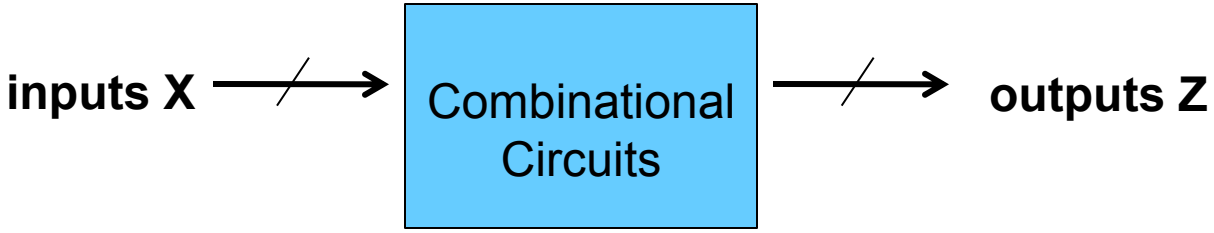
Digital Logic Design
Sequential Circuits

Dr. Basem ElHalawany

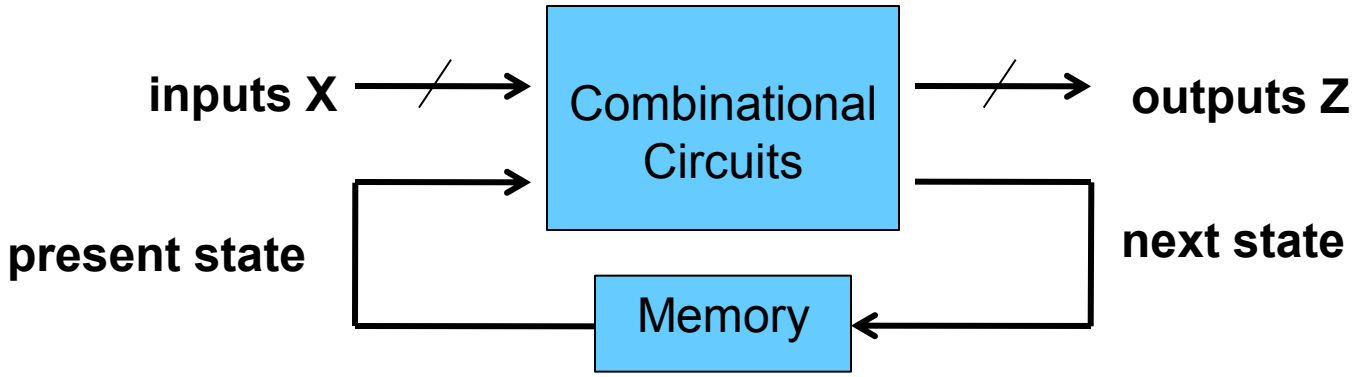
Combinational vs Sequential



A **combinational** circuit:

- At any time, outputs depends only on inputs
  - Changing inputs changes outputs
- No regard for previous inputs
  - No memory (history)

Combinational vs Sequential



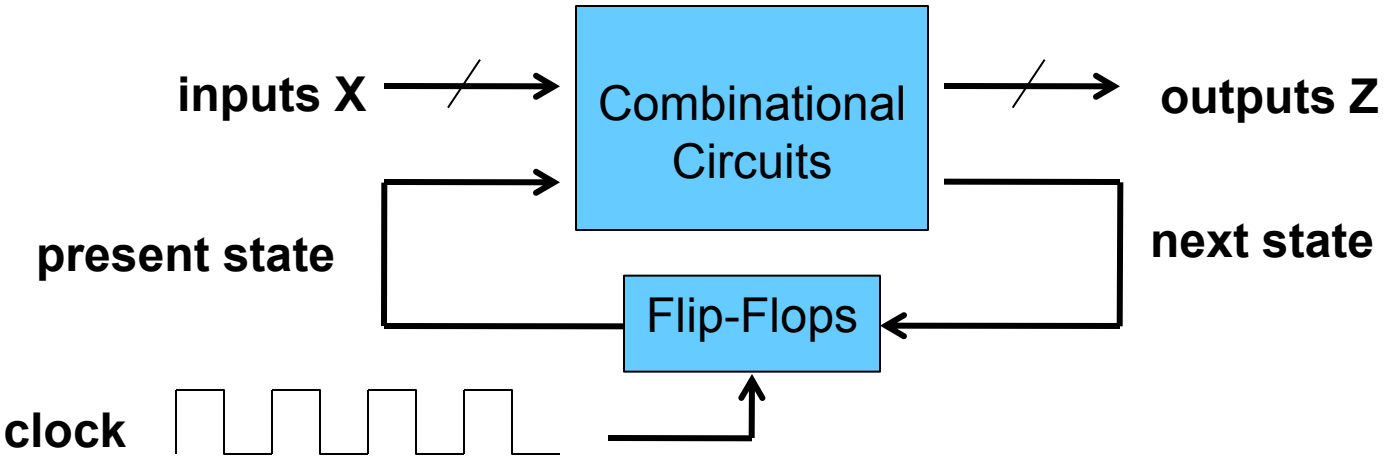
A **sequential** circuit:

- A combinational circuit with **feedback** through **memory**
  - The stored information at any time defines a **state**
- Outputs depends on inputs and previous inputs
  - Previous inputs are stored as binary information into memory
- Next state depends on inputs and present state

Types of Sequential Circuits

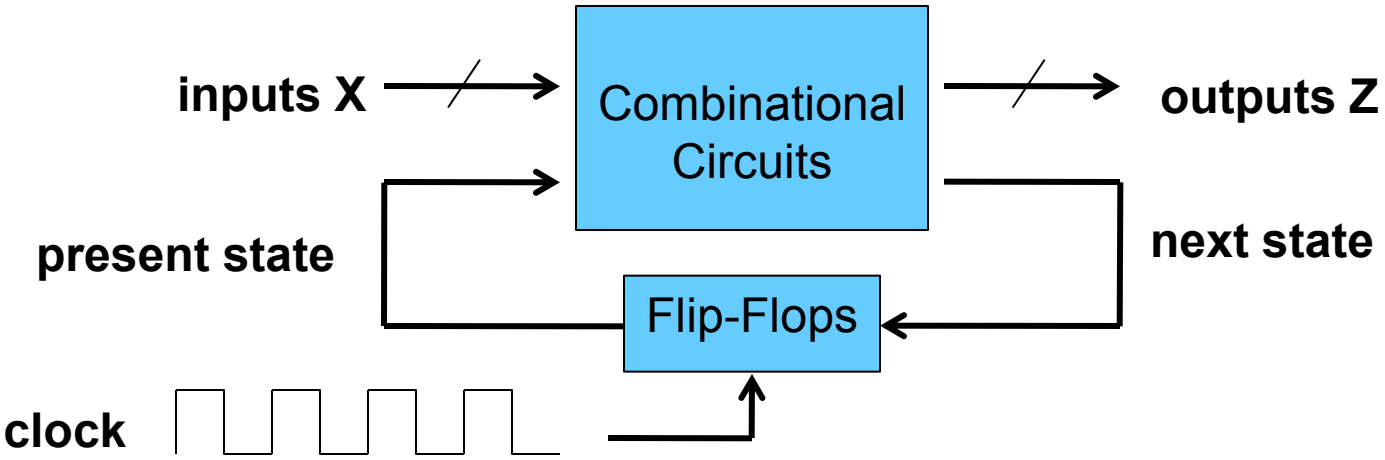
- Two types of sequential circuits:
  - **Synchronous**: The behavior of the circuit depends on the input signal at discrete instances of time *(also called clocked)*
  - **Asynchronous**: The behavior of the circuit depends on the input signals at any instance of time and the order of the inputs change
    - A combinational circuit with feedback

