Synchronous Sequential Logic

ENCS2340 - Digital Systems

Dr. Ahmed I. A. Shawahna

Electrical and Computer Engineering Department

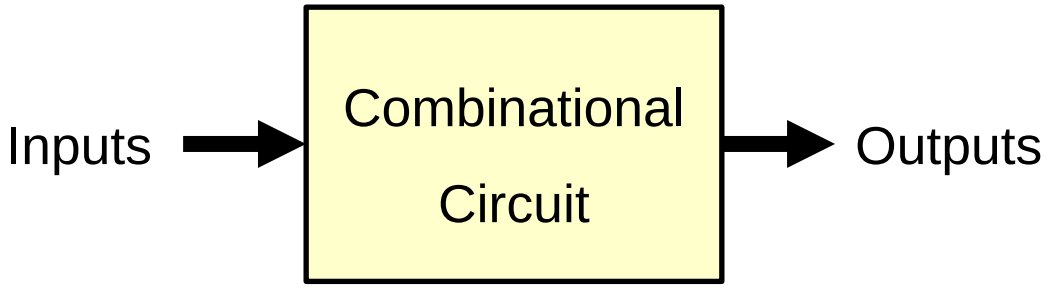
Birzeit University

Presentation Outline

- **Introduction to Sequential Circuits**
  - **Combinational versus Sequential Circuits**
  - **Synchronous versus Asynchronous Sequential Circuits**
- Storage Elements
  - Latches
  - Flip-Flops
- Analysis of Clocked Sequential circuits
  - State and Output Equations
  - State Table
  - State Diagram
- Mealy versus Moore Sequential Circuits

Combinational versus Sequential

- Two classes of digital circuits
  - Combinational Circuits
  - Sequential Circuits
- Combinational Circuit
  - Outputs = F(Inputs)
  - Function of Inputs only
  - NO internal memory



- Sequential Circuit
  - Outputs is a function of Inputs and internal Memory
  - There is an internal memory that stores the state of the circuit
  - Time is very important: memory changes with time

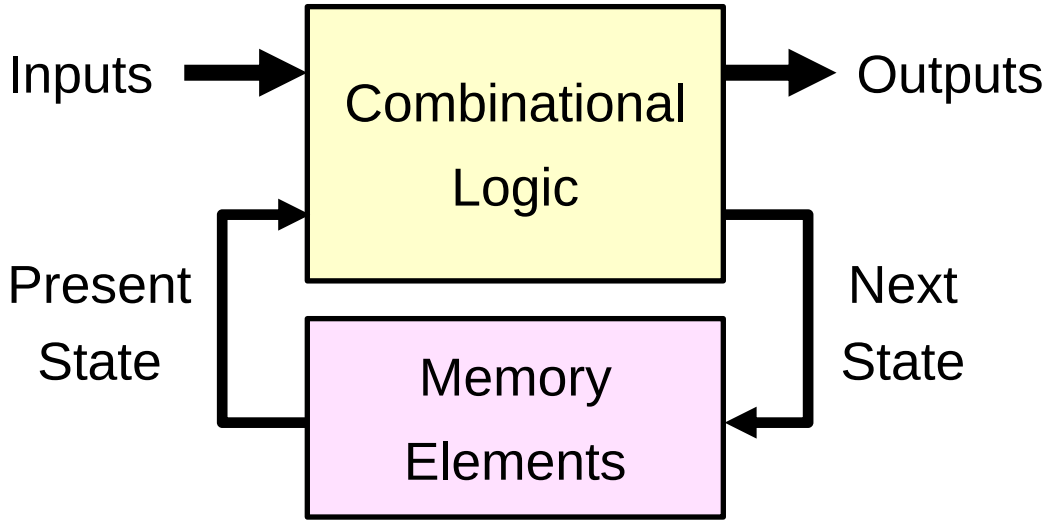
Introduction to Sequential Circuits

A Sequential circuit consists of:

