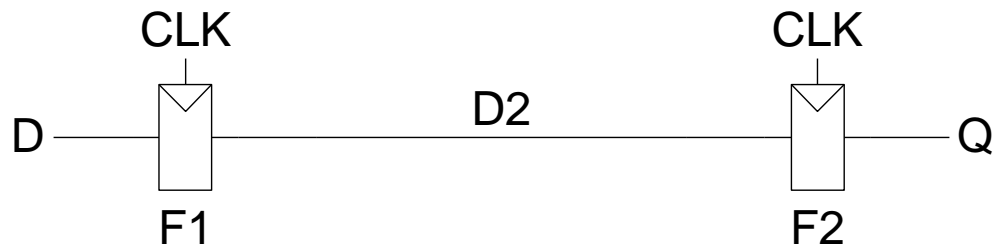
$$P(\text{failure}) = (T_0/T_c) e^{-(T_c - t_{\text{setup}})/\tau}$$

- If asynchronous input changes once per second, probability of failure per second is $P(\text{failure})$.
- If input changes N times per second, probability of failure per second is:

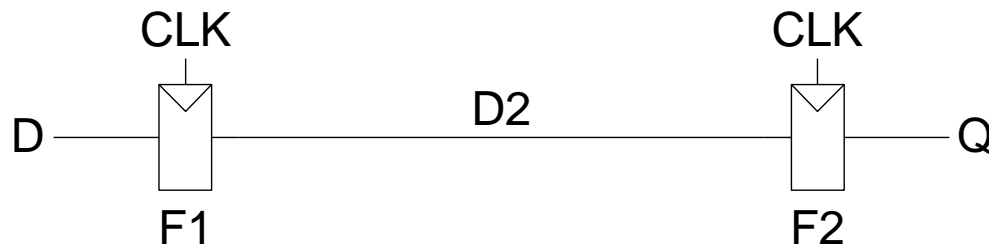
$$P(\text{failure})/\text{second} = (NT_0/T_c) e^{-(T_c - t_{\text{setup}})/\tau}$$

- Synchronizer fails, on average, $1/[P(\text{failure})/\text{second}]$
- Called *mean time between failures*, MTBF:

$$\text{MTBF} = 1/[P(\text{failure})/\text{second}] = (T_c/NT_0) e^{(T_c - t_{\text{setup}})/\tau}$$



- Suppose: $T_c = 1/500 \text{ MHz} = 2 \text{ ns}$ $\tau = 200 \text{ ps}$
 $T_0 = 150 \text{ ps}$ $t_{\text{setup}} = 100 \text{ ps}$
 $N = 10 \text{ events per second}$
- What is the probability of failure? MTBF?



- Suppose: $T_c = 1/500 \text{ MHz} = 2 \text{ ns}$ $\tau = 200 \text{ ps}$
 $T_0 = 150 \text{ ps}$ $t_{\text{setup}} = 100 \text{ ps}$
 $N = 10 \text{ events per second}$

- What is the probability of failure? MTBF?

$$P(\text{failure}) = (150 \text{ ps} / 2 \text{ ns}) e^{-(1.9 \text{ ns}) / 200 \text{ ps}}$$

$$= \mathbf{5.6 \times 10^{-6}}$$

$$P(\text{failure})/\text{second} = 10 \times (5.6 \times 10^{-6})$$

$$= 5.6 \times 10^{-5} / \text{second}$$

$$\text{MTBF} = 1/[P(\text{failure})/\text{second}] \approx \mathbf{5 \text{ hours}}$$

Parallelism



Two types of parallelism:

- **Spatial parallelism (空间并行)**

- duplicate hardware performs multiple tasks at once

- **Temporal parallelism(时间并行)**

task is broken into multiple stages, also called pipelining. For example, an assembly line



- **Token:** Group of inputs processed to produce group of outputs
- **Latency:** Time for one token to pass from start to end

延迟：是指从某个输入建立到与该输入相关联的输出变成有效之间的延迟时间。

- **Throughput:** Number of tokens produced per unit time

吞吐量：输出或者输入的处理速率

Parallelism increases throughput

- Ben Bitdiddle bakes cookies to celebrate traffic light controller installation
- 5 minutes to roll cookies
- 15 minutes to bake
- What is the latency and throughput without parallelism?

- Ben Bitdiddle bakes cookies to celebrate traffic light controller installation
- 5 minutes to roll cookies
- 15 minutes to bake
- What is the latency and throughput without parallelism?

