



CSE 205: DIGITAL LOGIC DESIGN

Dr. Tanzima Hashem

Associate Professor

CSE, BUET

TEXTBOOK

- Digital Design (5th Edition)
 - M. Morris Mano
 - Michael D. Ciletti



COMBINATIONAL LOGIC

- Combinational Logic:
 - Output depends only on current input
 - Has no memory



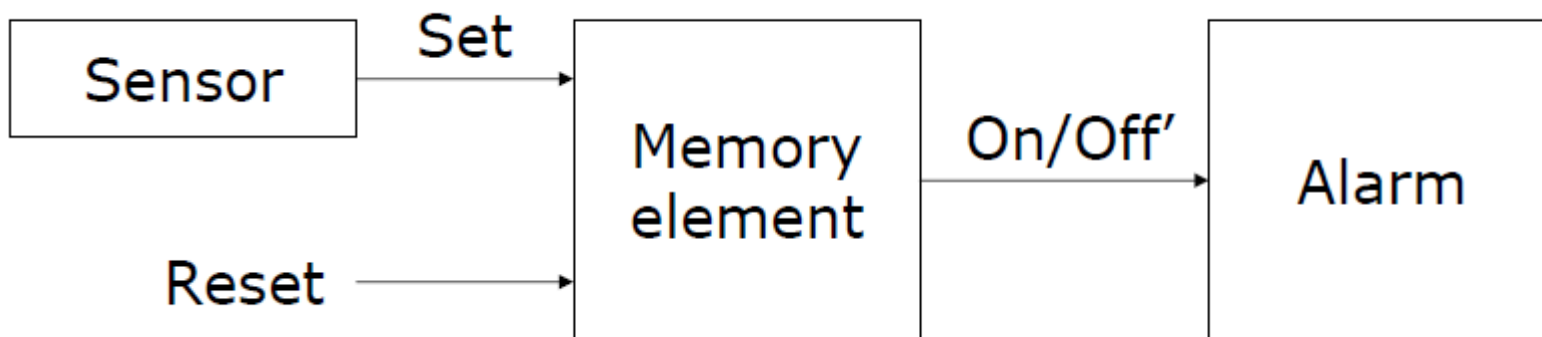
SEQUENTIAL LOGIC

- Sequential Logic:
 - Output depends not only on current input but also on past input values, e.g., design a counter
 - Need some type of memory to remember the past input values



ALARM CONTROL SYSTEM

- Suppose we wish to construct an alarm circuit such that the output remains active (on) even after the sensor output that triggered the alarm goes off
- The circuit requires a memory element to remember that the alarm has to be active until a reset signal arrives



SEQUENTIAL CIRCUITS

- Consist of a combinational circuit to which storage elements are connected to form a feedback path
- State: the state of the memory devices now, also called current state
- Next states and outputs are functions of inputs and present states of storage elements

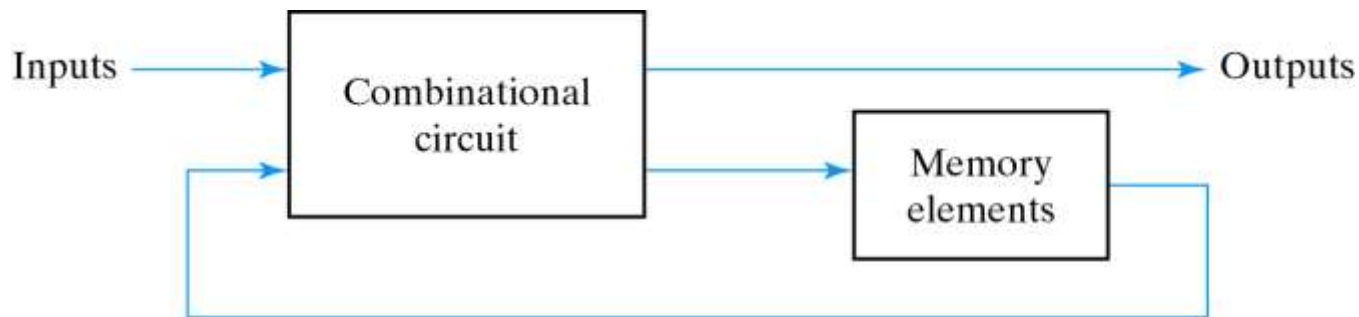


Fig. 5-1 Block Diagram of Sequential Circuit

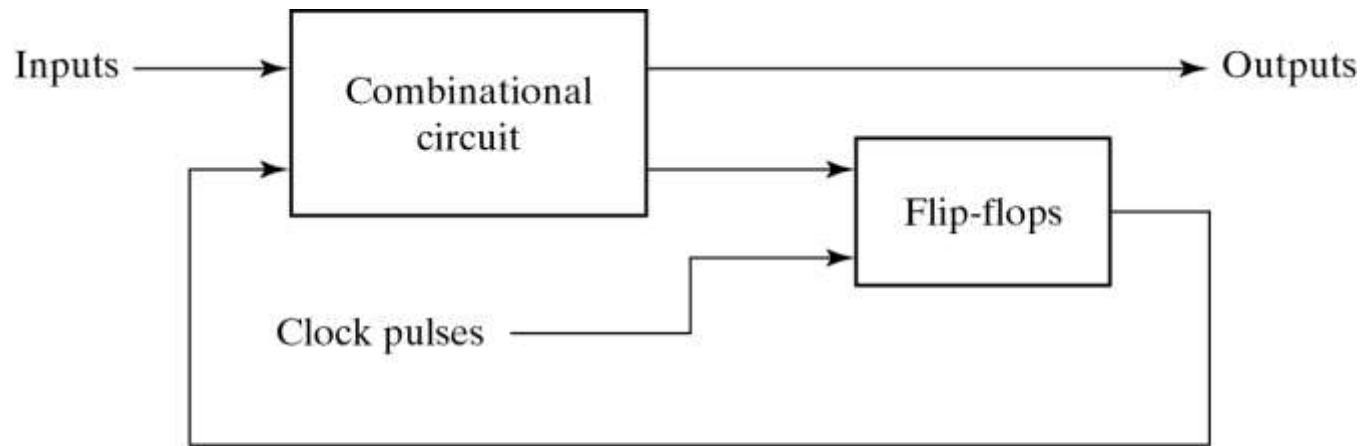


TWO TYPES OF SEQUENTIAL CIRCUITS

- Synchronous sequential circuit
 - circuit output changes only at some discrete instants of time.
 - Synchronized by a periodic train of clock pulses
 - Much easier to design (preferred design style)
- Asynchronous sequential circuit
 - circuit output can change at **any** time (clockless)
 - May have better performance but hard to design



SYNCHRONOUS SEQUENTIAL CIRCUITS



(a) Block diagram



(b) Timing diagram of clock pulses

Fig. 5-2 Synchronous Clocked Sequential Circuit

MEMORY ELEMENTS

- A digital circuit that can maintain a binary state indefinitely, until directed by an input signal to switch states
- Differences among different types of storage elements depends on
 - The number of inputs they possess
 - The manner in which the inputs affect the binary state



MEMORY ELEMENTS

- Latch - □ a level-sensitive memory element
 - SR latches
 - D latches
- Flip-Flop □ an edge-triggered memory element
 - Master-slave flip-flop
 - Edge-triggered flip-flop
- RAM and ROM □ a mass memory element



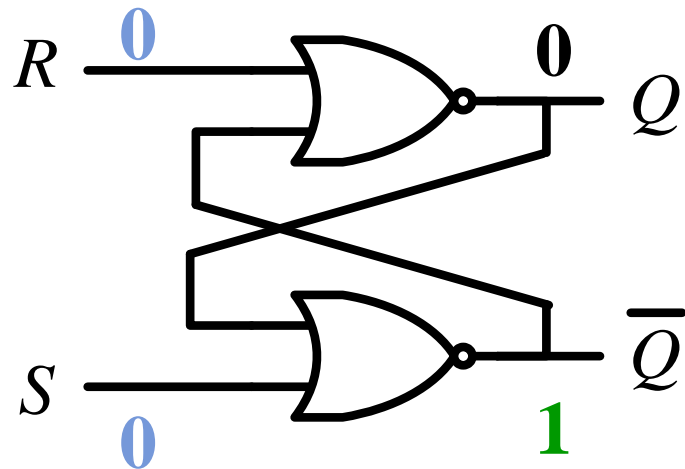
LATCHES

- A latch is binary storage element
- Can store a 0 or 1
- The most basic memory
- Easy to build
- Built with gates (NORs, NANDs, NOT)



LATCHES

SR Latch



S	R	Q_0	Q	Q'
0	0	0	0	1

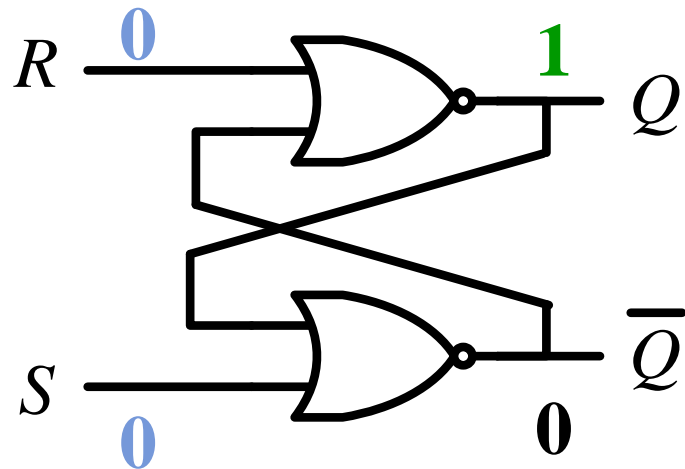
$Q = Q_0$

Initial Value



LATCHES

- SR Latch



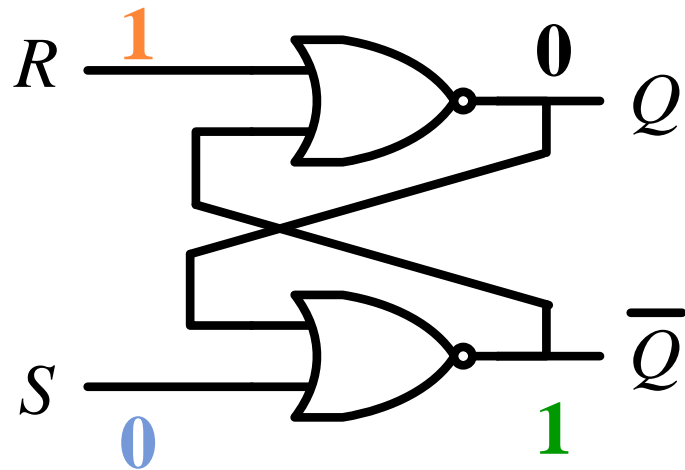
S	R	Q_0	Q	Q'
0	0	0	0	1
0	0	1	1	0

$Q = Q_0$
 $Q = Q_0$



LATCHES

- SR Latch



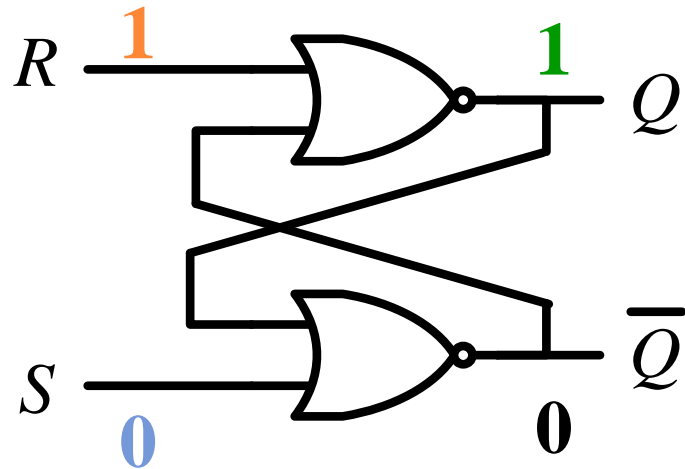
S	R	Q_0	Q	Q'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1

} $Q = Q_0$
 $Q = 0$



LATCHES

- SR Latch



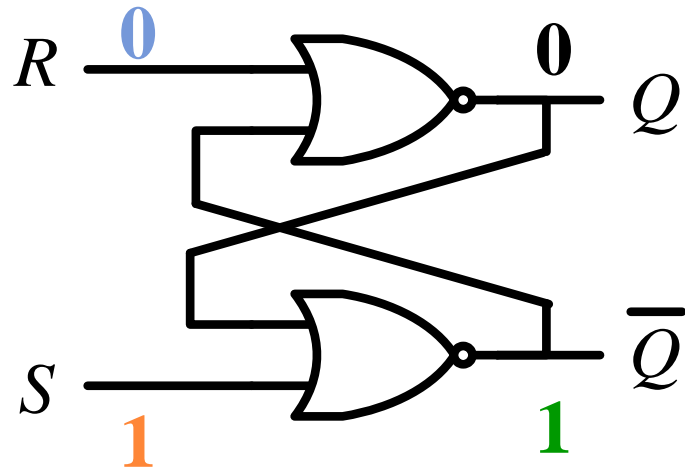
S	R	Q_0	Q	Q'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1

$\left. \begin{array}{l} \text{Row 1: } Q = Q_0 \\ \text{Row 2: } Q = Q_0 \end{array} \right\} Q = Q_0$
 $Q = 0$
 $Q = 0$



LATCHES

- SR Latch

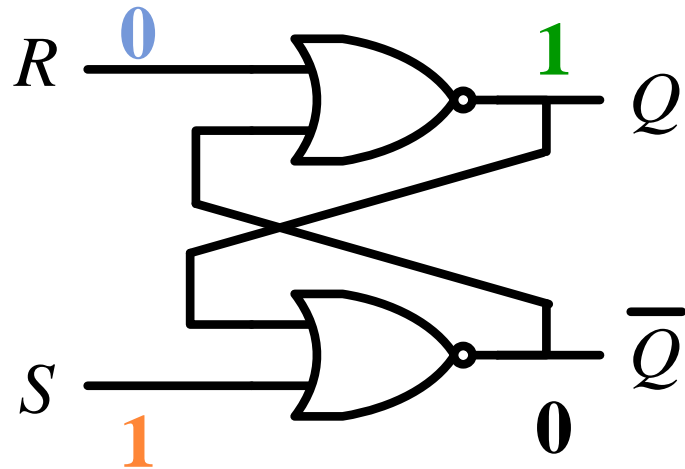


S	R	Q_0	Q	Q'	
0	0	0	0	1	} $Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	} $Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$



LATCHES

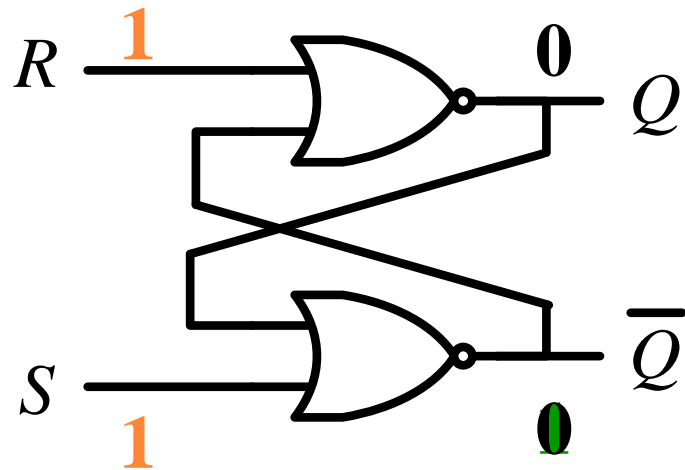
- SR Latch



S	R	Q_0	Q	Q'	
0	0	0	0	1	} $Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	} $Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	$Q = 1$

LATCHES

- SR Latch

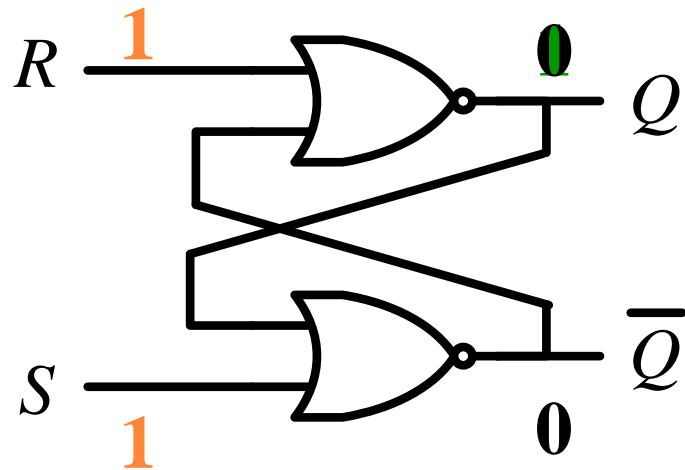


S	R	Q_0	Q	Q'	
0	0	0	0	1	} $Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	} $Q = 0$
0	1	1	0	1	
1	0	0	1	0	} $Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$