1. Memory elements:
   - ✧ **Latches** or **Flip-Flops**
   - ✧ Store the **Present State**
2. Combinational Logic
   - ✧ Computes the **Outputs** of the circuit

     Outputs depend on Inputs and Current State
   - ✧ Computes the **Next State** of the circuit

     Next State also depends on the Inputs and the Present State



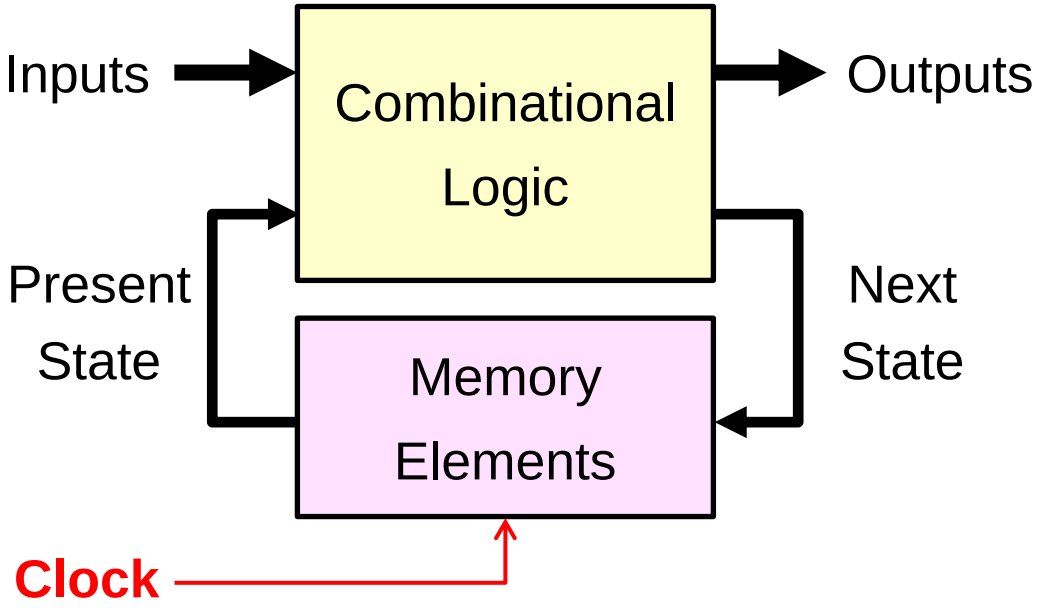
Two Types of Sequential Circuits

1. **Synchronous** Sequential Circuit
   - ✧ Uses a clock signal as an additional input
   - ✧ Changes in the memory elements are controlled by the clock
   - ✧ Changes happen at discrete instances of time
2. **Asynchronous** Sequential Circuit
   - ✧ No clock signal
   - ✧ Changes in the memory elements can happen at any instance of time

❖ Our focus will be on Synchronous Sequential Circuits

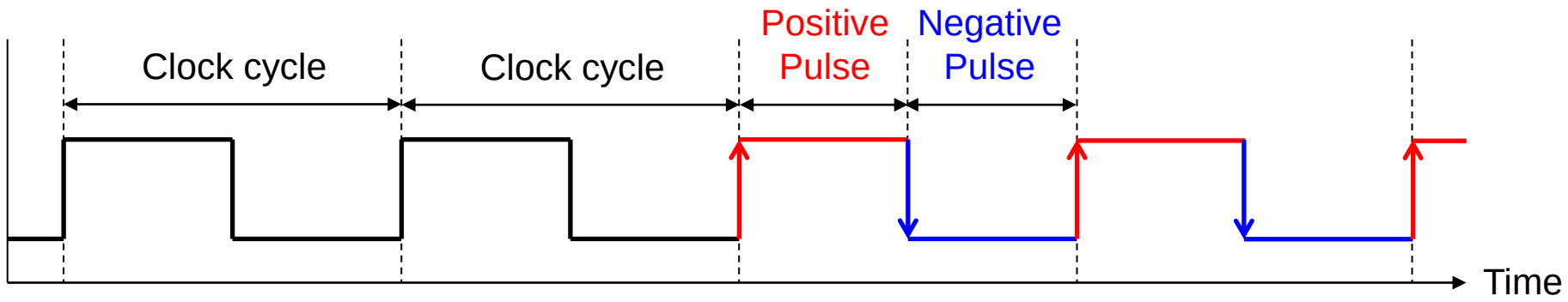
- ✧ Easier to design and analyze than asynchronous sequential circuits