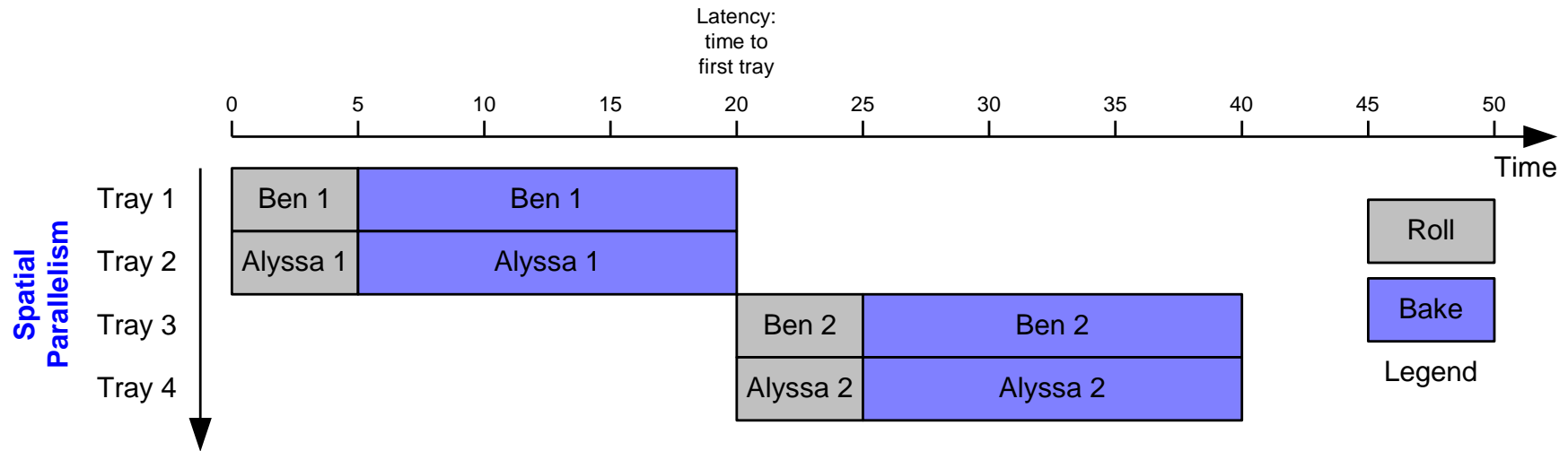
Latency = $5 + 15 = 20$ minutes = **1/3 hour**

Throughput = $1 \text{ tray} / 1/3 \text{ hour} = \mathbf{3 \text{ trays/hour}}$



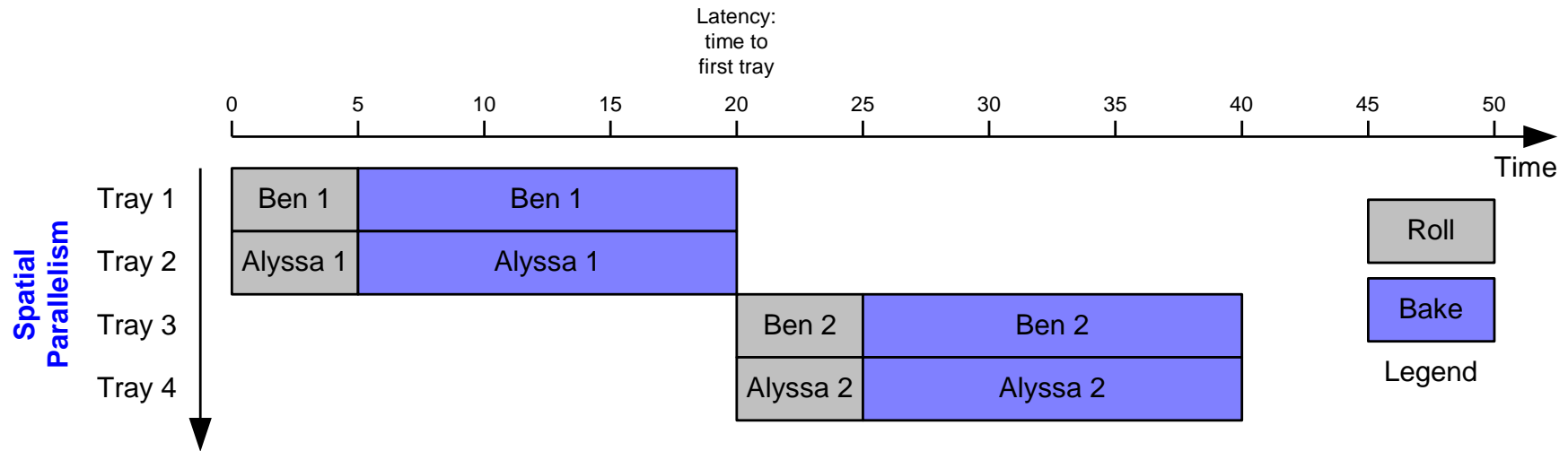
What is the latency and throughput if Ben uses parallelism?

- **Spatial parallelism:** Ben asks Allysa P. Hacker to help, using her own oven
- **Temporal parallelism:**
 - two stages: rolling and baking
 - He uses two trays
 - While first batch is baking, he rolls the second batch, etc.



