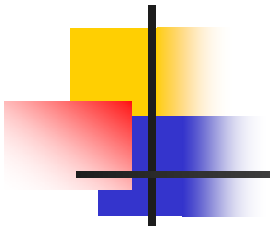


Chapter 5

Sequential Logic

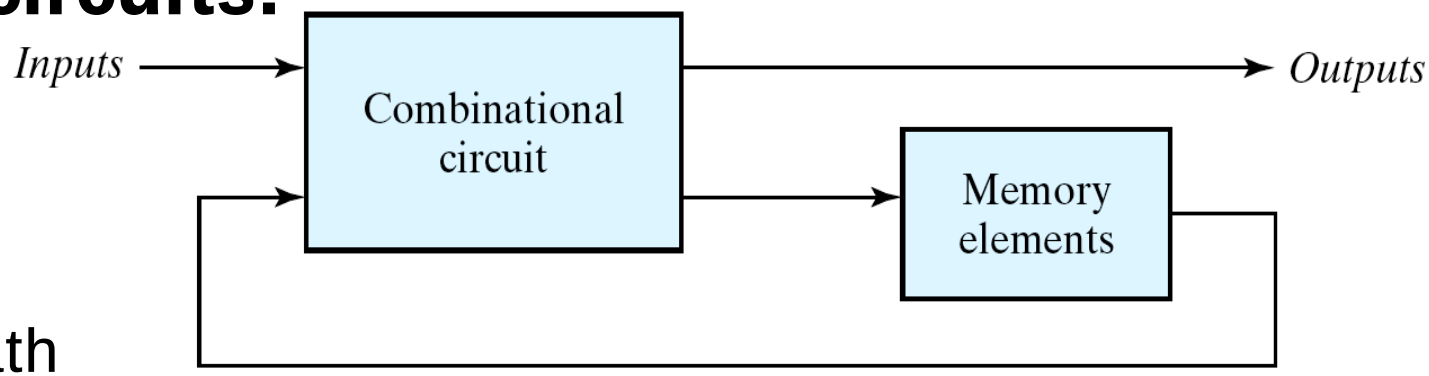
Sequential Circuits (1/2)

Combinational circuits:

a. contain no memory elements

b. the outputs depends on the current inputs

Sequential circuits:



- a feedback path
- outputs depends on present inputs and present states (pre. inputs)
- (inputs, current state) $\Rightarrow$ (outputs, next state)
- synchronous: the transition happens at discrete instants of time
- asynchronous: at any instant of time

Sequential Circuits (2/2)

- A sequential circuit is specified by a time sequence of inputs, outputs and internal states

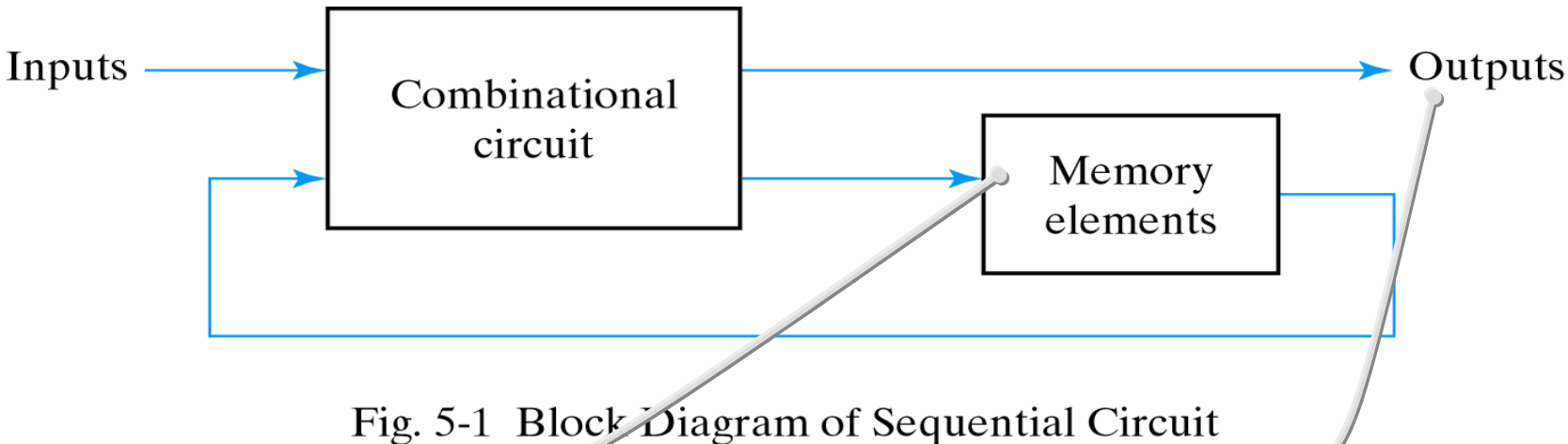


Fig. 5-1 Block Diagram of Sequential Circuit

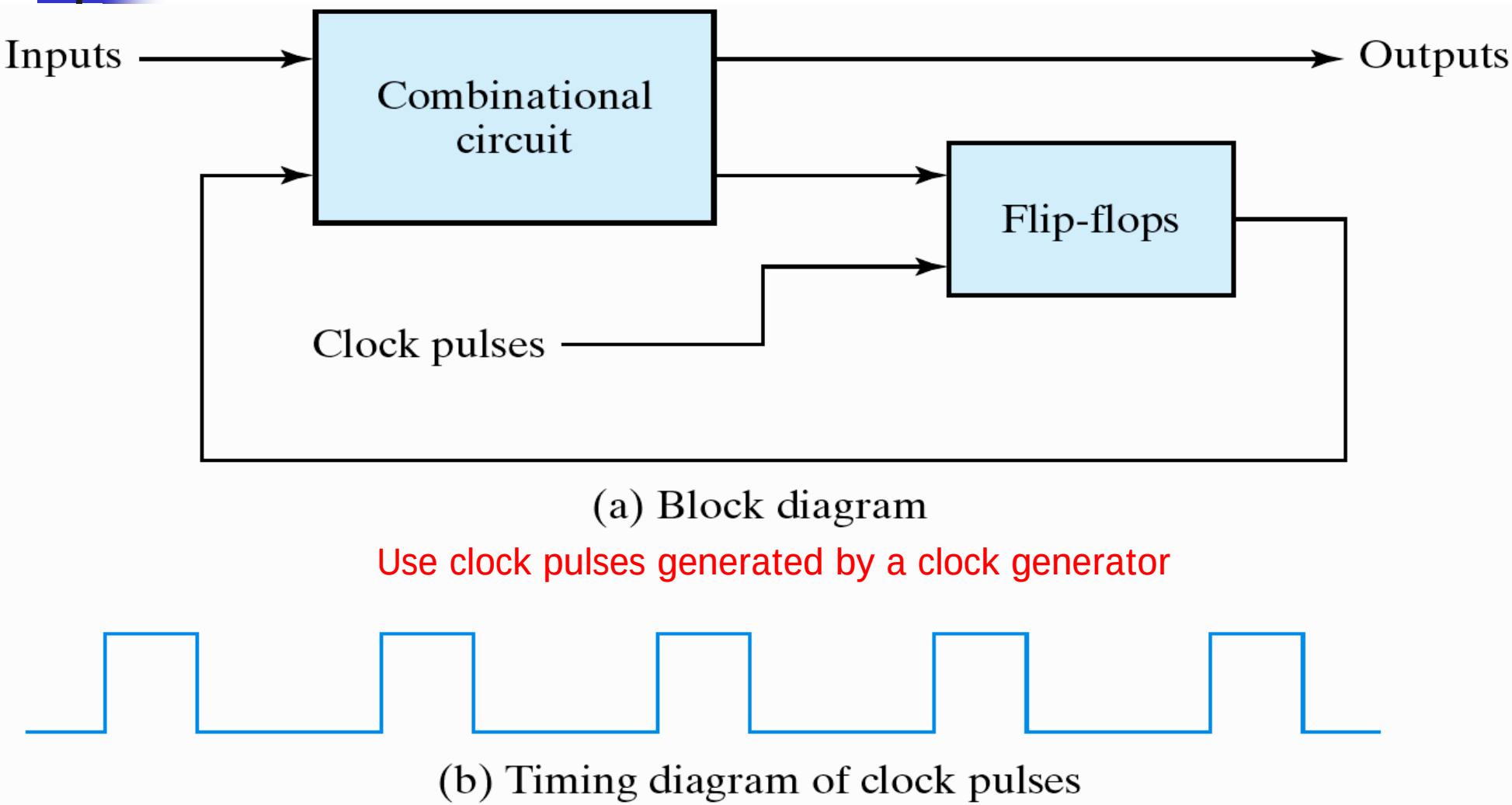
Sequential circuits must be able to remember the past history

Flip-flops: most commonly used memory devices

A function of

- present inputs &
- present state of memory elements (the past sequence of inputs)

Clock



(a) Block diagram

Use clock pulses generated by a clock generator

(b) Timing diagram of clock pulses

Fig. 5.2
Synchronous clocked sequential circuit

Types of Sequential Circuits

depending on the timing of their signals

- Synchronous (同步) sequential circuits
  - Storage elements are affected at discrete time instants
    - Use clock pulses in the inputs of storage elements
- Asynchronous (非同步) sequential circuits
  - Storage elements are affected at any time instant

Synchronous Sequential Circuits (1/2)

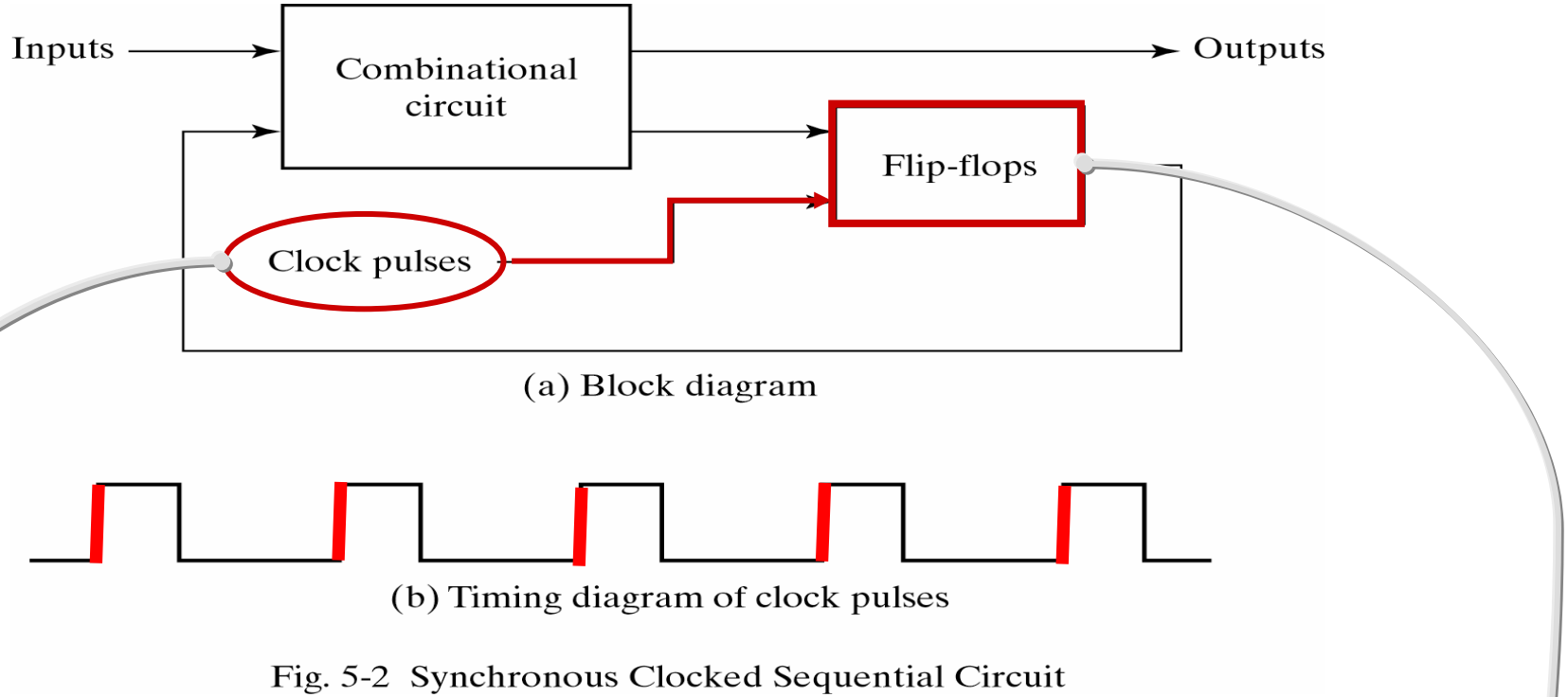


Fig. 5-2 Synchronous Clocked Sequential Circuit

Synchronous

- *Use clock pulses in the inputs of storage elements*

Storage elements

- *are affected only with the arrival of each pulse*
- *The storage elements used in the clocked sequential circuits are called "flip-flops"*

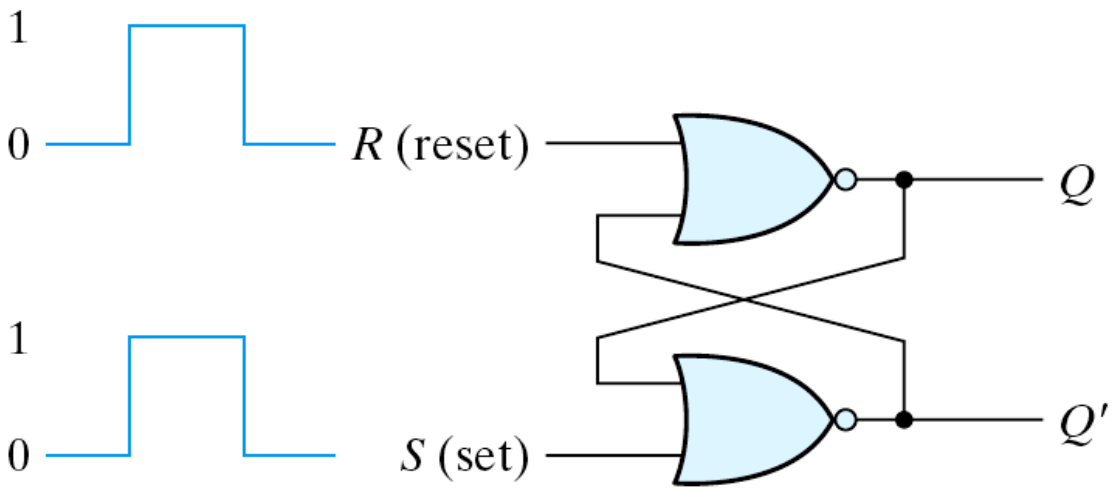
Synchronous Sequential Circuits (2/2)

- Synchronous sequential circuits
  - a master-clock generator to generate a periodic train of clock pulses
  - the clock pulses are distributed throughout the system
  - clocked sequential circuits (most popular)
  - no instability problems
  - the memory elements: flip-flops
    - binary cells capable of storing one bit of information
    - two outputs: one for the normal value and one for the complement value
    - maintain a binary state indefinitely until directed by an input signal to switch states

Latches (1/3)

- The most basic types of flip-flops operate with signal levels ➔latch
- All FFs are constructed from the latches introduced here

  A FF can maintain a binary state indefinitely until directed by an input signal to switch states



(a) Logic diagram

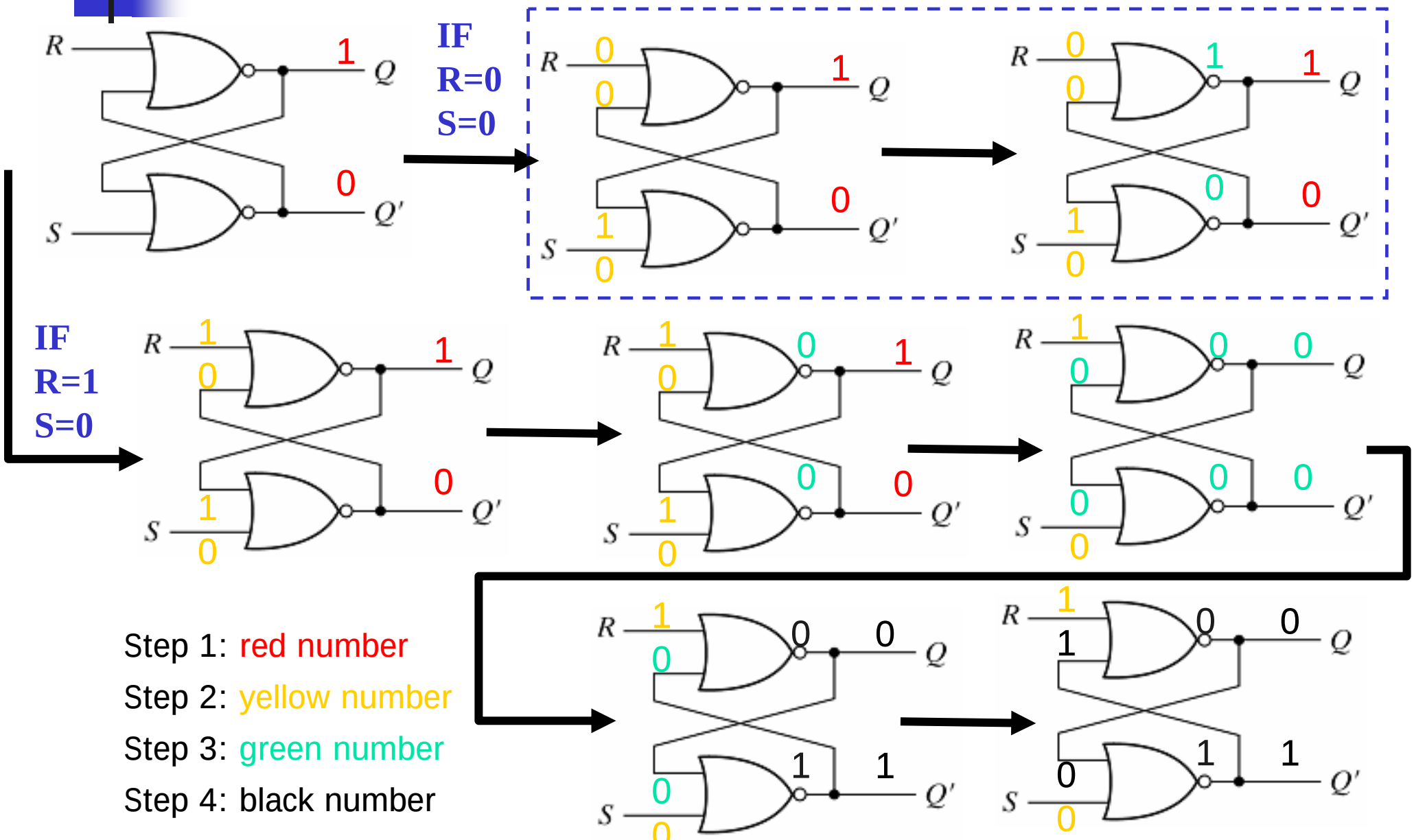
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

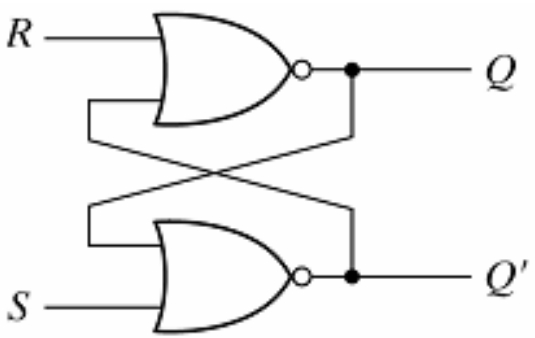
Two NOR gates

Set➔1, Reset➔0

Latches (2/3)


R
S
Q
Q'
IF
R=0
S=0
IF
R=1
S=0
Step 1: red number
Step 2: yellow number
Step 3: green number
Step 4: black number

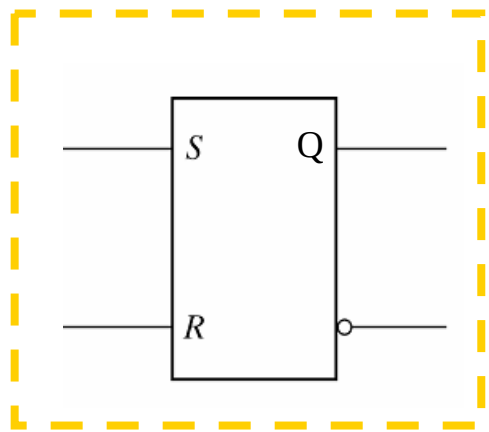
Latches (3/3)