Synchronous Sequential Circuits



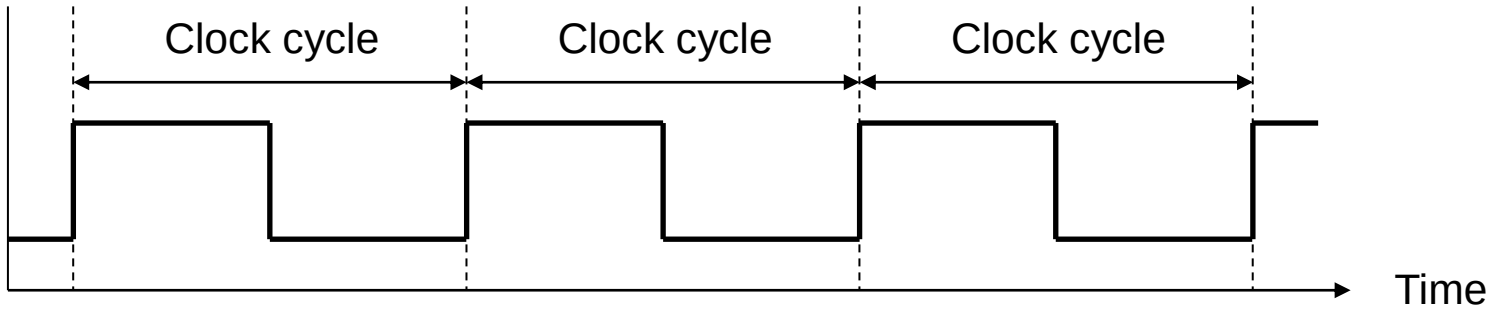
- Synchronous sequential circuits use a **clock signal**
- The clock signal is an input to the memory elements
- The clock determines **when** the memory should be updated
- The **present state** = output value of memory (stored)
- The **next state** = input value to memory (not stored yet)

The Clock



- Clock is a periodic signal = Train of pulses (1's and 0's)
- The same clock cycle repeats indefinitely over time
- **Positive Pulse**: when the **level** of the clock is **1**
- **Negative Pulse**: when the **level** of the clock is **0**
- **Rising Edge**: when the clock goes **from 0 to 1**
- **Falling Edge**: when the clock goes **from 1 down to 0**

Clock Cycle versus Clock Frequency



- Clock cycle (or period) is a time duration
  - Measured in seconds, milli-, micro-, nano-, or pico-seconds
  - 1 ms = 10^{-3} sec, 1 µs = 10^{-6} sec, 1 ns = 10^{-9} sec, 1 ps = 10^{-12} sec
- Clock frequency = number of cycles per second (Hertz)
  - 1 Hz = 1 cycle/sec, 1 KHz = 10^{3} Hz, 1 MHz = 10^{6} Hz, 1 GHz = 10^{9} Hz
- Clock frequency = 1 / Clock Cycle
  - Example: Given the clock cycle = 0.5 ns = 0.5 ×10^{-9} sec
  - Then, the clock frequency = 1/(0.5×10^{-9}) = 2×10^{9} Hz = 2 GHz

Next . . .