LATCHES

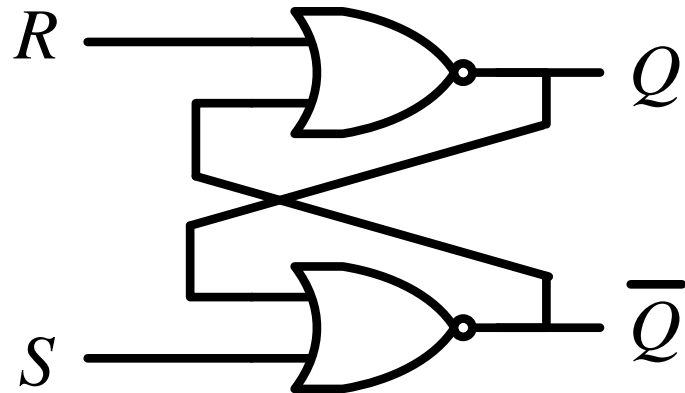
- SR Latch



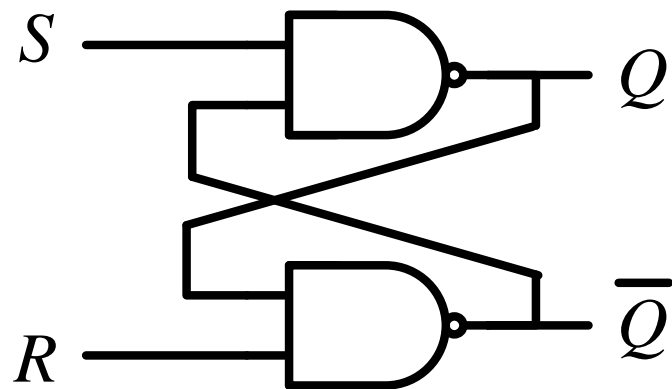
S	R	Q_0	Q	Q'	
0	0	0	0	1	} $Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	} $Q = 0$
0	1	1	0	1	
1	0	0	1	0	} $Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$
1	1	1	0	0	$Q = Q'$



SR LATCH

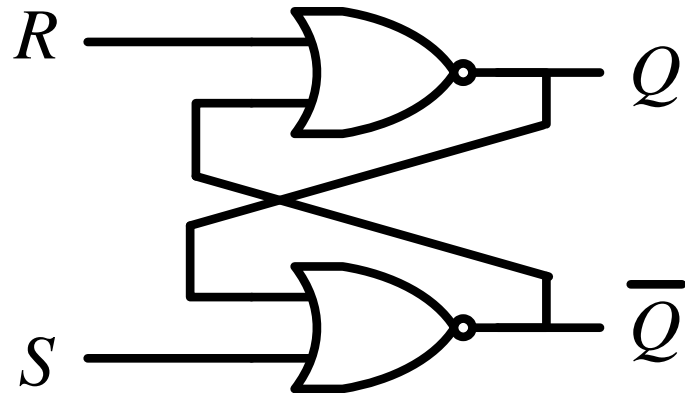


S	R	Q	
0	0	Q_0	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q=Q'=0$	Invalid

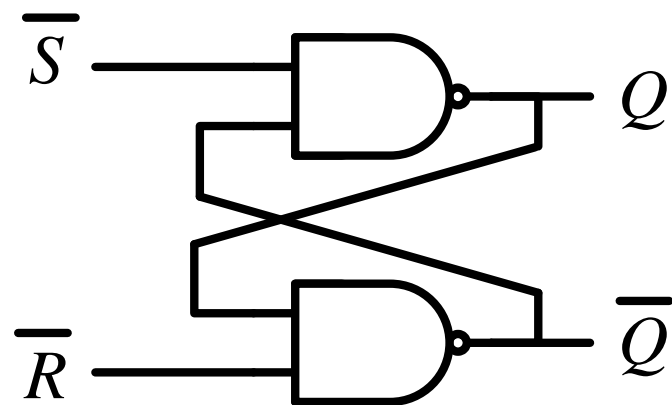


S	R	Q	
0	0	$Q=Q'=1$	Invalid
0	1	1	Set
1	0	0	Reset
1	1	Q_0	No change

SR LATCH



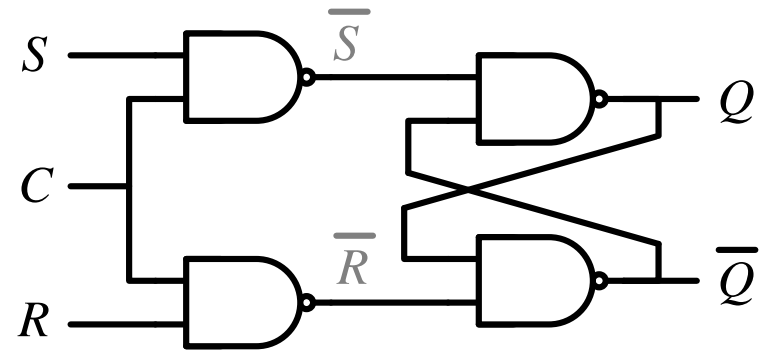
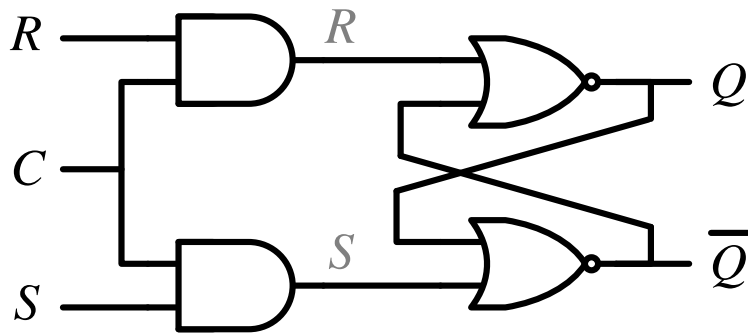
S	R	Q	
0	0	Q_0	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q=Q'=0$	Invalid



S'	R'	Q	
0	0	$Q=Q'=1$	Invalid
0	1	1	Set
1	0	0	Reset
1	1	Q_0	No change

CONTROLLED LATCHES

SR Latch with Control Input



C	S	R	Q
0	x	x	Q_0
1	0	0	Q_0
1	0	1	0
1	1	0	1
1	1	1	$Q=Q'$

No change

No change

Reset

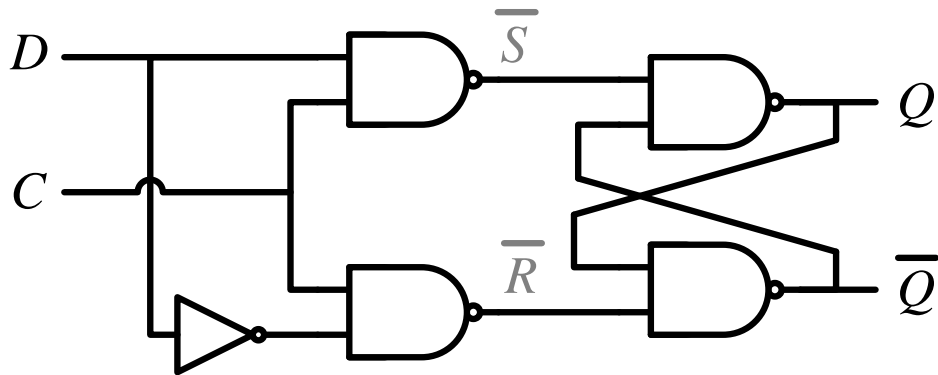
Set

Invalid



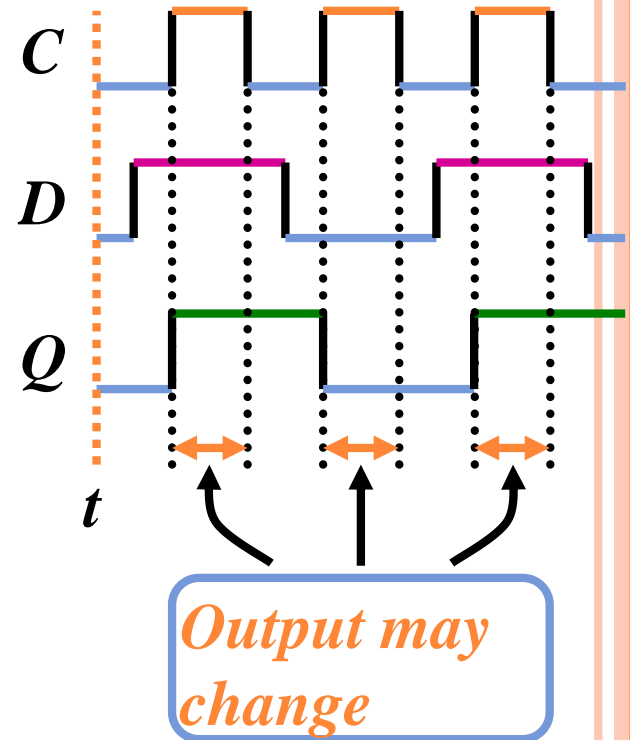
CONTROLLED LATCHES

- D Latch (D = Data)



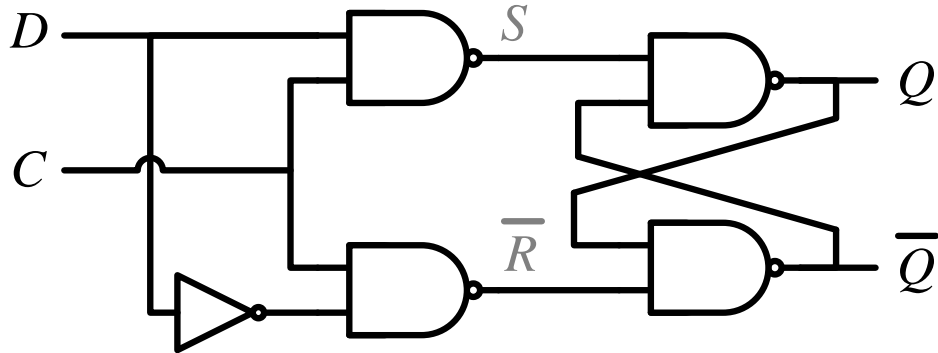
C	D	Q	
0	x	Q_0	No change
1	0	0	Reset
1	1	1	Set

Timing Diagram



CONTROLLED LATCHES

- D Latch (D = Data)



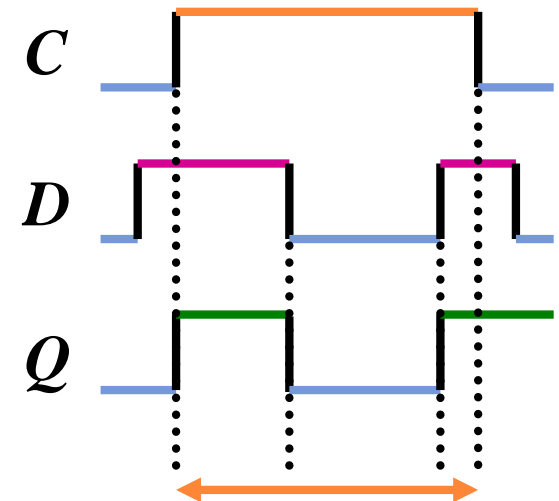
C	D	Q
0	x	Q_0
1	0	0
1	1	1

No change

Reset

Set

Timing Diagram



Output may change



GRAPHIC SYMBOLS FOR LATCHES

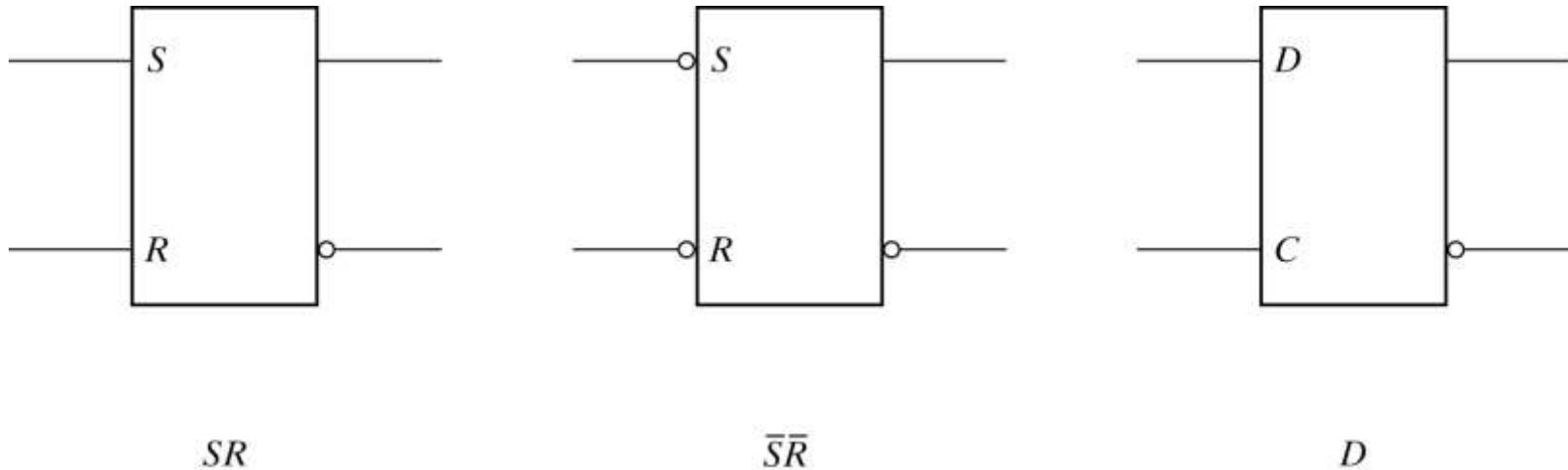


Fig. 5-7 Graphic Symbols for Latches



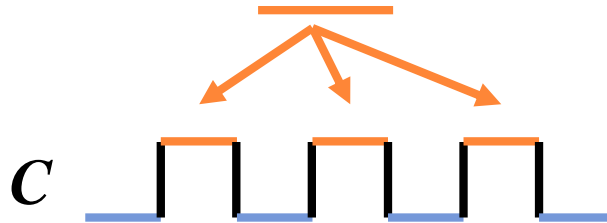
LEVEL VERSUS EDGE SENSITIVITY

- Since the output of the D latch is controlled by the level (0 or 1) of the clock input, the latch is said to be *level sensitive*
 - All of the latches we have seen have been level sensitive
- It is possible to design a storage element for which the output only changes at the point in time when the clock changes from one value to another
 - Such circuits are said to be *edge triggered*