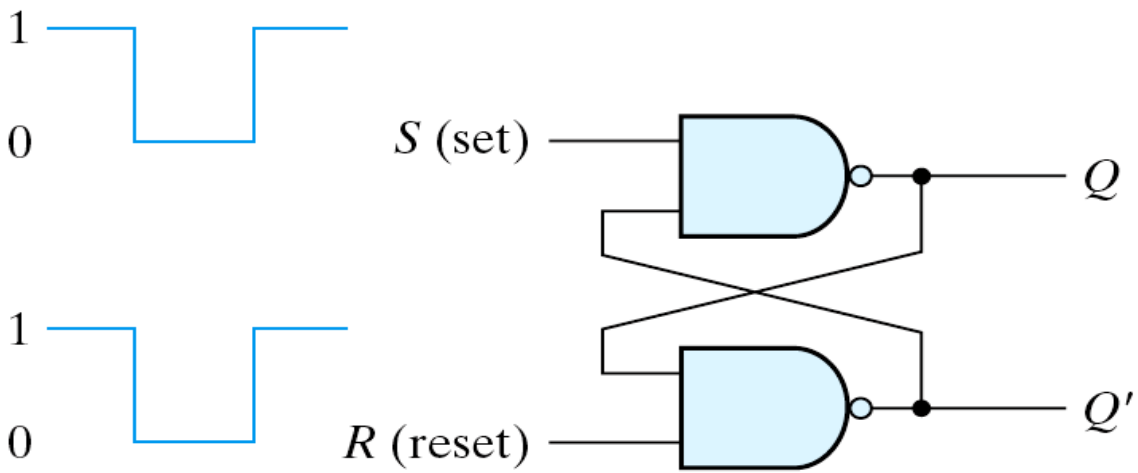
S	R	P=Q'	Q	
0	0	*	*	// a stable state in the previous state
1	0	0	1	// change to another stable state "Set"
0	0	0	1	// remain in the previous state
0	1	1	0	// change to another stable state "Reset"
0	0	1	0	// remain in the previous state
1	1	0	0	// oscillate (unpredictable) if next SR=00

- more complicated types can be built upon it
- an asynchronous sequential circuit
- (S,R)= (0,0): no operation
  (S,R)=(0,1): reset (Q=0, the clear state)
  (S,R)=(1,0): set (Q=1, the set state)
  (S,R)=(1,1): indeterminate state (Q=Q'=0)
- consider (S,R) = (1,1) ⇒ (0,0)

the condition should be avoided



SR Latch with NAND gates

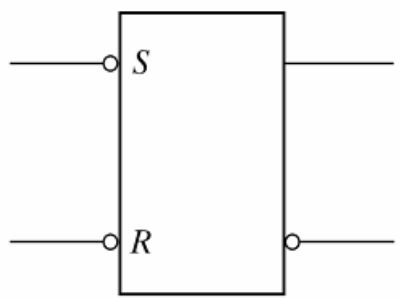


(a) Logic diagram

S	$\overline{R}$	Q	Q'	
1	0	0	1	reset
1	1	0	1	set (after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

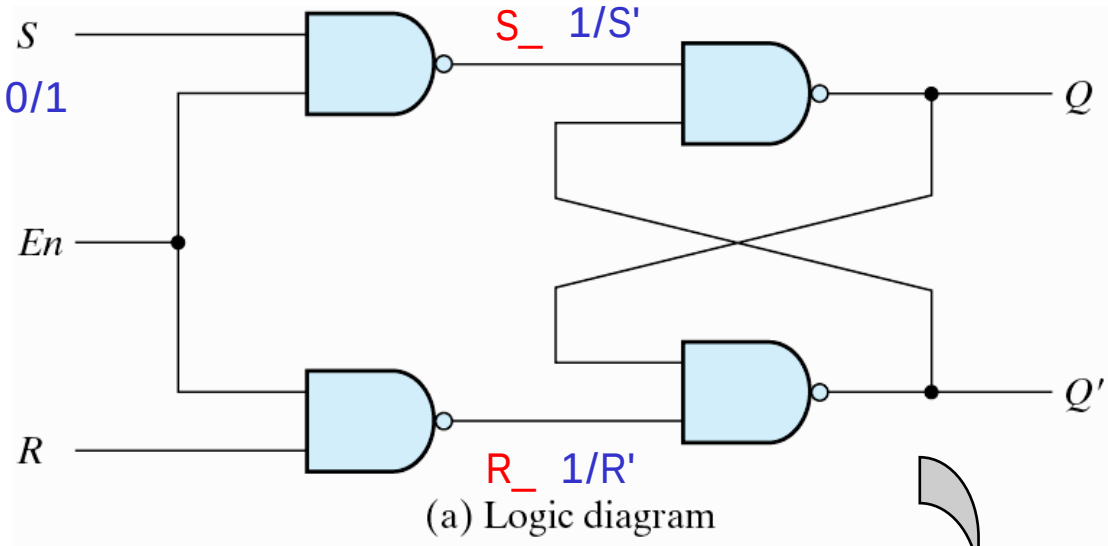
S'R' latch



(c) Graphic symbol

SR Latch with Control Input

The complement output of the previous R'S' latch.



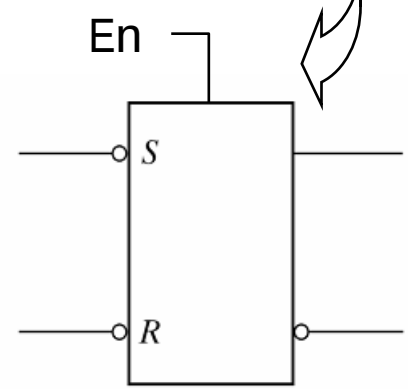
(a) Logic diagram

En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(b) Function table

En=0, no change

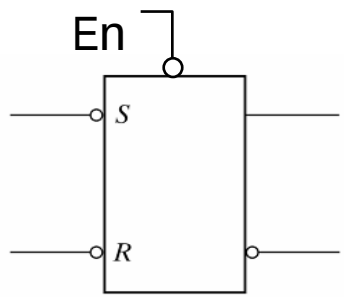
En=1, enable



(c) Graphic symbol

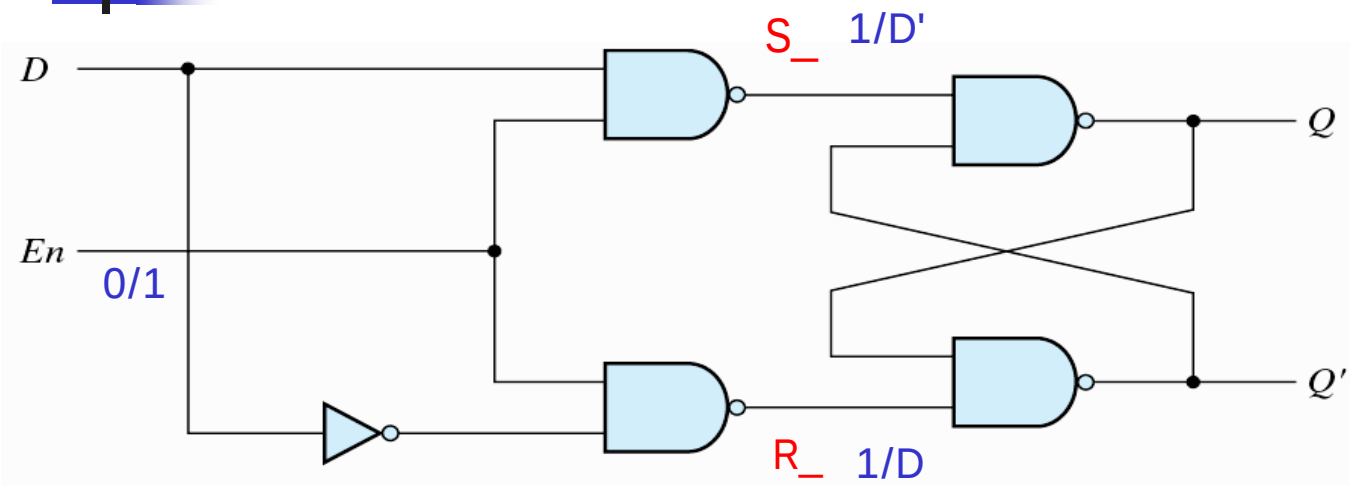
En=1, no change

En=0, enable



(c) Graphic symbol

D Latch (Transparent Latch)

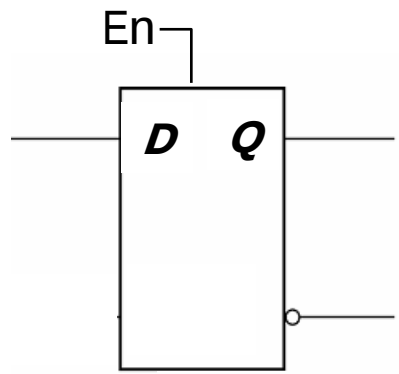


(a) Logic diagram

En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

(b) Function table

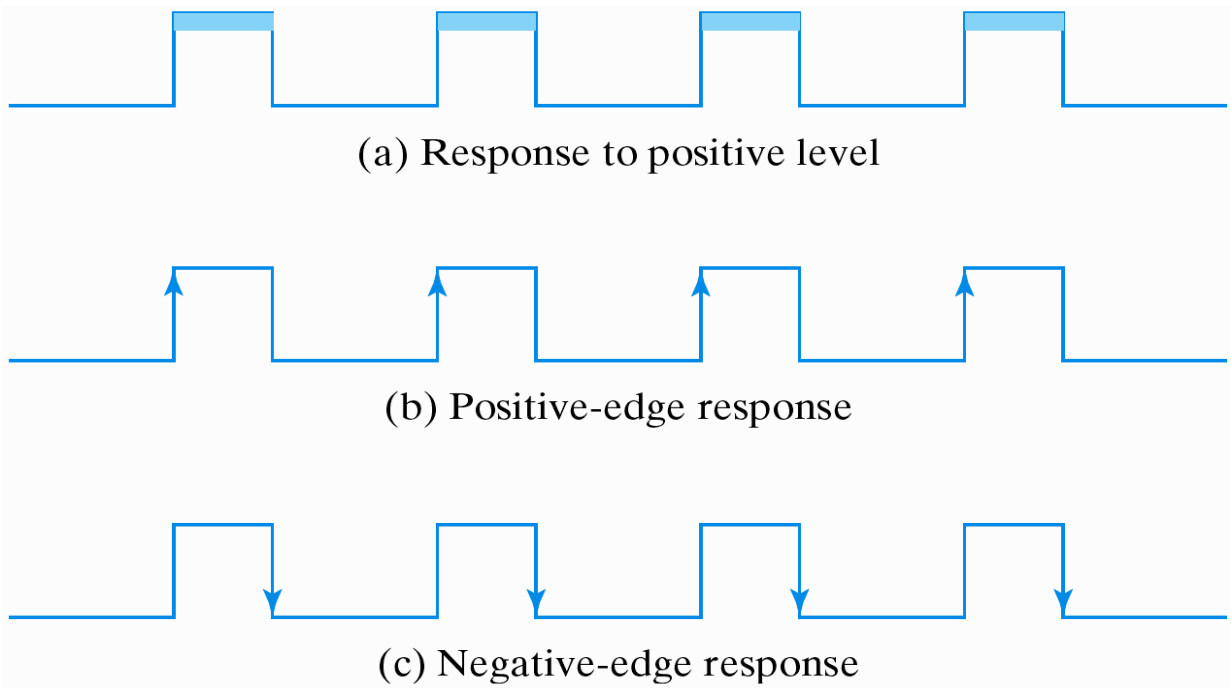
- eliminate the undesirable conditions of the indeterminate state in the RS flip-flop
- D: data
- gated D-latch
- D $\Rightarrow$ Q when En=1; no change when En=0



level triggered
(level-sensitive)

Flip-Flops

- A trigger
  - The state of a latch or flip-flop is switched by a change of the control input
- Level triggered – latches
- Edge triggered – flip-flops

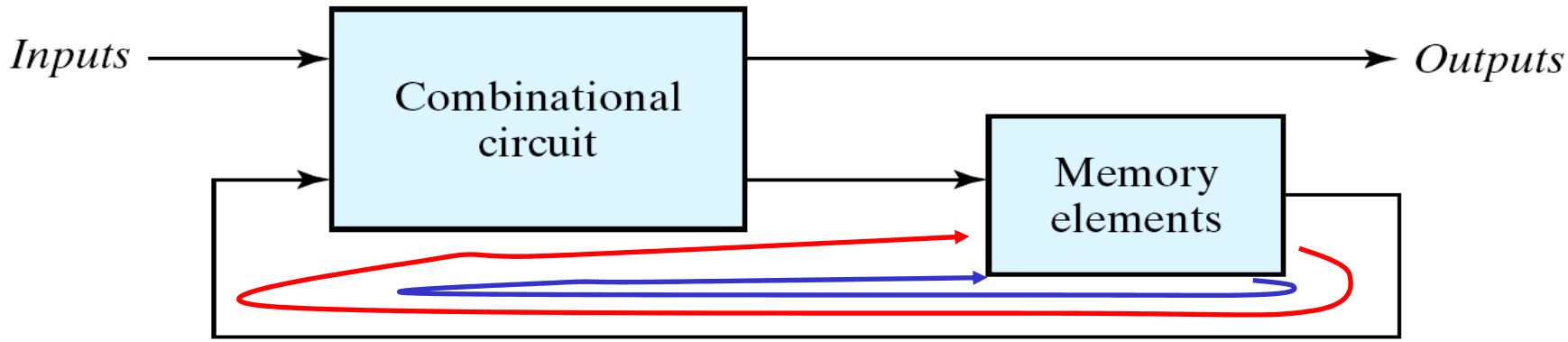


Level triggered

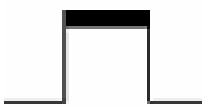
Edge triggered

Edge triggered

Problem of Latch



- If level-triggered flip-flops are used
  - the feedback path may cause instability problem (since the time interval of logic-1 is too long)
  - multiple transitions might happen during logic-1 level



- Edge-triggered flip-flops
  - the state transition happens only at the edge
  - eliminate the multiple-transition problem

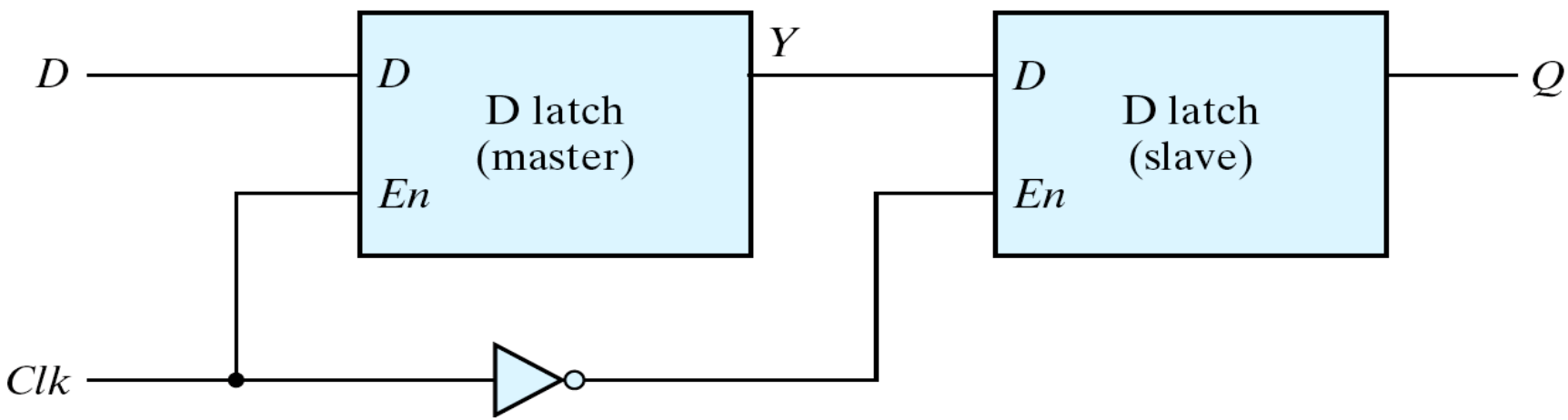
Edge-Triggered D Flip-Flop

Two designs to solve the problem of latch:

a. Master-slave D flip-flop

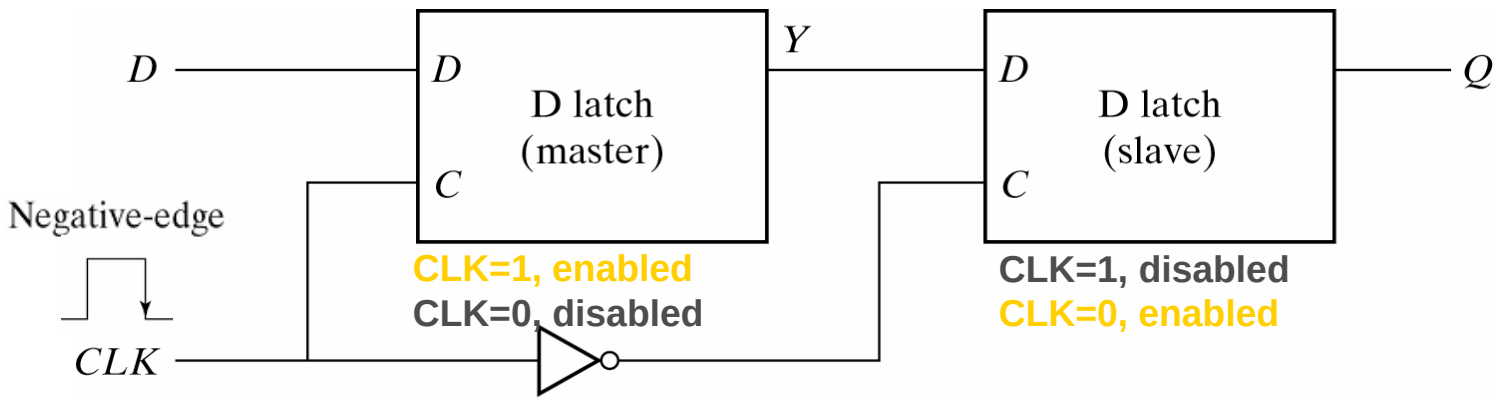
b. Edge-trigger D flip-Flop

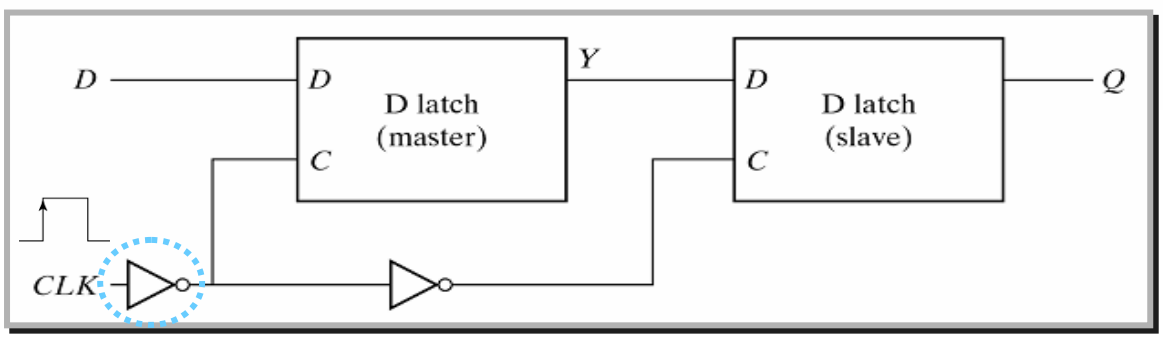
- Master-slave D flip-flop
  - two separate flip-flops
  - a master latch (positive-level triggered)
  - a slave latch (negative-level triggered)



Master-slave D flip-flop (1/2)