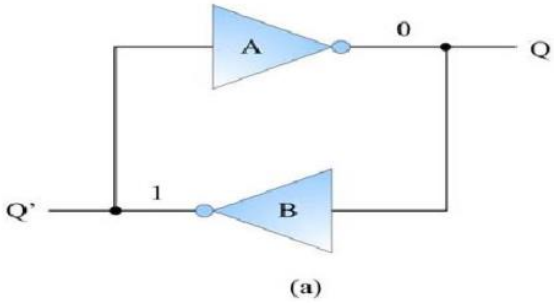
- Introduction to Sequential Circuits
  - Combinational versus Sequential Circuits
  - Synchronous versus Asynchronous Sequential Circuits
- **Storage Elements**
  - **Latches**
  - Flip-Flops
- Analysis of Clocked Sequential circuits
  - State and Output Equations
  - State Table
  - State Diagram
- Mealy versus Moore Sequential Circuits

Memory Elements

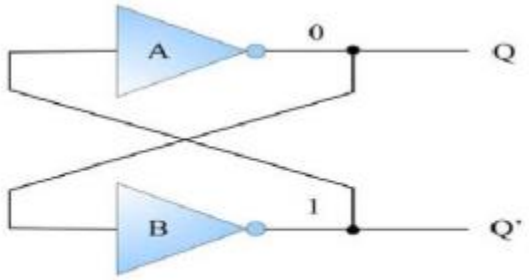
- Memory can store and maintain binary state (0's or 1's)
  - Until directed by an input signal to change state
- Main difference between memory elements
  - Number of inputs they have
  - How the inputs affect the binary state
- Two main types:
  - **Latches** are level-sensitive (sensitive to the level of the clock)
  - **Flip-Flops** are edge-sensitive (sensitive to the edge of the clock)
- Flip-Flips are used in synchronous sequential circuits
- Flip-Flops are built with latches

Latch

- A **latch**, shown in **Figure (a)**, is a memory element that can store one bit (**0** or **1**)
  - The latch circuit consists of two inverters; with the output of one connected to the input of the other
  - The latch circuit has two outputs, one for the stored value (**Q**) and one for its complement (**Q'**)
  - **Figure (b)** shows the same latch circuit re-drawn to illustrate the two complementary outputs



(a)

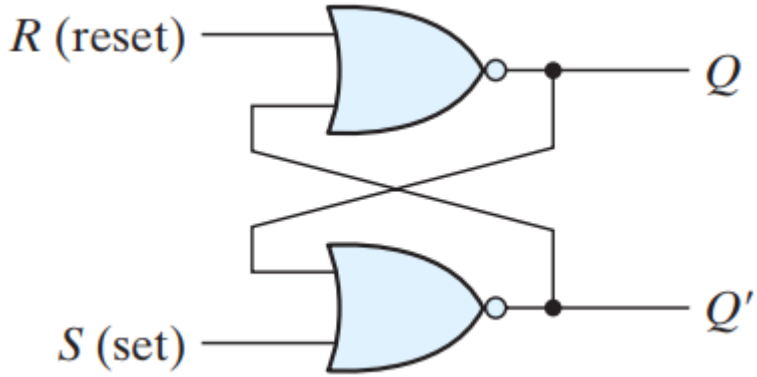


(b)

- The problem with the latch formed by NOT gates is that we can't change the stored value
  - For example, if the output of inverter B has logic 1, then it will be latched forever; and there is no way to change this value

SR Latch

- An **SR Latch** can be built using two **cross-coupled** NOR gates
- Two inputs: S (Set) and R (Reset)
- Two outputs: Q and $\overline{Q}$
  - If $S = 1$ and $R = 0$ then **Set** ($Q = 1$, $\overline{Q} = 0$)
  - If $S = 0$ and $R = 1$ then **Reset** ($Q = 0$, $\overline{Q} = 1$)
  - When $S = R = 0$, Q and $\overline{Q}$ are **unchanged**
  - When $S = R = 1$, Q and $\overline{Q}$ are **undefined** (should never be used)