FLIP-FLOPS

- Controlled latches are **level**-triggered

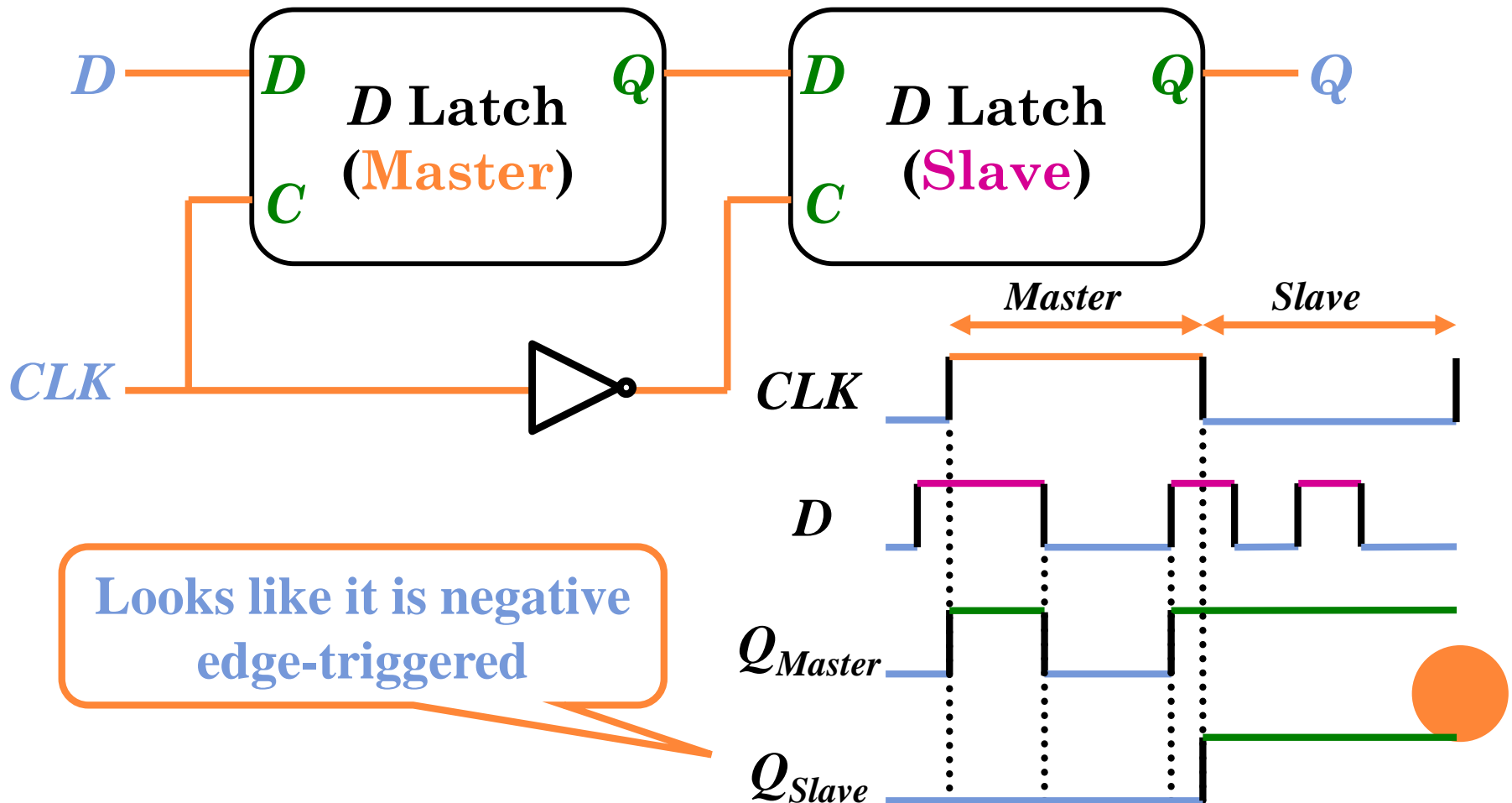


- Flip-Flops are **edge**-triggered



FLIP-FLOPS

- Master-Slave *D* Flip-Flop



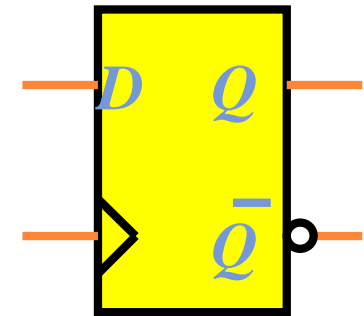
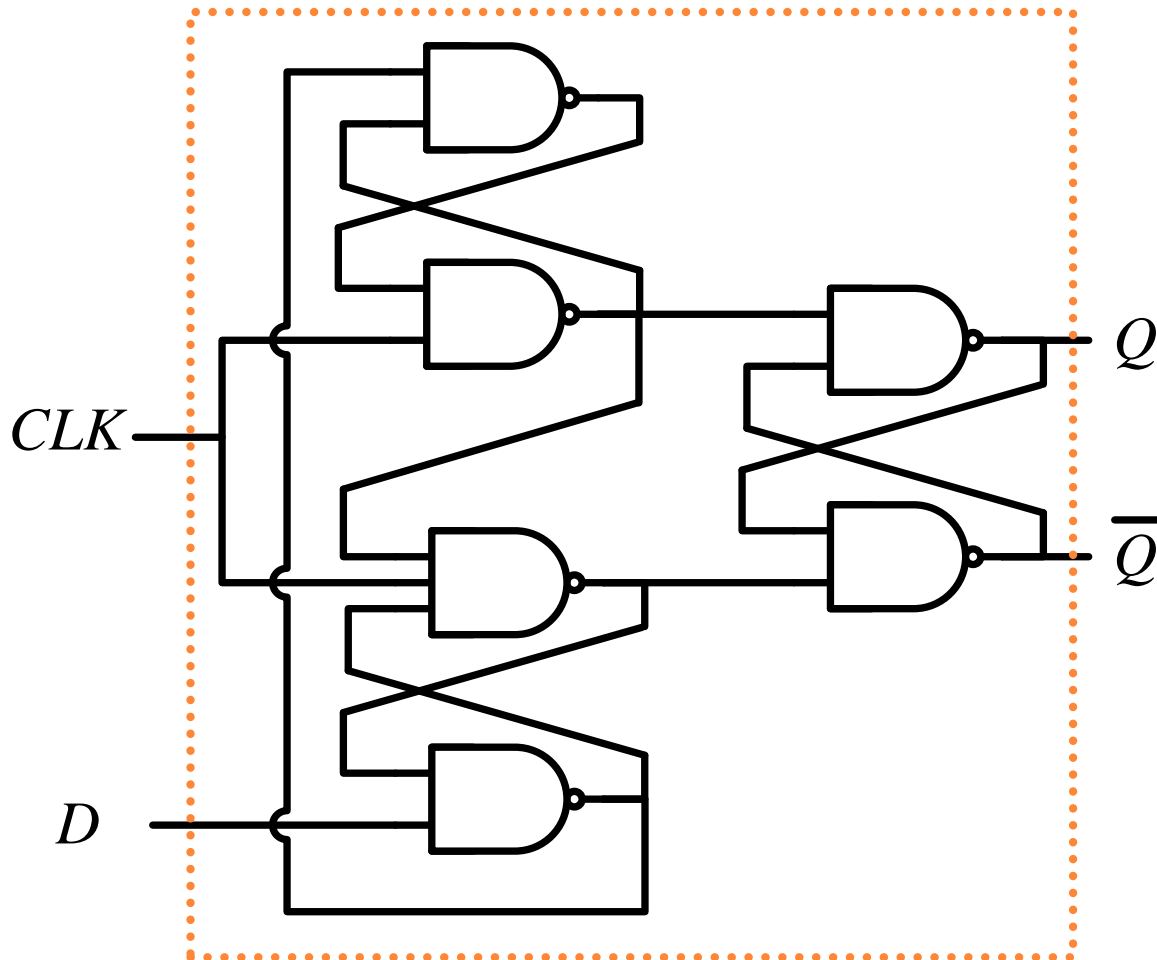
FLIP-FLOPS

- The circuit samples the D input and changes its output at the negative edge of the clock, CLK .
- When the clock is 0, the output of the inverter is 1. The slave latch is enabled and its output Q is equal to the master output Y . The master latch is disabled ($CLK = 0$).
- When the CLK changes to high, D input is transferred to the master latch. The slave remains disabled as long as CLK is low. Any change in the input changes Y , but not Q .
- The output of the flip-flop can change when CLK makes a transition $1 \rightarrow 0$

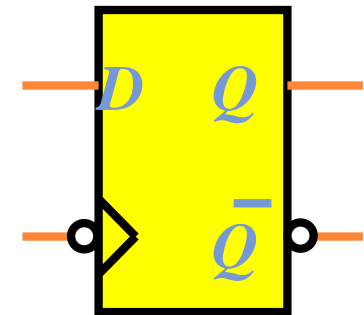


FLIP-FLOPS

- Edge-Triggered D Flip-Flop



Positive Edge

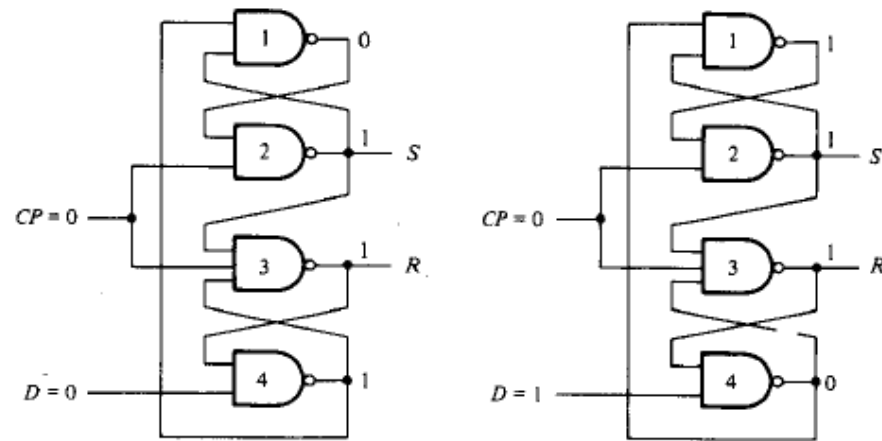


Negative Edge

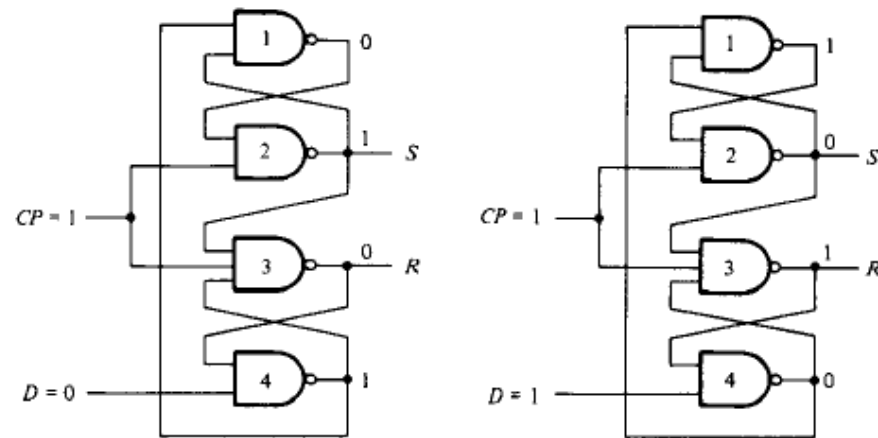


FLIP-FLOPS:

EDGE-TRIGGERED D FLIP-FLOP



(a) With $CP = 0$



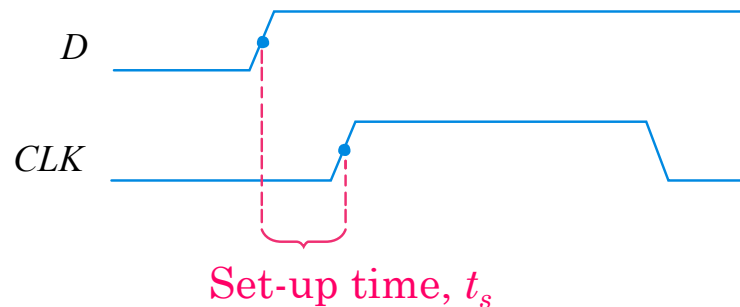
(b) With $CP = 1$



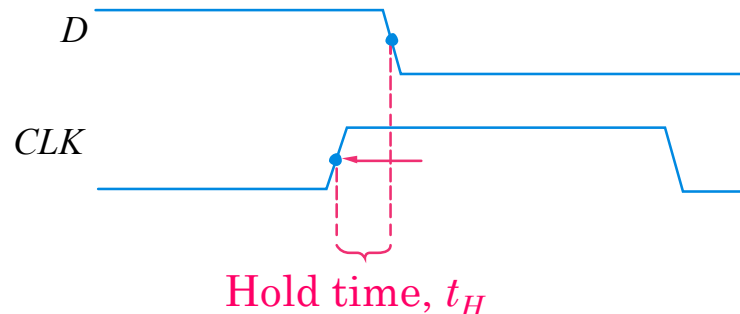
FLIP-FLOPS

Set-up time and **hold time** are times required before and after the clock transition that data must be present to be reliably clocked into the flip-flop.

Setup time is the minimum time for the data to be present *before* the clock.



Hold time is the minimum time for the data to *remain* after the clock.



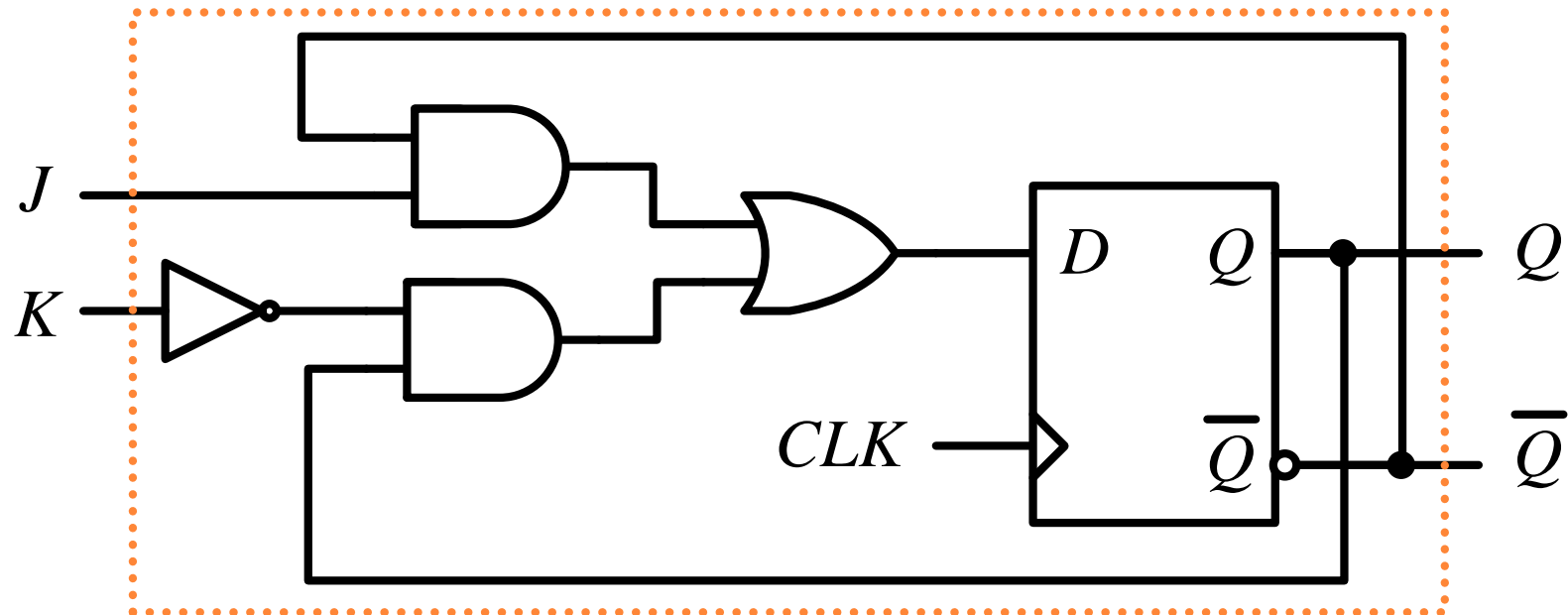
JK FLIP-FLOPS

- JK Flip-Flop : $D = JQ' + K'Q$
- When $J = 1$ and $K = 0$, $D = 1 \rightarrow$ next clock edge sets output to 1.
- When $J = 0$ and $K = 1$, $D = 0 \rightarrow$ next clock edge resets output to 0.
- When $J = 1$ and $K = 1$, $D = Q' \rightarrow$ next clock edge complements output.
- When $J = 0$ and $K = 0$, $D = Q \rightarrow$ next clock edge leaves output unchanged.

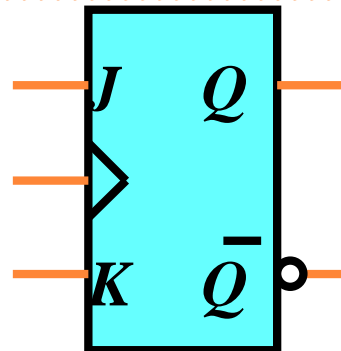


JK FLIP-FLOPS

JK Flip-Flop

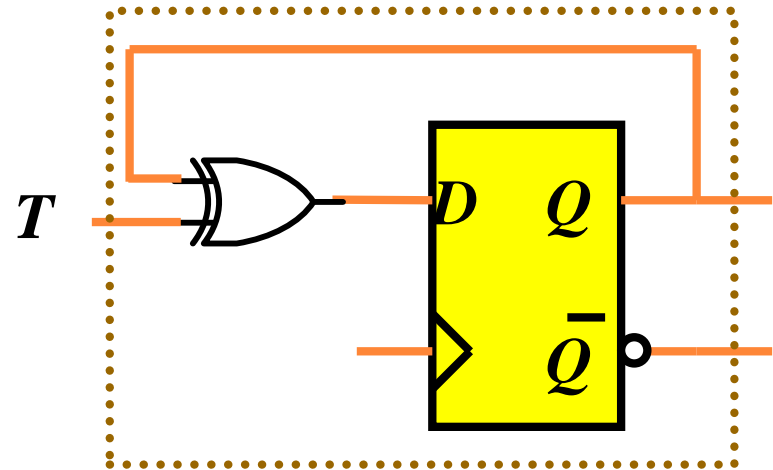
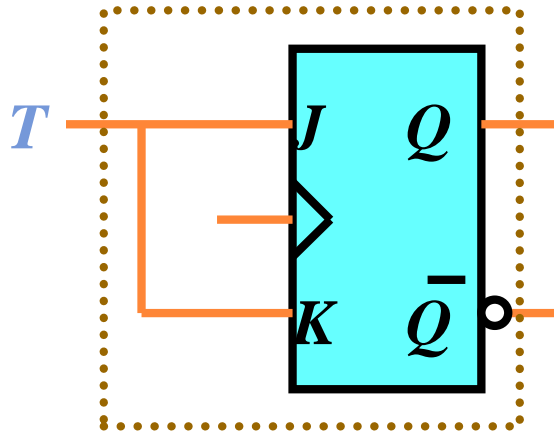


$$D = JQ' + K'Q$$



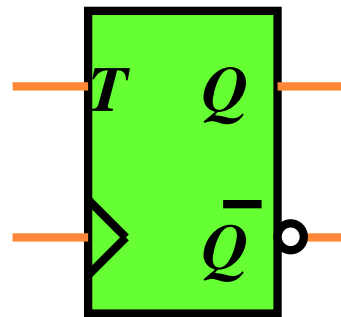
T FLIP-FLOPS

- T Flip-Flop

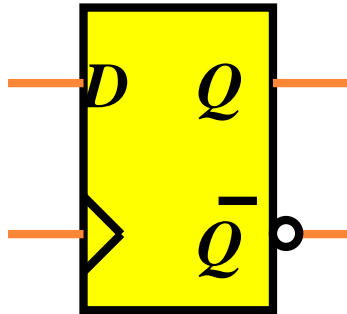


$$D = JQ' + K'Q$$

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



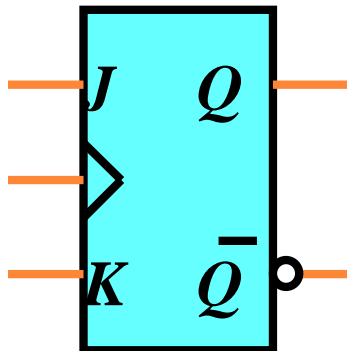
FLIP-FLOP CHARACTERISTIC TABLES



D	$Q(t+1)$
0	0
1	1

Reset

Set



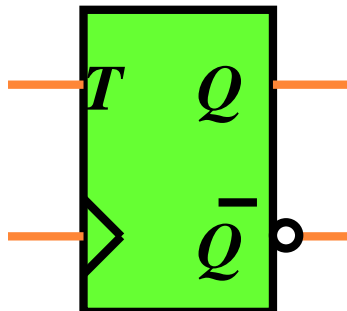
J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