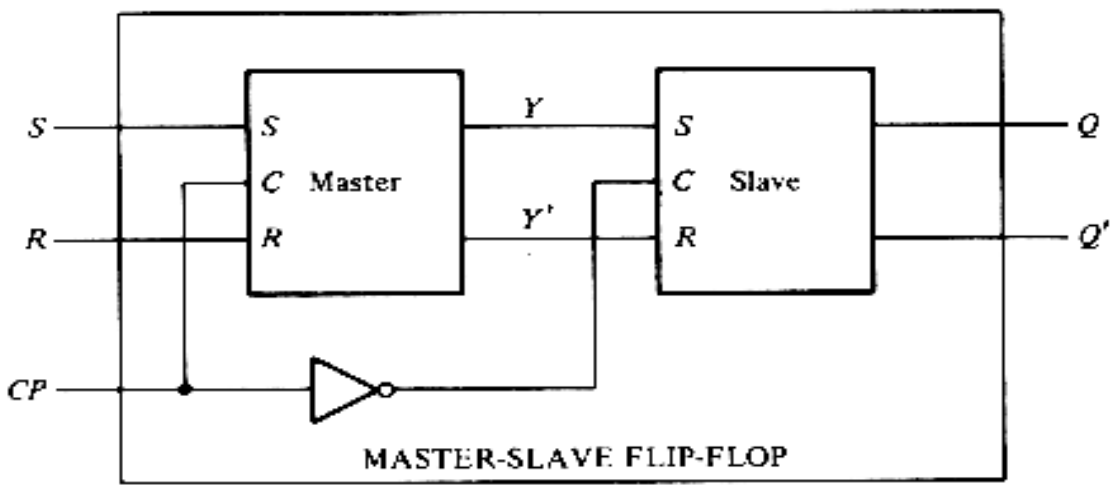
- Two D latches and one inverter
- The circuit samples D input and changes its output Q only at the negative-edge of CLK
- isolate the output of FF from being affected while its input is changing

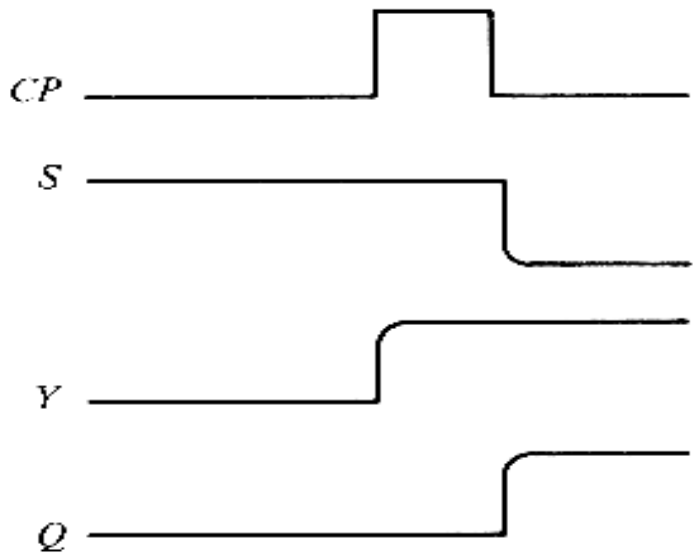




Master-slave D flip-flop (2/2)

- CP = 1: (S,R) ⇒ (Y,Y'); (Q,Q') holds
- CP = 0: (Y,Y') holds; (Y,Y') ⇒ (Q,Q')
- (S,R) could not affect (Q,Q') directly
- the state changes coincide with the negative-edge transition of CP

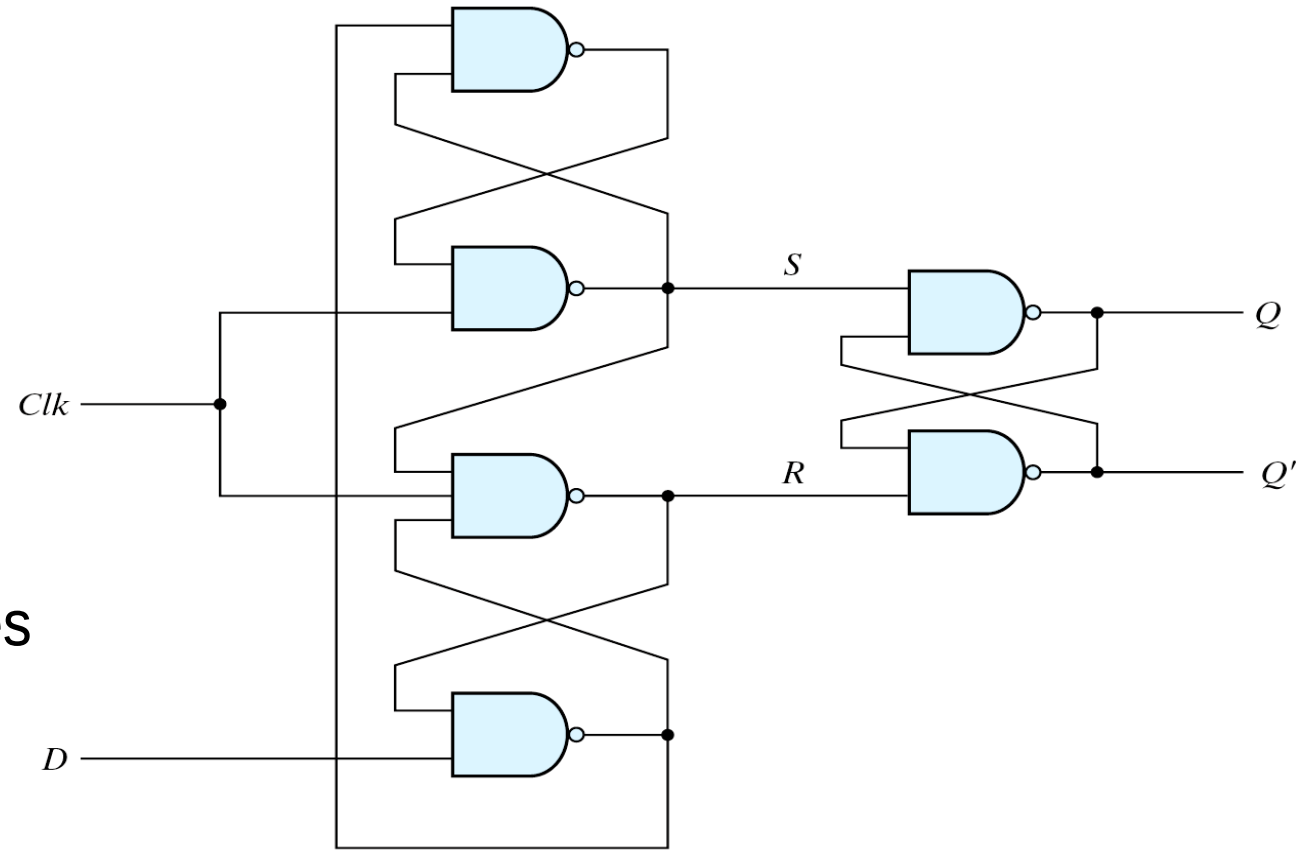




Edge-Triggered Flip-Flops (1/2)

the state changes during a clock-pulse transition

A D-type positive-edge-triggered flip-flop

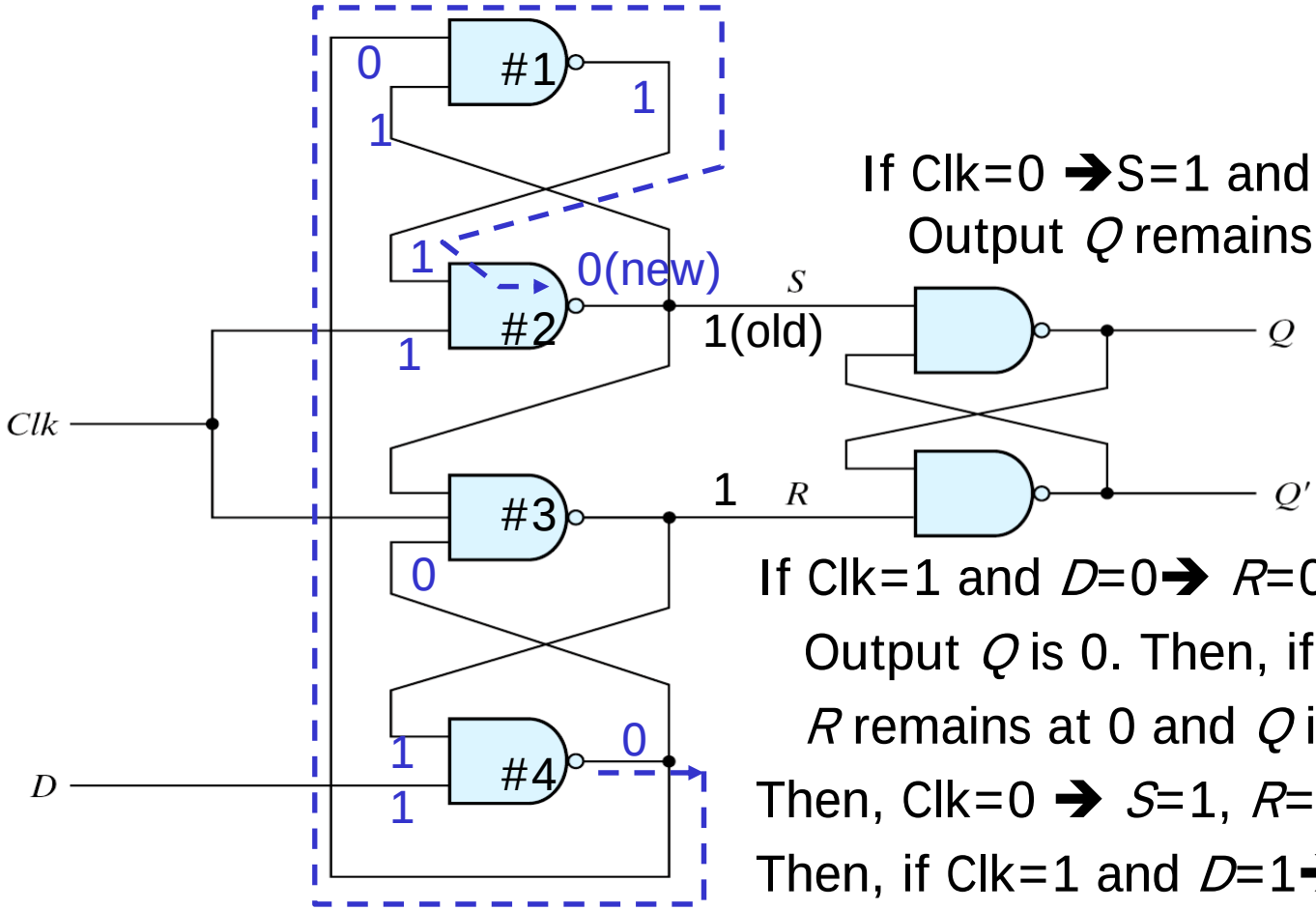


Three SR latches

Edge-Triggered Flip-Flops (2/2)

(S,R) = (0,1): Q = 1 (S,R) = (1,0): Q = 0

(S,R) = (1,1): no operation (S,R) = (0,0): should be avoided



If Clk=0 ➔S=1 and R=1 ➔ no operation.
Output *Q* remains in the present state.

If Clk=1 and *D*=0➔ *R*=0 ➔ Reset.
Output *Q* is 0. Then, if *D* changes to 1,
R remains at 0 and *Q* is 0.

Then, Clk=0 ➔ *S*=1, *R*=1 ➔ no operation (Q=0)

Then, if Clk=1 and *D*=1➔ *S*=0➔ Set.
Output *Q* is 1. (see the blue dot-line flow)

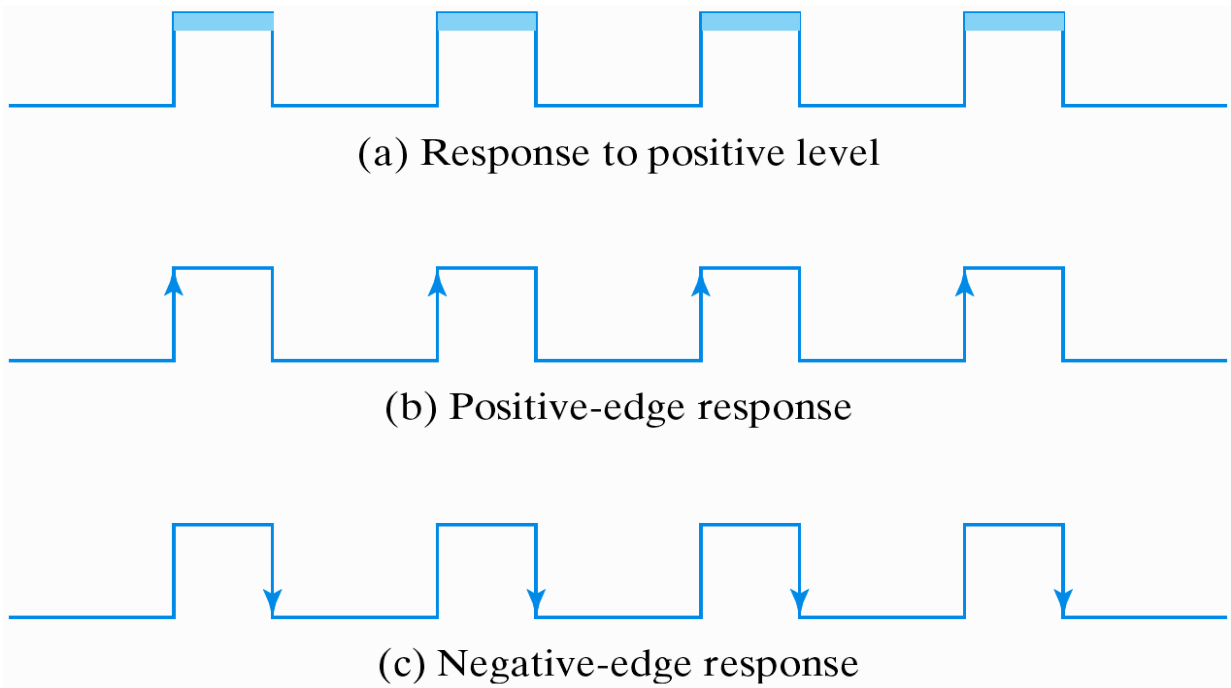
Then, if *D* changes to 0, *S* remains at 0 and *Q*=1;

Positive-Edge-Triggered Flip-Flops

- Summary
  - Clk=0: (S,R) = (1,1), no state change
  - Clk=↑: state change once
  - Clk=1: state holds
  - eliminate the feedback problems in sequential circuits
- All flip-flops must make their transition at the same time

Flip-Flops

- A trigger
  - The state of a latch or flip-flop is switched by a change of the control input
- Level triggered – latches
- Edge triggered – flip-flops



Level triggered

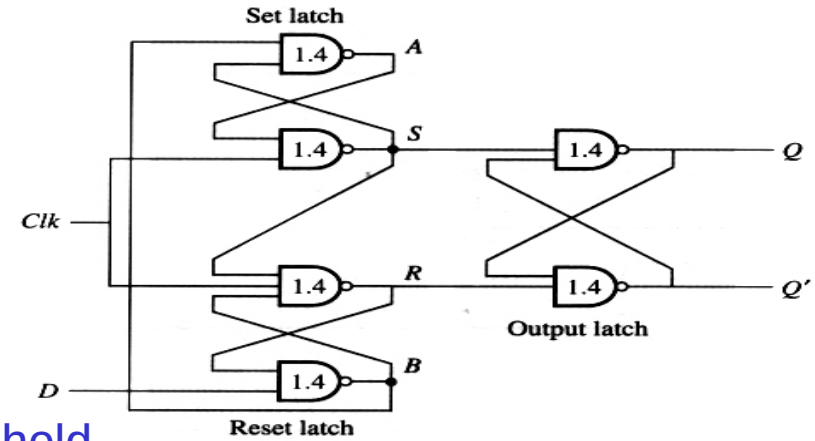
Edge triggered

Edge triggered

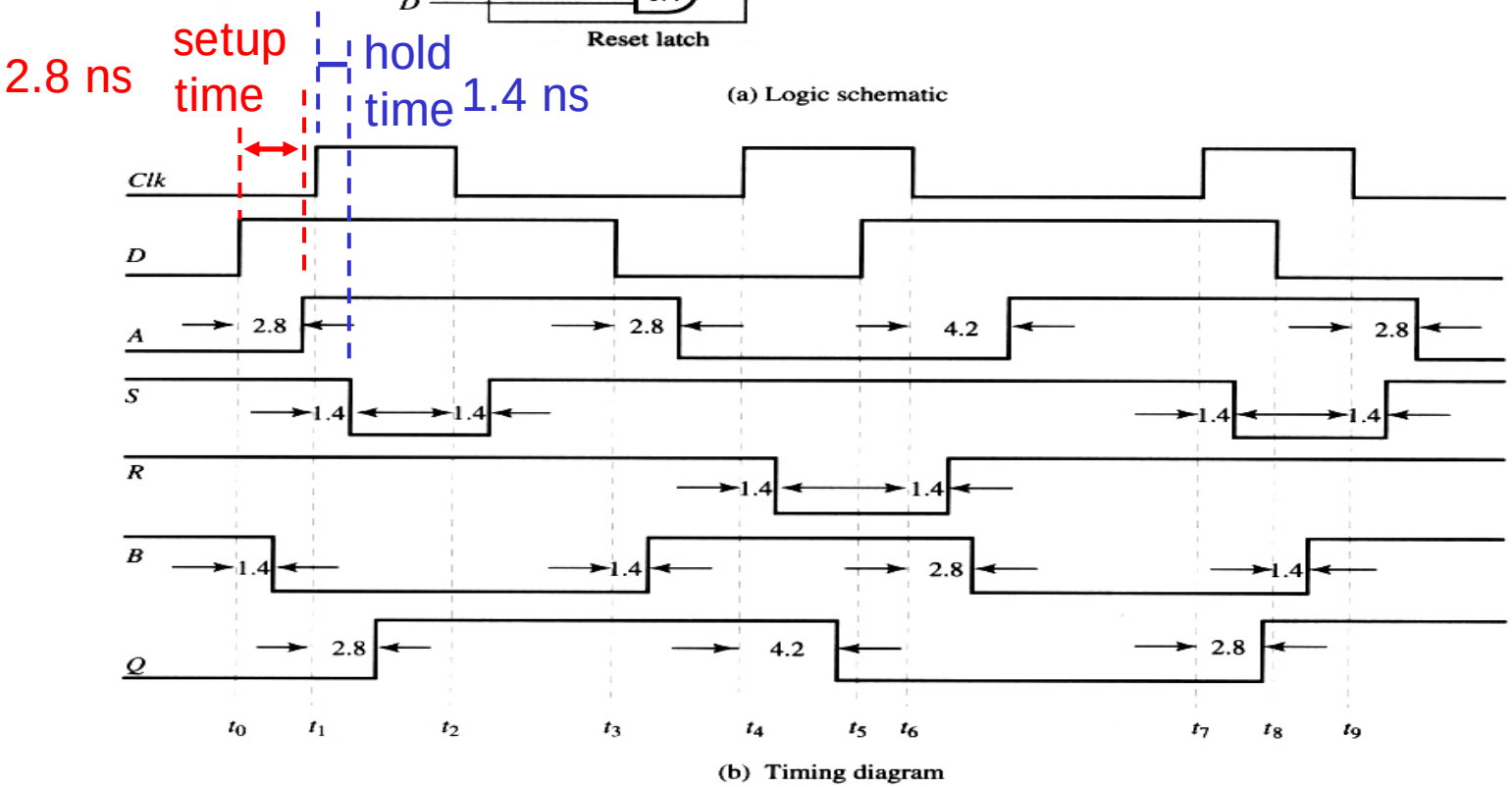
Setup Time and Hold Time

- The setup time
  - D input must be maintained at a constant value prior to the application of the positive Clk pulse
  - = the propagation delay through gates 4 and 1
  - data to the internal latches
- The hold time
  - D input must not changes after the application of the positive Clk pulse
  - = the propagation delay of gate 3 (try to understand)
  - clock to the internal latch