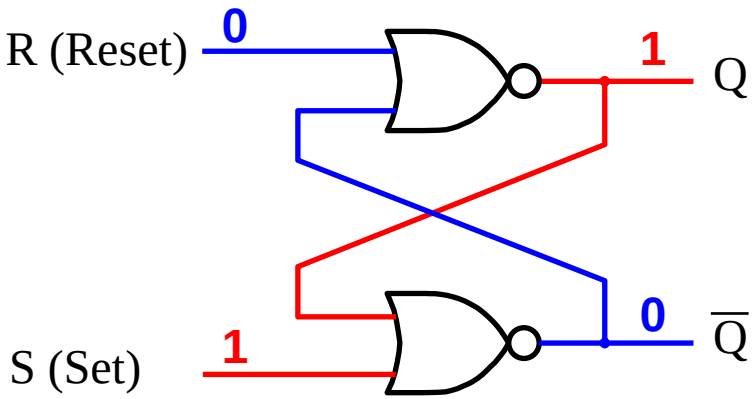
(a) Logic diagram

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

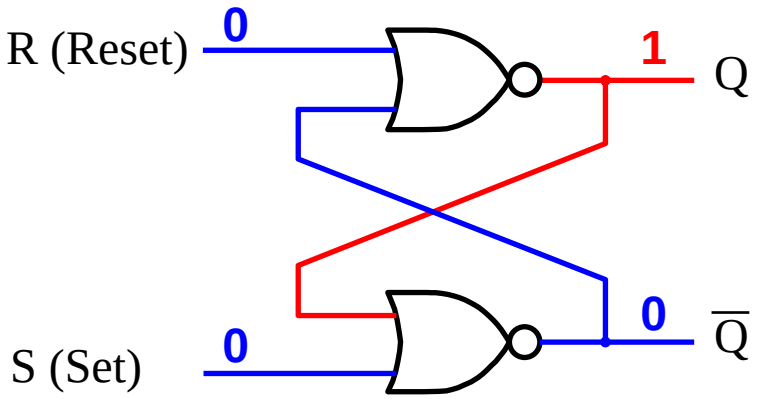
(b) Function table

SR Latch Operation

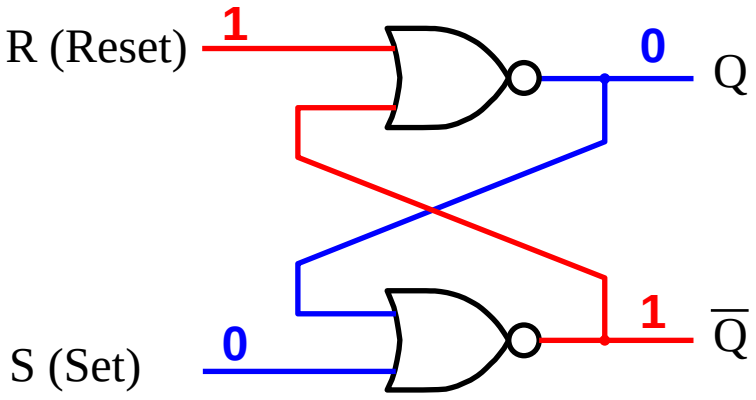
Set Operation



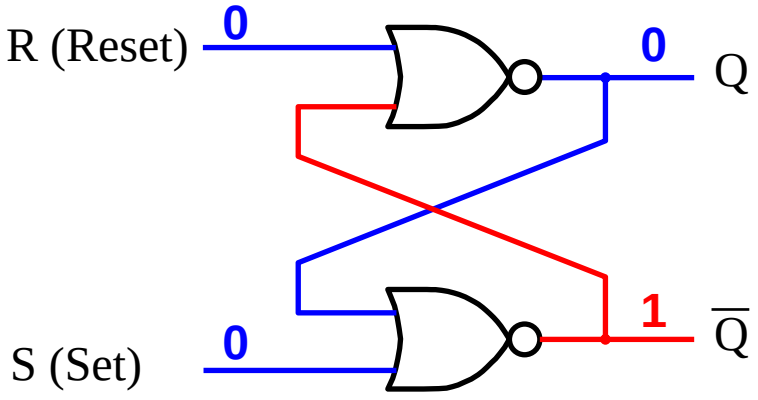
Store Operation



Reset Operation