No change

Reset

Set

Toggle



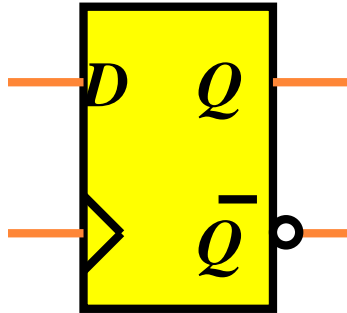
T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

No change

Toggle

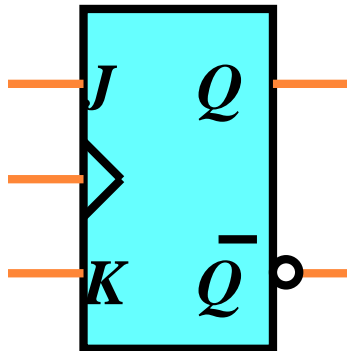


FLIP-FLOP CHARACTERISTIC EQUATIONS



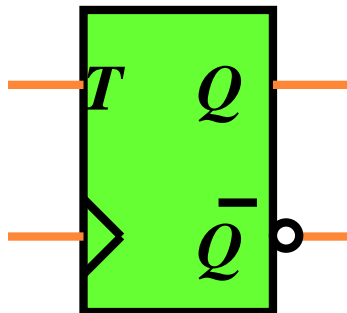
D	$Q(t+1)$
0	0
1	1

$$Q(t+1) = D$$



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

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



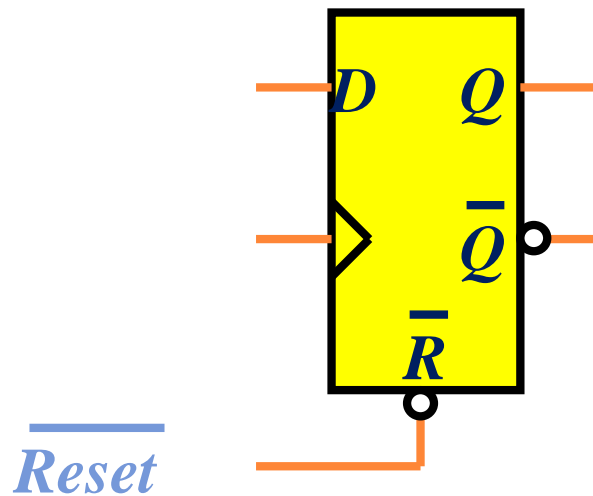
T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

$$Q(t+1) = T \oplus Q$$



FLIP-FLOPS WITH ASYNCHRONOUS/DIRECT INPUTS

- Asynchronous Reset

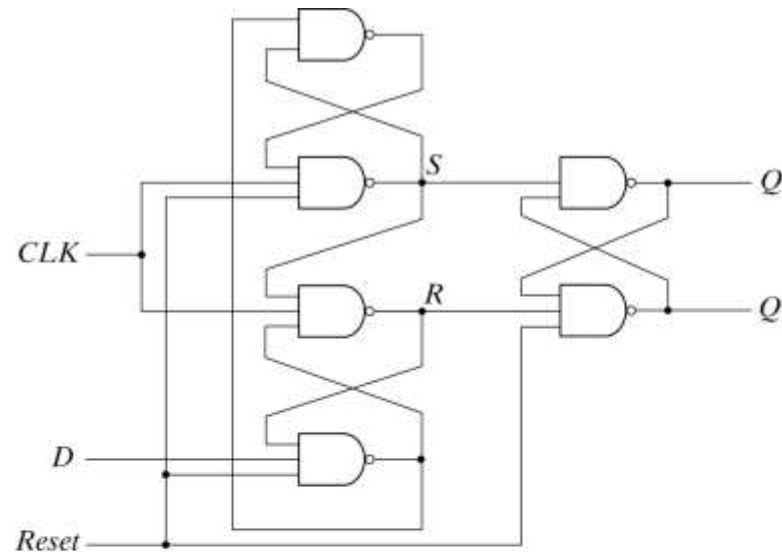


R'	D	CLK	$Q(t+1)$
0	x	x	0
1	0	↑	0
1	1	↑	1

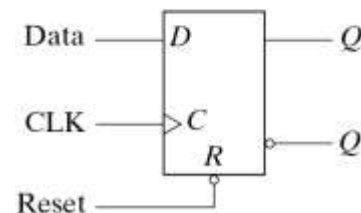


FLIP-FLOPS WITH DIRECT INPUTS

- Asynchronous Reset



(a) Circuit diagram



(b) Graphic symbol

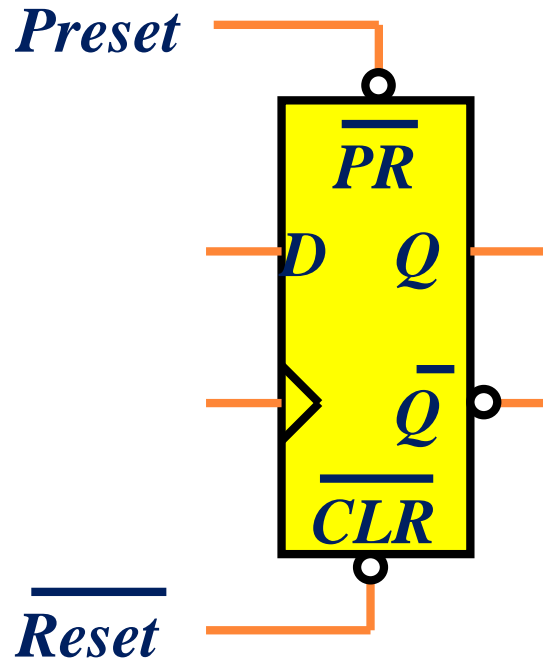
R	C	D	Q	Q'
0	X	X	0	1
1	\uparrow	0	0	1
1	\uparrow	1	1	0

(b) Function table

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

FLIP-FLOPS WITH DIRECT INPUTS

- Asynchronous Preset and Clear



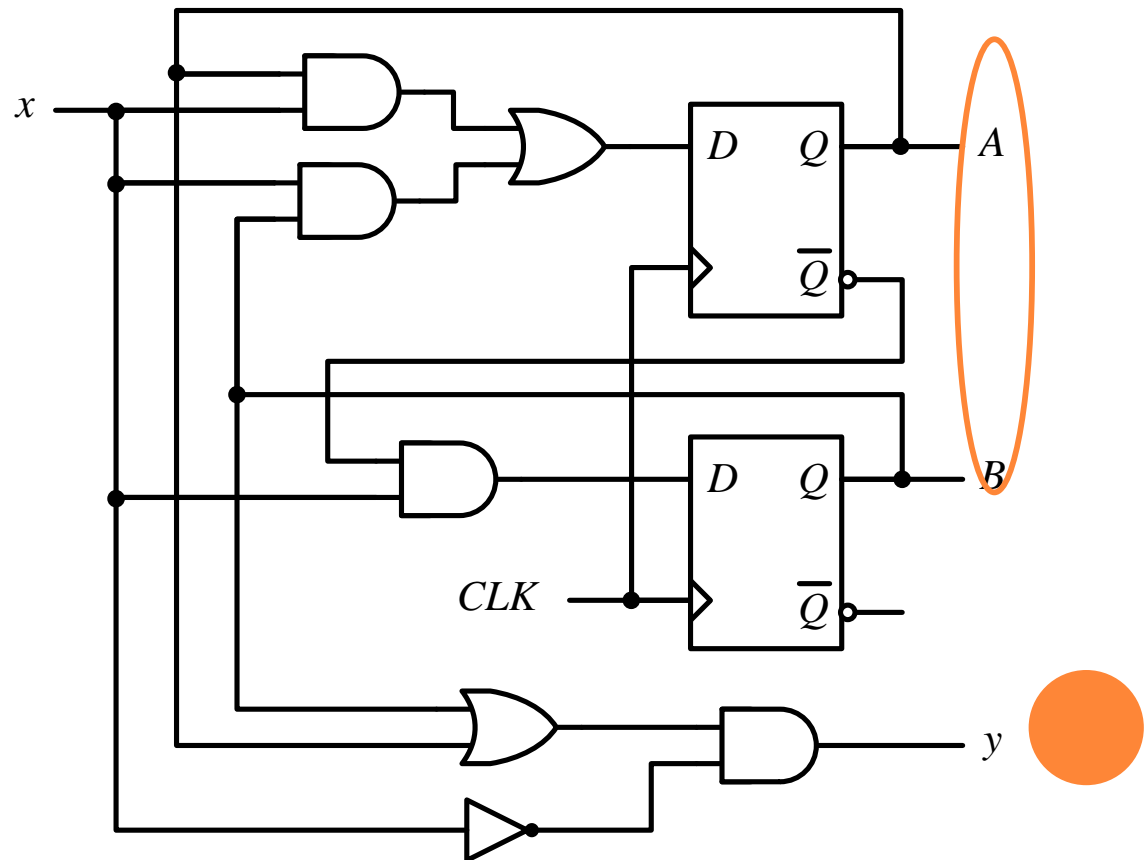
PR'	CLR'	D	CLK	$Q(t+1)$
1	0	x	x	0
0	1	x	x	1
1	1	0	\uparrow	0
1	1	1	\uparrow	1

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: THE STATE

- State = Values of all Flip-Flops

Example

$A B = 0 0$



ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: TERMINOLOGY

- *State Equation*: A state equation (transition equation) specifies the next state as a function of the present state and inputs.
- *State Table*: A state table (transition table) consists of: present state, input, next state and output.
- *State Diagram*: The information in a state table can be represented graphically in a state diagram. The state is represented by a circle and the transitions between states are indicated by directed lines connecting the circles.

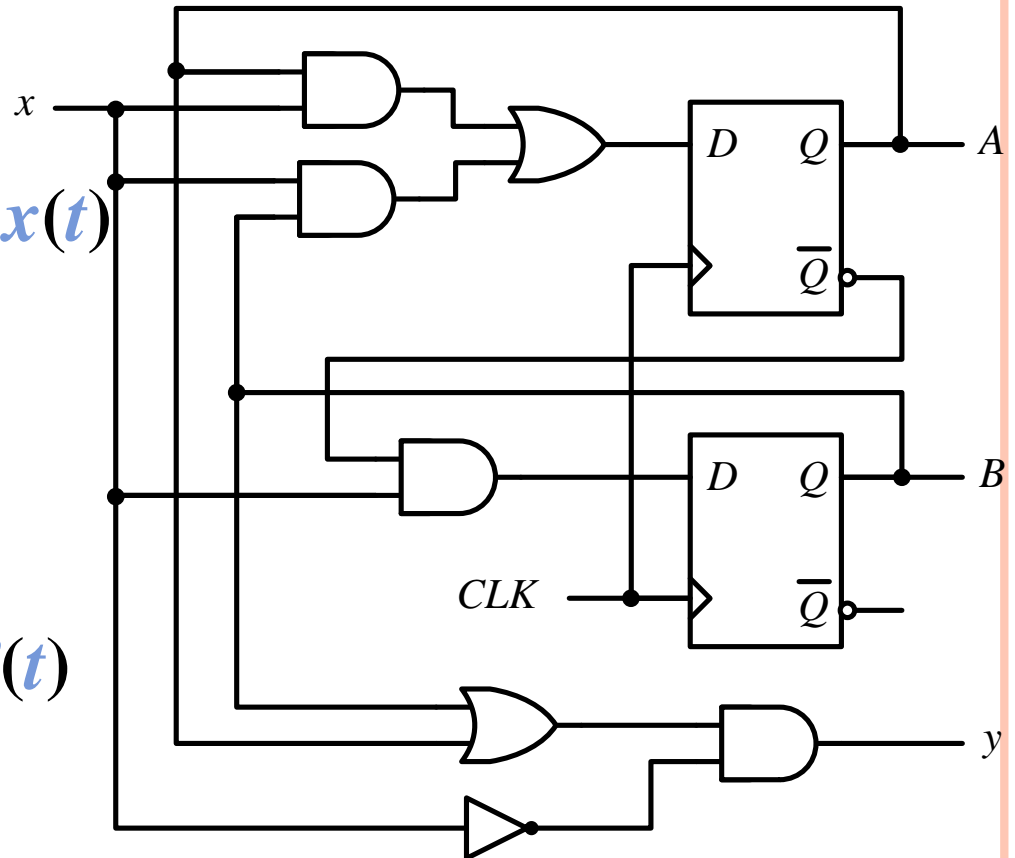


ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: STATE/TRANSITION EQUATIONS

$$\begin{aligned} A(t+1) &= D_A \\ &= A(t) x(t) + B(t) x(t) \\ &= A x + B x \end{aligned}$$

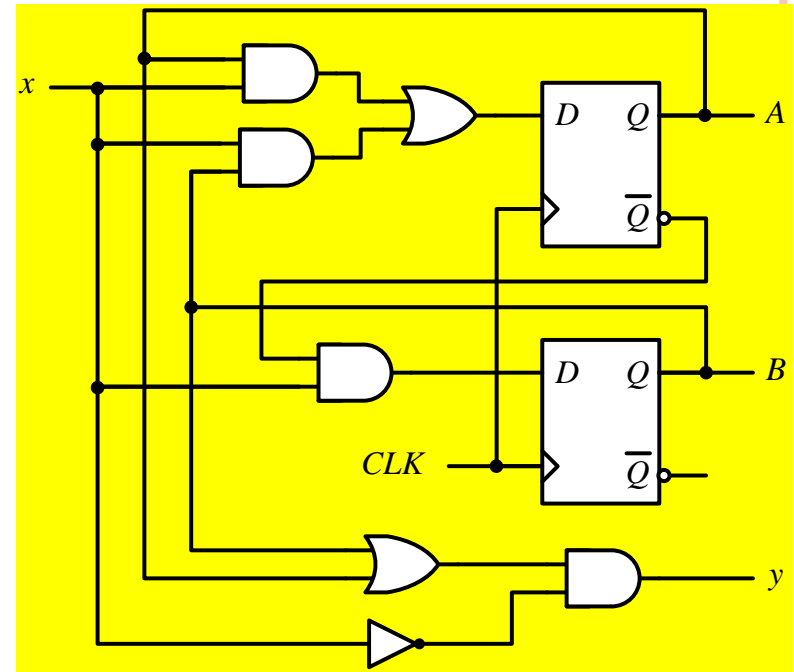
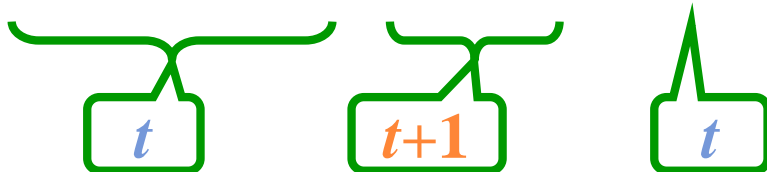
$$\begin{aligned} B(t+1) &= D_B \\ &= A'(t) x(t) \\ &= A' x \end{aligned}$$

$$\begin{aligned} y(t) &= [A(t) + B(t)] x'(t) \\ &= (A + B) x' \end{aligned}$$



ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: STATE /TRANSITION TABLE

Present State		Input	Next State		Output
<i>A</i>	<i>B</i>	<i>x</i>	<i>A</i>	<i>B</i>	<i>y</i>
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