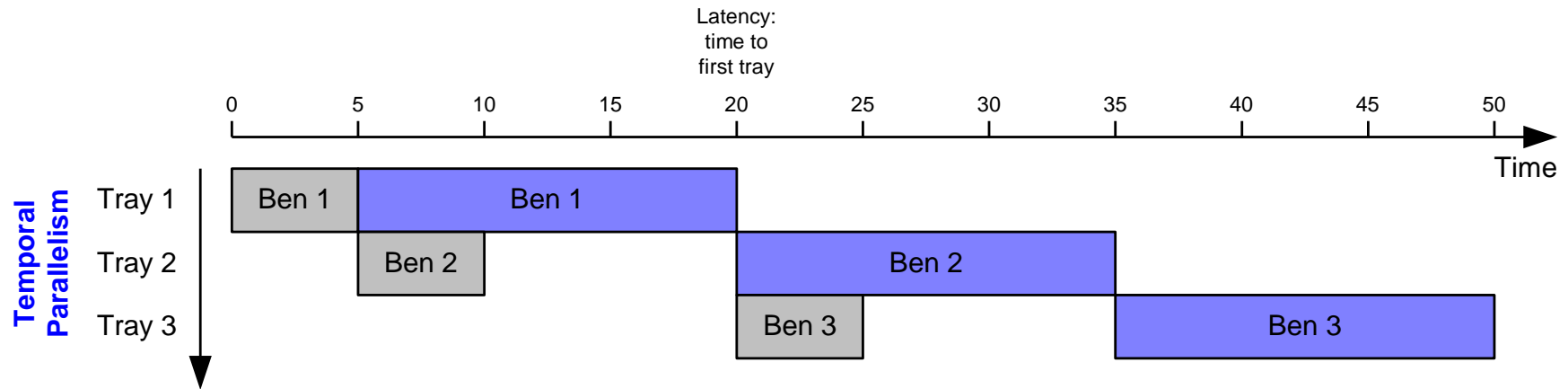
Latency = ?

Throughput = ?



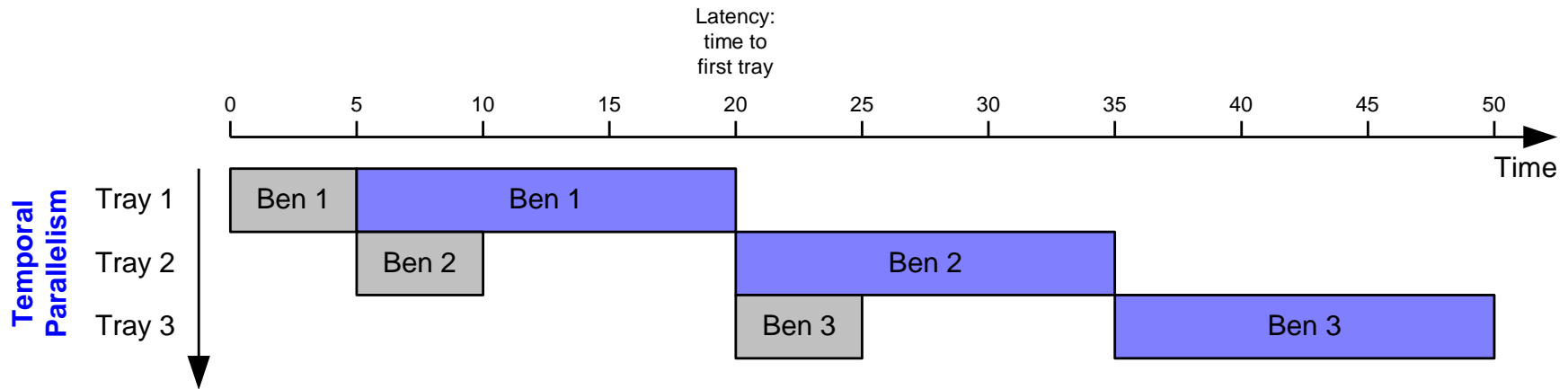
Latency = 5 + 15 = 20 minutes = **1/3 hour**

Throughput = 2 trays/ 1/3 hour = **6 trays/hour**



Latency = ?

Throughput = ?