Timing Diagram



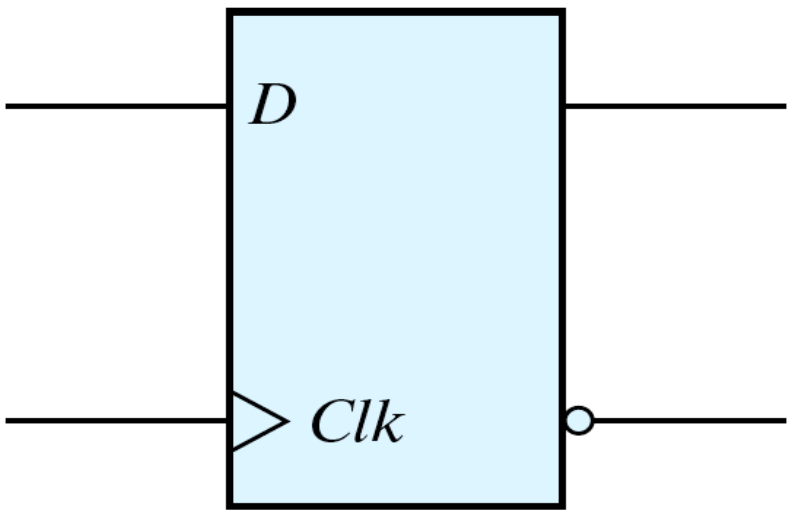
(a) Logic schematic



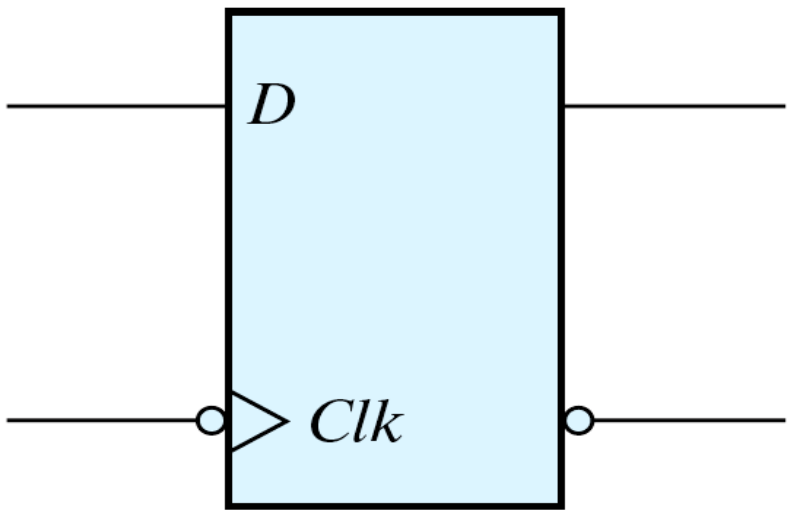
(b) Timing diagram

Positive-Edge vs. Negative-Edge

- The edge-triggered D flip-flops
  - The most economical and efficient
  - The most popular flip-flop
  - Positive-edge and negative-edge



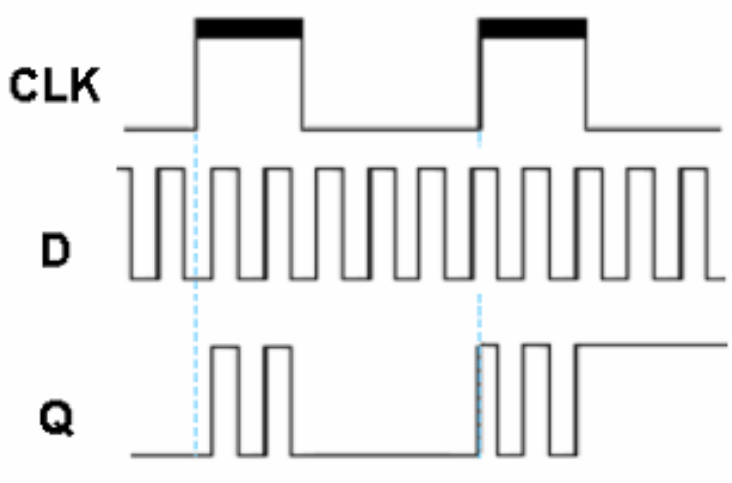
(a) Positive-edge



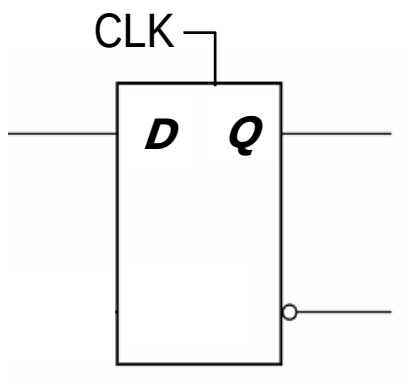
(a) Negative-edge

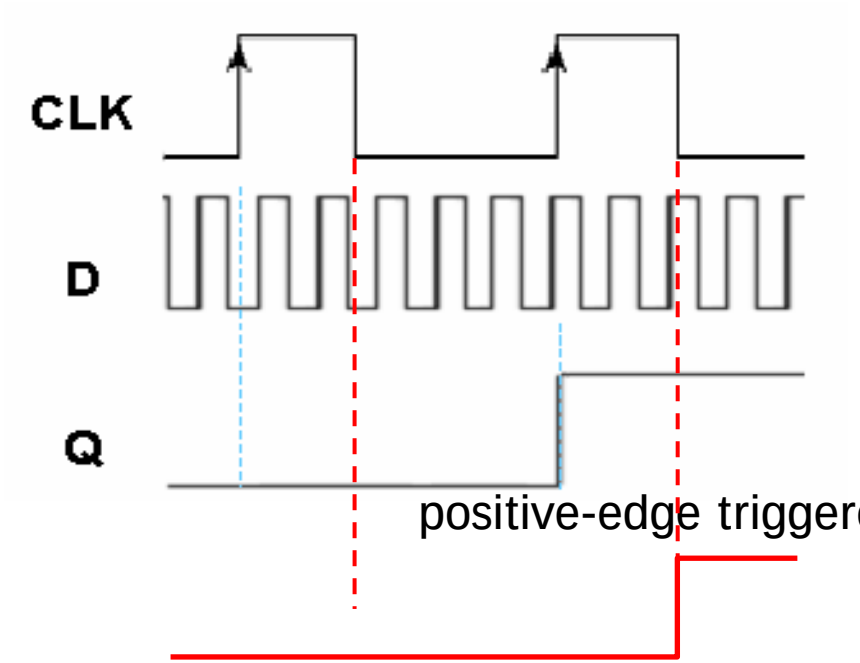
Latch vs. Flip-Flop

- Level triggered



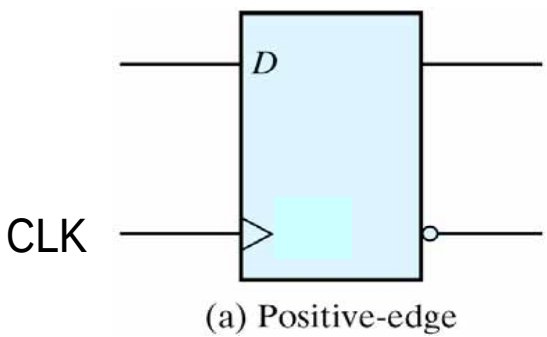
Latch



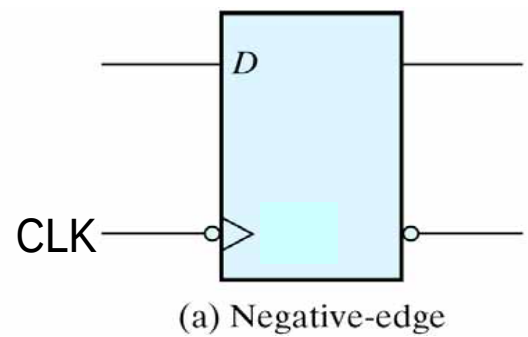


positive-edge triggered

negative-edge triggered

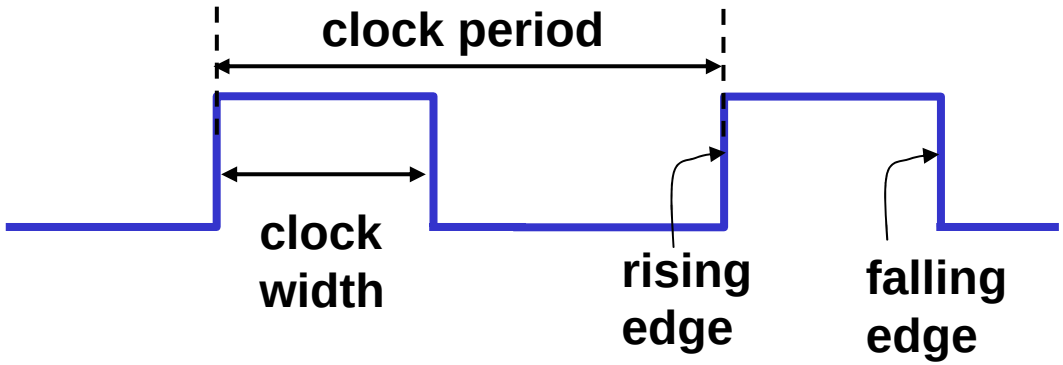


(a) Positive-edge



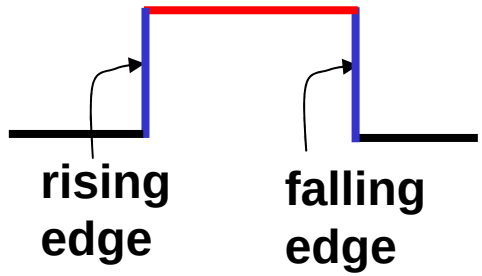
(a) Negative-edge

Clock Period



- Clock period (measured in micro or nanoseconds) is the time between successive transitions in the same direction
- Clock frequency (measured in MHz or GHz) is the reciprocal of clock period
- Clock width is the time interval during which clock is equal to 1
- Duty cycle is the ratio of the clock width and clock period
- Clock signal is active high if the changes occur at the rising edge or during the clock width. Otherwise, it is active low

Latch and Flip-Flop



Latches are level-sensitive since they respond to input changes during clock width. ⇨ Latches are difficult to work with for this reason.

Flip-Flops respond to input changes only during the change in clock signal (the rising edge or the falling edge).

They are easy to work with though more expensive than latches.

Two basic styles of flip-flops are available:

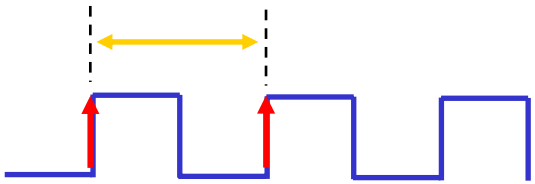
(1) master-slave (2) edge-triggered

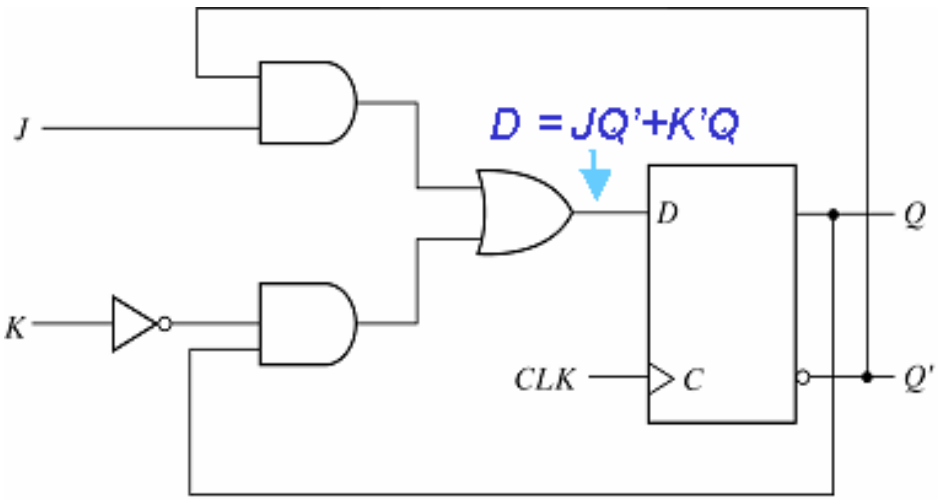
JK Flip-Flop

(*clear=1)

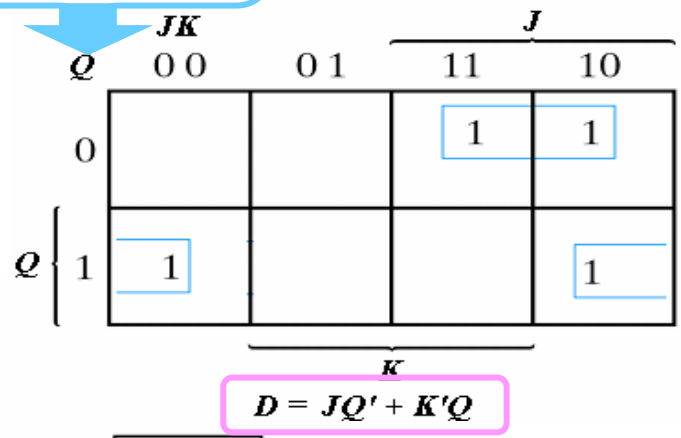
Inputs	J	K	D	Q(t+1)	Function
J, K disabled	0	0	Q(t)	Q(t)	no change
K enabled	0	1	0	0	reset FF to 0
J enabled	1	0	1	1	set FF to 1
J, K enabled	1	1	Q'(t)	Q'(t)	complement output

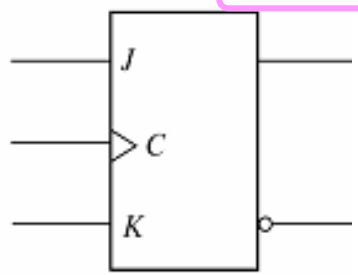
All operations must be finished in the interval





(a) Circuit diagram





(b) Graphic symbol

positive-edge

D flip-flop + external logic

Fig. 5-12 *JK* Flip-Flop

T(Toggle) Flip-Flop

Characteristic Table

T	Q(t+1)	
0	Q(t)	no change
1	Q'(t)	complement

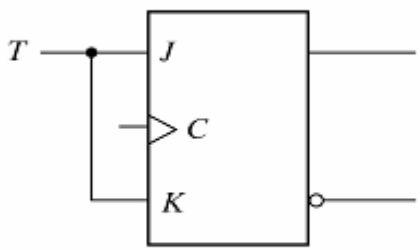
"*Complementing FF*"

- *T=**1**: a clock edge complements the output*
- *useful for designing binary counters*

(a) based on JK FF

J	K	Q(t+1)	
0	**0**	**Q(t)**	**no change**
0	1	0	reset
1	0	1	set
1	**1**	**Q'(t)**	**complement**

- tie J,K together



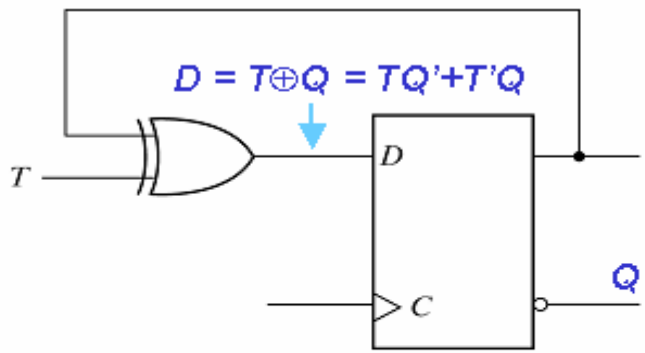
(a) From *JK* flip-flop

(b) based on D FF

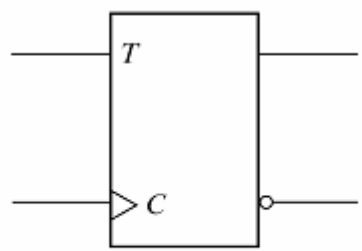
$D = T \oplus Q = TQ' + T'Q$

T	D	Q(t+1)	
0	Q	Q(t)	no change
1	Q'	Q'(t)	complement

Q \ T	0	1
0		1
1	1	



(b) From *D* flip-flop



(c) Graphic symbol

Fig. 5-13 T Flip-Flop

Characteristic Equations/Tables of FFs

- Characteristic Equations
  - define next state Q(t+1) as a function of inputs and present state algebraically
- Characteristic Tables
  - define next state Q(t+1) as a function of inputs and present state Q(t) in tabular form

Table 5.1
Flip-Flop Characteristic Tables

***JK* Flip-Flop**

$Q(t+1) = JQ' + K'Q$

J	*K*	*Q*(*t* + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

***D* Flip-Flop**

$Q(t+1) = D$

D	*Q*(*t* + 1)	
0	0	Reset
1	1	Set

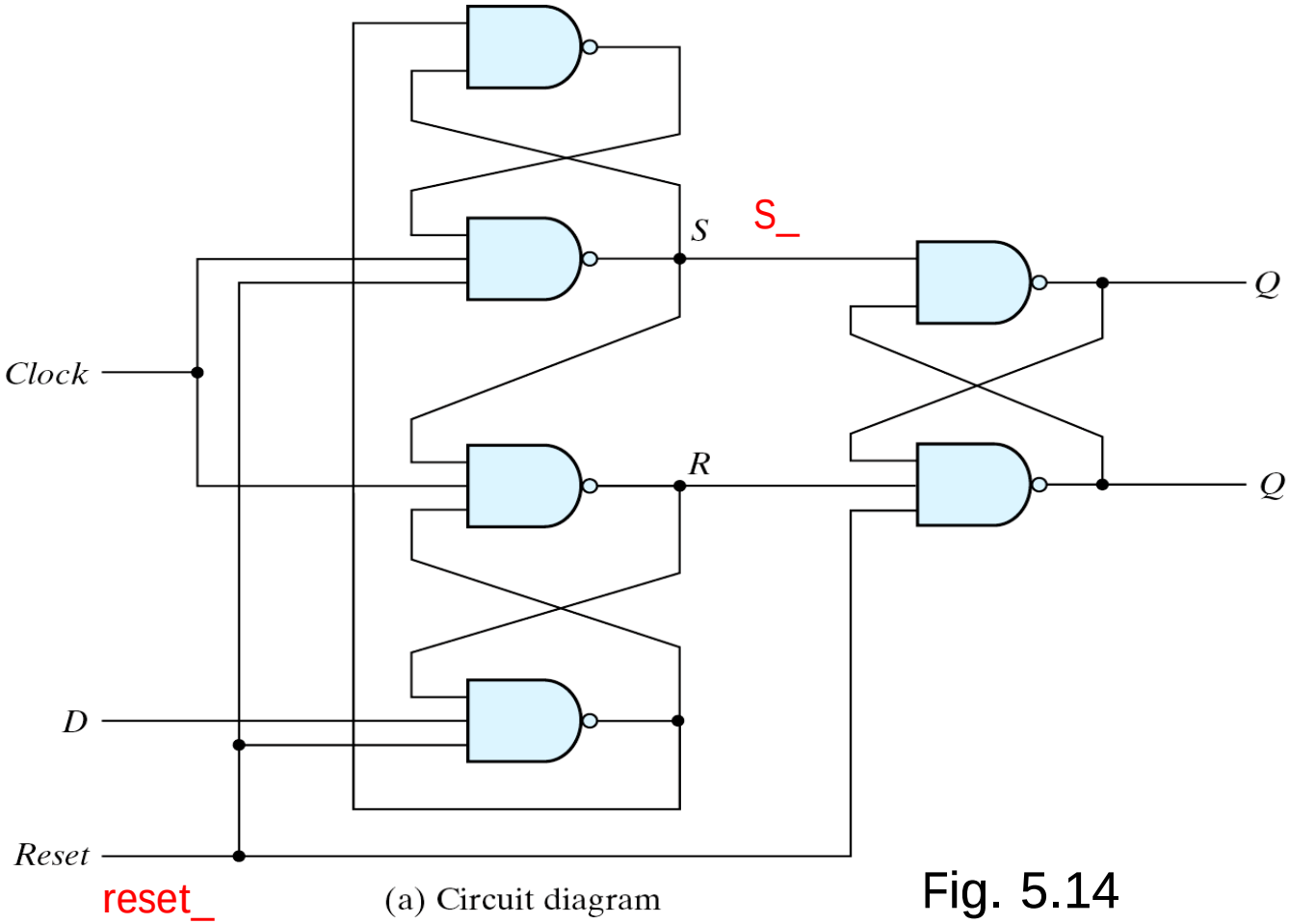
***T* Flip-Flop**

$Q(t+1) = T \oplus Q = TQ' + T'Q$

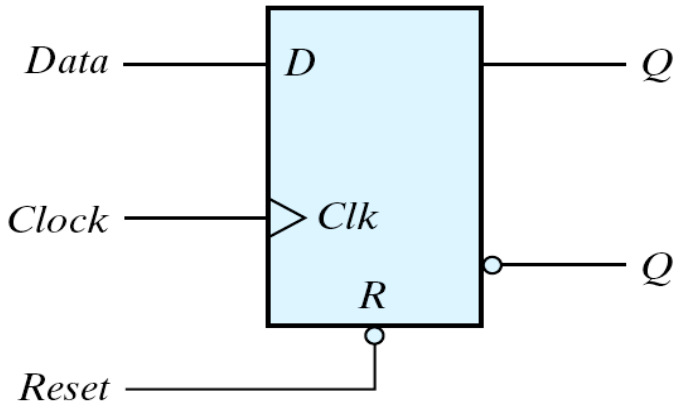
T	*Q*(*t* + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Direct inputs