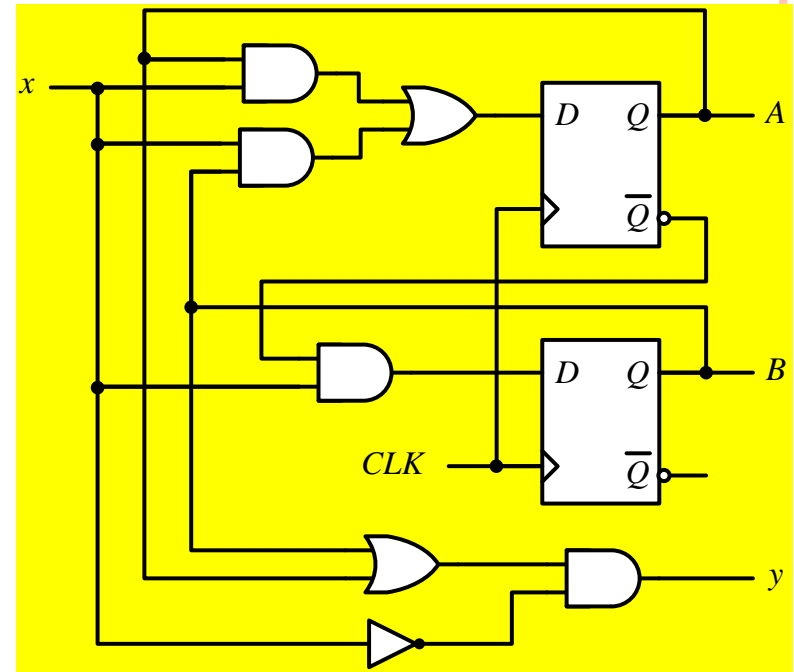
$$A(t+1) = Ax + Bx$$

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

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

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: STATE/TRANSITION 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

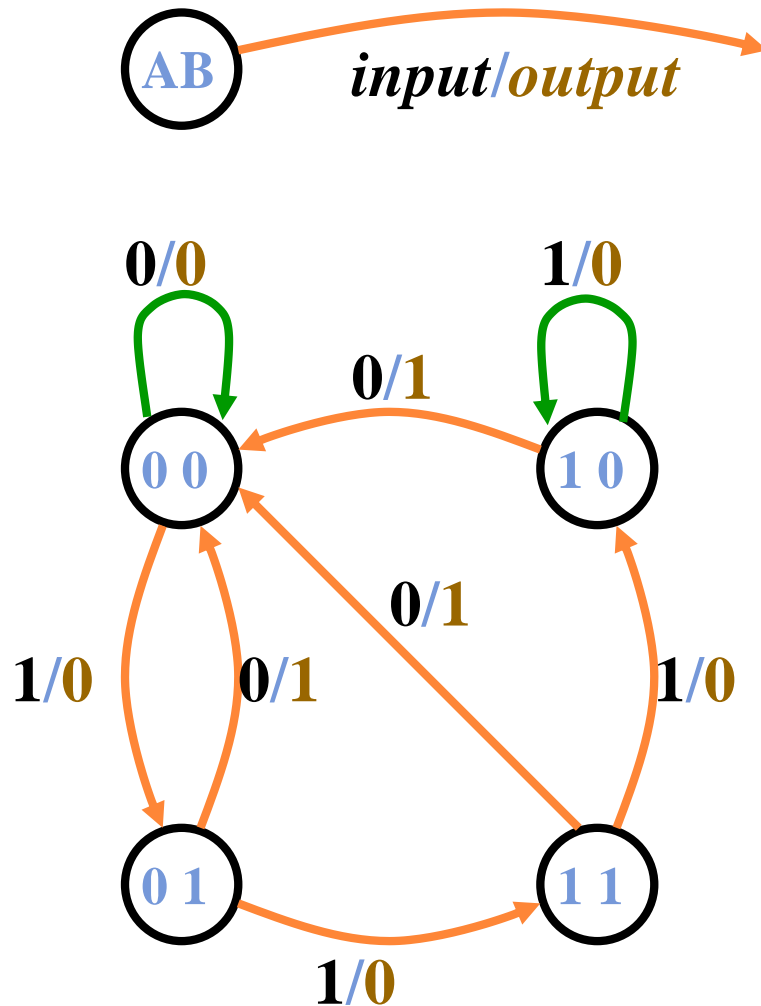


$$A(t+1) = A x + B x$$

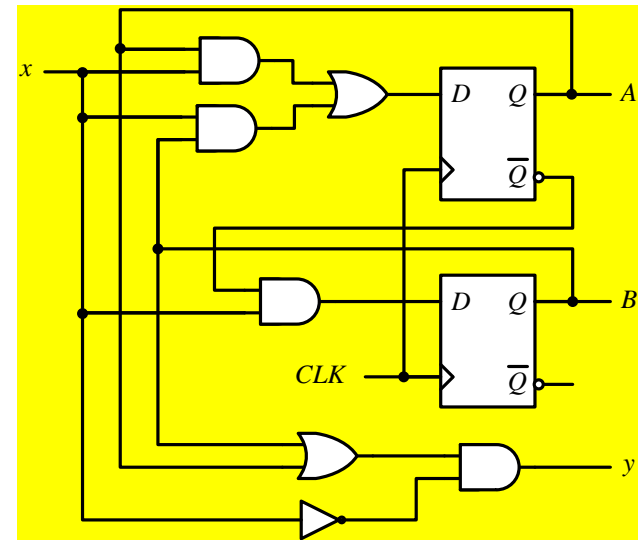
$$B(t+1) = A'x$$

$$y(t) = (A + B) x,$$

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: STATE DIAGRAM



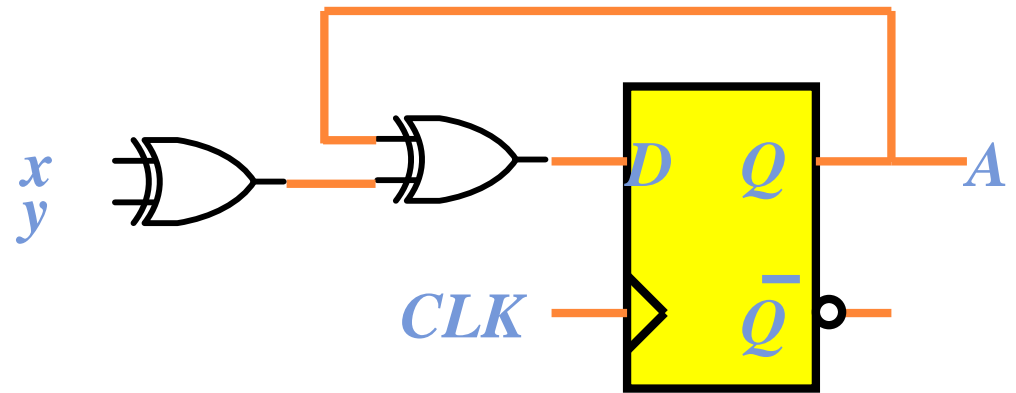
Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
$A \ B$	$A \ B$	$A \ B$	y	y
0 0	0 0	0 1	0	0
0 1	0 0	1 1	1	0
1 0	0 0	1 0	1	0
1 1	0 0	1 0	1	0



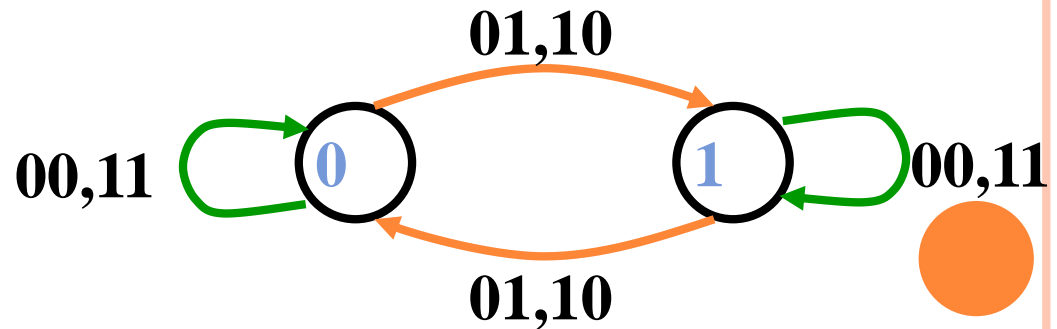
ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: *D* FLIP-FLOPS

Example:

Present State	Input		Next State
<i>A</i>	<i>x</i>	<i>y</i>	<i>A</i>
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



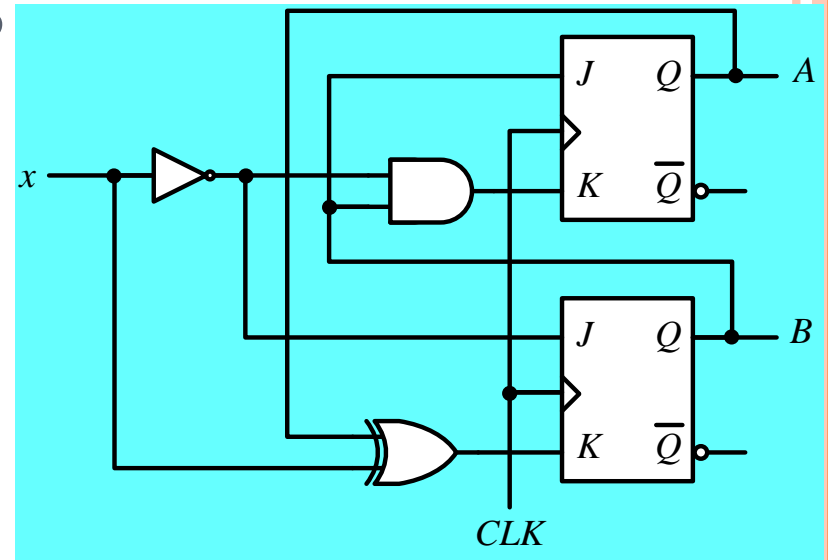
$$A(t+1) = D_A = A \oplus x \oplus y$$



ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: *JK* FLIP-FLOPS

Example:

Present State		I/P	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



$$J_A = B$$

$$K_A = Bx'$$

$$J_B = x'$$

$$K_B = A \oplus x$$

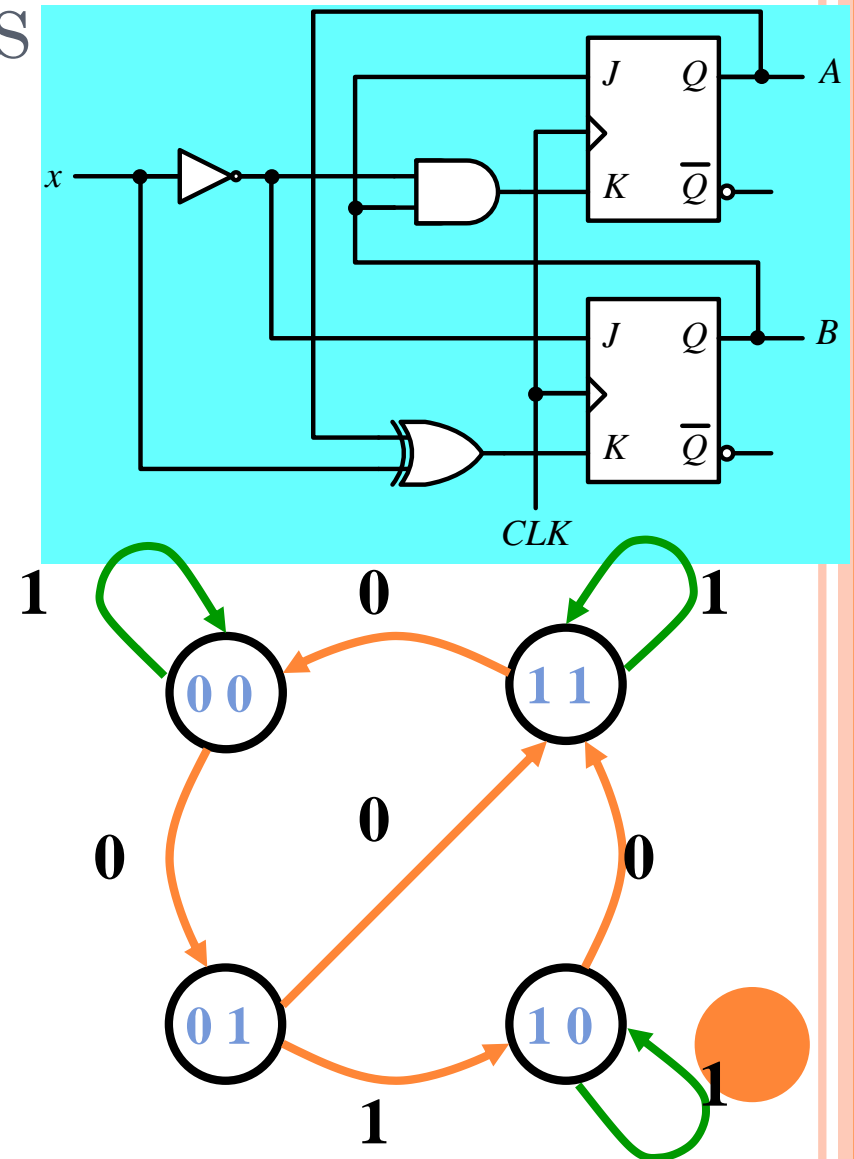
$$\begin{aligned} A(t+1) &= J_A Q'_A + K'_A Q_A \\ &= A'B + AB' + Ax \end{aligned}$$

$$\begin{aligned} B(t+1) &= J_B Q'_B + K'_B Q_B \\ &= B'x' + ABx + A'Bx' \end{aligned}$$

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: JK FLIP-FLOPS

Example:

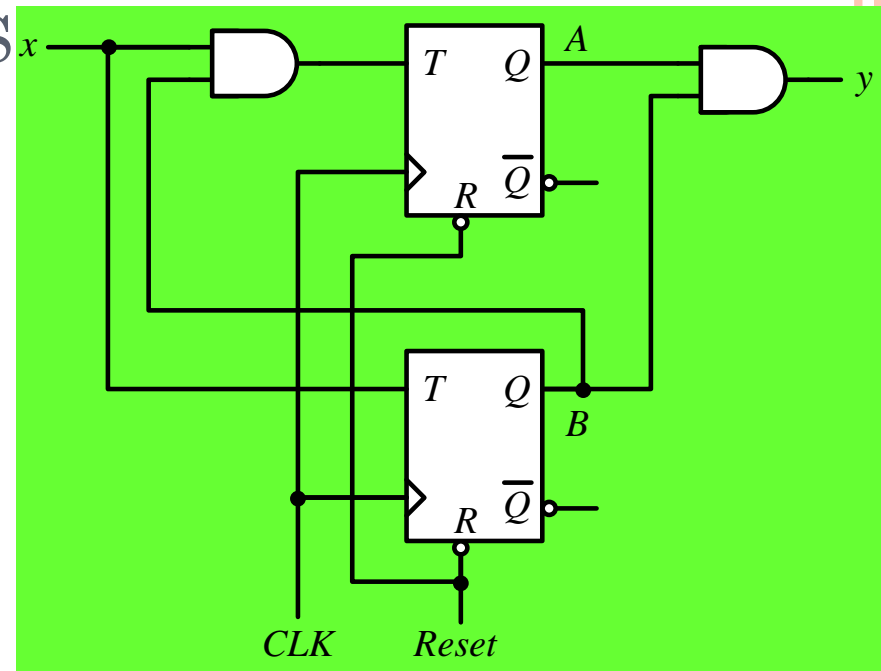
Present State		I/P	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0



ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: *T* FLIP-FLOPS

Example:

Present State		I/P	Next State		F.F Inputs		O/P
A	B	x	A	B	T_A	T_B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	0	0	0
0	1	1	1	0	1	1	0
1	0	0	1	0	0	0	0
1	0	1	1	1	0	1	0
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	1



$$T_A = Bx$$

$$T_B = x$$

$$y = AB$$

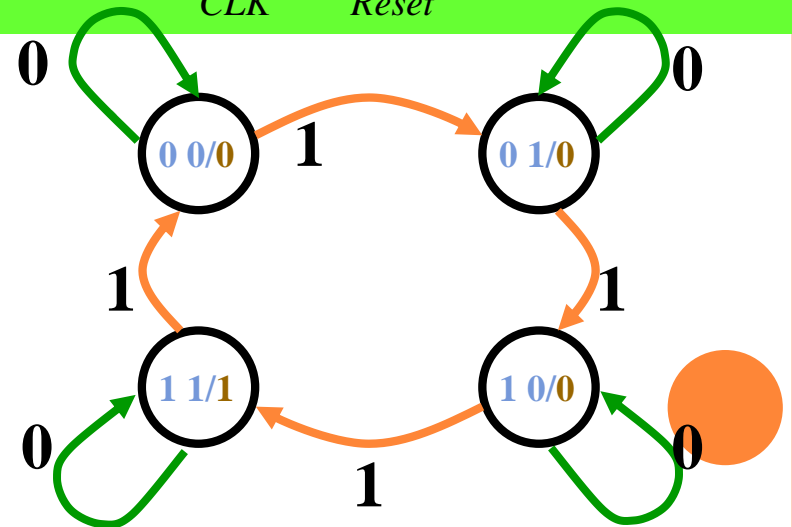
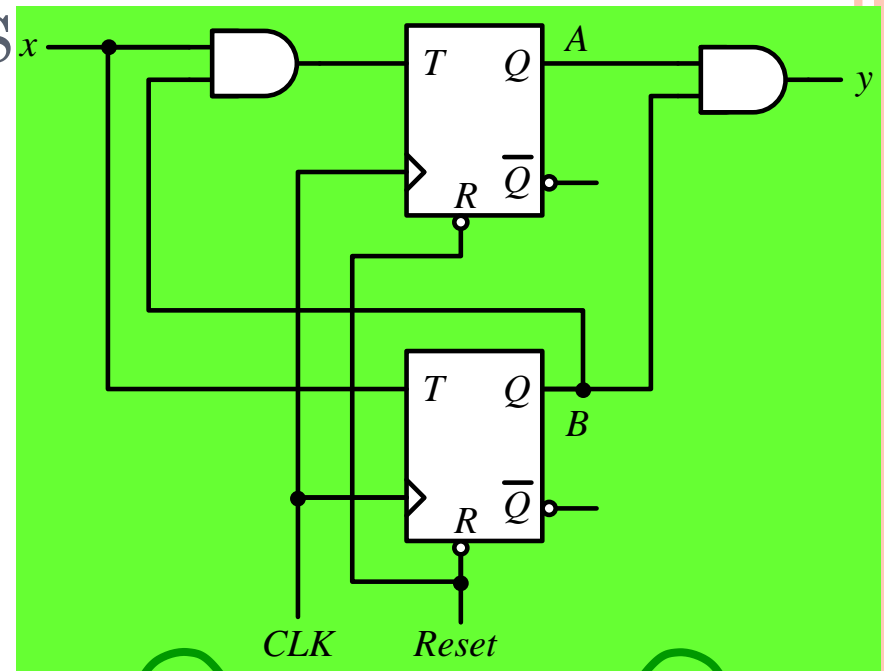
$$\begin{aligned} A(t+1) &= T_A Q'_A + T'_A Q_A \\ &= AB' + Ax' + A'Bx \end{aligned}$$

$$\begin{aligned} B(t+1) &= T_B Q'_B + T'_B Q_B \\ &= x \oplus B \end{aligned}$$

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS: *T* FLIP-FLOPS

Example:

Present State		I/P	Next State		F.F Inputs		O/P
A	B	x	A	B	T_A	T_B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	0	0	0
0	1	1	1	0	1	1	0
1	0	0	1	0	0	0	0
1	0	1	1	1	0	1	0
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	1



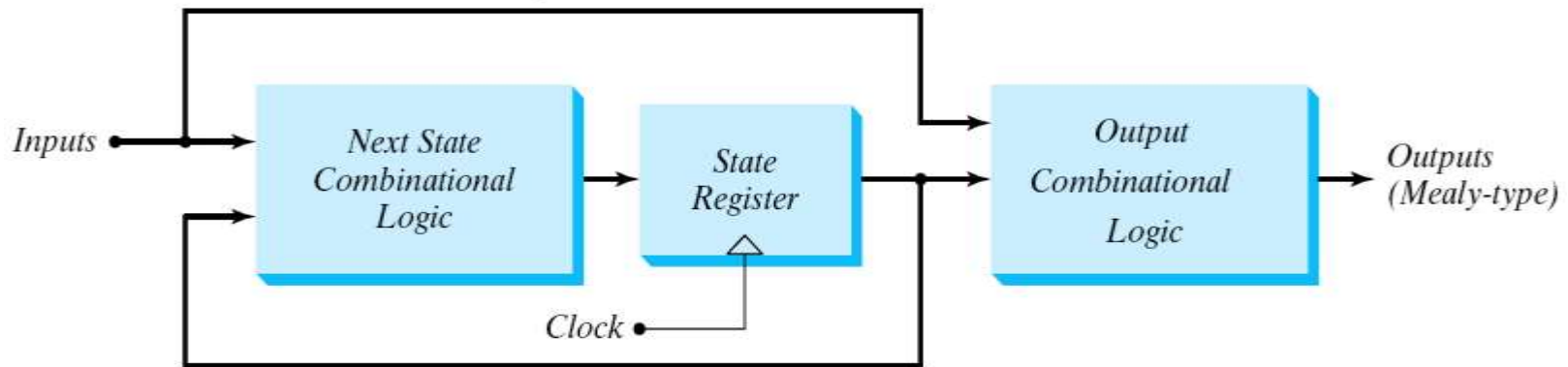
MEALY AND MOORE MODELS

- **The Mealy model:** the outputs are functions of both the present state and inputs
 - The outputs may change if the inputs change during the clock pulse period.
- **The Moore model:** the outputs are functions of the present state only
 - The outputs are synchronous with the clocks.



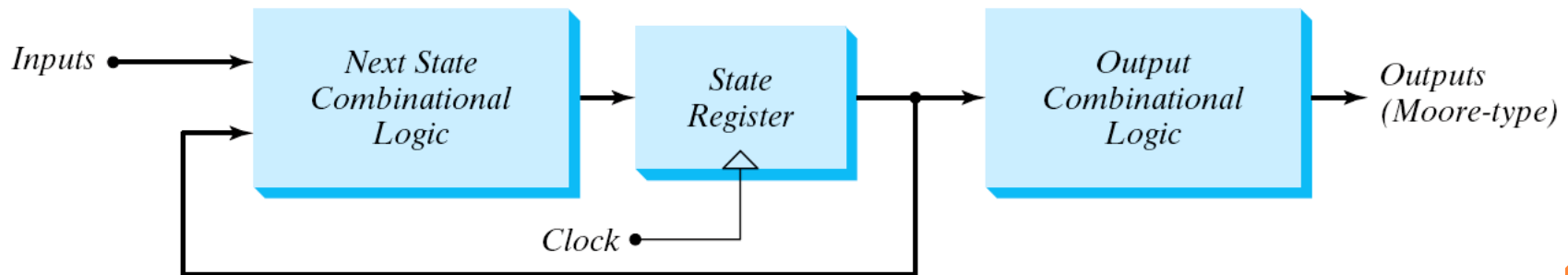
MEALY AND MOORE MODELS

Mealy Machine



(a)

Moore Machine



(b)

Block diagram of Mealy and Moore state machine



MEALY AND MOORE MODELS

Mealy

Present State		I/P	Next State		O/P
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

For the same *state*,
the *output* changes with the *input*

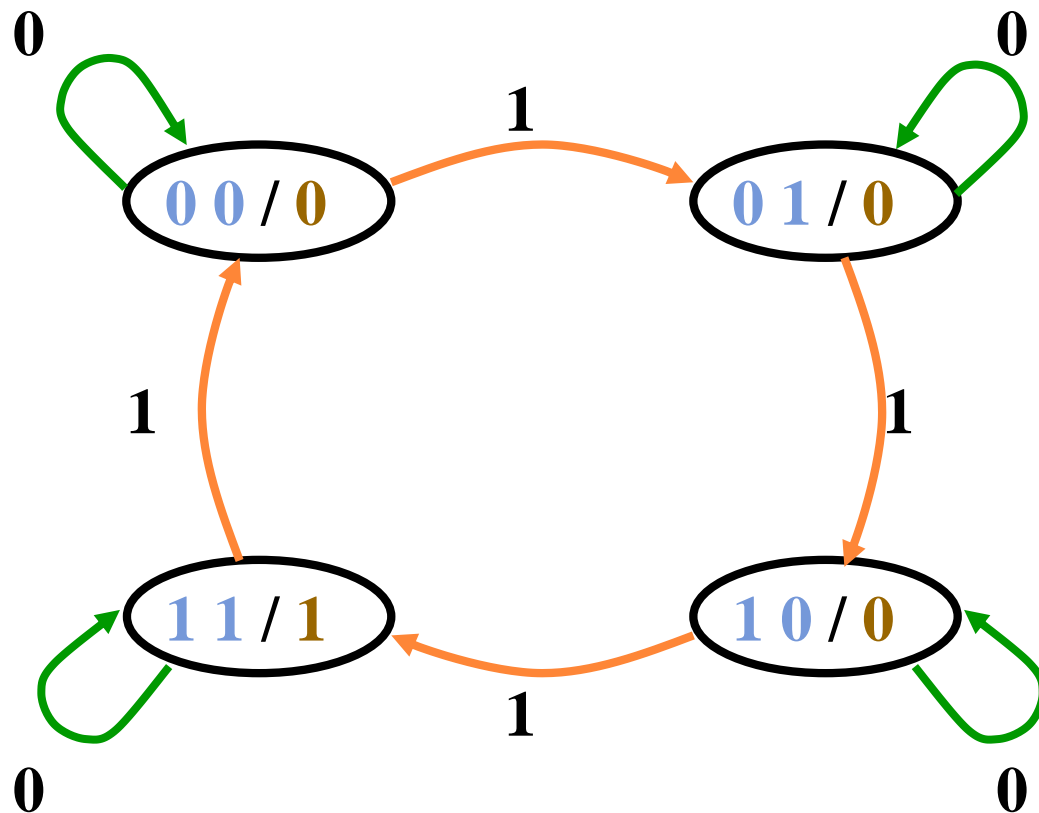
Moore

Present State		I/P	Next State		O/P
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

For the same *state*,
the *output* does not change with the *input*

MOORE STATE DIAGRAM

State / Output



STATE REDUCTION

- Sequential circuit analysis

Circuit diagram \longrightarrow state table (or state diagram)

- Sequential circuit design

State diagram (state table) \longrightarrow circuit diagram

- Redundant state may exist in a state diagram (or table)

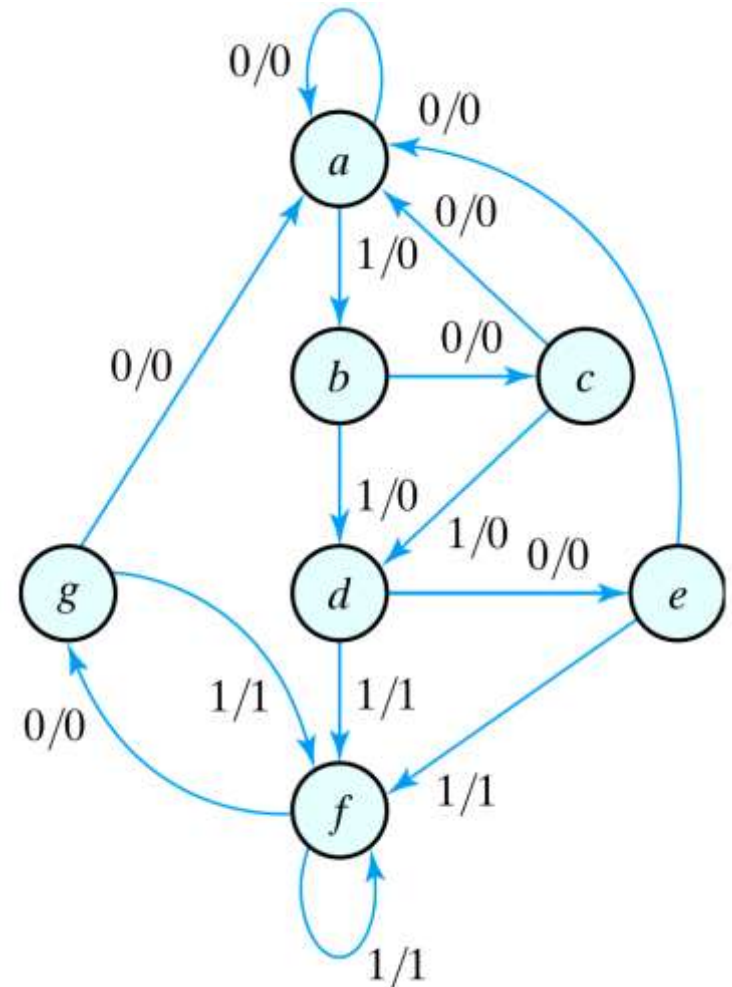
- By eliminating them \longrightarrow may reduce the # of flip-flops



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.



State diagram

STATE REDUCTION

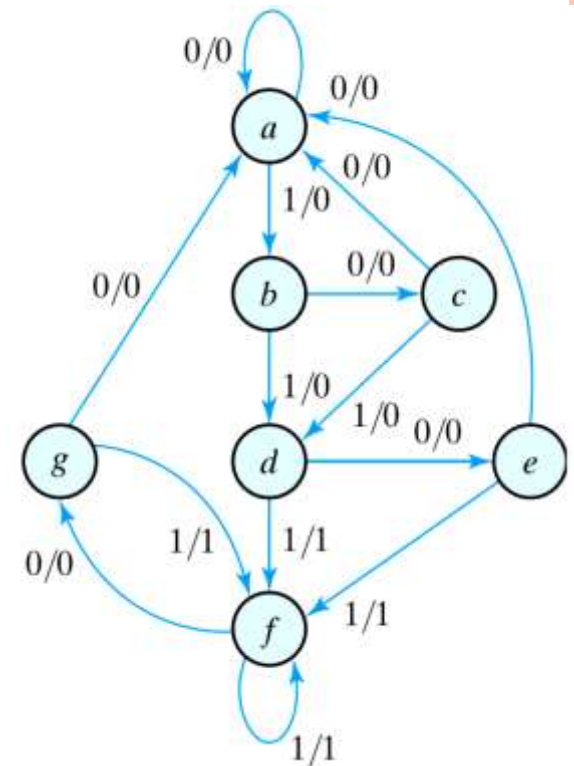
- Equivalent states
 - Two states are said to be equivalent
 - For each member of the set of inputs, they give exactly the same output and send the circuit to the same state or to an equivalent state.
 - One of them can be removed.



STATE REDUCTION

Table 5.6
State Table

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



STATE REDUCTION

- Reducing the state table
 - $e = g$ (remove g);
 - $d = f$ (remove f);

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



STATE REDUCTION

- The reduced finite state machine

Table 5.8

Reduced State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>d</i>	0	1
<i>e</i>	<i>a</i>	<i>d</i>	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: IMPLICATION TABLE

- The state-reduction procedure for completely specified state tables is based on the algorithm that two states in a state table can be combined into one if they can be shown to be equivalent.
- There are occasions when a pair of states do not have the same next states, but, nonetheless, go to equivalent next states
- The checking of each pair of states for possible equivalence in a table with a large number of states can be done systematically by means of an *implication table*.



STATE REDUCTION: IMPLICATION TABLE

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

- (a, b) imply (c, d) and (c, d) imply (a, b) . Both pairs of states are equivalent; i.e., a and b are equivalent as well as c and d .



STATE REDUCTION:IMPLICATION TABLE

<i>b</i>	<i>d, e</i> ✓					
<i>c</i>	×	×				
<i>d</i>	×	×	×			
<i>e</i>	×	×	×	✓		
<i>f</i>	<i>c, d</i> ×	<i>c, e</i> × <i>a, b</i>	×	×	×	
<i>g</i>	×	×	×	<i>d, e</i> ✓	<i>d, e</i> ✓	×
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>

Present State	Next State		Output	
	<i>x</i> = 0	<i>x</i> = 1	<i>x</i> = 0	<i>x</i> = 1
<i>a</i>	<i>d</i>	<i>g</i>	0	0
<i>b</i>	<i>e</i>	<i>a</i>	0	0
<i>c</i>	<i>g</i>	<i>f</i>	0	1
<i>d</i>	<i>a</i>	<i>d</i>	1	0
<i>e</i>	<i>a</i>	<i>d</i>	1	0
<i>f</i>	<i>c</i>	<i>b</i>	0	0
<i>g</i>	<i>a</i>	<i>e</i>	1	0



STATE REDUCTION: IMPLICATION TABLE

- Finally, all the squares that have no crosses are recorded with check marks. The equivalent states are: (a, b) , (d, e) , (d, g) , (e, g) .
- We now combine pairs of states into larger groups of equivalent states. The last three pairs can be combined into a set of three equivalent states (d, e, g) because each one of the states in the group is equivalent to the other two.



STATE REDUCTION: IMPLICATION TABLE

- The final partition of these states consists of the equivalent states found from the implication table, together with all the remaining states in the state table that are not equivalent to any other state: (a, b) (c) (d, e, g) (f)

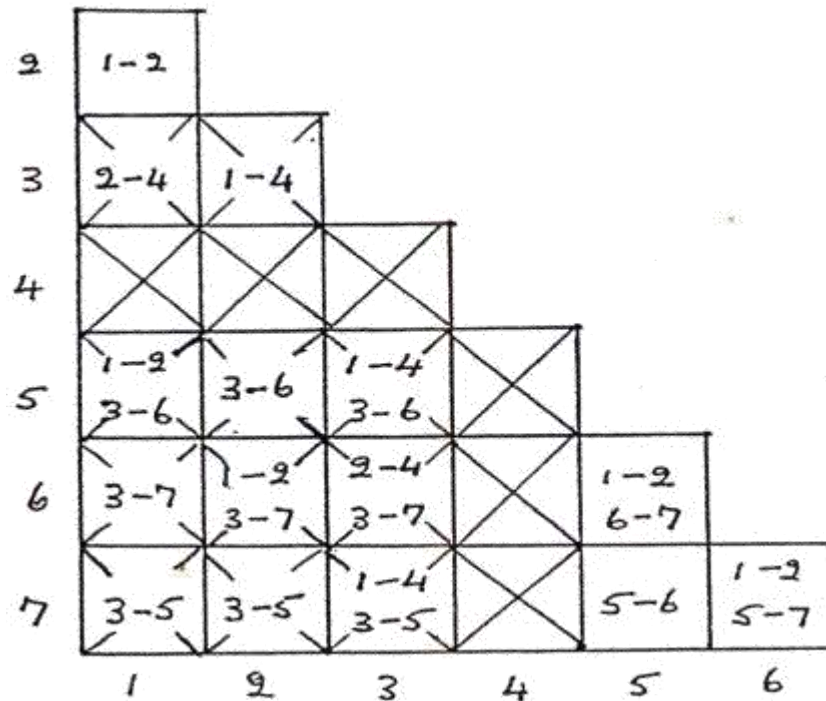
Table 9.5
Reduced State Table

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



IMPLIED EQUIVALENCES EXTENDING OVER A NUMBER OF STATES

PS	NS		Z
	x = 0	x = 1	
①	2	3	0
②	1	3	0
③	4	3	0
④	4	5	1
⑤	1	6	0
⑥	2	7	0
⑦	1	5	0



- **5 = 6** if 6 = 7, **5 = 7** if 5 = 6, **6 = 7** if 5 = 7
- **1 = 2** and **5 = 6**, **5 = 7**, **6 = 7** i.e. **5 = 6 = 7**



IMPLIED EQUIVALENCES EXTENDING OVER A NUMBER OF STATES

PS	NS		Z
	x = 0	x = 1	
①	2	3	0
②	1	3	0
③	4	3	0
④	4	5	1
⑤	1	6	0
⑥	2	7	0
⑦	1	5	0

PS	NS		Z
	x = 0	x = 1	
①	1	3	0
③	4	3	0
④	4	5	1
⑤	1	5	0

- **5 = 6** if **6 = 7**, **5 = 7** if **5 = 6**, **6 = 7** if **5 = 7**
- **1 = 2** and **5 = 6**, **5 = 7**, **6 = 7** i.e. **5 = 6 = 7**



IMPLIED EQUIVALENCES EXTENDING OVER A NUMBER OF STATES

Alternative definition of equivalence:

- Two circuit states with the same output conditions can be made equivalent, unless the equivalence depends upon a known non-equivalence.