Synchronous Sequential Circuits



- Synchronous circuits employs a synchronizing signal called **clock** (a periodic train of pulses; 0s and 1s)
- A clock determines **when** computational activities occur
- Other signals determines **what** changes will occur

Synchronous Sequential Circuits



- The storage elements (memory) used in clocked sequential circuits are called **flip-flops**
  - Each flip-flop can store one bit of information 0,1
  - A circuit may use many flip-flops; together they define the circuit state
- Flip-Flops (memory/state) update **only** with the clock

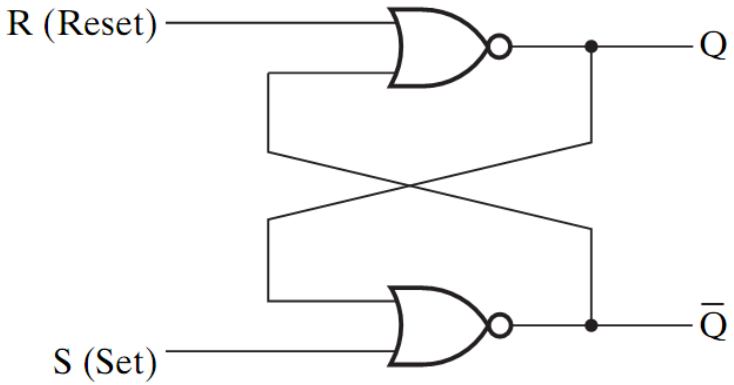
Storage Elements (Memory)

- A storage element can maintain a binary state (0,1) indefinitely, until directed by an input signal to switch state
- Main difference between storage elements:
  - Number of inputs they have
  - How the inputs affect the binary state
- Two main types:
  - **Latches** (level-sensitive)
  - **Flip-Flops** (edge-sensitive)
- Latches are useful in asynchronous sequential circuits
- Flip-Flips are built with latches

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)

SR Latch (Active High)

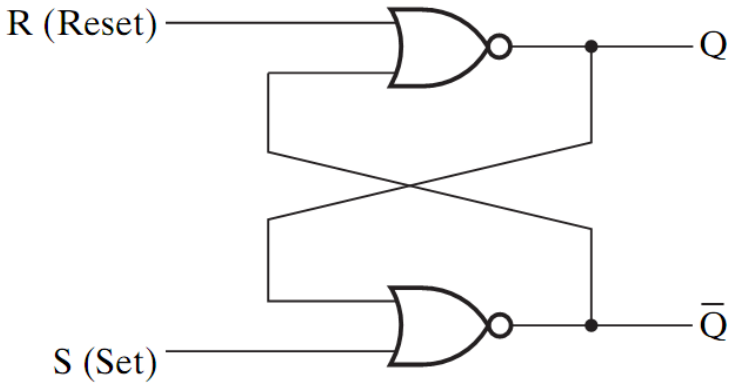


(a) Logic diagram

S	R	Q	$\overline{Q}$
1	0		
0	0		
0	1		
0	0		
1	1		

(b) Function table

SR Latch (Active High)



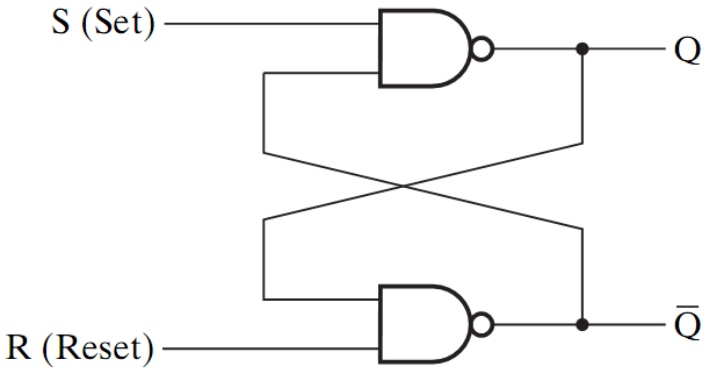
(a) Logic diagram

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table

- Two states: Set (Q = 1) and Reset (Q = 0)
- When S=R=0, Q remains the same, S=R=1 is not allowed!
- Normally, S=R=0 unless the state need to be changed (memory?)
- State of the circuit depends not only on the current inputs, but also on the recent history of the inputs

S’ R’ Latch (Active Low)

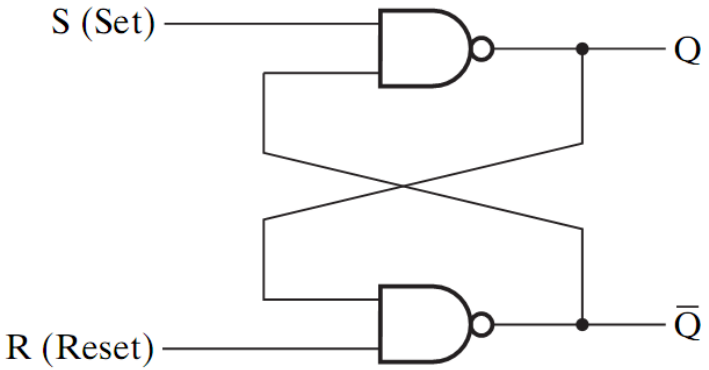


(a) Logic diagram

S	R	Q	$\overline{Q}$
0	1		
1	1		
1	0		
1	1		
0	0		

(b) Function table

S' R' Latch (Active Low)



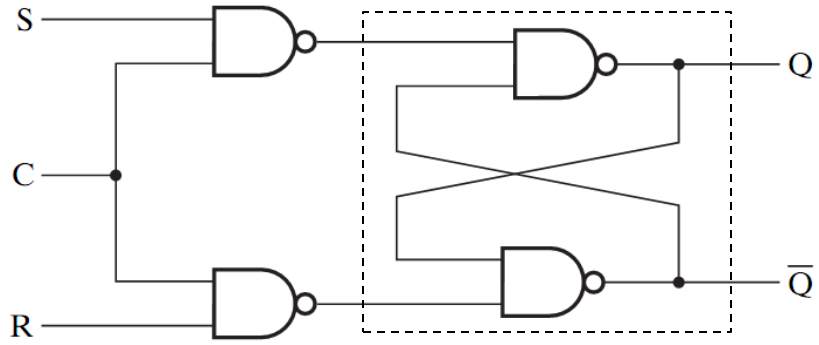
(a) Logic diagram

S	R	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

- Similar to SR latch (complemented)
- When S=R=1, Q remains the same
- S=R=0 is not allowed!

SR Latch with Clock



(a) Logic diagram

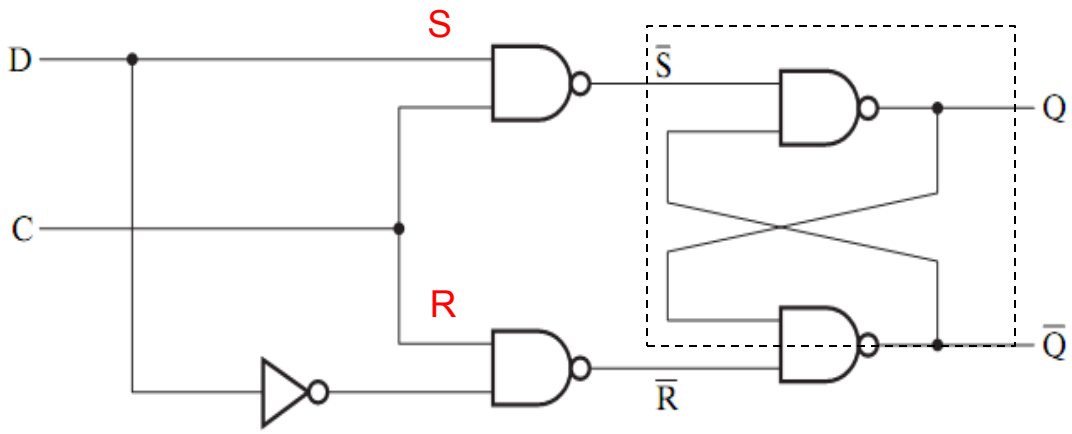
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

- An SR Latch can be modified to control **when** it changes
- An additional input signal Clock (C)
- When C=0, the S and R inputs have no effect on the latch
- When C=1, the inputs affect the state of the latch and possibly the output

How can we eliminate the undefined state?