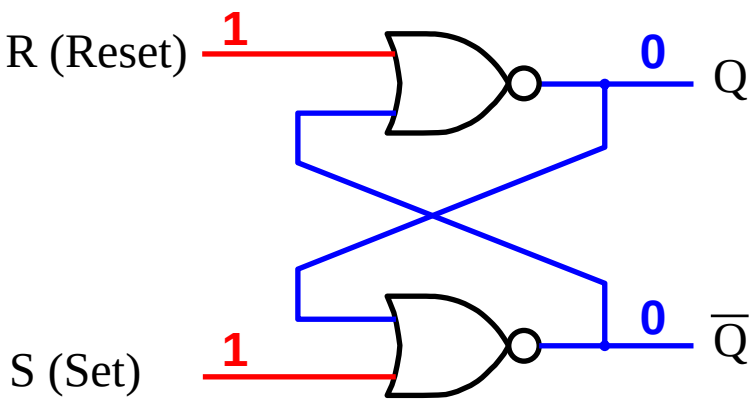


Store Operation

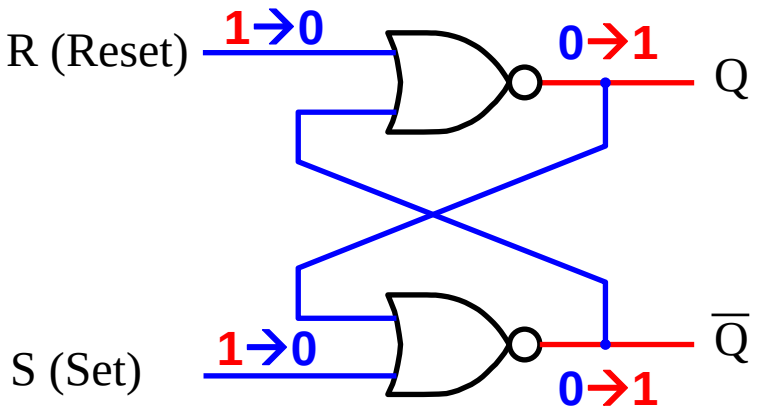


SR Latch Invalid Operation

Invalid Operation



Race Condition

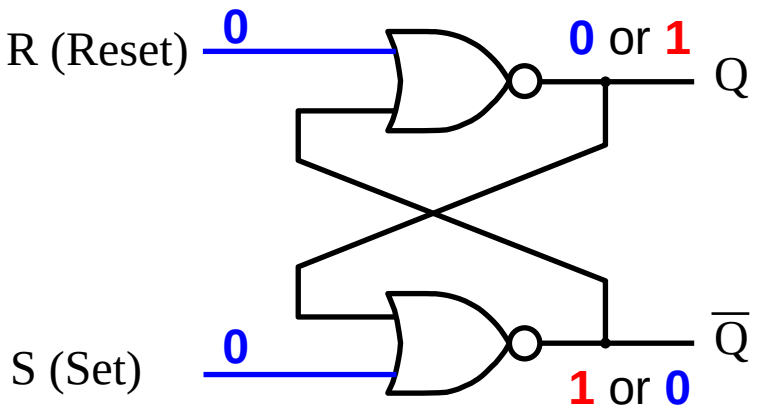


S = R = 1 should never be used

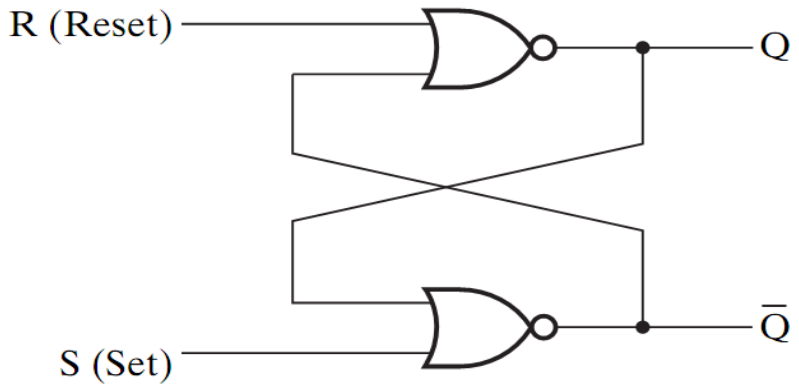
If S and R change from 1 → 0 simultaneously then race condition (oscillation) occurs