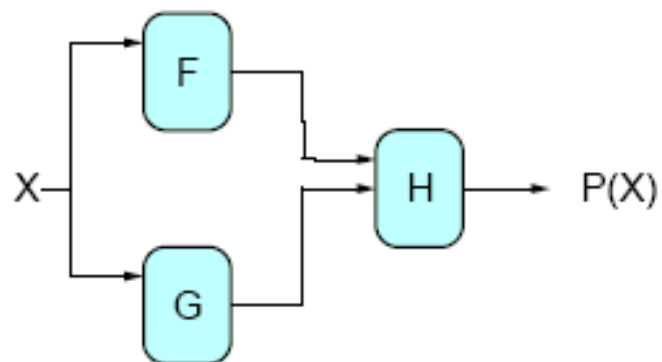


Latency = 20 minutes = **1/3 hour**

Throughput = 1 trays/ 1/4 hour = **4 trays/hour**

Using both techniques, the throughput would be **8 trays/hour**

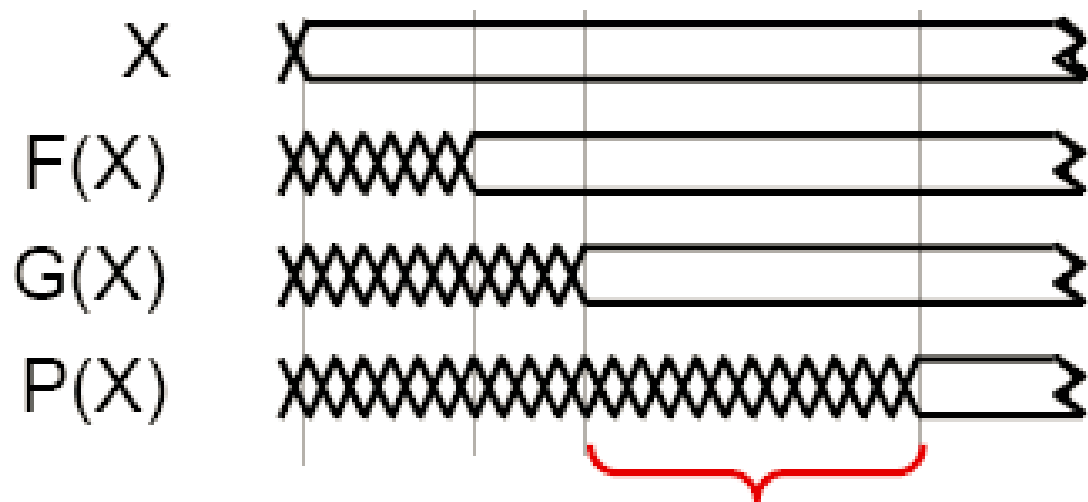
对于电路：



对于组合逻辑：

延时= t_{PD} ,

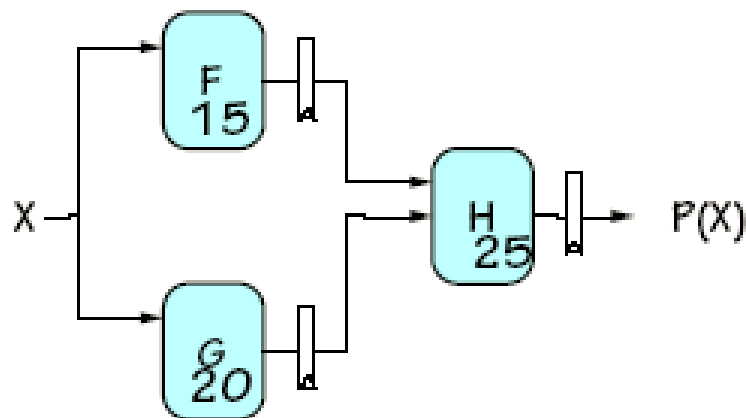
吞吐量= $1/t_{PD}$ 。



当 H 执行计算时， F 和 G 处于闲置状态，只是保持其输出不变。

流水线电路

插入寄存器



在H对 X_i 执行计算的过程中，F和G就可以工作在输入 X_{i+1} 上。

如果在时钟周期j时有一个有效的输入X，那么P(X)在时钟周期j+2将是有效的。

假定F、G、H的传播延迟分别为15、20、25 ns，我们使用的是理想的0延迟寄存器：

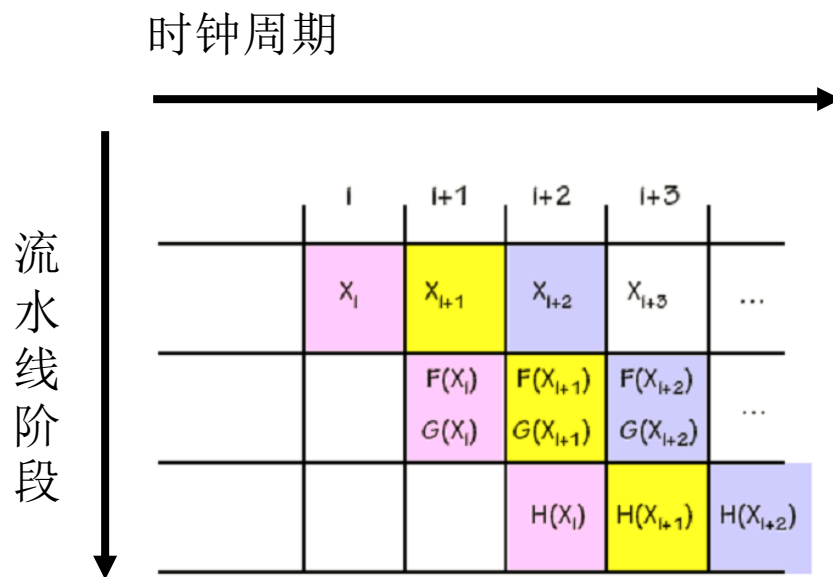
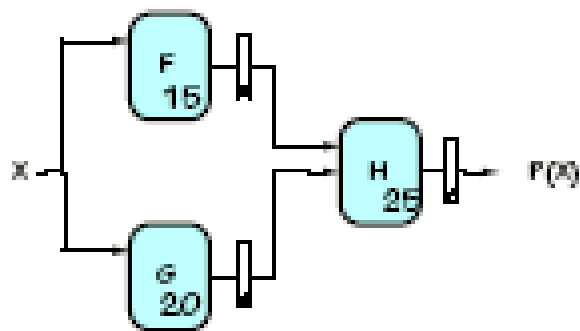
延时 吞吐量