D Latch



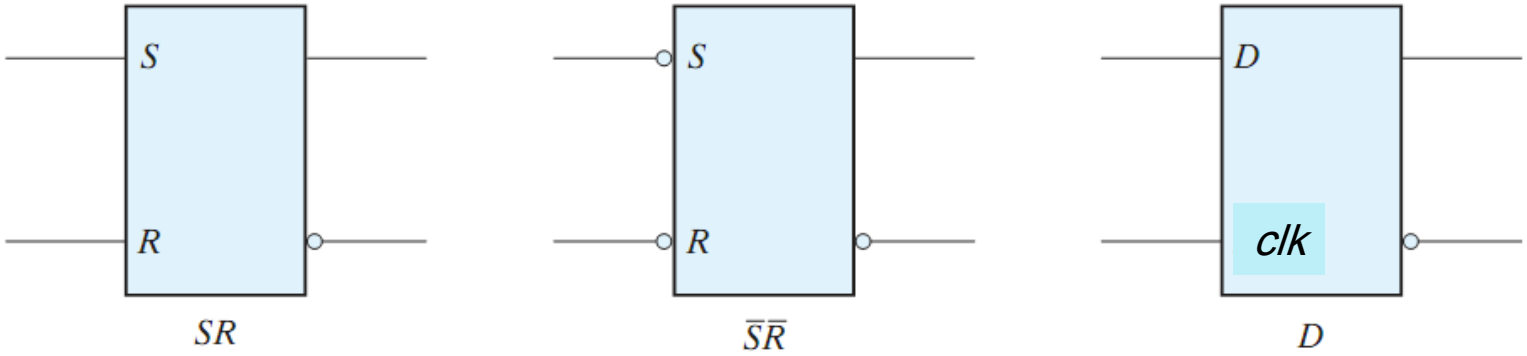
(a) Logic diagram

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

(b) Function table

- Ensure S and R are never equal to 1 at the same time
- Add inverter
- Only one input (D)
  - D connects to S
  - D' connects to R
- D stands for data
- Output follows the input when C = 1
  - Transparent
- When C = 0, Q remains the same

Graphic Symbols for Latches



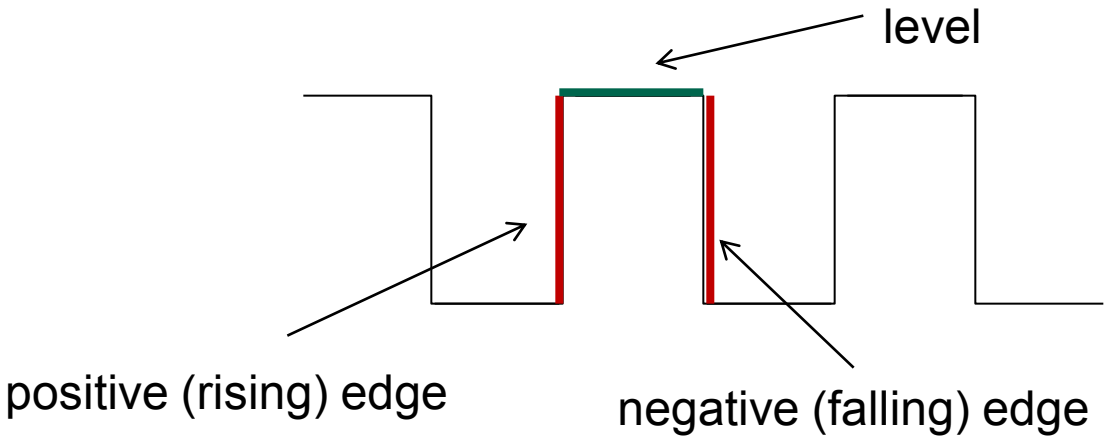
- A latch is designated by a rectangular block with inputs on the left and outputs on the right
- One output designates the normal output, the other (with the bubble) designates the complement
- For S'R' (SR built with NANDs), bubbles added to the input

Problem with Latches

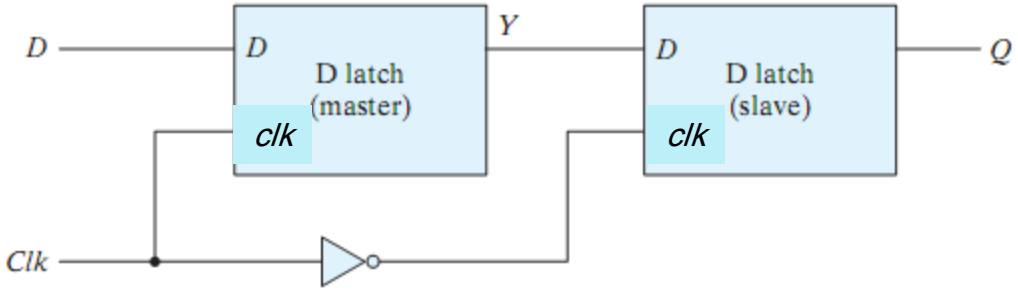
- **A latch is a level sensitive device.**
- **Problem:** A latch is **transparent**; state keep changing as long as the clock remains active
- Due to this uncertainty, latches can not be reliably used as storage elements.

Flip Flops

- A **flip-flop** is a one bit memory similar to latches
- Solves the issue of latch transparency
- Latches are **level** sensitive memory element
  - Active when the clock = 1 (whole duration)
- Flip-Flops are **edge-triggered** or **edge-sensitive** memory element
  - Active only at transitions; i.e. either from 0 → 1 or 1 → 0