Final Q and $\overline{Q}$ are unknown

Unknown State



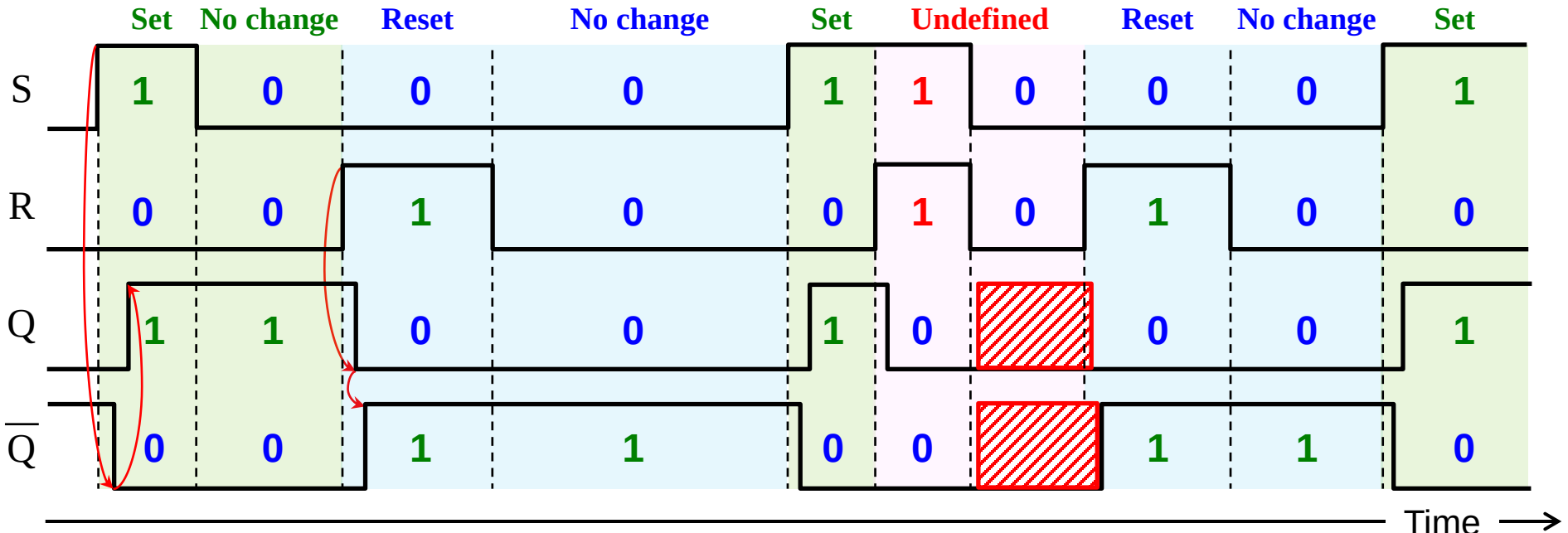
Timing Diagram of an SR Latch



(a) Logic diagram

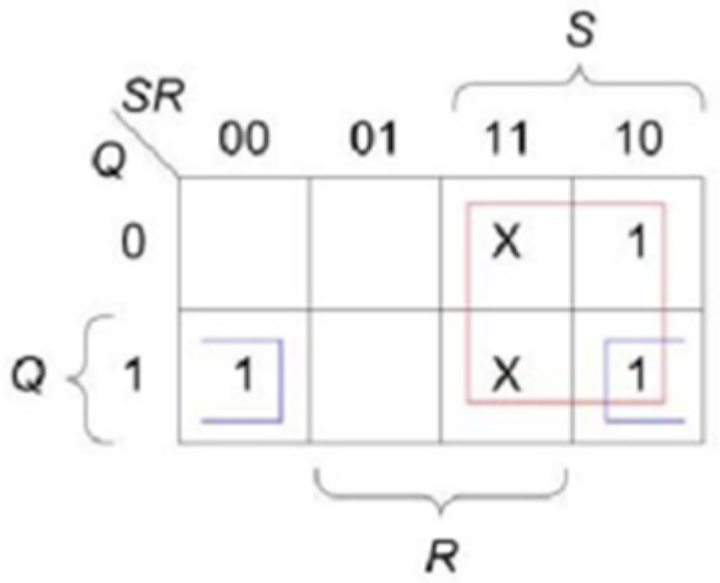
S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table



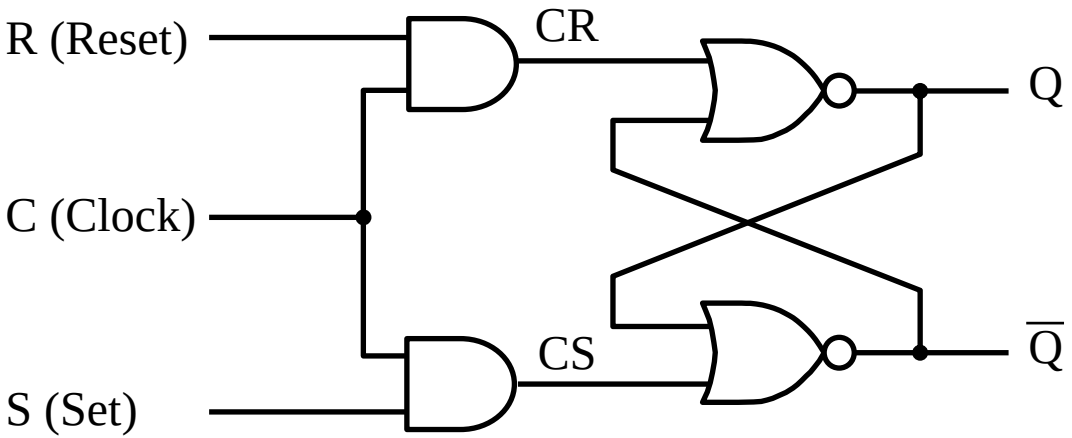
Characteristic Equation of the SR Latch

Q(t)	S	R	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate



$Q(t+1) = S + R'Q$