非流水线： 45 1/45

2阶段流水线： 50 1/25

创建一个 2阶段流水线

流水线时空图



与某组特定输入数据相关的结果沿该图的对角线方向移动，每个时钟周期通过一个流水阶段。

流水线约定

定义：

K阶段流水线（“K流水线”）是一个非循环电路，对应于从输入到输出的每一条路径恰好有K个寄存器。

因此，组合电路是0阶段流水线。

约定：

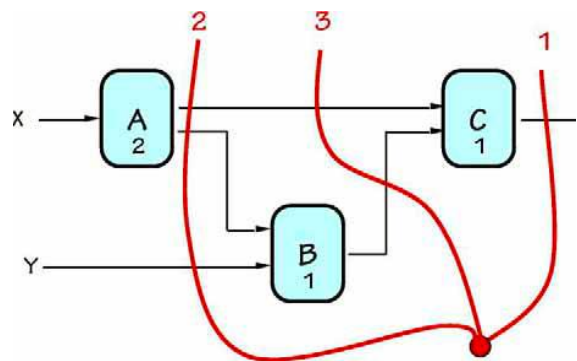
在K阶段流水线中，每个流水线阶段的输出（而不是输入）都有一个寄存器。

通常：

对于所有寄存器公用的时钟来说，必须有足够长的周期时间，包括：组合路径中的传播延迟 + （输入）寄存器的 t_{PD} 时间 + （输出）寄存器的 t_{SETUP} 时间。

K阶段流水线的延时是对所有寄存器公用的时钟周期的K倍。

流水线技术举例



观察：

- 1阶段流水线在延时或者吞吐量方面都没有改进。
- 通过对长的组合路径进行拆分来改进吞吐量，使得时钟速度变得更快；
- 流水线阶段太多只会增加延时开销，而不会改进吞吐量。
- 要想保持合适的流水线，通常需要紧紧连接在一起的许多寄存器。

	延时	吞吐量
0阶段流水线:	4	$1/4$
1阶段流水线:	4	$1/4$
2阶段流水线:	4	$1/2$
3阶段流水线:	6	$1/2$



流水线技术小结

优点：

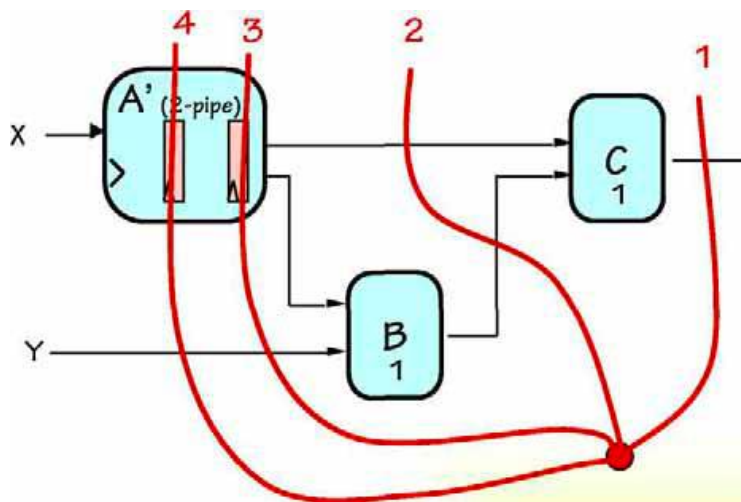
——通过拆分长的组合路径这种方法来增加吞吐量，从而增加了时钟频率。

缺点：

——可能会增加延时.....