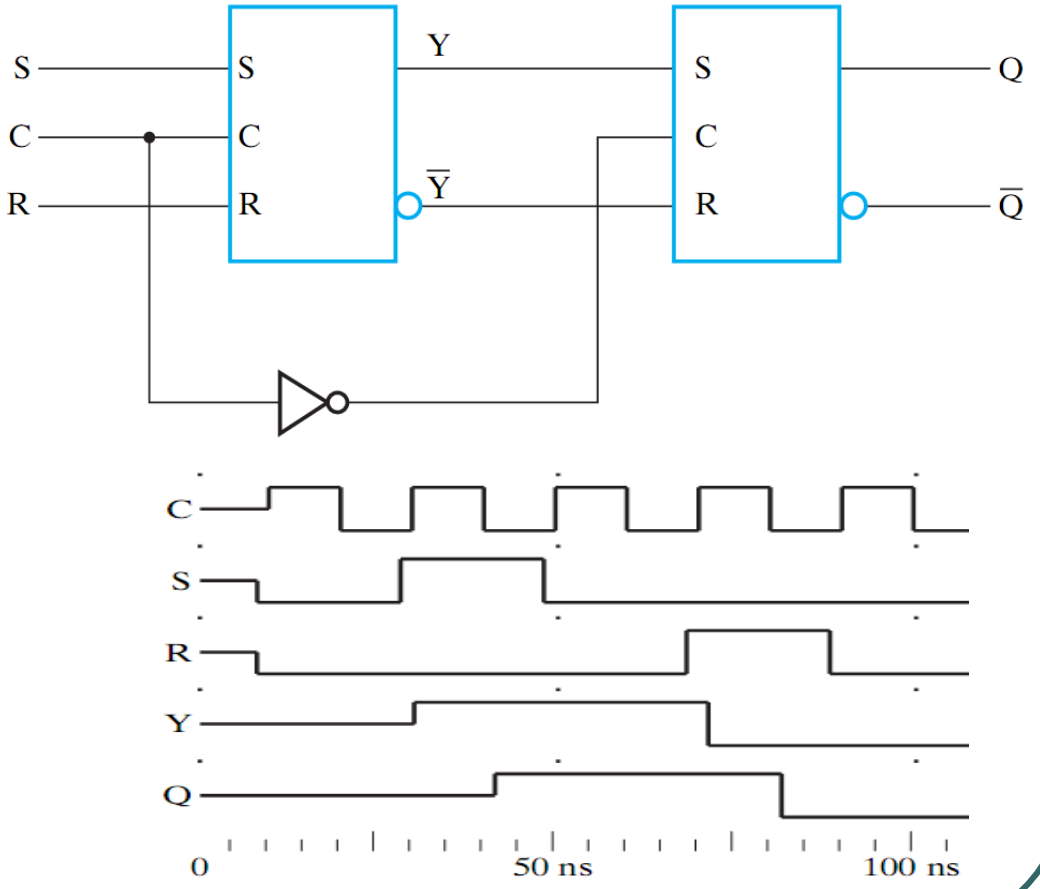
Flip Flops



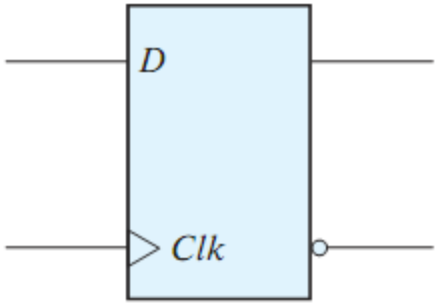
- A flip flop can be built using two latches in a **master-slave** configuration
- A master latch receives external inputs
- A slave latch receives inputs from the master latch
- Depending on the clock signal, only one latch is active at any given time
  - If clk=1, the master latch is enabled and the inputs are latched
  - if clk=0, the master is disabled and the slave is activated to generate the outputs

SR Flip Flop

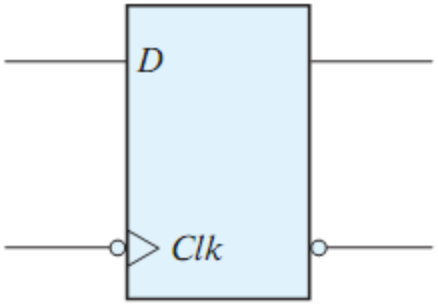
- Built using two latches (Master and Slave)
  - C = 1, master is active
  - C = 0, slave is active
- Q is sampled at the falling edge
- Data is entered on the rising edge of the clock pulse, but the output does not reflect the change until the falling edge of the clock pulse.



Graphic Symbols for Flip Flops



(a) Positive-edge



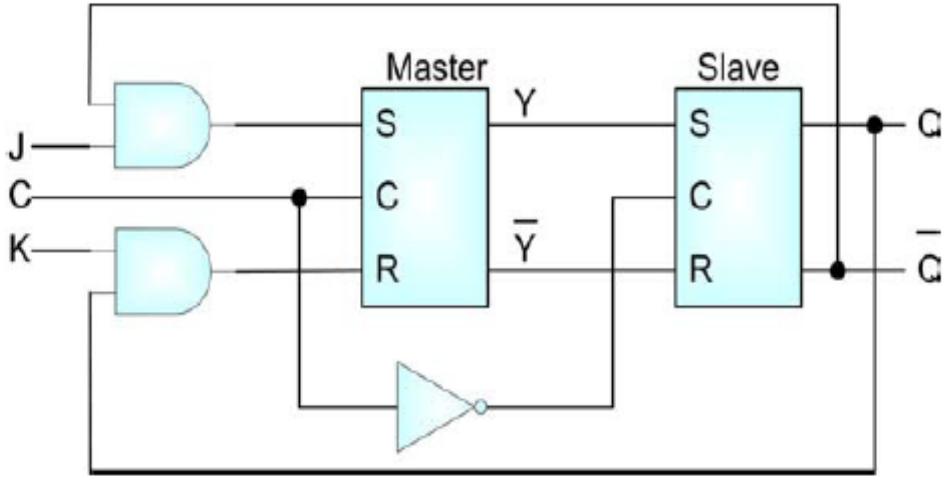
(a) Negative-edge

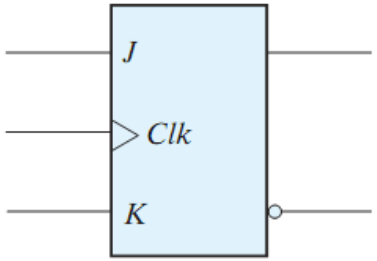
- A Flip Flop is designated by a rectangular block with inputs on the left and outputs on the right (similar to latches)
- The clock is designated with an arrowhead
- A bubble designates a negative-edge triggered flip flops

Other Flip Flops

JK Flip Flop

- JK Flip Flop can be built with SR latches as shown
- Used to force the forbidden state SR =11 to produce a 4th allowed state (Toggle/Complement)



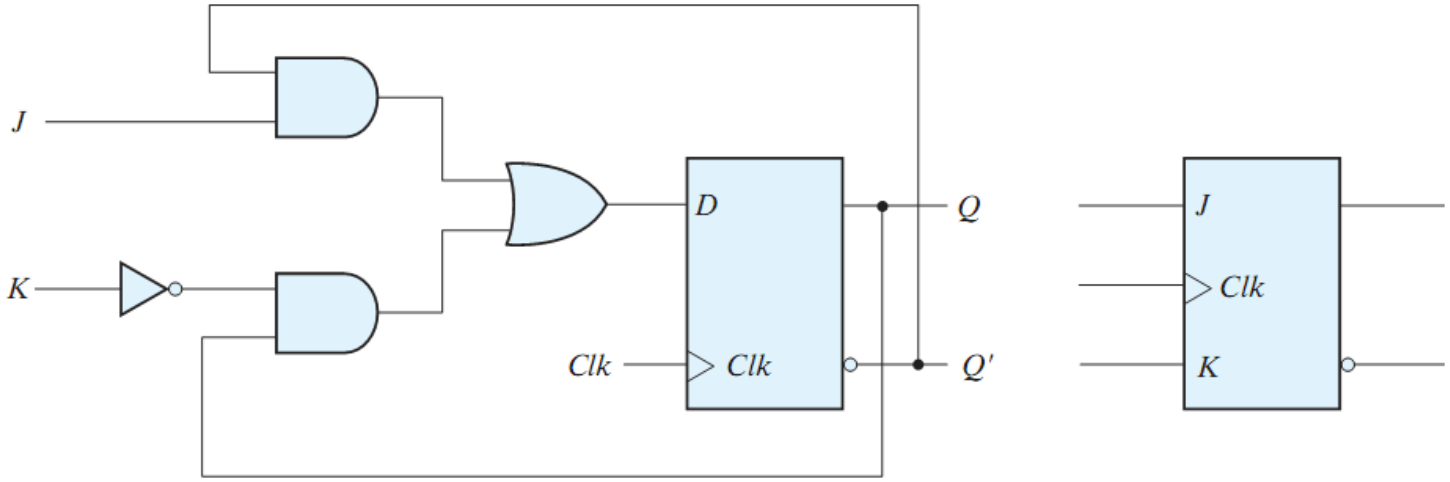


(b) Graphic symbol

- J sets the flip flop (1)
- K reset the flip flop (0)
- When J = K = 1, the output is complemented