- asynchronous set and/or asynchronous



(a) Circuit diagram



(b) Graphic symbol

R	Clk	D	Q	Q′
0	X	X	0	1
0	↑	0	0	1
0	↑	1	1	0

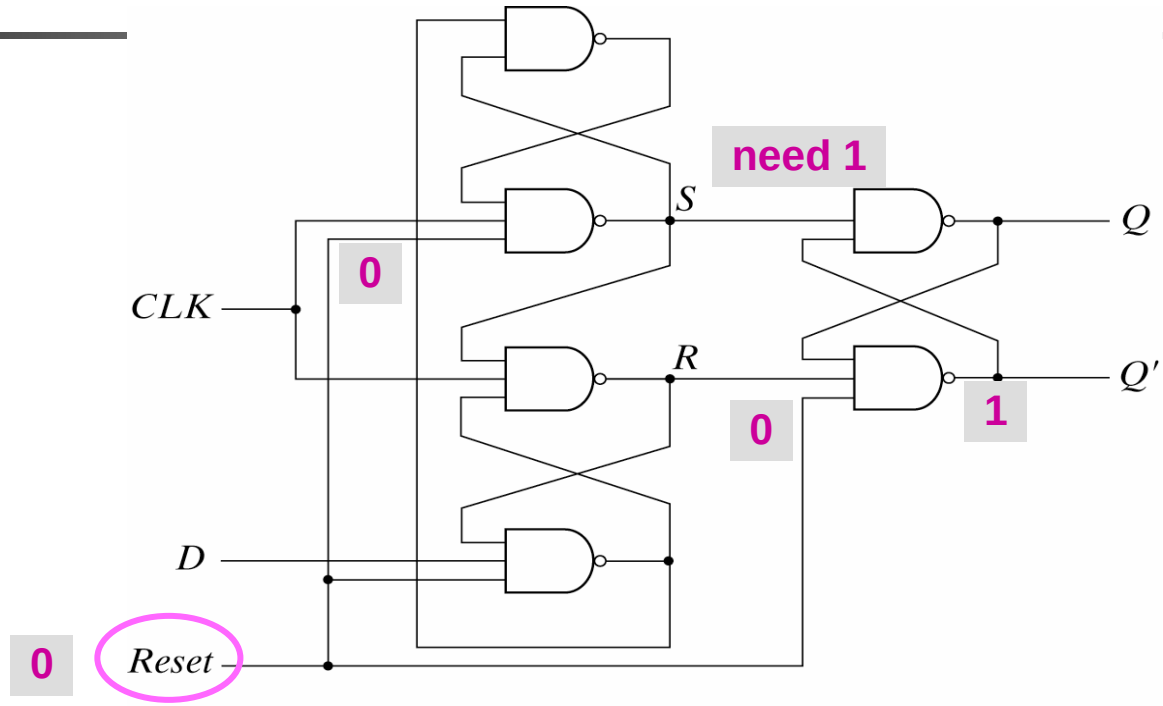
(b) Function table

Fig. 5.14
D flip-flop with asynchronous reset

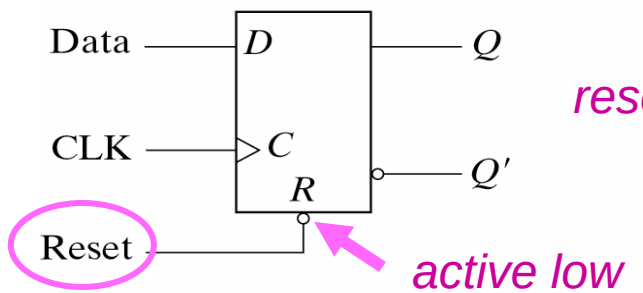
Direct Input

- Preset (PRE)
  - an asynchronous input that sets the FF
  - “direct set”
- Clear (CLR)
  - an asynchronous input that clears the FF
  - “direct reset”
- Purpose
  - Can be used to bring all FFs in a system to a known state prior to the clocked operation
- Asynchronous set: Set as soon as preset =1
- Synchronous set: Set when preset=1 and CLK↑

D Flip-Flop with Asynchronous Reset



(a) Circuit diagram



(b) Graphic symbol

reset →

R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

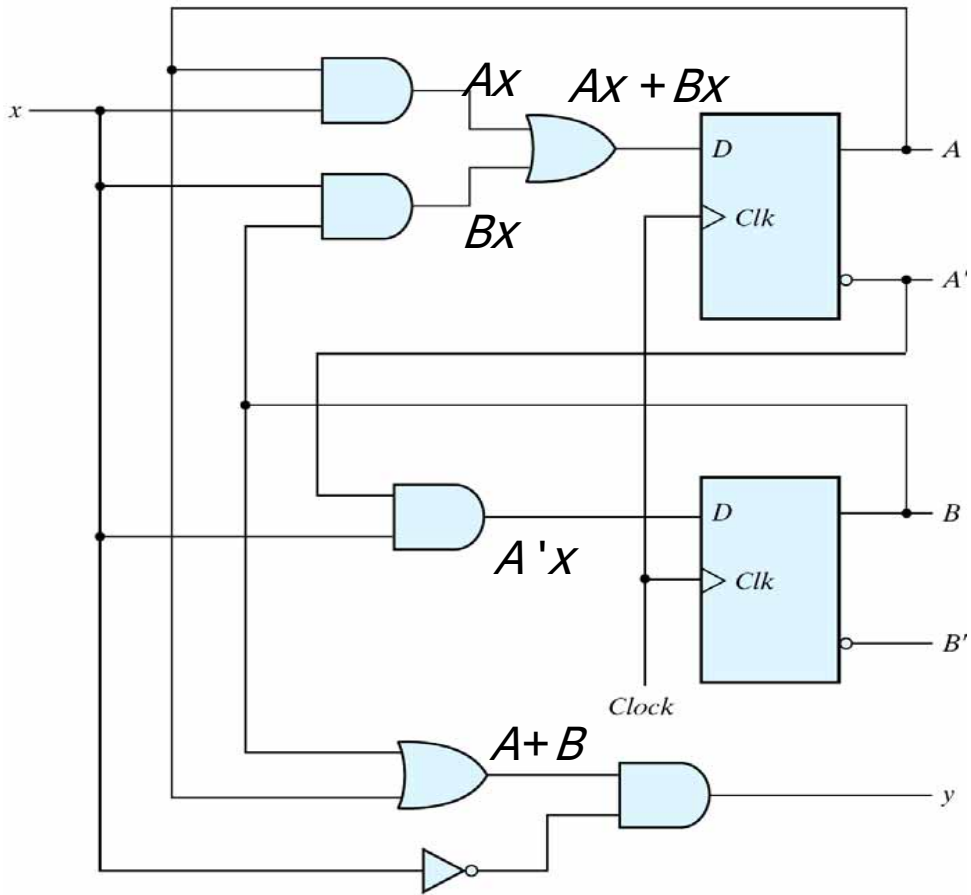
FF triggers on the positive edge of CLK

(b) Function table

Fig. 5-14 *D* Flip-Flop with Asynchronous Reset

Analysis of Clocked Sequential Ckts

- A sequential circuit
  - (inputs, current state) ⇒ (output, next state)
  - a state transition table or state transition diagram



- State (transition) equation
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = A'(t)x(t)$
- A compact form
  - $A(t+1) = Ax + Bx$
  - $B(t+1) = A'x$
- The output equation
  - $y(t) = (A(t)+B(t))x'(t)$
  - $y = (A+B)x'$

State table 1

Table 5.2
State Table for the Circuit of Fig. 5.15

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

$A(t+1) = Ax + Bx$

$B(t+1) = A'x$

$y = Ax' + Bx'$

State table 2

$A(t + 1) = Ax + Bx$
$B(t + 1) = A'x$
$y = Ax' + Bx'$

Table 5.3
Second Form of the State Table

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

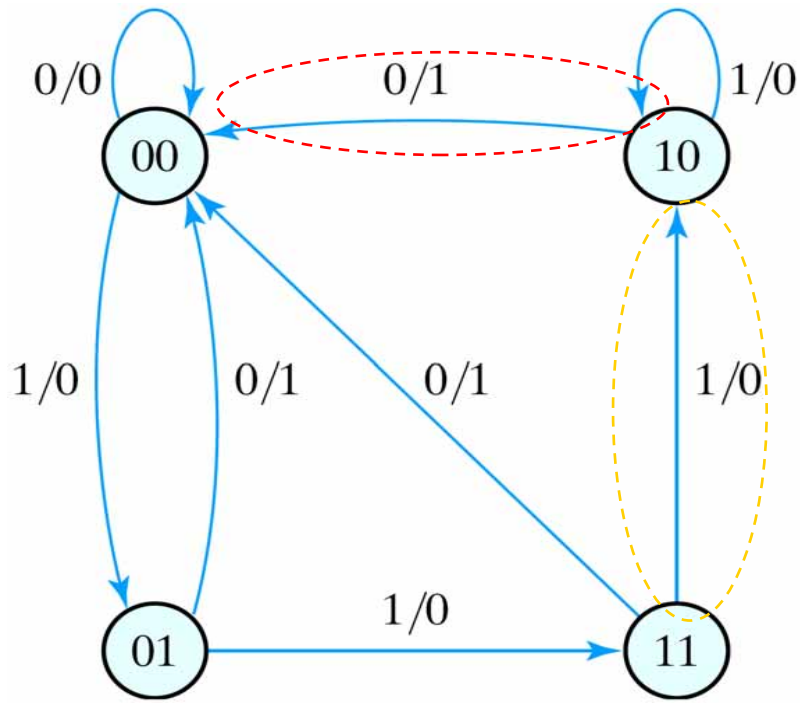
State diagram

- State transition diagram
  - a circle: a state
  - a directed lines connecting the circles: the transition between the states
    - Each directed line is labeled “inputs/outputs”

state: *A B*
input: *x*

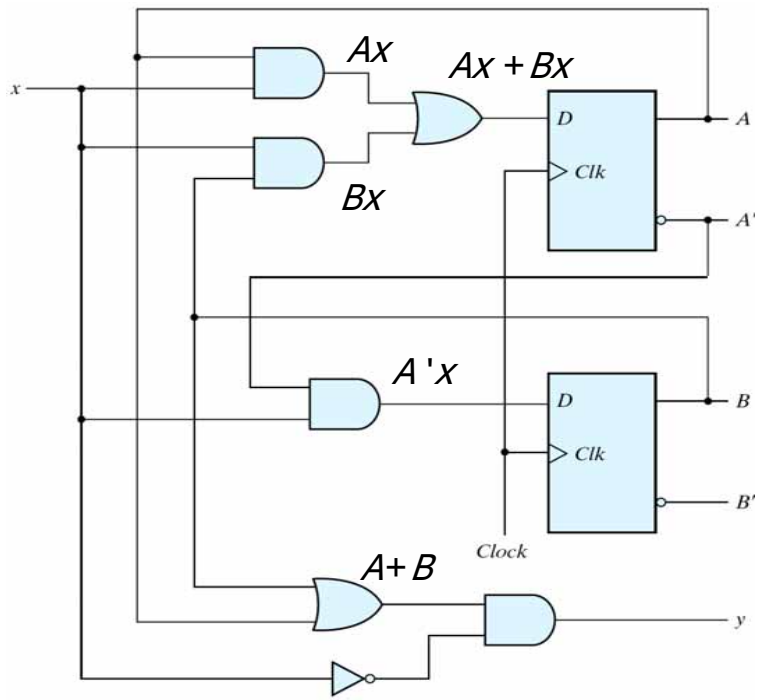
Table 5.3
Second Form of the State Table

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0



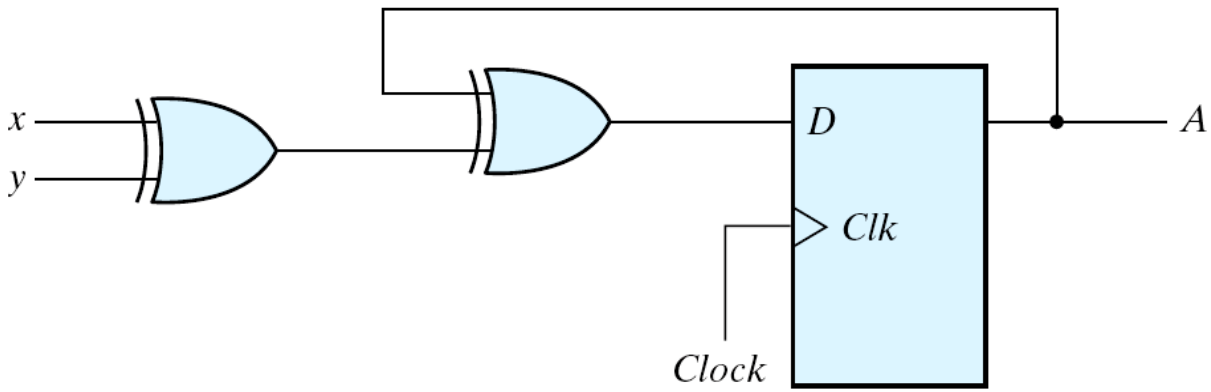
Flip-Flop Input Equations

- The part of circuit that generates the inputs to flip-flops
  - Also called excitation functions
  - DA = Ax +Bx
  - DB = A'x
- The output equations
  - to fully describe the sequential circuit
  - y = (A+B)x'



Analysis with D flip-flops

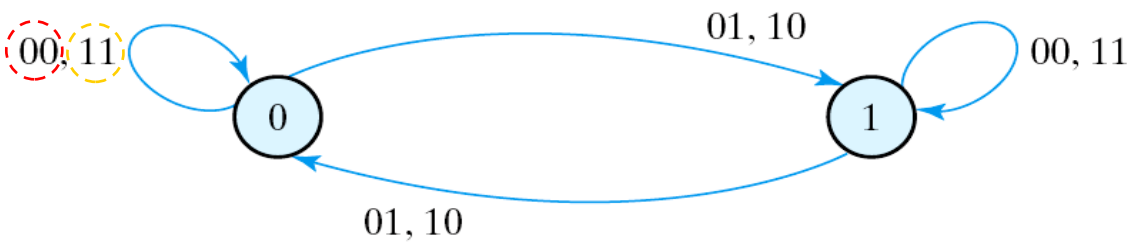
- The input equation
  - $D_A = A \oplus x \oplus y$
- The state equation
  - $A(t+1) = A \oplus x \oplus y$



(a) Circuit diagram

Present state	Inputs		Next state
A	*x*	*y*	*A*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table



(c) State diagram

Analysis with JK flip-flops

- Determine the flip-flop input function in terms of the present state and input variables
- Used the corresponding flip-flop characteristic table to determine the next state

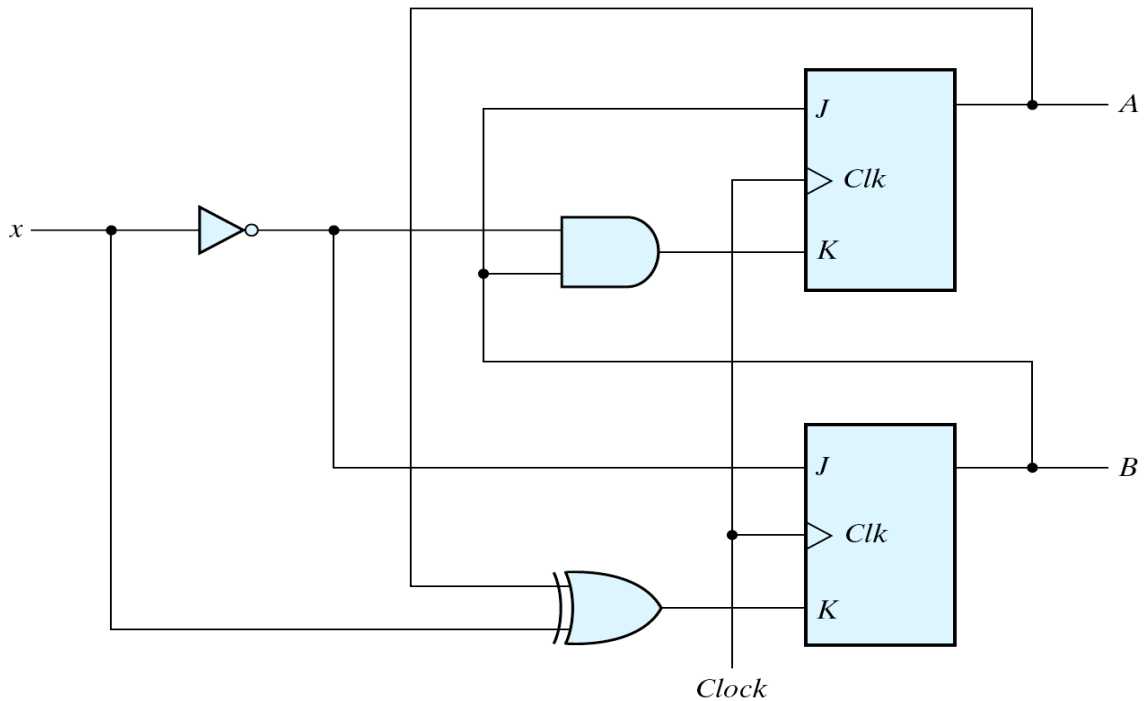


Fig. 5-18
Sequential circuit with *JK* flip-flop

$J_A = B,\ K_A = Bx'$

$J_B = x',\ K_B = A'x + A$

State Table for Fig. 5-18

$J_A = B,\ K_A = Bx'$

$J_B = x',\ K_B = A'x + Ax'$

Table 5.4

State Table for Sequential Circuit with JK Flip-Flops

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

State Transition Diagram for Fig. 5-18

The characteristic equation of JK FF is

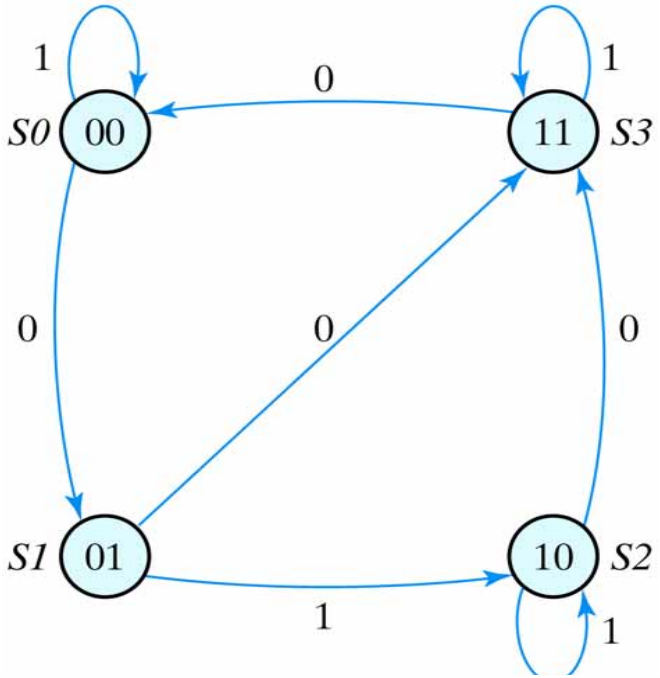
Method 1

$$A(t+1) = JA' + K'A$$
$$B(t+1) = JB' + K'B$$

State equation for *A* and *B*:

$$A(t+1) = BA' + (Bx')'A = A'B + AB' + Ax$$

$$B(t+1) = x'B + (A \oplus x)'B = B'x + ABx + A'Bx'$$



Method 2

$A(t+1)$

x / AB	00	01	11	10	
0	0	1	0	1	AB'
1	0	1	1	1	
		$A'B$	Ax		

Using *K*-map, we also can derive $A(t+1)$.