STATE ASSIGNMENT

- Assign coded binary values to the states for physical implementation
- For a circuit with m states, the codes must contain n bits where $2^n \geq m$
- Unused states are treated as don't care conditions during the design
 - Don't cares can help to obtain a simpler circuit
- There are many possible state assignments
 - Have large impacts on the final circuit size



POPULAR STATE ASSIGNMENT

Table 5.9

Three Possible Binary State Assignments

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



STATE ASSIGNMENT

- Any binary number assignment is satisfactory as long as each state is assigned a unique number
- Use binary assignment 1

Table 5.10

Reduced State Table with Binary Assignment 1

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



DESIGN PROCEDURE

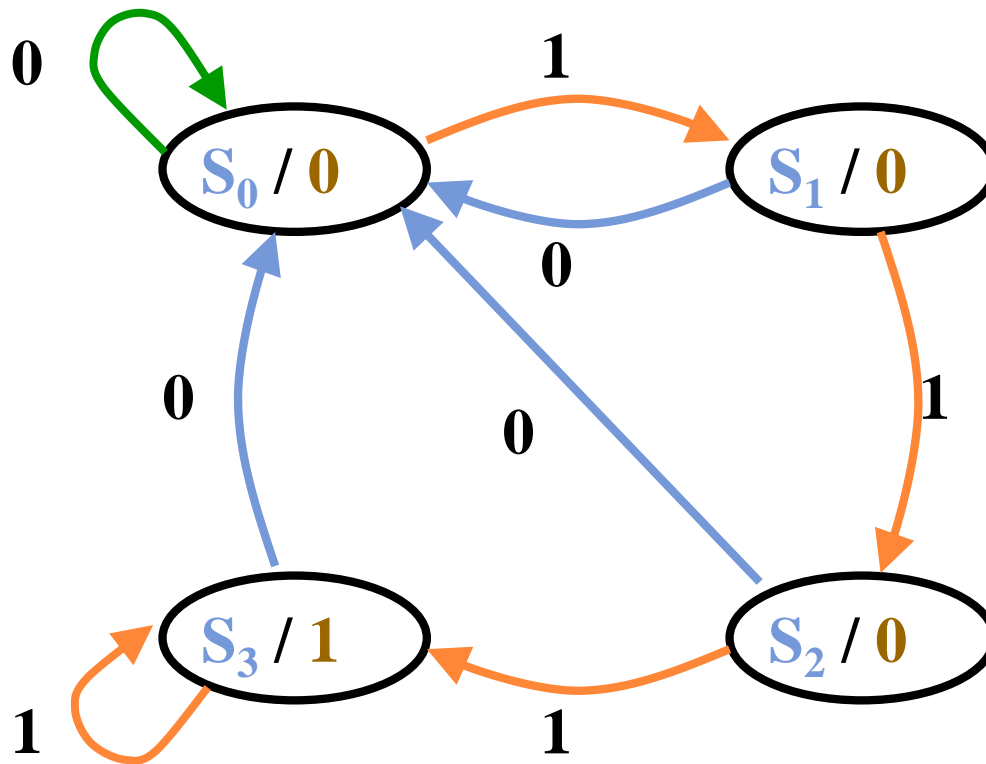
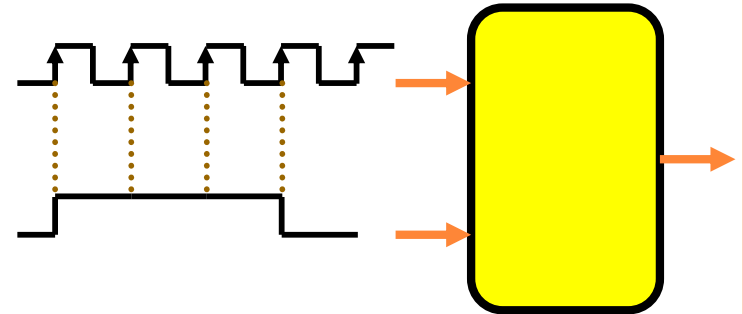
- Derive a state diagram for the circuit from specifications
- Reduce the number of states if necessary
- Assign binary values to the states
- Obtain the binary-coded state table
- Choose the type of flip-flop to be used
- Derive the simplified flip-flop input equations and output equations
- Draw the logic diagram



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS

○ *Example:*

Detect 3 or more consecutive 1's



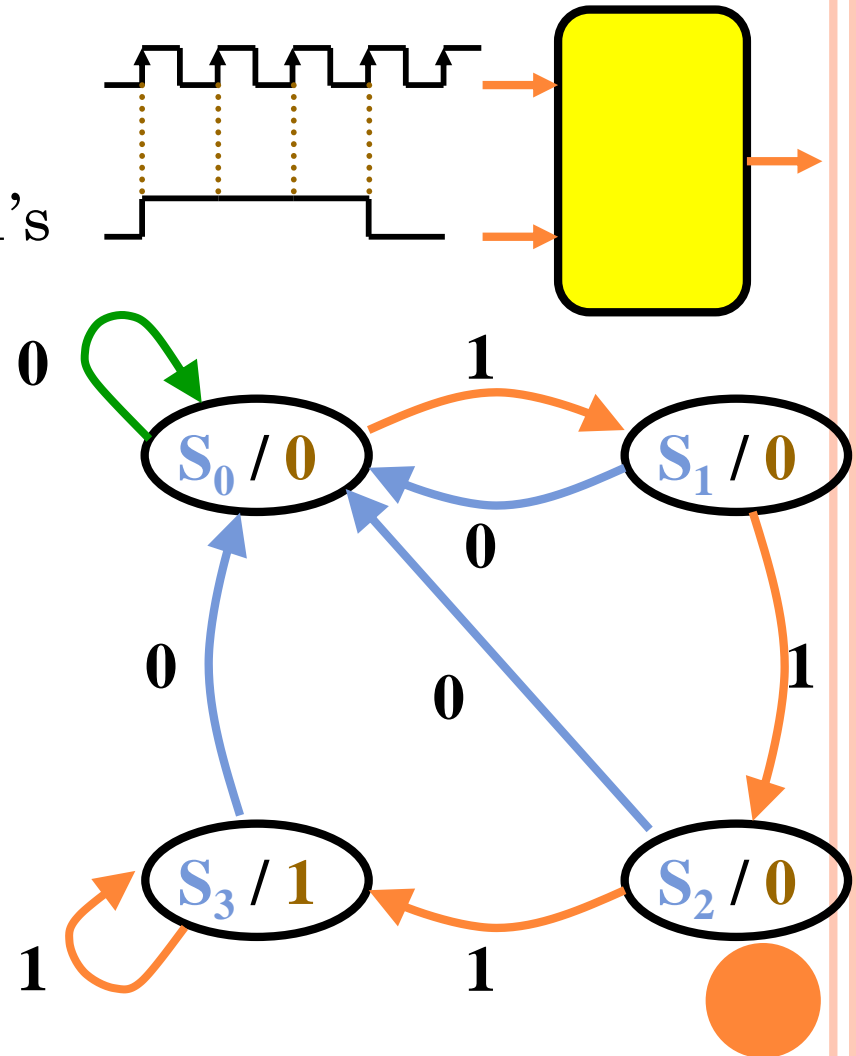
State	A	B
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

DESIGN OF CLOCKED SEQUENTIAL CIRCUITS

○ *Example:*

Detect 3 or more consecutive 1's

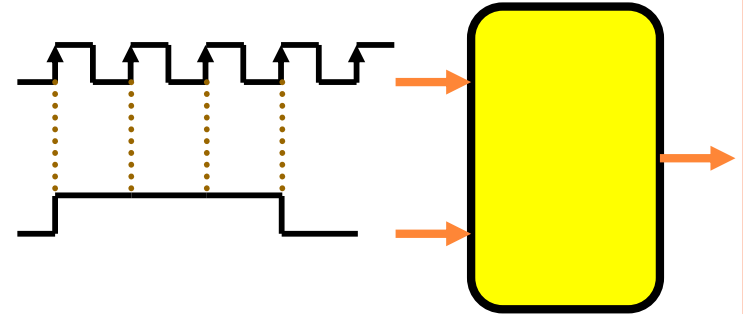
Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS

○ *Example:*

Detect 3 or more consecutive 1's



Present State		Input	Next State		Output
<i>A</i>	<i>B</i>	<i>x</i>	<i>A</i>	<i>B</i>	<i>y</i>
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

Synthesis using *D* Flip-Flops

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

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

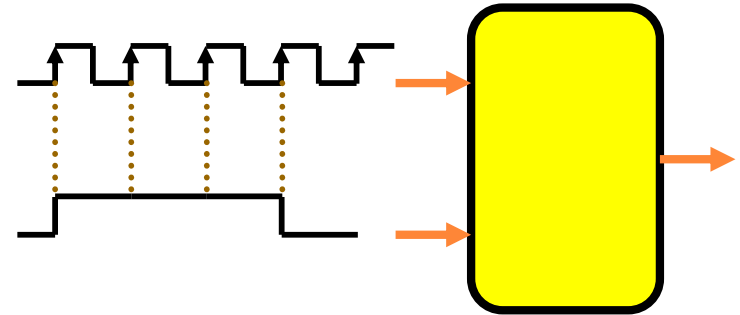
$$y(A, B, x) = \sum (6, 7)$$



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *D* F.F.

○ *Example:*

Detect 3 or more consecutive 1's



Synthesis using *D* Flip-Flops

$$D_A(A, B, x) = \sum (3, 5, 7) \\ = Ax + Bx$$

$$D_B(A, B, x) = \sum (1, 5, 7) \\ = Ax + B'x$$

$$y(A, B, x) = \sum (6, 7) \\ = AB$$

	<i>B</i>			
	0	0	1	0
<i>A</i>	0	1	1	0
	<i>x</i>			

	<i>B</i>			
	0	1	0	0
<i>A</i>	0	1	1	0
	<i>x</i>			

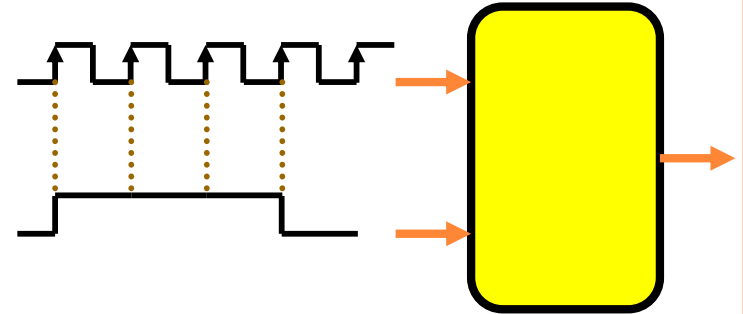
	<i>B</i>			
	0	0	0	0
<i>A</i>	0	0	1	1
	<i>x</i>			



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *D* F.F.

○ *Example:*

Detect 3 or more consecutive 1's

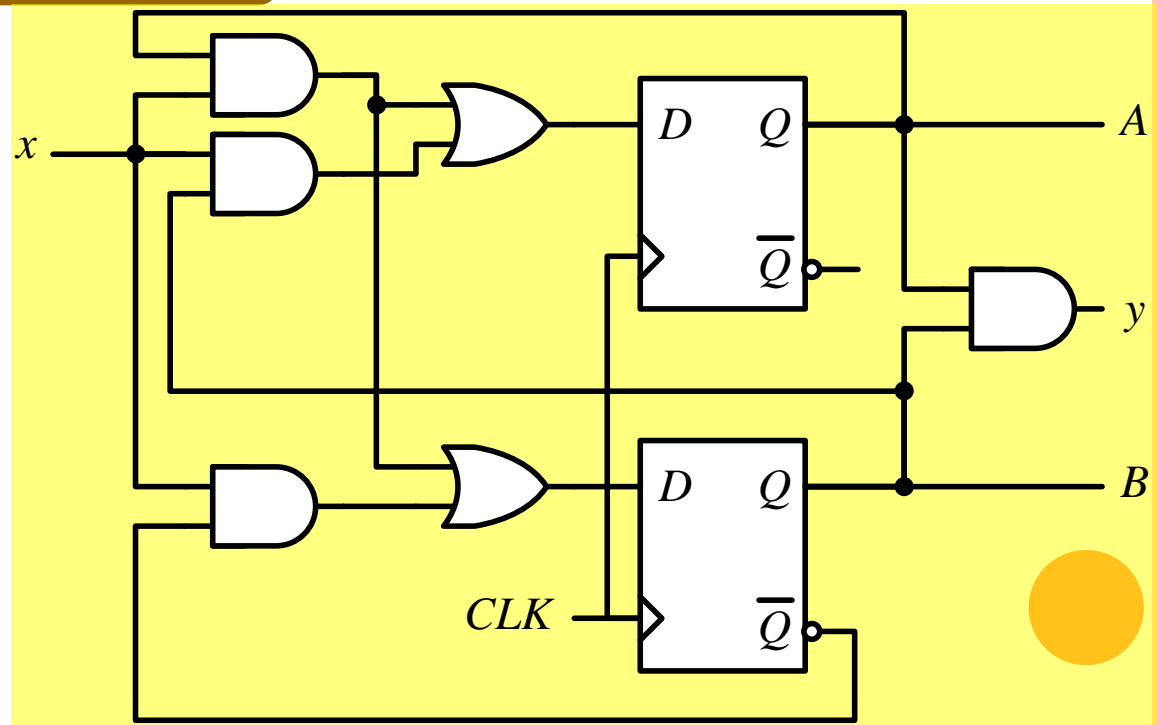


Synthesis using *D* Flip-Flops

$$D_A = A x + B x$$

$$D_B = A x + B' x$$

$$y = A B$$



FLIP-FLOP EXCITATION TABLES

Present State	Next State	F.F. Input
$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

Present State	Next State	F.F. Input	
$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

0 0 (No change)

0 1 (Reset)

1 0 (Set)

1 1 (Toggle)

0 1 (Reset)

1 1 (Toggle)

0 0 (No change)

1 0 (Set)

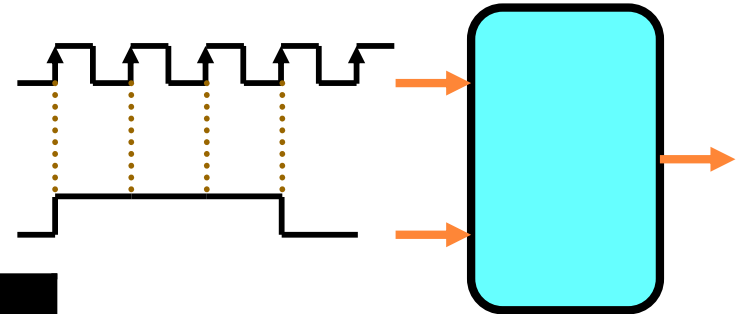
$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH JK F.F.

Example:

Detect 3 or more consecutive 1's



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	0	0	0	x	x	1
0	1	1	1	0	1	x	x	1
1	0	0	0	0	x	1	0	x
1	0	1	1	1	x	0	1	x
1	1	0	0	0	x	1	x	1
1	1	1	1	1	x	0	x	0

Synthesis using JK F.F.

$$J_A(A, B, x) = \sum (3)$$

$$d_{JA}(A, B, x) = \sum (4, 5, 6, 7)$$

$$K_A(A, B, x) = \sum (4, 6)$$

$$d_{KA}(A, B, x) = \sum (0, 1, 2, 3)$$

$$J_B(A, B, x) = \sum (1, 5)$$

$$d_{JB}(A, B, x) = \sum (2, 3, 6, 7)$$

$$K_B(A, B, x) = \sum (2, 3, 6)$$

$$d_{KB}(A, B, x) = \sum (0, 1, 4, 5)$$

DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *JK* F.F.