Other Flip Flops

JK Flip Flop

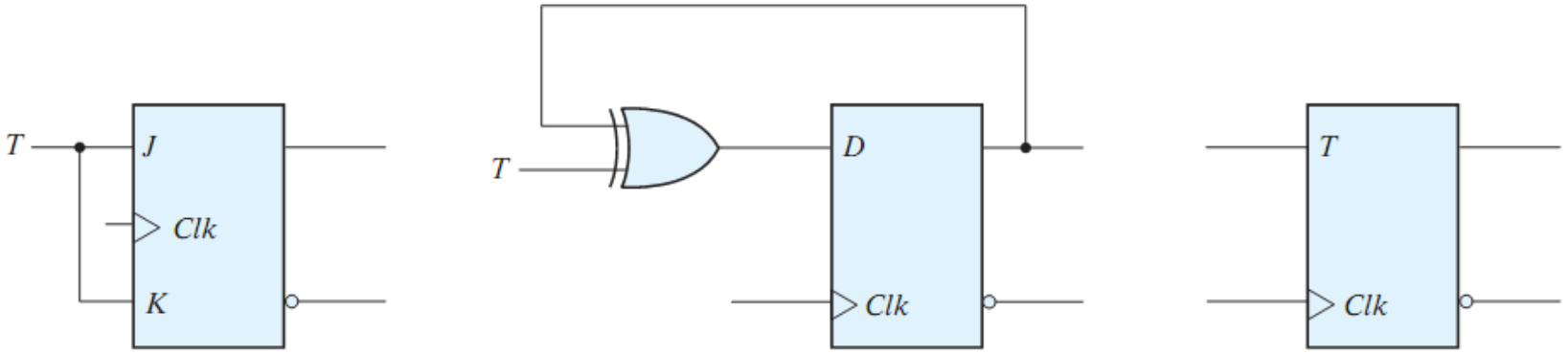


(a) Circuit diagram (b) Graphic symbol

- D = J Q' + K' Q
- J sets the flip flop (1)
- K reset the flip flop (0)
- When J = K = 1, the output is complemented

Other Flip Flops (cont.)

T Flip Flop



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

- T (toggle) flip flop is a complementing flip flop
- Built with a JK or D flip flop (as shown above)
- T = 0, no change,
- T = 1, complement (toggle)
- For D-FF implementation, D = T $\oplus$ Q

Functional Table & Characteristic Table

- Tables that define the operation of a flip flop in a tabular form

- Next state is defined in terms of the current state and the inputs
  - Q(t) refers to current state (before the clock arrives)
  - Q(t+1) refers to next state (after the clock arrives)

- Example: (D Latch)

Table 6: Functional Table of the D-Latch

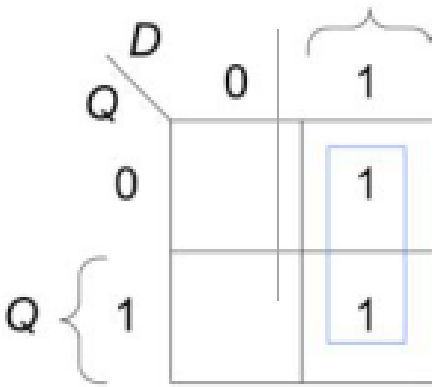
C	D	Next State of Q
0	X	No Change
1	0	Q = 0; Reset State
1	1	Q = 1; Set State

Table 7: Characteristic Table of the D-Latch

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

Characteristic Equation of the D-Latch

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



A **characteristic equation** defines the operation of a flip flop in an algebraic form

Functional Table & Characteristic Table

- JK

Table 8: Functional Table of the Clocked JK-Latch

C	J	K	Next State of Q
0	X	X	Q *(No Change)*
1	0	0	Q *(No Change)*
1	0	1	0 *(Reset State)*
1	1	0	1 *(Set State)*
1	1	1	Q' *(Complement)*

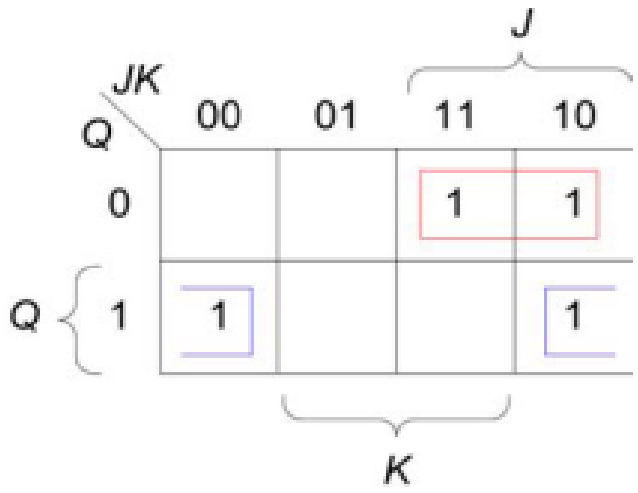


Table 9: Characteristic Table of the JK-Latch

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

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

Characteristic Equation of the JK-Latch

Functional Table & Characteristic Table

- T

Table 11: Functional Table of the Clocked T-Latch

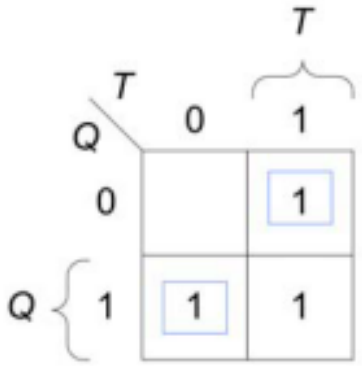
C	T	Next State of Q
0	X	*No Change*
1	0	*No Change*
1	1	*Q'*

Table 12: Characteristic Table of the T-Latch

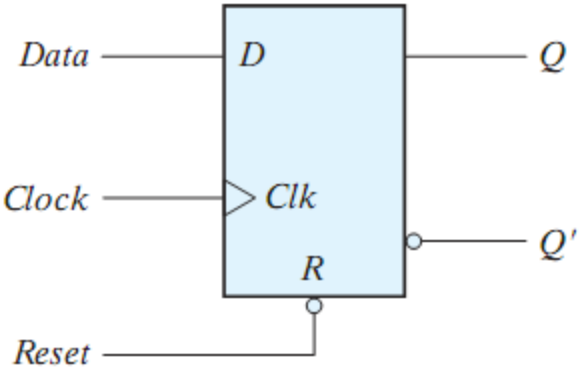
Q(t)	T	Q(t + 1)
0	0	0
0	1	1
1	0	1
1	1	0

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

Characteristic Equation of the T-Latch



Direct Inputs



(b) Graphic symbol

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

(b) Function table

- Some flip-flops have asynchronous inputs to set/reset their states independently of the clock.
- **Preset** or **direct set**, sets the flip-flop to 1
- **Clear** or **direct reset**, set the flip-flop to 0
- When power is turned on, a flip-flop state is unknown; Direct inputs are useful to put in a known state
- Figure shows a positive-edge D-FF with active-low asynchronous reset.

Analysis & Design of Synchronous Sequential Circuits

Analysis of Sequential Circuits

- Analysis is describing what a given circuit will do
- The behavior of a clocked (synchronous) sequential circuit is determined from the inputs, the output, and the states of FF

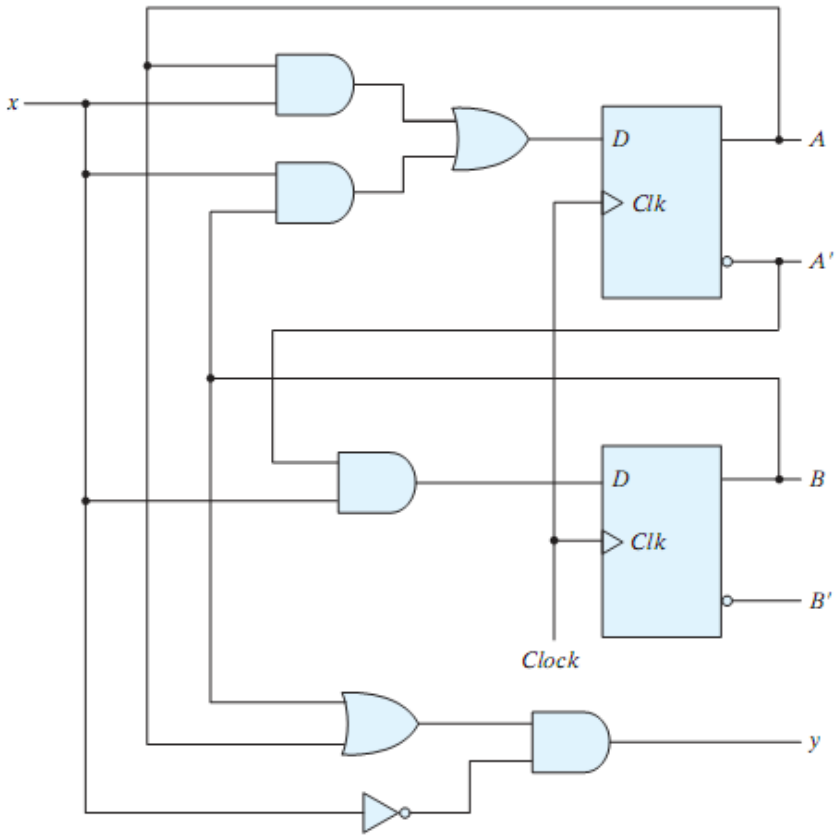
Steps:

- Obtain state equations
  - FF input equations
  - Output equations
- Fill the state table
  - Put all combinations of inputs and current states
  - Fill the next state and output
- Draw the state diagram

Example 1

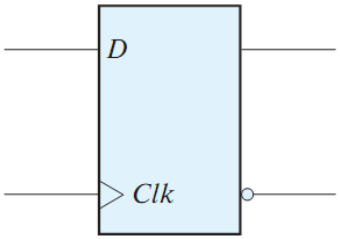
Analyze this circuit?

- Is this a sequential circuit? Why?
- How many inputs?
- How many outputs?
- How many states?
- What type of memory?

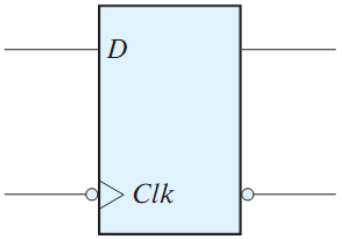


Example 1 (cont.)

D Flip Flop (review)



(a) Positive-edge



(a) Negative-edge

Characteristic Tables and Equations

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

D	Q(t+1)
0	0
1	1

Q(t+1) = D

Example 1 (cont.)

State equations:

$D_A = AX + BX$

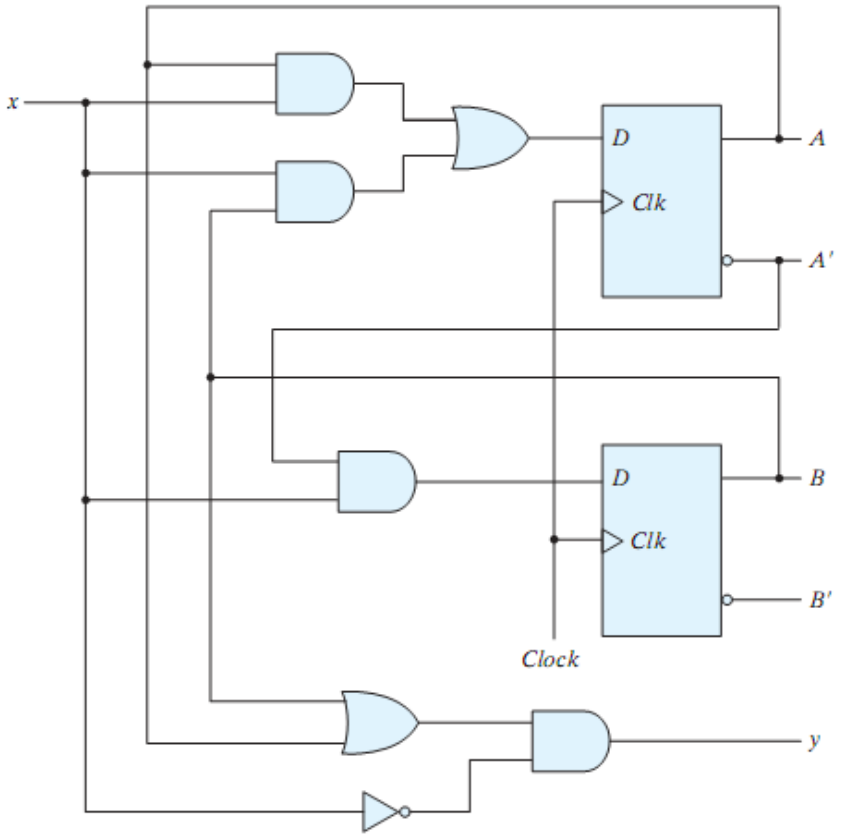
$D_B = A' X$

$Y = (A + B) X'$

Substitute (to get N.S.):

$A(t+1) = D_A =$ $AX + BX$

$B(t+1) =$ $D_B = A' X$



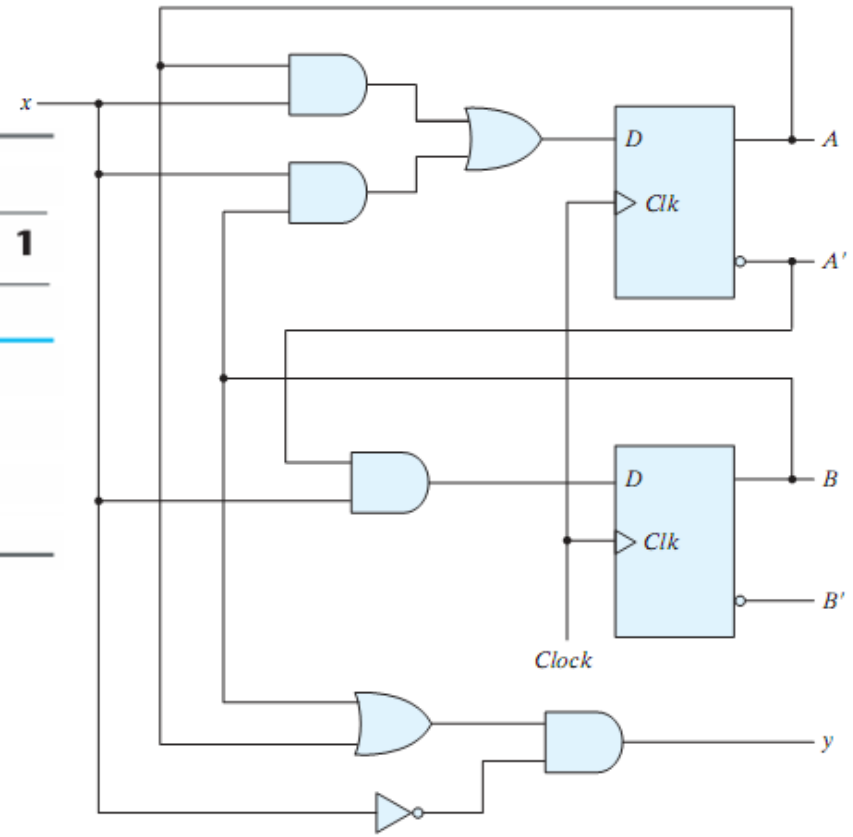
State table:

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

Example 1 (cont.)