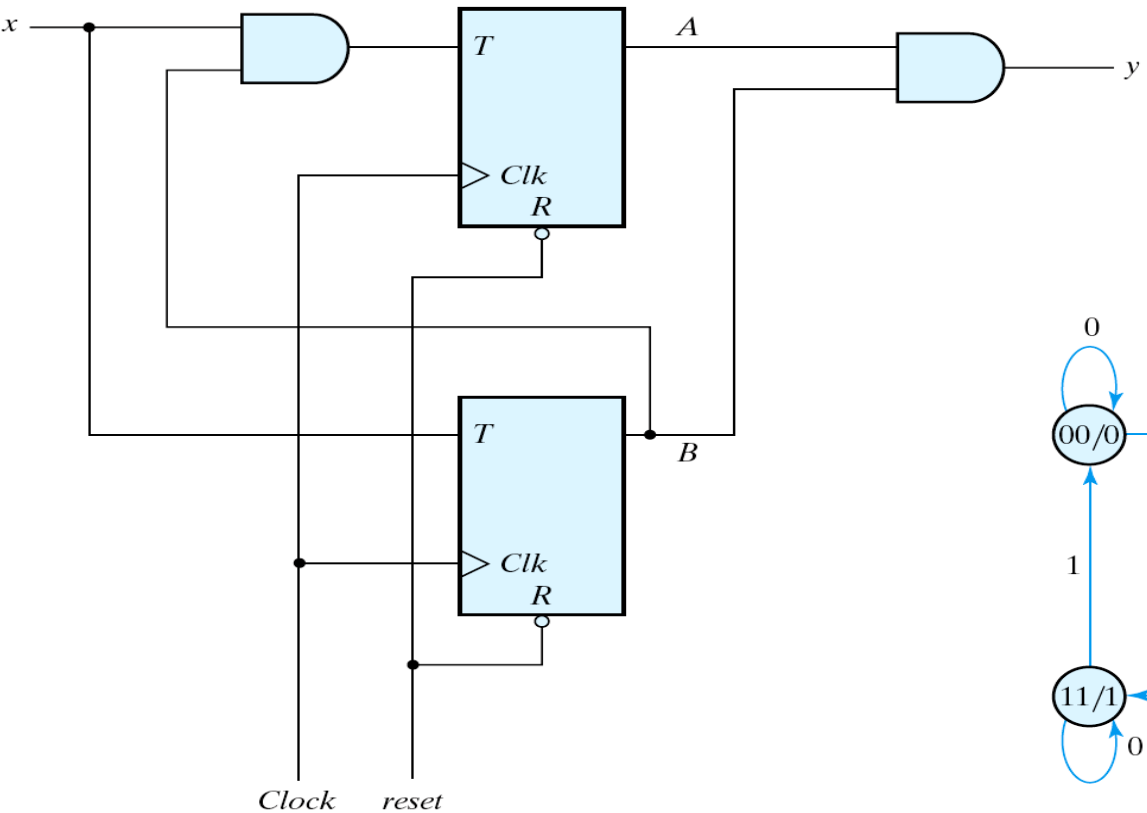
$A(t+1) = A'B + AB' + Ax$

Table 5.4
State Table for Sequential Circuit with JK Flip-Flops

Present State		Input	Next State		Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Analysis with T Flip-Flops

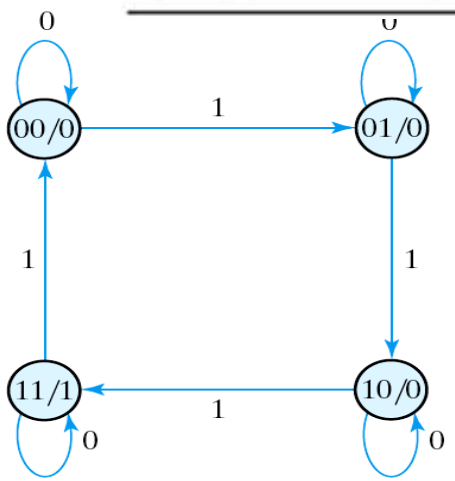
- The characteristic equation
  - $Q(t+1) = T \oplus Q = TQ' + T'Q$



(a) Circuit diagram

Table 5.5
State Table for Sequential Circuit with T Flip-Flops

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

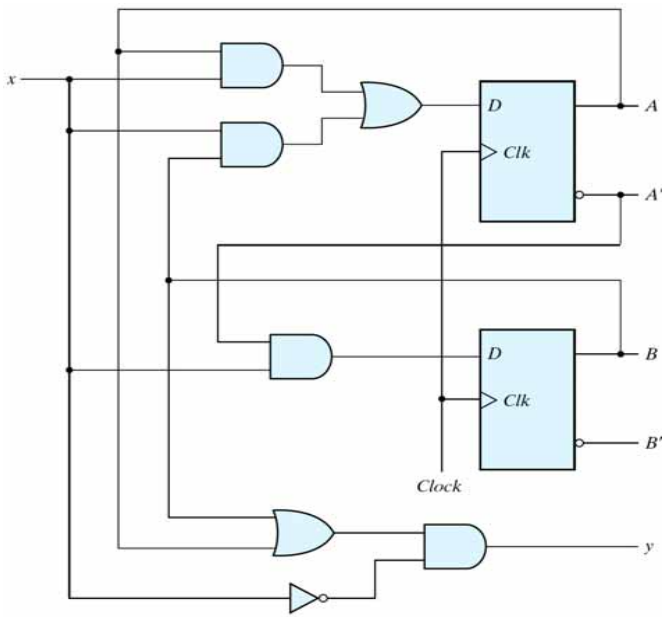


(b) State diagram

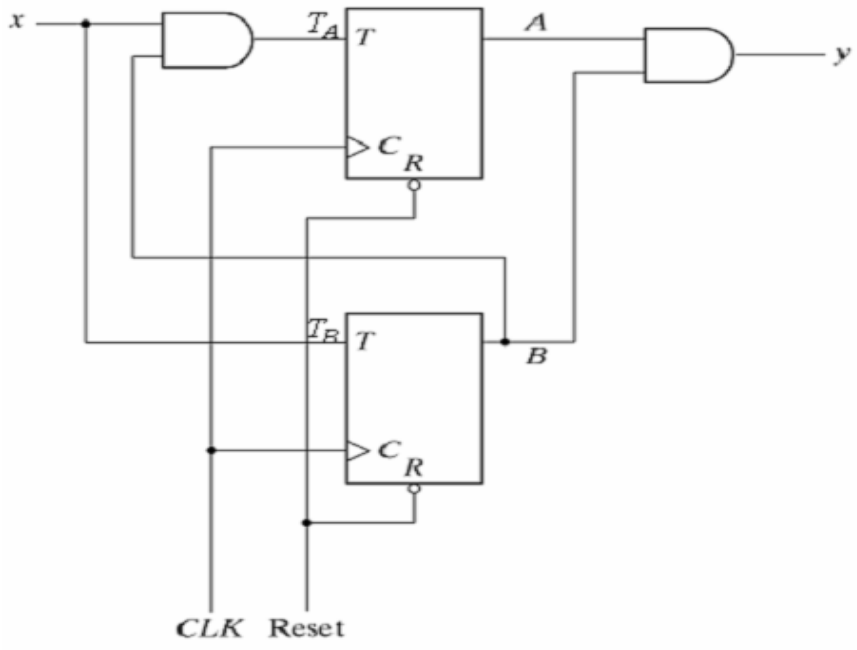
Finite State Machine (FSM)

The inputs, outputs and states of a sequential circuit can be described as the FSM. There are two different FSMs:

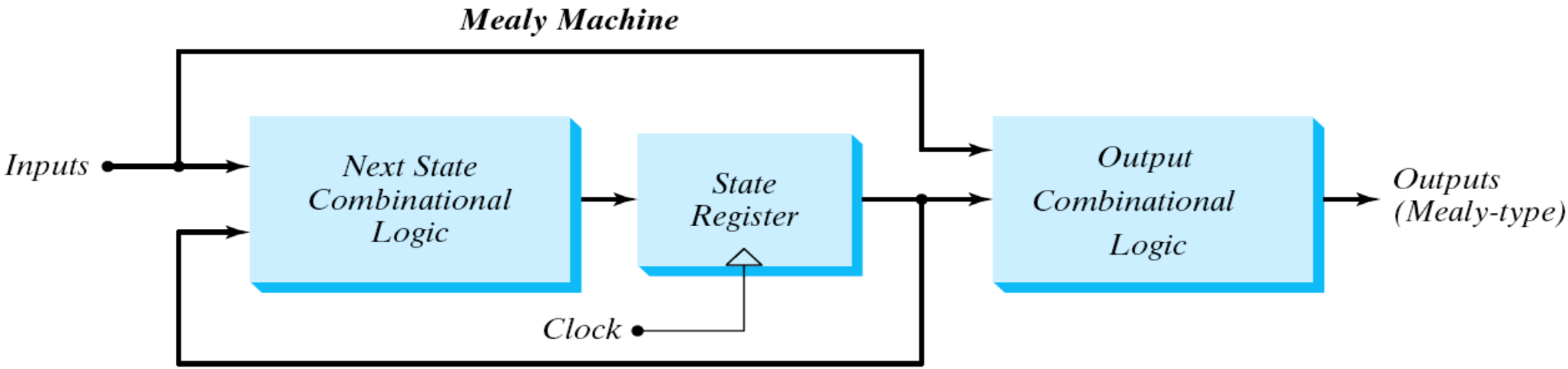
(a) Mealy machine: the outputs are functions of both the present state and inputs



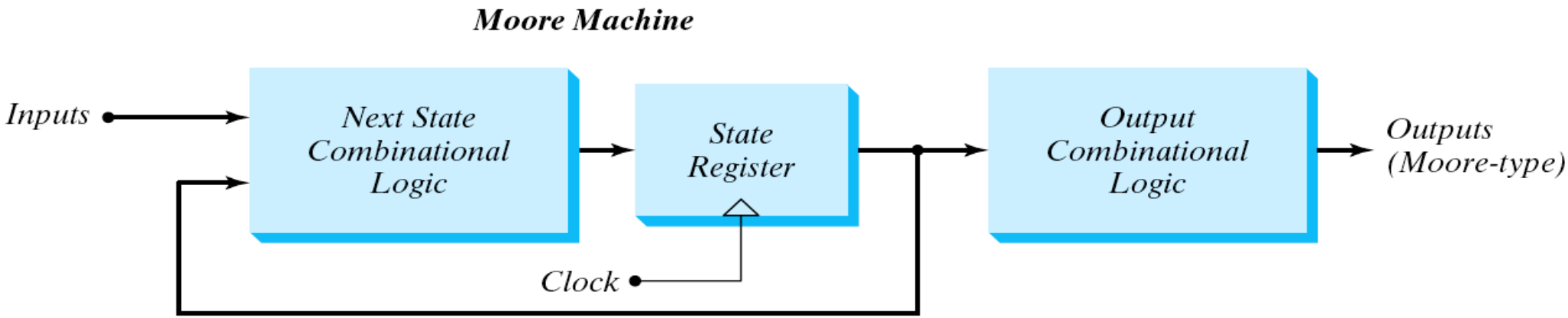
(b) Moore machine: the outputs are functions of the present state only



Mealy Machine vs. Moore Machine



(a)

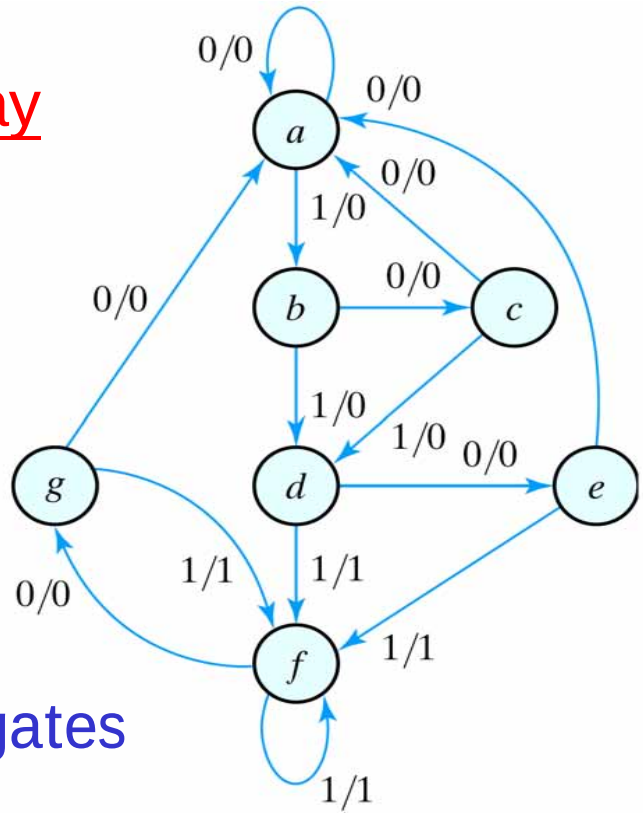


(b)

State Reduction and Assignment

State Reduction

- reductions on the number of flip-flops and the number of gates
- a reduction in the number of states may result in a reduction in the number of flip-flops
- How to reduce the necessary states?



State reduction:

does not guarantee a saving in #FFs or #gates

State Reduction

- state a a b c d e f f g f g a
  input 0 1 0 1 0 1 1 0 1 0 0
  output 0 0 0 0 0 1 1 0 1 0 0
- only the input-output sequences are important
- two circuits are equivalent
  - have identical outputs for all input sequences
  - the number of states is not important

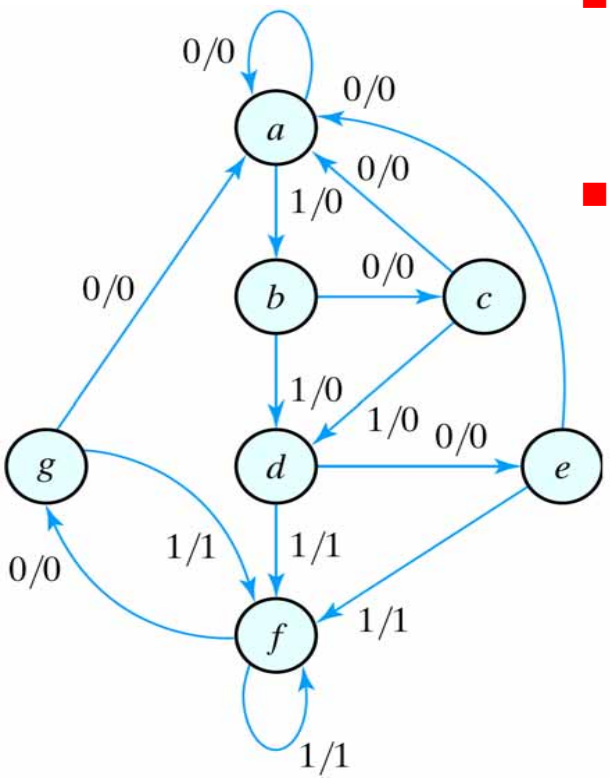


Fig. 5.25 State diagram

Equivalent States

- Two states are said to be equivalent
  - for each member of the set of inputs, they give exactly the same output and send the circuit to the same state or to an equivalent state
  - one of them can be removed

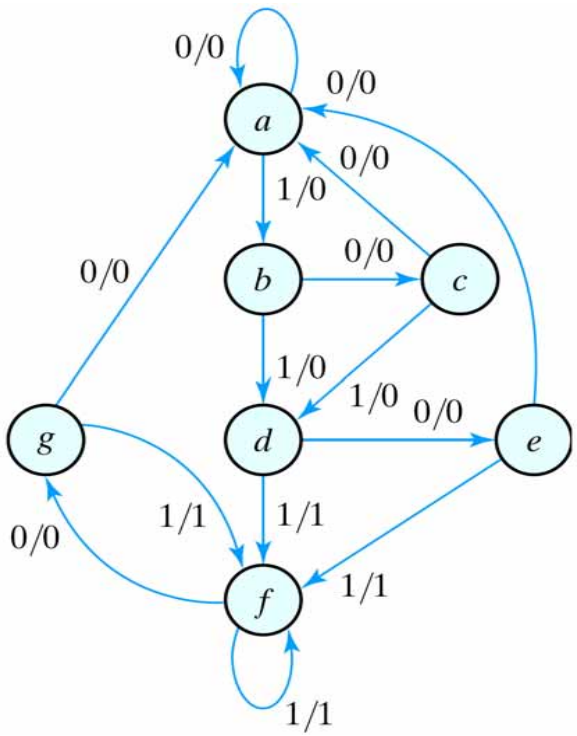


Table 5.6
State Table

Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Reducing State Table

Table 5.7
Reducing the State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Reduced Finite State Machine

Table 5.8
Reduced State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

- state a a b c d e d d e d e a
  input 0 1 0 1 0 1 1 0 1 0 0
  output 0 0 0 0 0 1 1 0 1 0 0

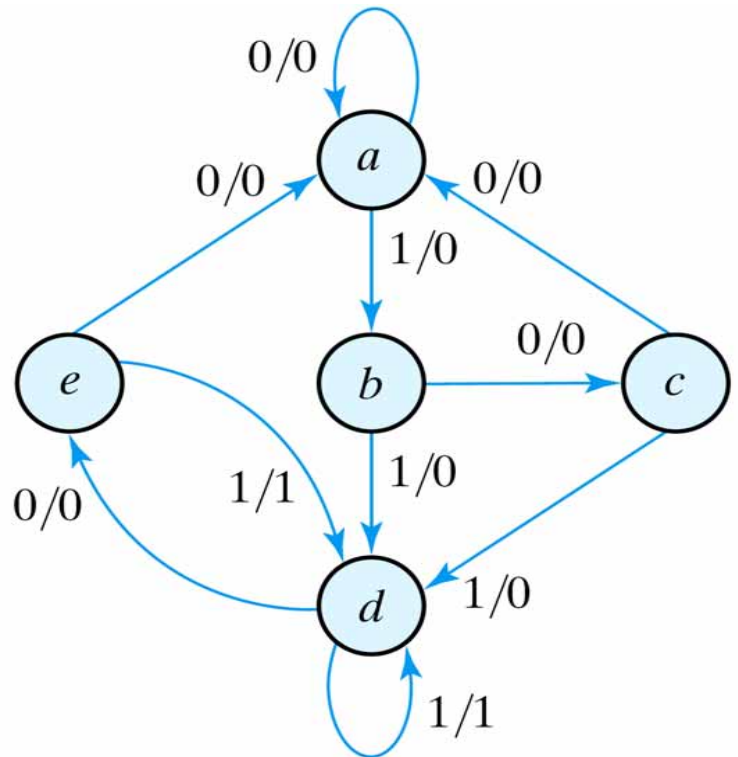
State Reduction

- the checking of each pair of states for possible equivalence can be done systematically (9-5)
- the unused states are treated as don't-care condition ⇒ fewer combinational gates

This example:

reduce to

7 states ➔ 5 states



State Assignment

- to minimize the cost of the combinational circuits (not easy certainly ??)
- three possible binary state assignments

Table 5.9
Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Binary Assignment

- any binary number assignment is satisfactory as long as each state is assigned a unique number

  both OK ➔ 000,001,010,011,100 (OK v)

  ➔ 011,100,101,110,111 (OK v)

Table 5.10

Reduced State Table with Binary Assignment 1

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

The Three Assignments

- Binary code
  - n-bit code for m states, $2^n >= m$ (n FFs)
- Gray code
  - n-bit code for m states, $2^n >= m$ (n FFs)
  - More suitable for K-map simplification (more possible lower power)
- One-hot
  - m-bit code for m states (m FFs)
  - often used in control design

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Design Procedure

- specification ➔a state diagram (most challenging)
- state reduction if necessary
- assign binary values to the states
- obtain the binary-coded state table
- choose the type of flip-flops
- derive the simplified flip-flop input equations and output equations
- draw the logic diagram

Synthesis

The part of design that follows a well-defined procedure is called synthesis
Once a spec has been set down and the state diagram obtained, it is possible to use known synthesis procedure to complete the design

Synthesis using D flip-flops (1/2)

- An example state diagram and state table

  Design a circuit that detects one to three or more consecutive 1's in a input string