Example:

Detect 3 or more consecutive 1's

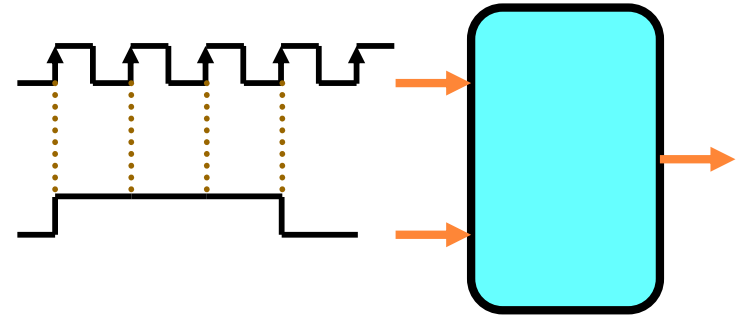
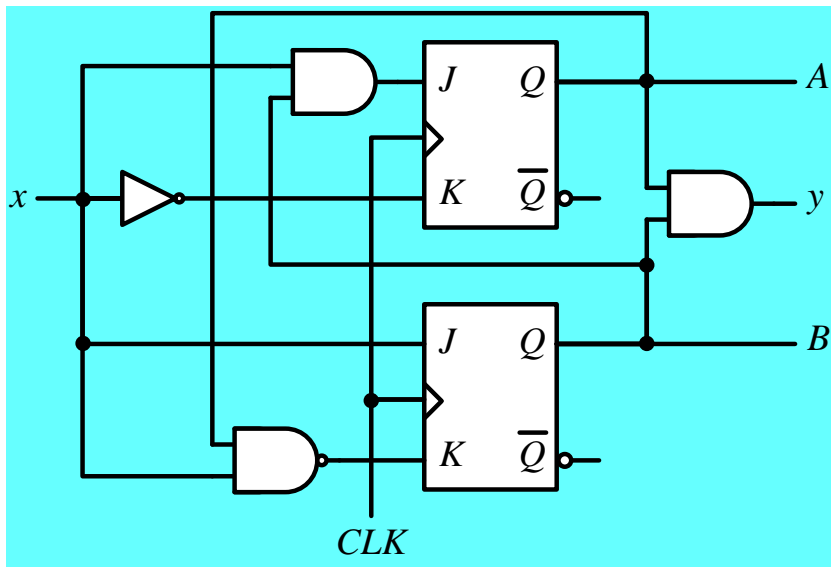
Synthesis using *JK* Flip-Flops

$$J_A = Bx$$

$$K_A = x'$$

$$J_B = x$$

$$K_B = A' + x'$$



	B			
	0	0	1	0
A	x	x	x	x
	x			

	B			
	x	x	x	x
A	1	0	0	1
	x			

	B			
	0	1	x	x
A	0	1	x	x
	x			

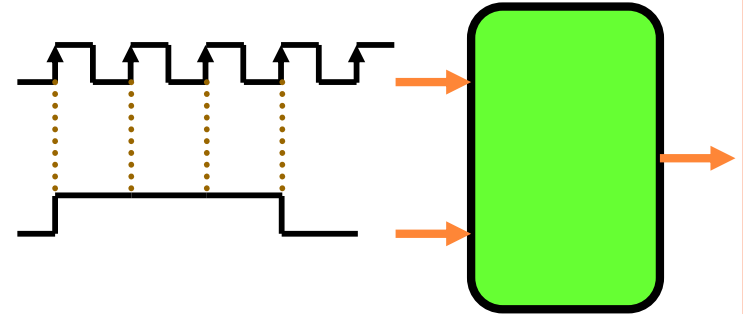
	B			
	x	x	1	1
A	x	x	0	1
	x			



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH T F.F.

○ *Example:*

Detect 3 or more consecutive 1's



Present State		Input	Next State		F.F. Input	
A	B	x	A	B	T_A	T_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	0	1
0	1	1	1	0	1	1
1	0	0	0	0	1	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	1	1	1	0	0

Synthesis using T Flip-Flops

$$T_A(A, B, x) = \sum (3, 4, 6)$$

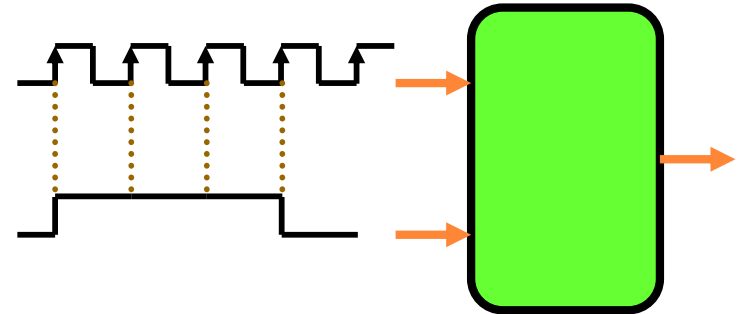
$$T_B(A, B, x) = \sum (1, 2, 3, 5, 6)$$



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *T* F.F.

○ *Example:*

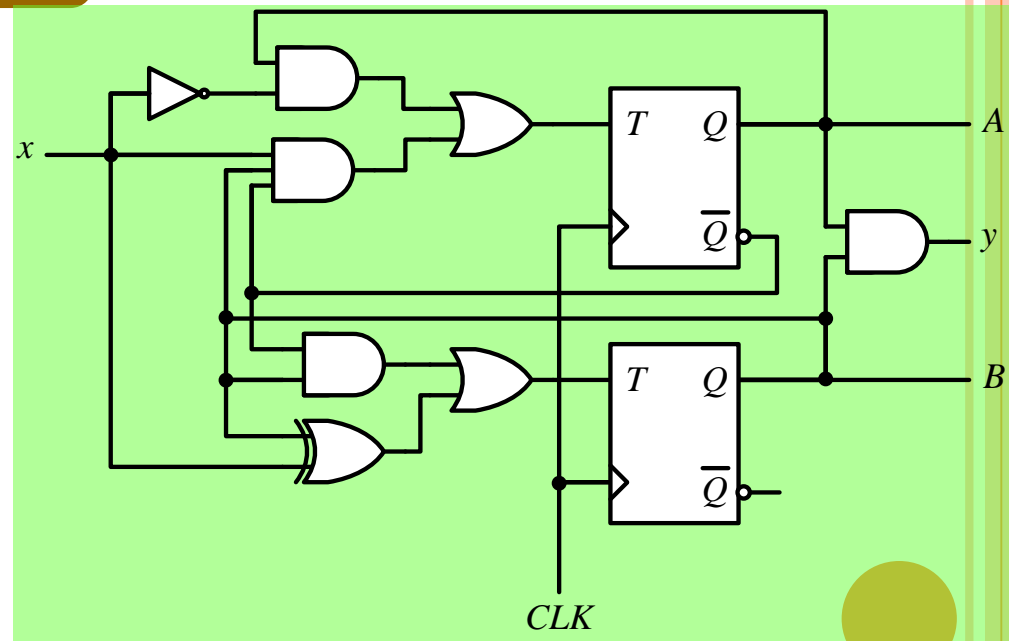
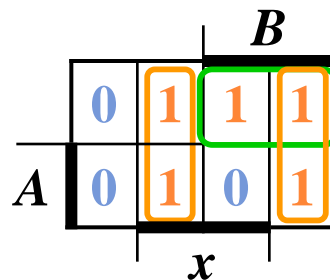
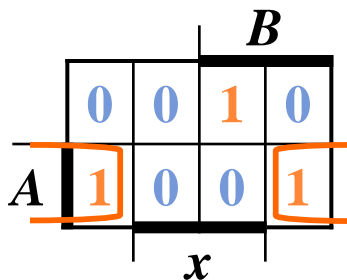
Detect 3 or more consecutive 1's



Synthesis using *T* Flip-Flops

$$T_A = Ax' + A'Bx$$

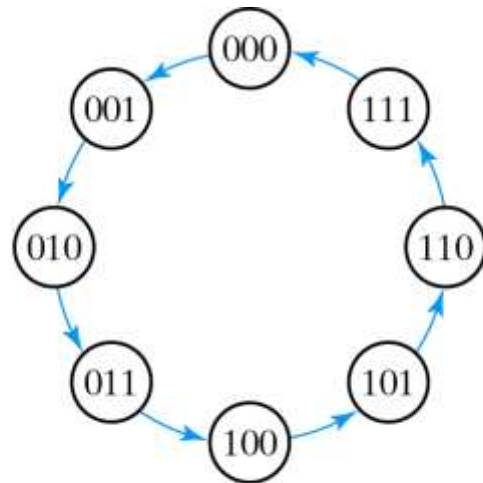
$$T_B = A'B + B \oplus x$$



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *T* F.F.

○ *Example:*

3-bit binary counter



Present State			Next State			Flip-Flop Inputs		
A2	A1	A0	A2	A1	A0	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	0	1
1	1	1	0	0	0	1	1	1

Fig. 5-29 State Diagram of 3-Bit Binary Counter



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *T* F.F.

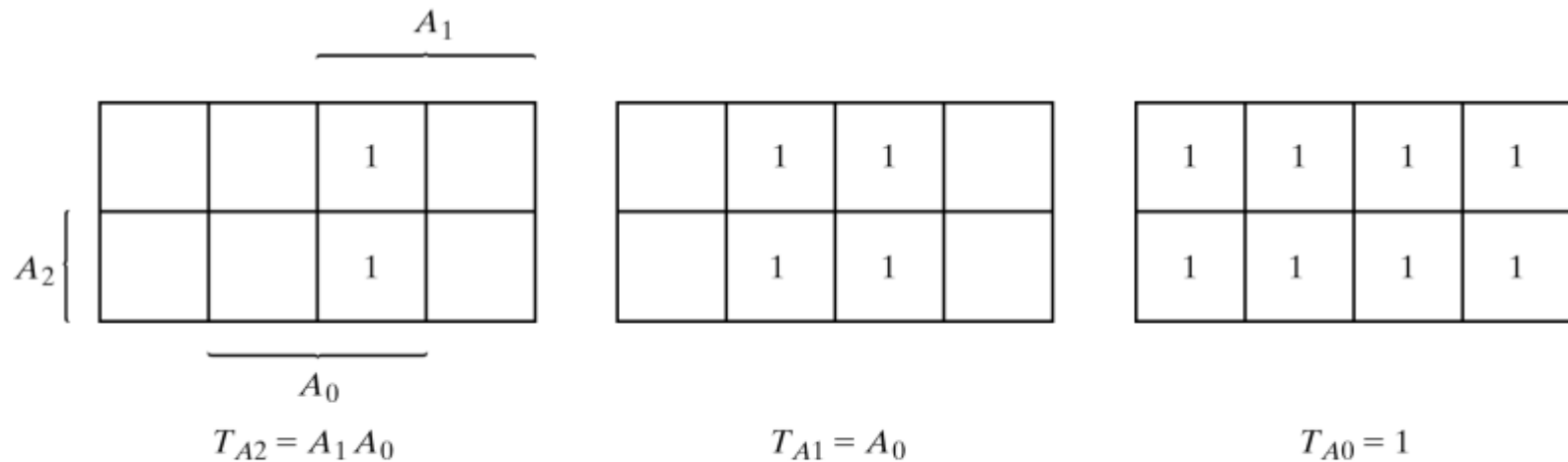


Fig. 5-30 Maps for 3-Bit Binary Counter

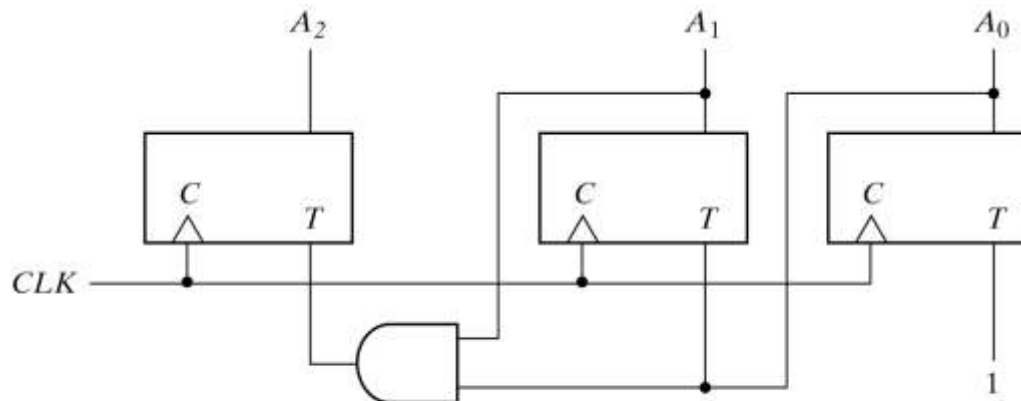


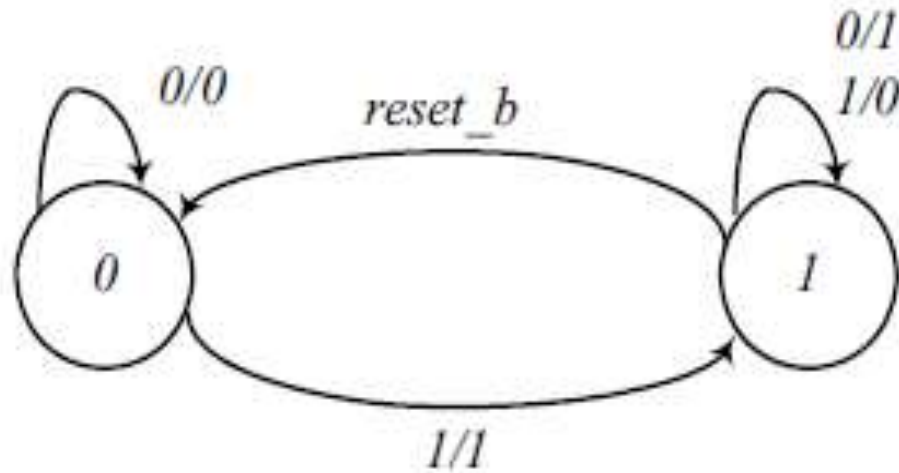
Fig. 5-31 Logic Diagram of 3-Bit Binary Counter

DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *D* F.F.

- Design a one-input, one-output serial 2's complementer. The circuit accepts a string of bits from the input and generates the 2's complement at the output. The circuit can be reset asynchronously to start and end the operation.



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *D* F.F.



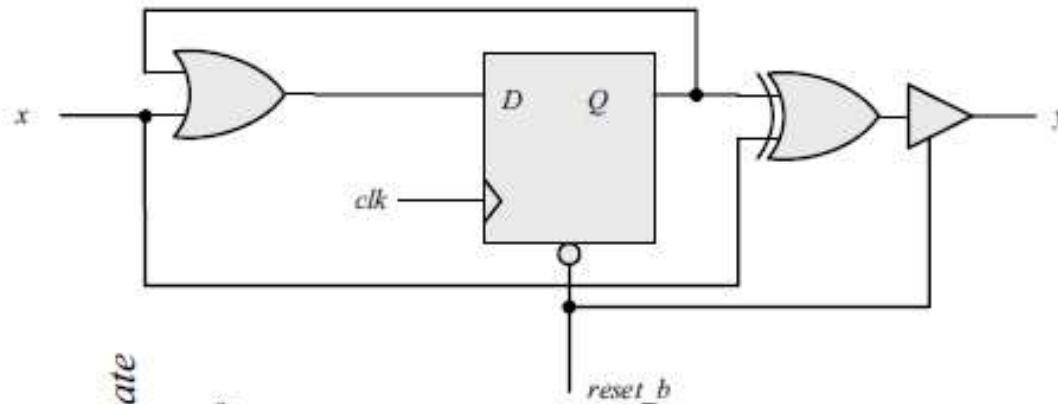
Present state	Input	Next state	Output
<i>A</i>	<i>x</i>	<i>A</i>	<i>y</i>
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

$$D_A = A + x$$

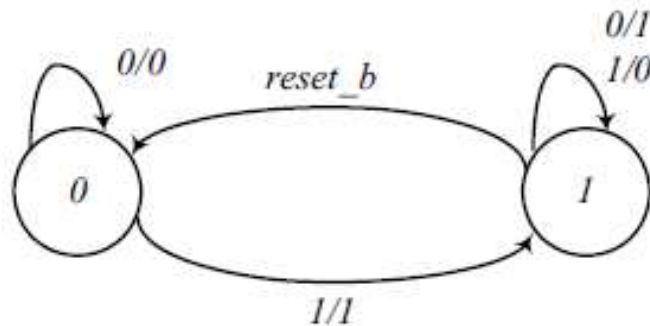
$$y = Ax' + A'x$$



DESIGN OF CLOCKED SEQUENTIAL CIRCUITS WITH *D* F.F.



Present state	Input	Next state	Output
<i>A</i>	<i>x</i>	<i>A</i>	<i>y</i>
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0



$$D_A = A + x$$

$$y = Ax' + A'x$$



SYLLABUS

- Chapter 5 (Excluding Section 5.6)