——性能仅取决于最坏的连接：最慢的步骤制约着系统的吞吐量。

流水线部件



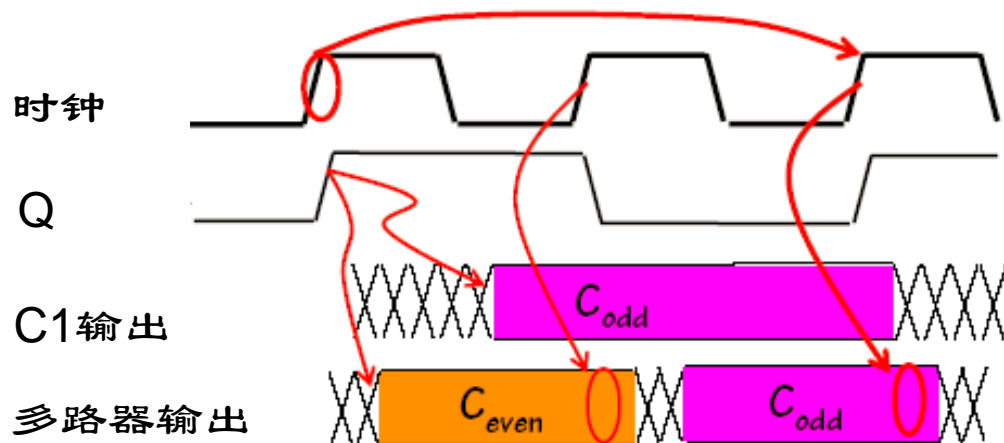
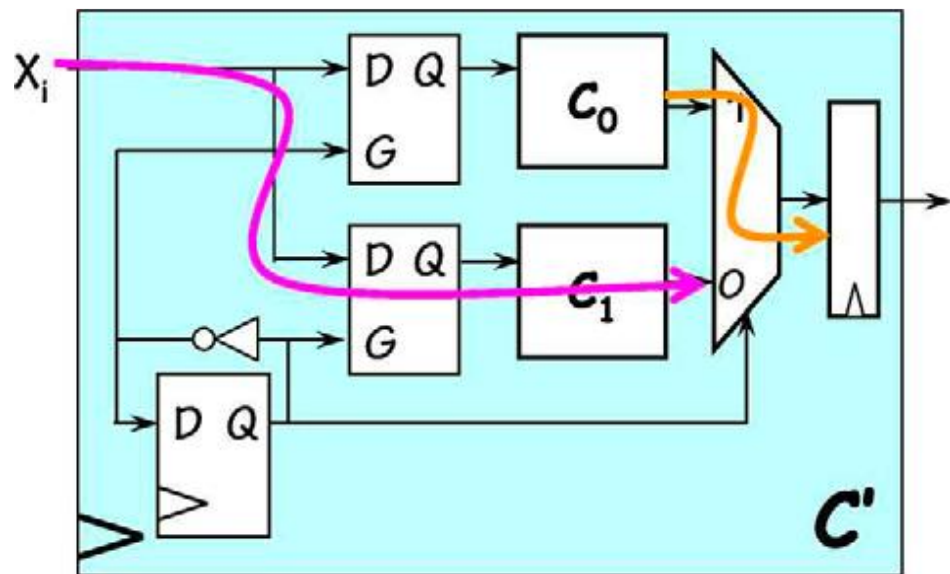
流水线系统是可以分层次的：

- 用一个K阶段流水线去替换某个较慢的组合部件，这样就可以增加时钟频率。
- 必须估计我们的计划中新流水线阶段情况。

4阶段流水线，吞吐量=1

空间并行

通过复制电路元件，并且交替改变不同副本之间的输入，就可以模拟一个较慢部件的流水线情况。



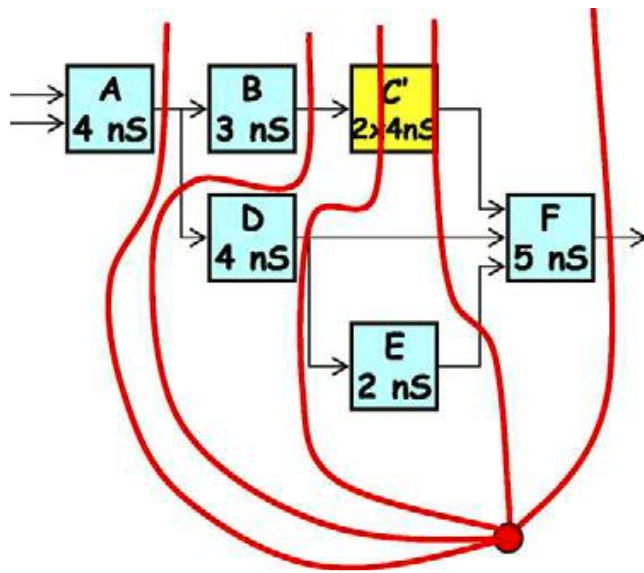
组合技术

---可以将空间并行与流水线技术结合起来

---C' 与带有8 ns传播延迟的两个C元件交织在一起。结果，C' 电路的吞吐量为4 ns分之一，延时为8 ns。可以将它看作是穿过C' 模块中间的附加流水级阶段。分隔线中必须有一条穿过该流水线阶段。

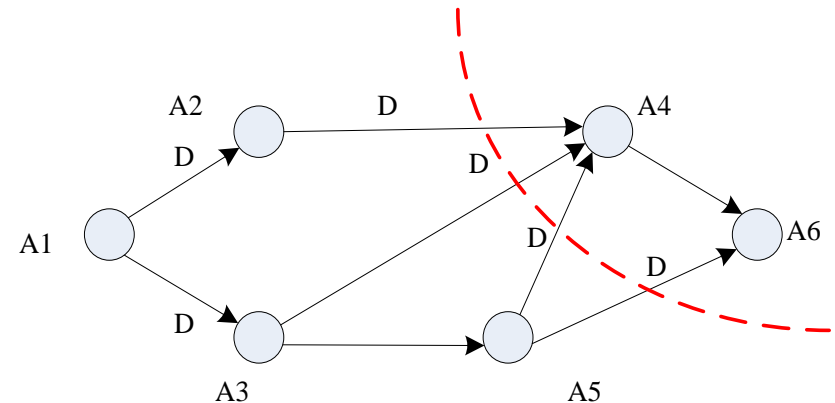
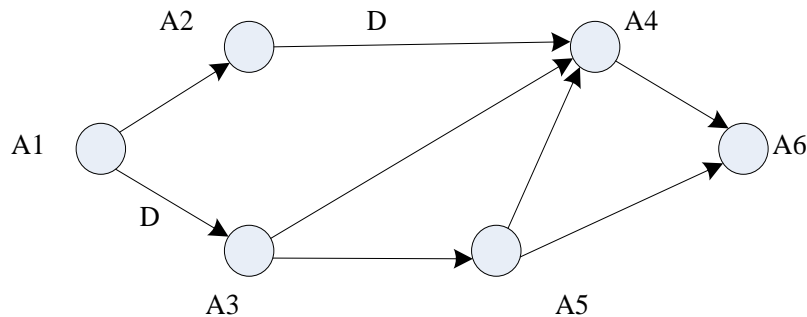
---通过将空间并行技术和流水线技术组合在一起，瓶颈从C元件转移到F元件。