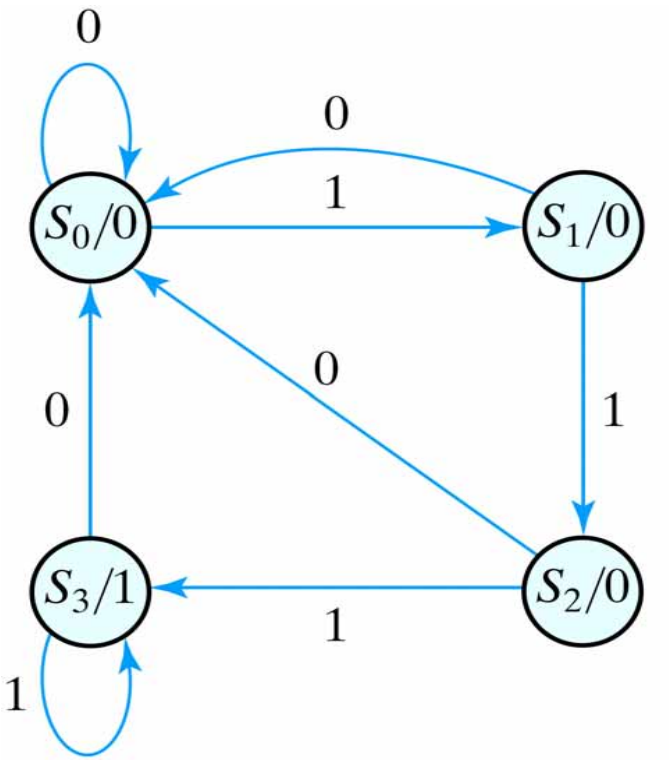


Table 5.11
State Table for Sequence Detector

Present State		Input		Next State		Output
A	*B*	*x*		*A*	*B*	*y*
0	0	0	m_0	0	0	0
0	0	1	m_1	0	1	0
0	1	0	m_2	0	0	0
0	1	1	m_3	1	0	0
1	0	0	m_4	0	0	0
1	0	1	m_5	1	1	0
1	1	0	m_6	0	0	1
1	1	1	m_7	1	1	1

Synthesis using D flip-flops (2/2)

- The flip-flop input equations

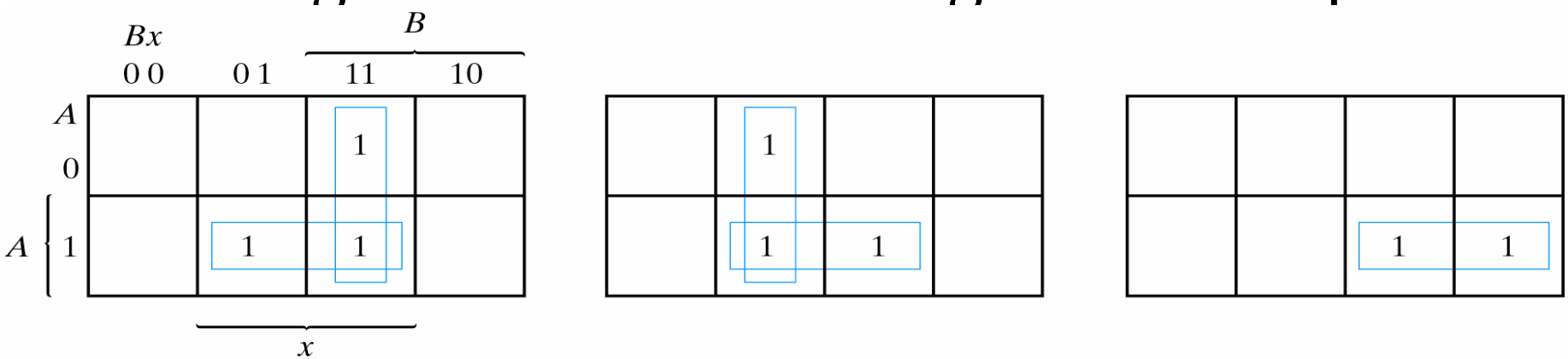
  (1) $A(t+1) = D_A(A,B,x) = \Sigma(3,5,7)$

  (2) $B(t+1) = D_B(A,B,x) = \Sigma(1,5,7)$

- The output equation

  (3) $y(A,B,x) = \Sigma(6,7)$

- Logic minimization using three K maps



Logic Diagram of Sequence Detector with D FF

- FF Input eqs.

$D_A(A,B,x) = Ax + Bx$

$D_B(A,B,x) = Ax + B'x$

- Output eq.

$y(A,B,x) = AB$

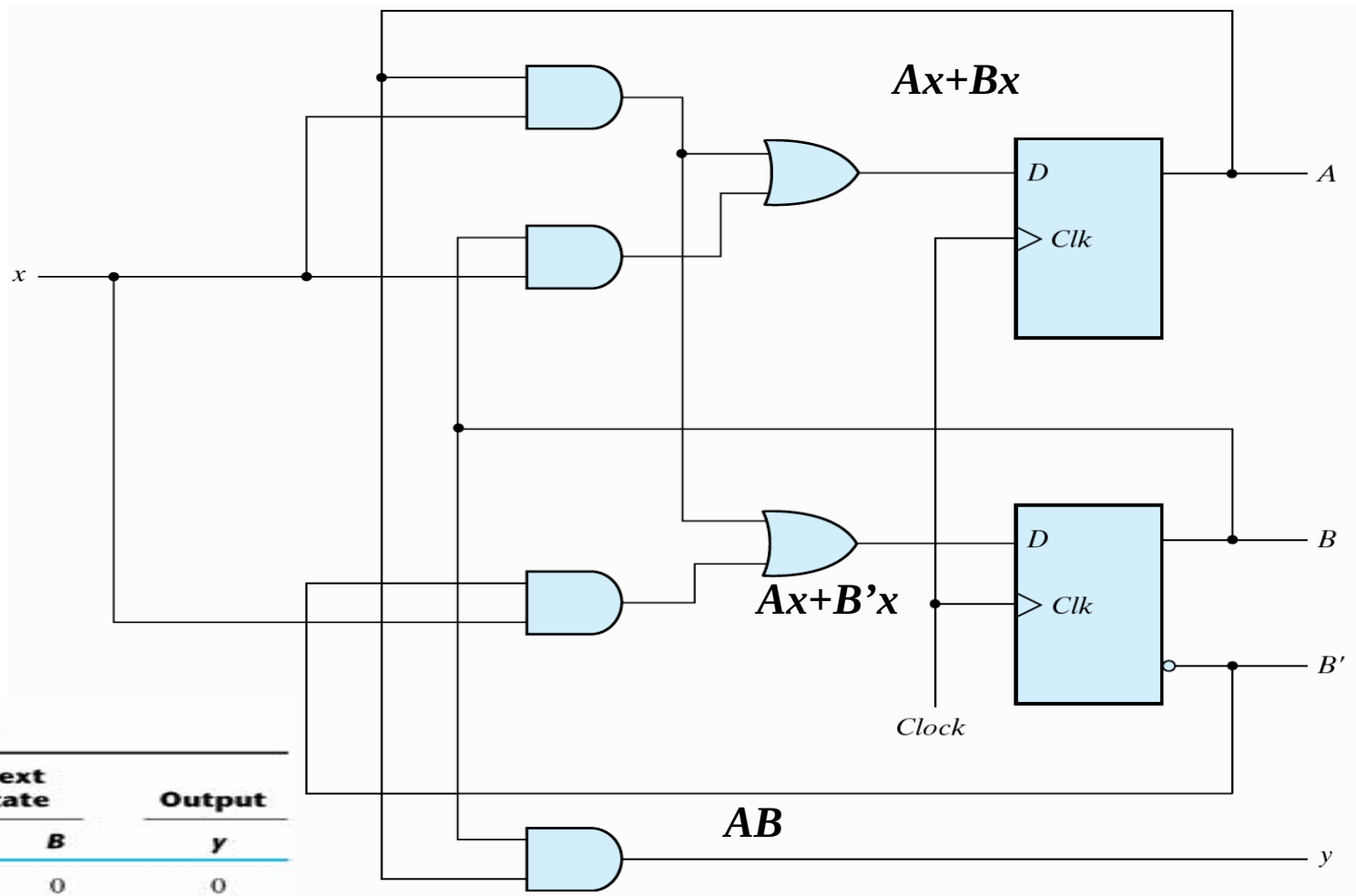


Table 5.11
State Table for Sequence Detector

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

back

Synthesis using JK flip-flops (1/4)

- A state diagram ⇒ flip-flop input functions
  - straightforward for D flip-flops
  - we need excitation tables for JK and T flip-flops

J	K	D	Q(t+1)	Function
0	0	Q(t)	Q(t)	no change
0	1	0	0	reset FF to 0
1	0	1	1	set FF to 1
1	1	Q'(t)	Q'(t)	complement output

T	D	Q(t+1)	
0	Q	Q(t)	no change
1	Q'	Q'(t)	complement

Table 5.12
Flip-Flop Excitation Tables

Q(t)	Q(t = 1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) *JK*

Q(t)	Q(t = 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T*

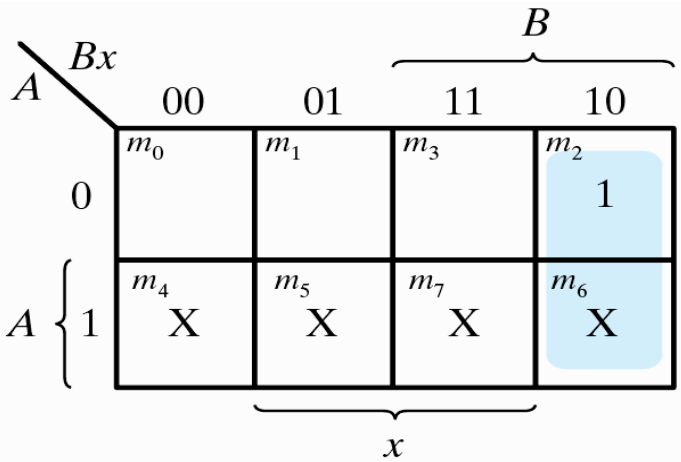
Synthesis using JK flip-flops (2/4)

- The same example
- The state table and JK flip-flop inputs

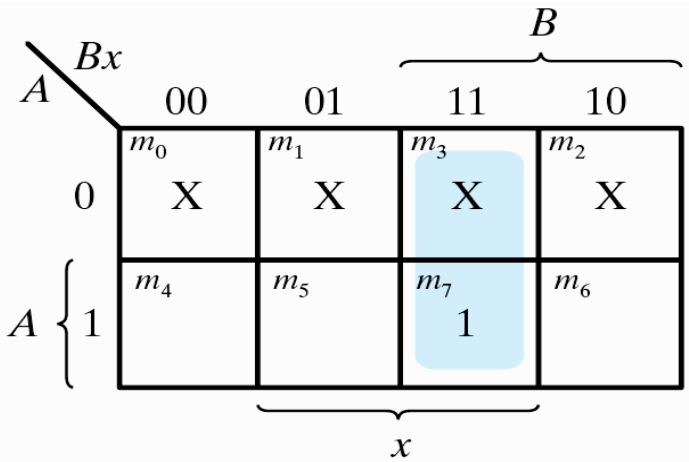
Table 5.13
State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

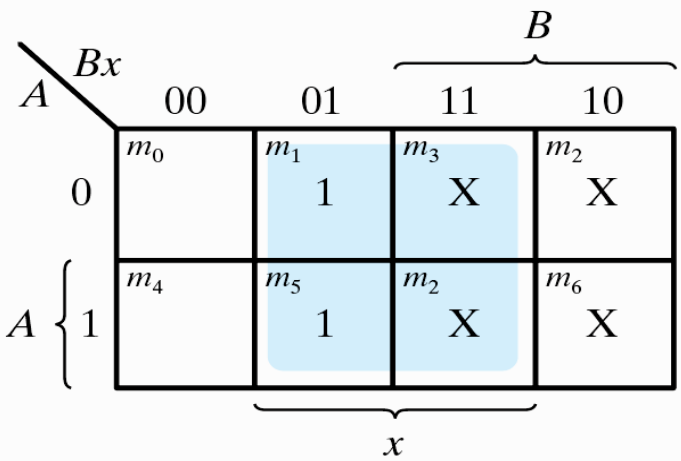
Synthesis using JK flip-flops (3/4)



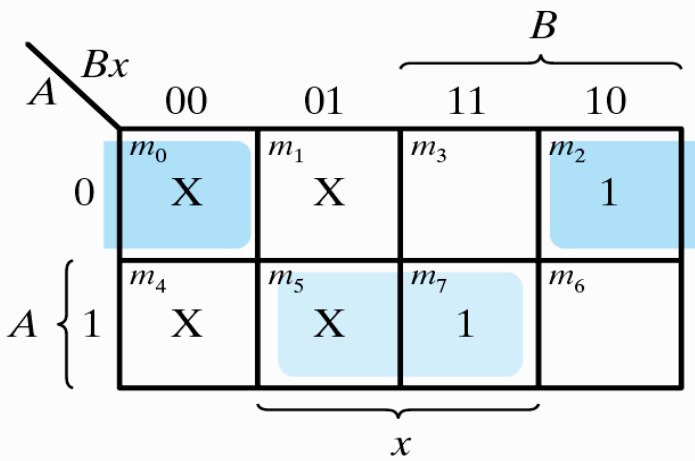
$J_A = Bx'$



$K_A = Bx$



$J_B = x$



$K_B = (A \oplus x)'$

Synthesis using JK flip-flops (4/4)

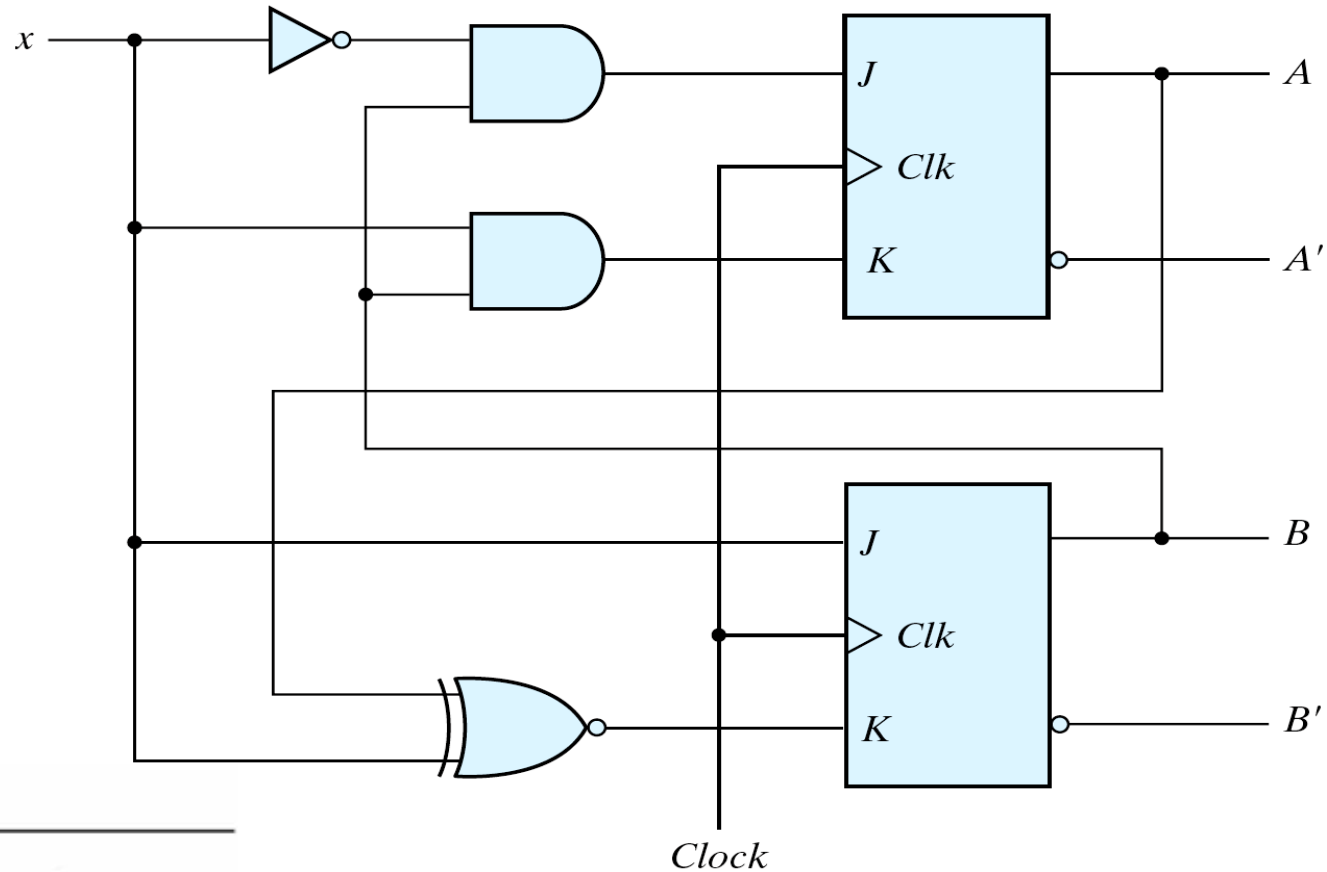


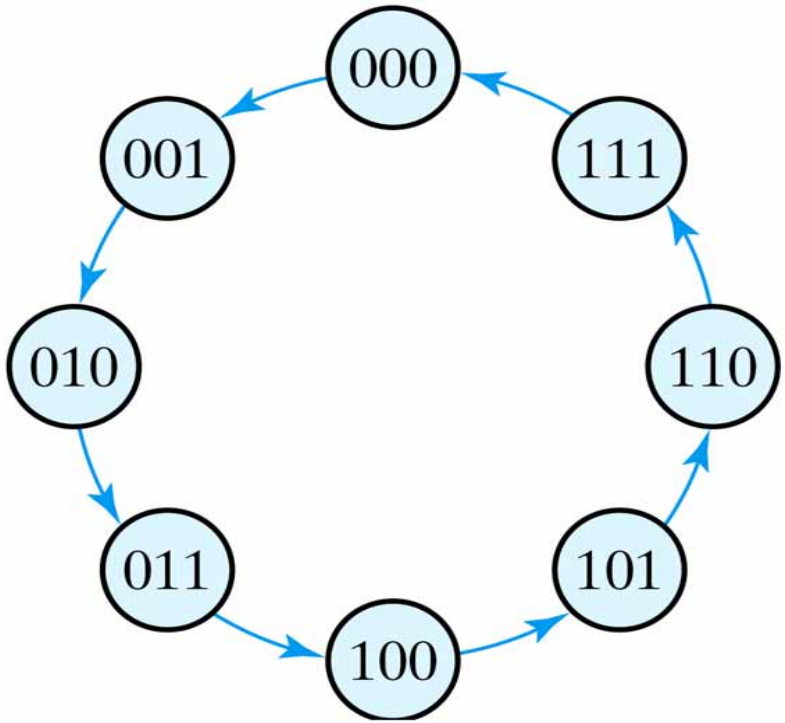
Table 5.13
State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Compare with D flip-flop

Synthesis using T flip-flops

- A n-bit binary counter
  - the state diagram



  - no inputs (except for the clock input)

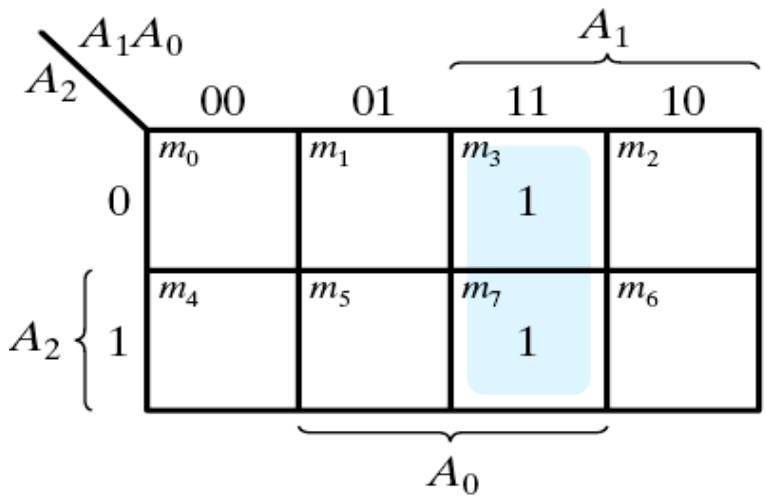
The state table and the flip-flop inputs