State table (2D):

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

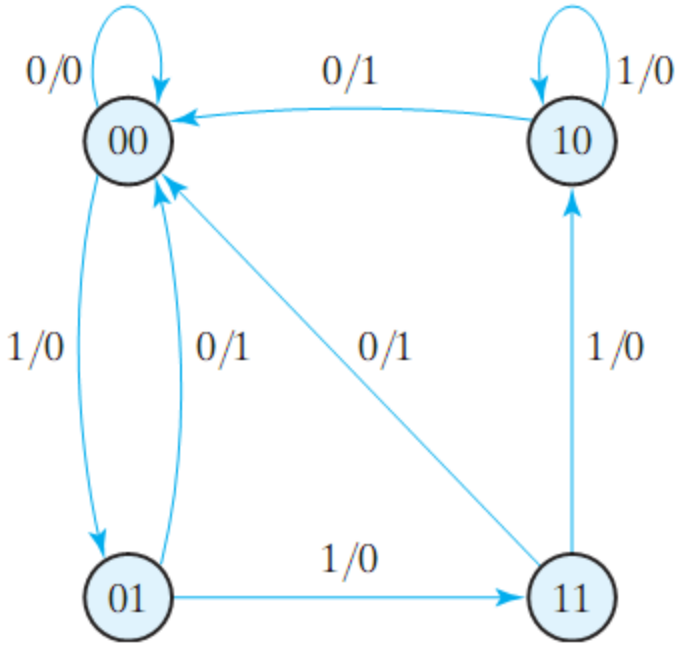


Example 1 (cont.)

State table:

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

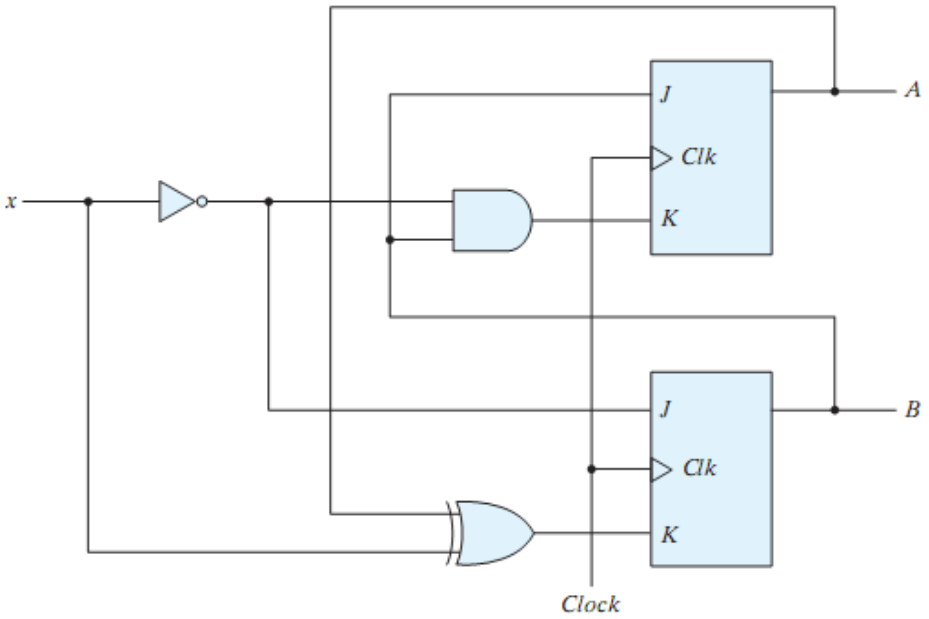
State diagram:



Example 2

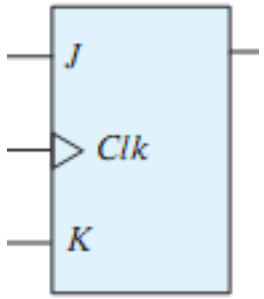
Analyze this circuit?

- Is this a sequential circuit? Why?
- How many inputs?
- How many outputs?
- How many states?
- What type of memory?



Example 2 (cont.)

JK Flip Flop (review)



Characteristic Tables and Equations

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

Example 2 (cont.)

State equations:

$J_A = B, K_A = B X'$

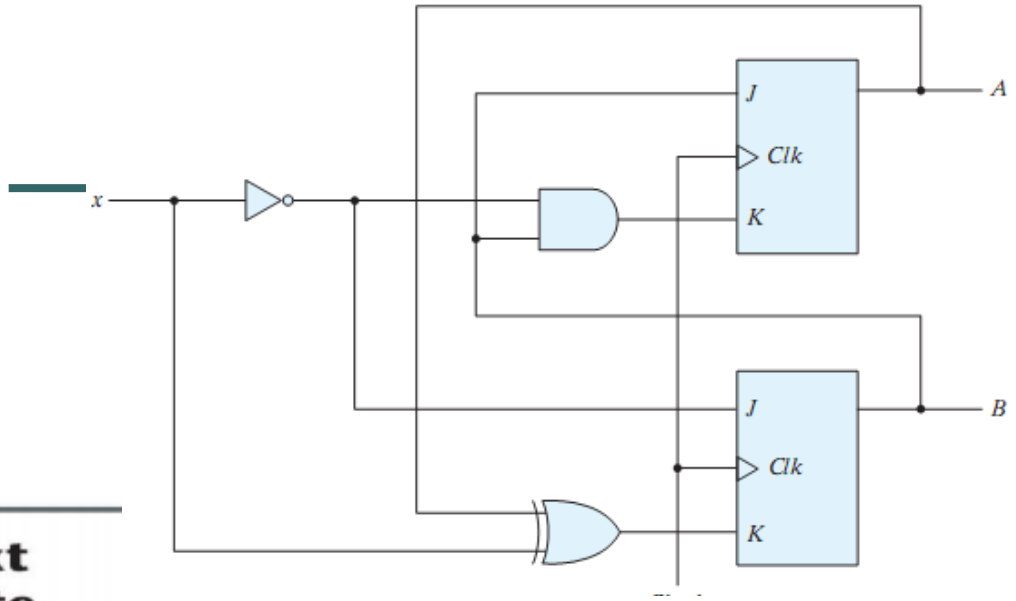
$J_B = X', K_B = A \oplus X$

by substitution:

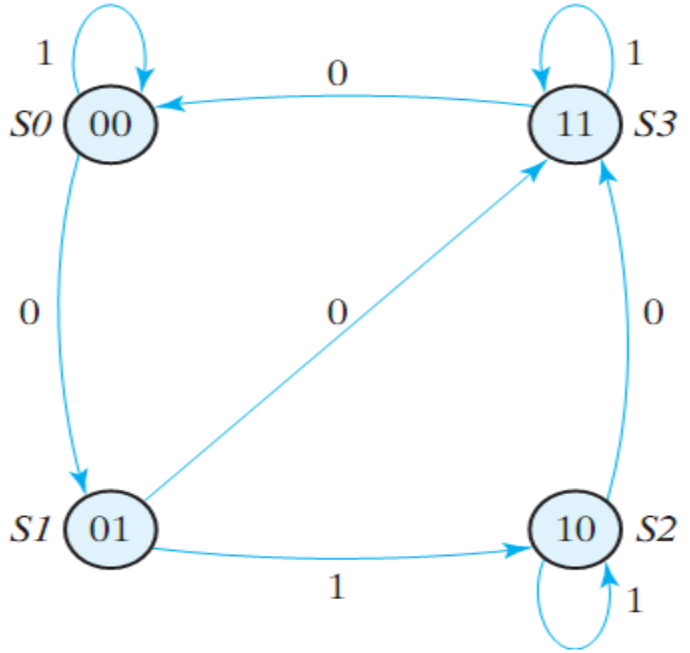
$A = J_A A' + K_A' A$

$= A' B + A B' + A X$

$B = B' X' + A B X + A' B X'$



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



Design of Synchronous Sequential Circuits

- The **design** of a clocked sequential circuit starts from a set of specifications and ends with a logic diagram (Analysis reversed!)
- Building blocks: flip-flops, combinational logic
- Need to choose type and number of flip-flops
- Need to design combinational logic together with flip-flops to produce the required behavior
- The combinational part is
  - flip-flop input equations
  - output equations

Design of Synchronous Sequential Circuits

Design Procedure:

- Obtain a state diagram from the word description
  - State reduction if necessary
- Obtain State Table
  - State Assignment
  - Choose type of flip-flops
  - Use FF's excitation table to complete the table
- Derive state equations
  - Obtain the FF input equations and the output equations
  - Use K-Maps
- Draw the circuit diagram