---吞吐量从原来的 $1/8\text{ns}$ 减少到 $1/5\text{ ns}$,延迟为25ns



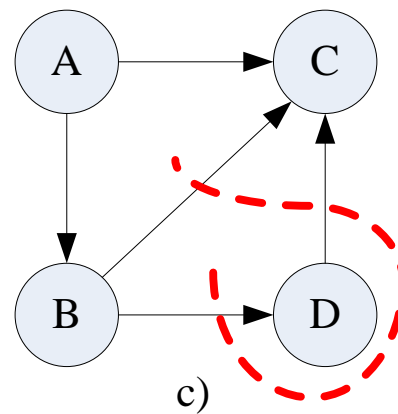
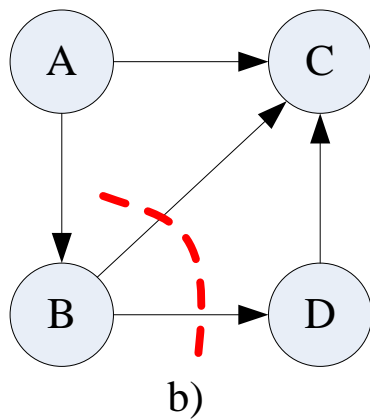
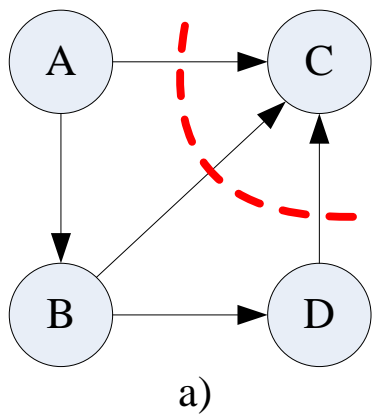
feed-forward cutset

- The pipelining registers/latches can only be placed across any *feed-forward cutset* of the graph.
 - Cutset: A cutset is a set of edges of a graph such that if these edges are removed from the graph, the graph becomes disjoint;
 - Feed-forward Cutset: A cutset is called a feed forward cutset if the data move in the forward direction on all the edges of the cutset.



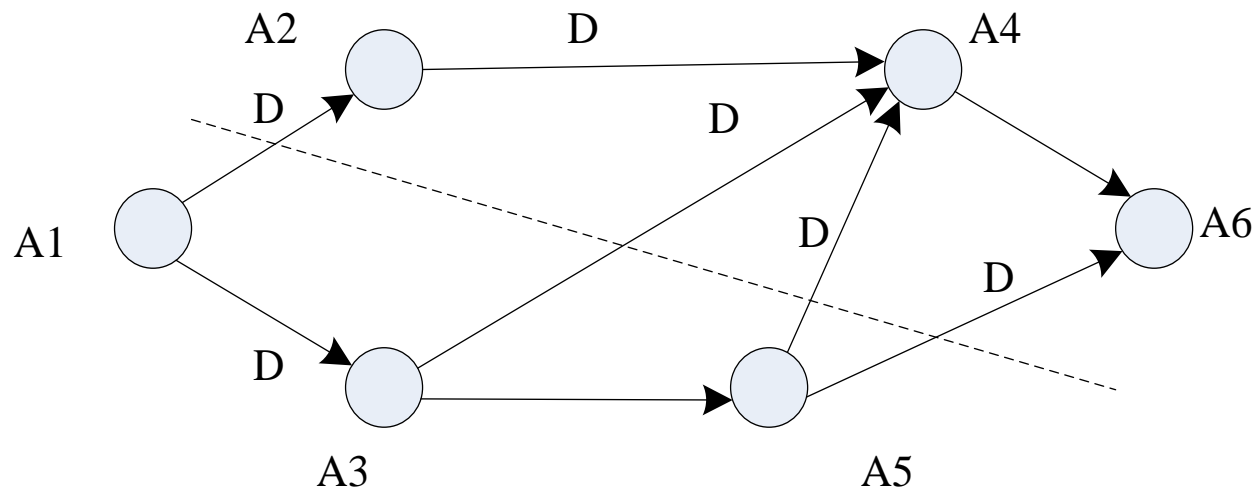
Question 1:

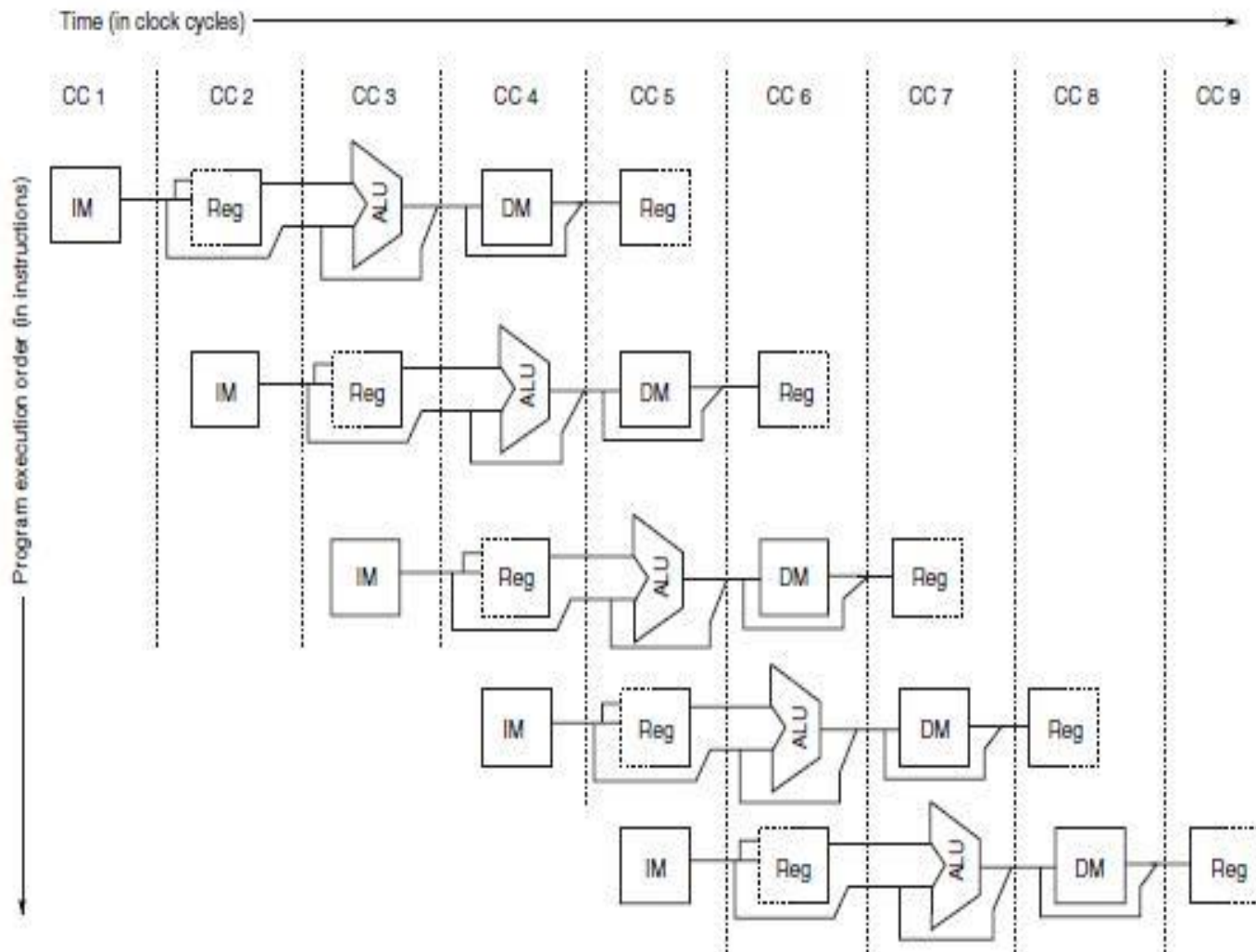
● 以下图形是割集吗，为什么？



feed-forward cutset

- 有效的割集
 - 不能改变电路功能
 - 应在关键路径上进行割集





小结

- 延时 (L , Latency) = 给定输入到达输出所需要的时间。
- 吞吐量 (T , Toughput) = 产生每个新输出的速率。
- 对于组合逻辑电路: L = 电路的 t_{PD} , $T = 1 / L$ 。
- K 阶段流水线 ($K > 0$) :
- 通常在输出端有寄存器;
- 从输入到输出的每个路径上有 K 个寄存器;
- 在时钟 i , 输入可用; 在时钟 $(i+K)$, 输出可用;
- $T = 1 / (t_{PD, REG} + \text{最慢流水线阶段的 } t_{PD} + t_{SETUP})$:
 - 更大的吞吐量 \rightarrow 将最慢的流水线阶段拆分;
 - 如果不能进一步拆分, 则可以使用复制/交织技术。
 - 流水线延时 \geq 组合延时。

流水线设计本质是通过增加系统的频率来提高吞吐量, 达到系统延迟和面积的折中