Table 5.14
State Table for Three-Bit Counter

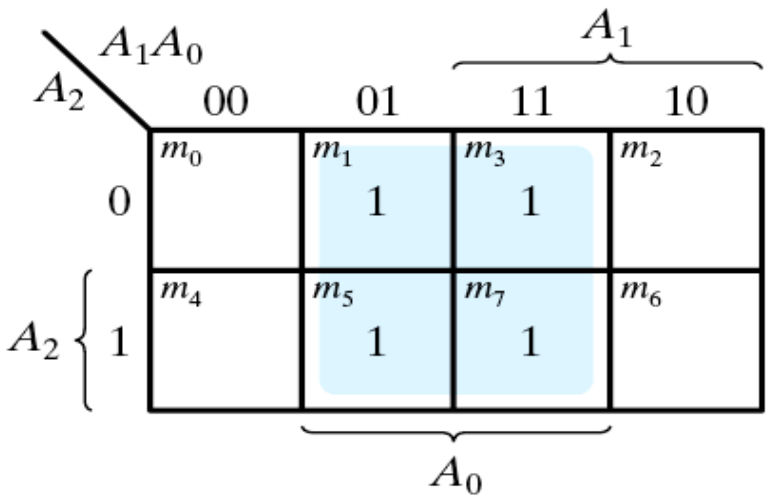
Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

No inputs (except for the clock input)

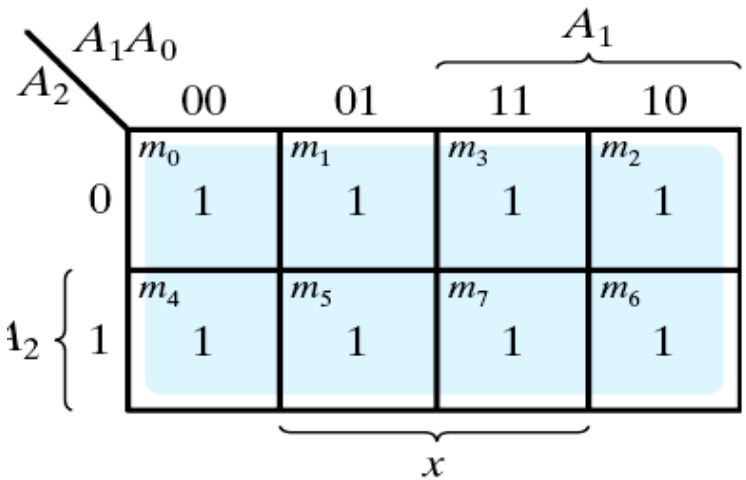
Logic Simplification using the K map



$T_{A2} = A_1A_0$

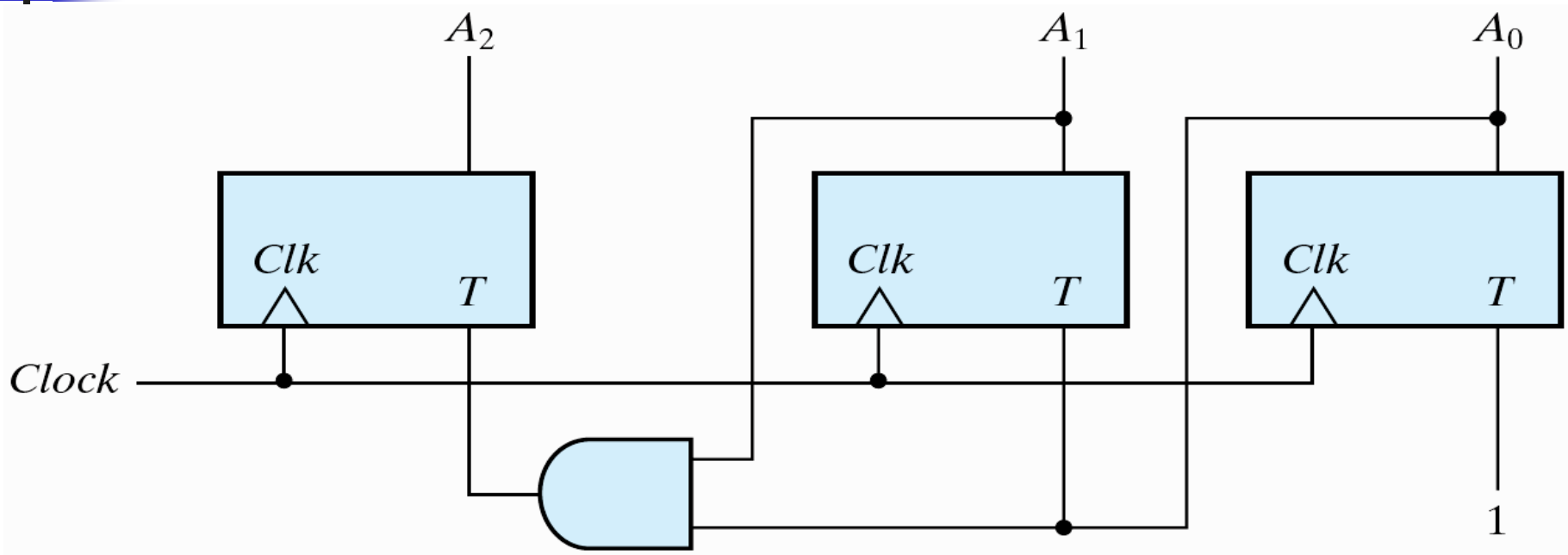


$T_{A1} = A_0$



$T_{A0} = 1$

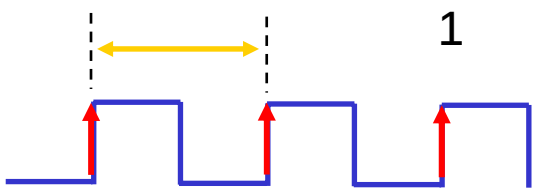
The Logic Diagram



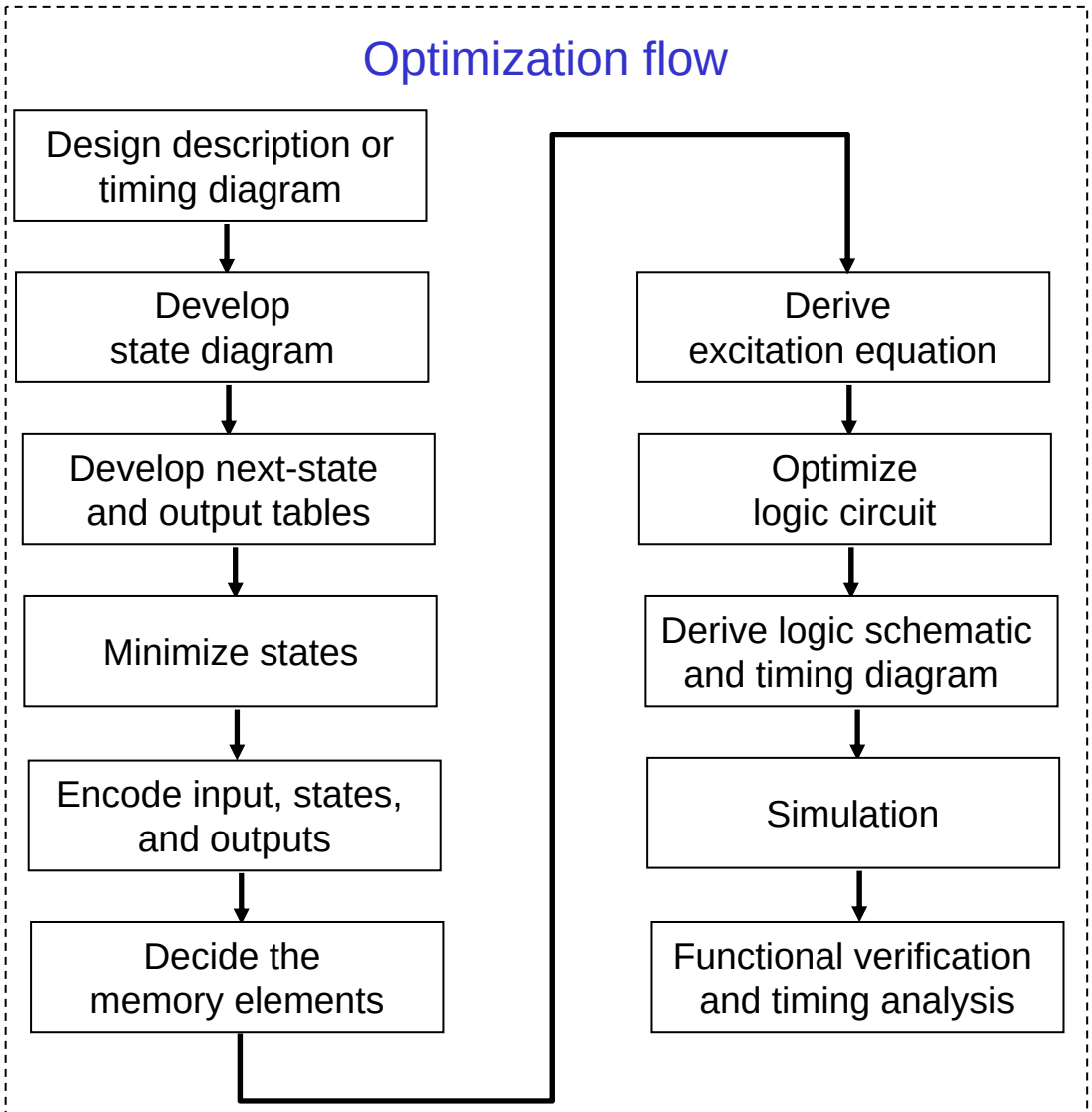
How to trace?

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

Note:



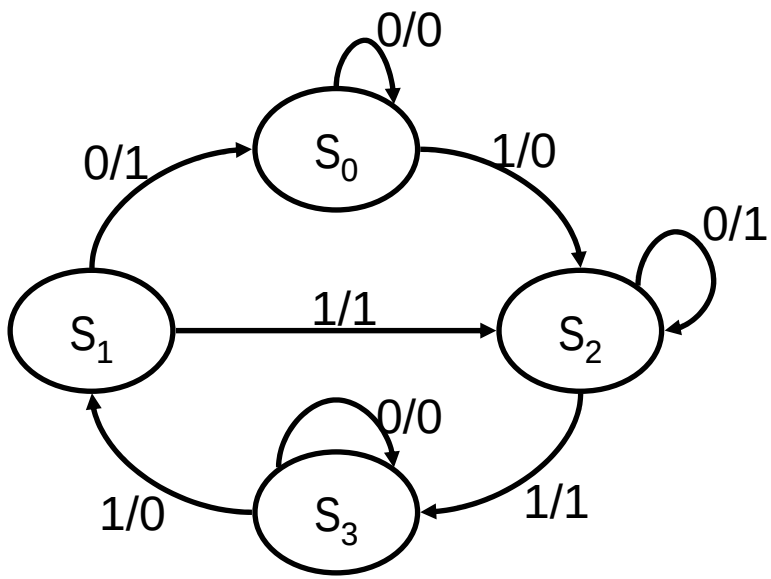
Moore Machine (1/4)



$S \rightarrow O \quad S:\text{state} \quad O:\text{output}$

Next-state and output tables (I=input)

Present State	Next State		Output	
	I=0	I=1	I=0	I=1
S_0	S_0	S_2	0	0
S_1	S_0	S_2	1	1
S_2	S_2	S_3	1	1
S_3	S_3	S_1	0	0



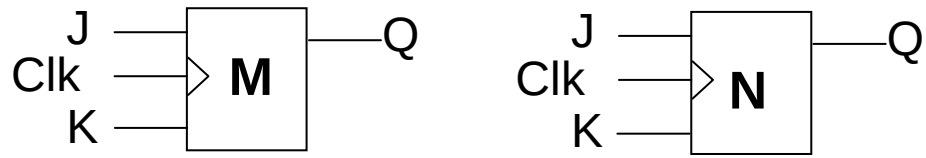
Moore Machine (2/4)

original state table

Present State	Next State		Output	
	I=0	I=1	I=0	I=1
S_0	S_0	S_2	0	0
S_1	S_0	S_2	1	1
S_2	S_2	S_3	1	1
S_3	S_3	S_1	0	0

Assume that we use JK flip-flops for storage

4 states ⇨ need 2 flip-flops (named M and N)



characteristic table

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

excitation table

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

I	Present State		Next State		M(JK)		N(JK)		Output
	M(t)	N(t)	M(t+1)	N(t+1)	MJ	MK	NJ	NK	
0	0	0	0	0	0	X	0	X	0
1	0	0	1	0	1	X	0	X	0
0	0	1	0	0	0	X	X	1	1
1	0	1	1	0	1	X	X	1	1
0	1	0	1	0	X	0	0	X	1
1	1	0	1	1	X	0	1	X	1
0	1	1	1	1	X	0	X	0	0
1	1	1	0	1	X	1	X	0	0

Moore Machine (3/4)

I \ MN	00	01	11	10
0	0	0	X	X
1	1	1	X	X

MJ=I

I \ MN	00	01	11	10
0	0	X	0	X
1	0	X	1	X

NJ=MI

I \ MN	00	01	11	10
0	X	X	0	0
1	X	X	1	0

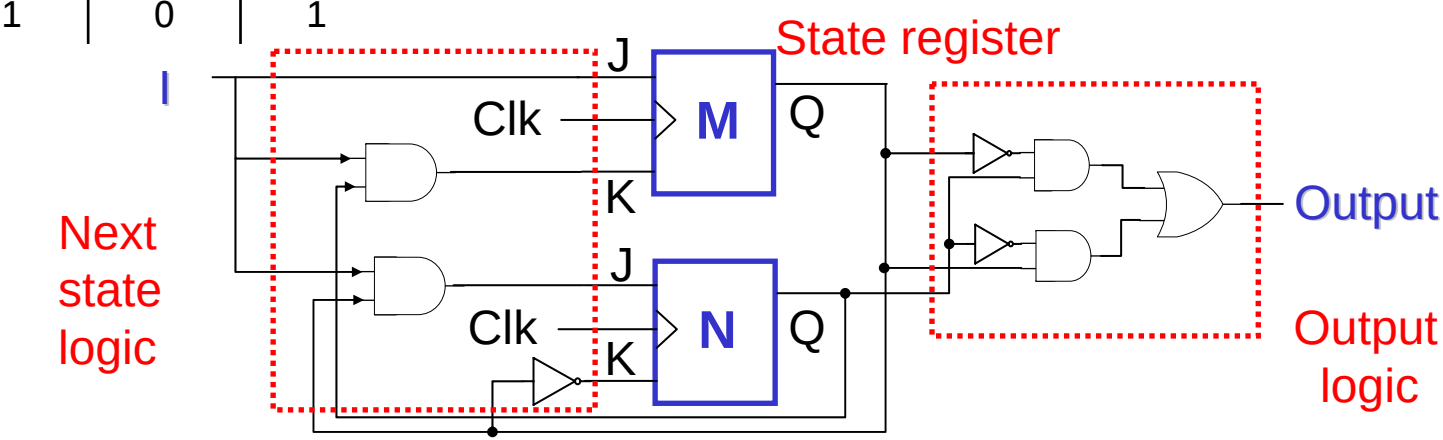
MK=NI

I \ MN	00	01	11	10
0	X	1	0	X
1	X	1	0	X

NK=M’

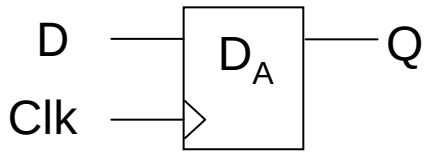
I \ MN	00	01	11	10
0	0	1	0	1
1	0	1	0	1

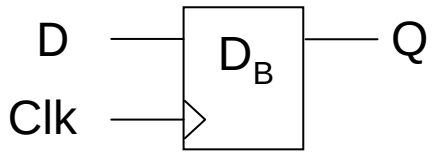
Output=M’N+MN’



Moore Machine (4/4)

How about D Flip-Flop?





Which implementation is better?

I	Present State		Next State		Output
	A	B	A	B	
0	0	0	0	0	0
1	0	0	1	0	0
0	0	1	0	0	1
1	0	1	1	0	1
0	1	0	1	0	1
1	1	0	1	1	1
0	1	1	1	1	0
1	1	1	0	1	0

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

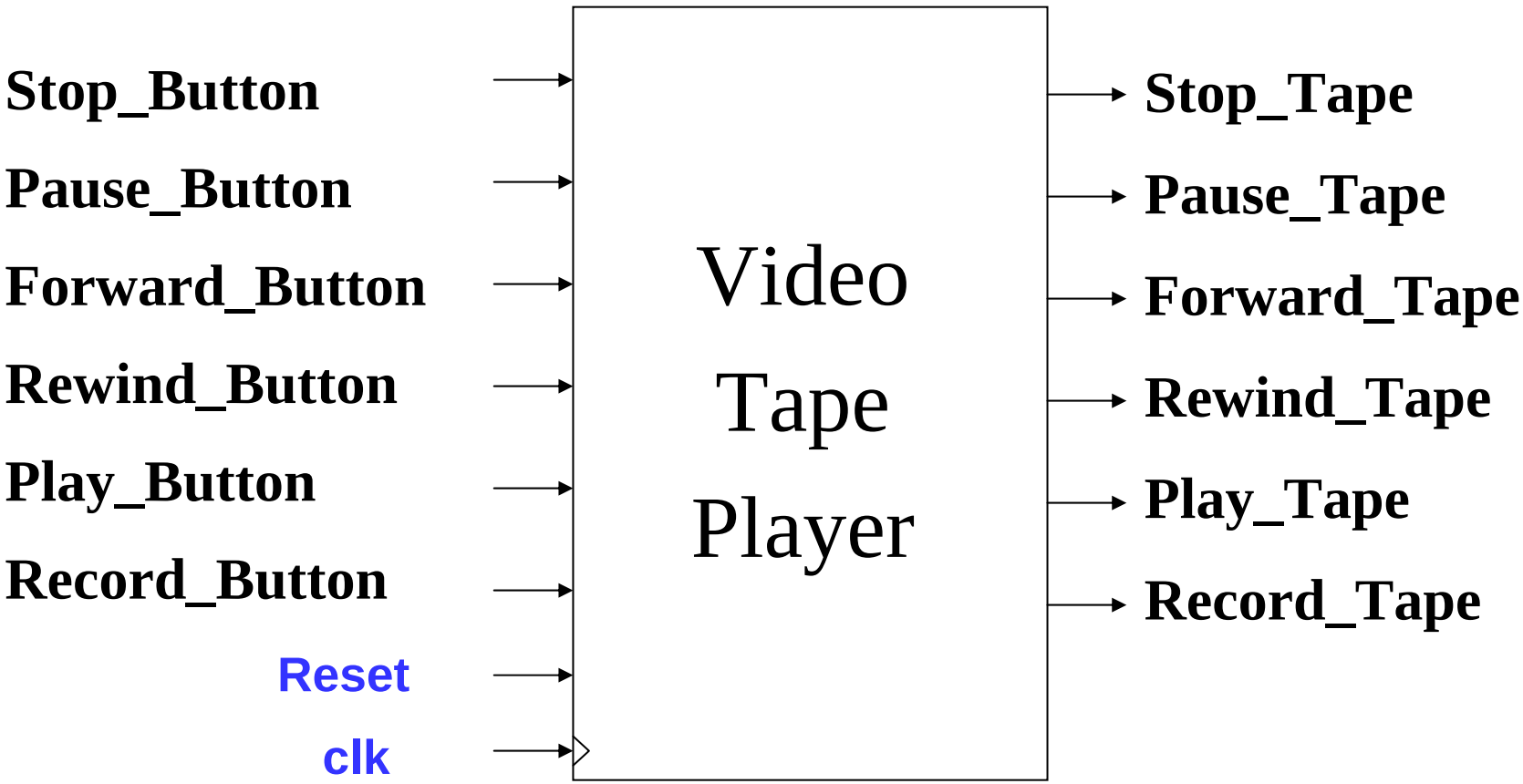
So, D_A=

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

So, D_B=

Output is the same as JK implementation.

Video Tape Player (1/2)



Video Tape Player (2/2)