Example 1

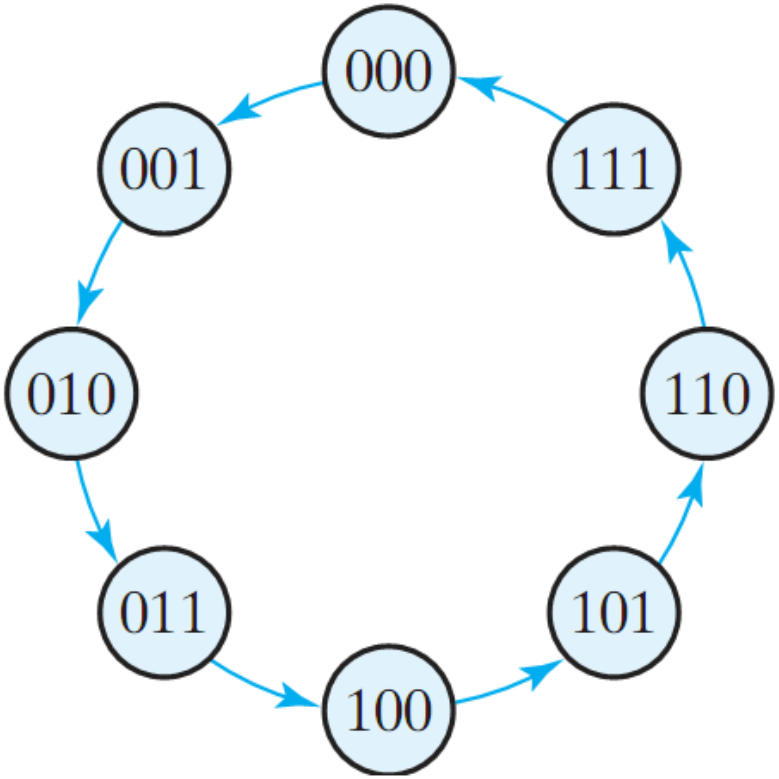
Problem: Design of A 3-bit Counter

Design a circuit that counts in binary form as follows
000, 001, 010, … 111, 000, 001, …

Example 1 (cont.)

Step1: State Diagram

- The outputs = the states
- Where is the input?



Example 1 (cont.)

Step2: State Table

No need for state assignment here

Present State			Next State		
A_2	A_1	A_0	A_2	A_1	A_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Example 1 (cont.)

Step2: State Table

We choose T-FF

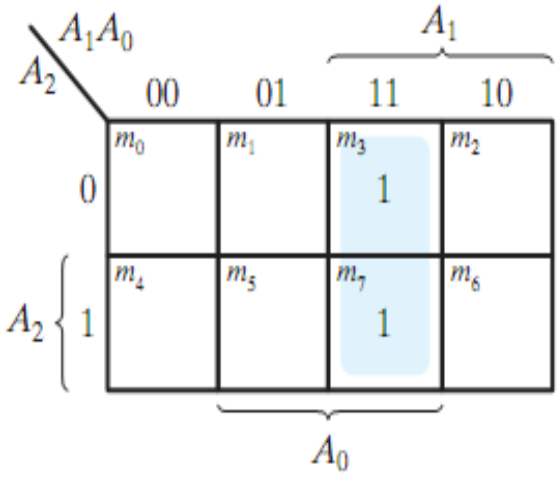
T–FF excitation table

Q(t +1)	T	Operation
$Q(t)$	0	No change
$\overline{Q}(t)$	1	Complement

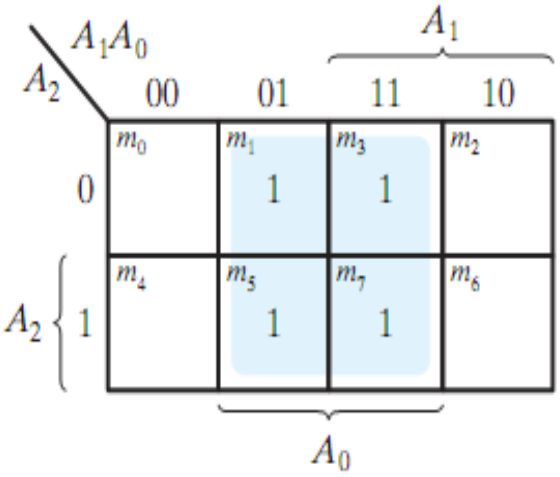
Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Example 1 (cont.)

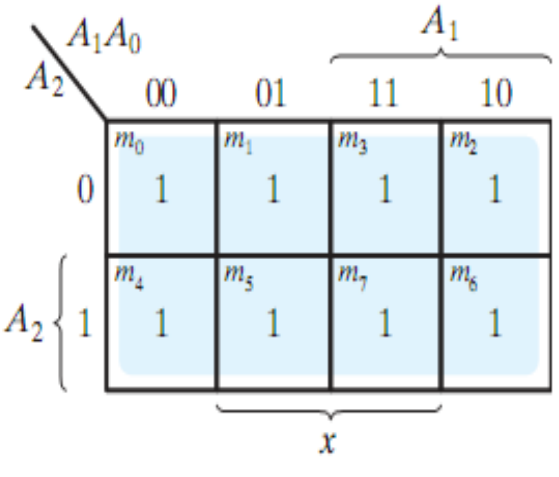
Step3: State Equations



$T_{A2} = A_1A_0$



$T_{A1} = A_0$



$T_{A0} = 1$

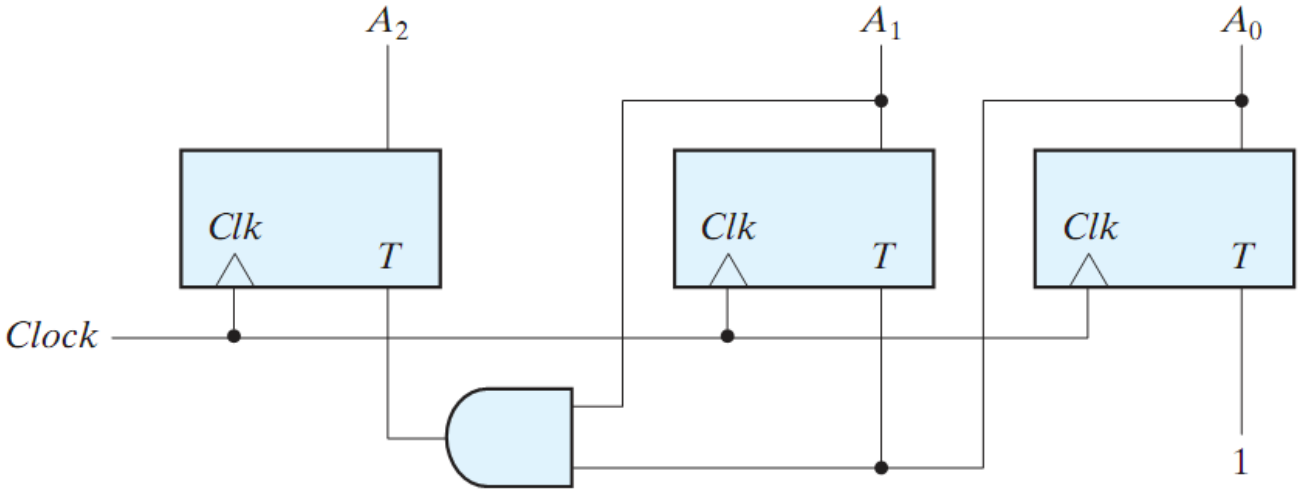
Example 1 (cont.)

Step4: Draw Circuit

$T_{A0} = 1$

$T_{A1} = A_0$

$T_{A2} = A_1A_0$



Excitation Table

T–FF excitation table

Q(t +1)	T	Operation
$Q(t)$	0	No change
$\overline{Q}(t)$	1	Complement

D–FF excitation table

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

JK–FF excitation table

Q(t)	Q(t +1)	J	K	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No Change