Gated SR Latch with Clock Enable

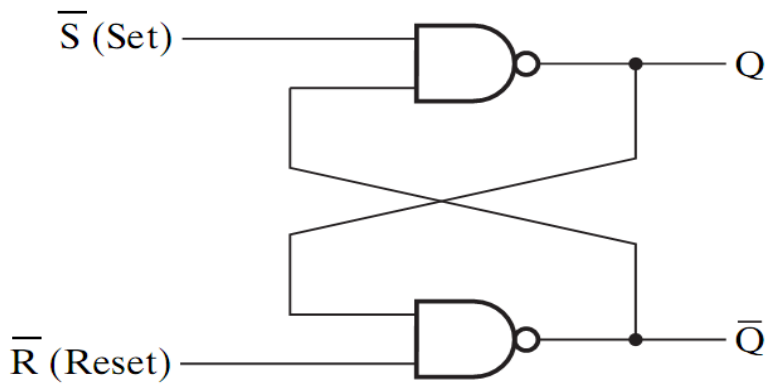


C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

- An additional Clock (enable) input signal **C** is used
- Clock controls **when** the state of the latch can be changed
- When **C=0**, the S and R inputs have no effect on the latch

  The latch will remain in the same state, regardless of S and R
- When **C=1**, then normal SR latch operation

$\overline{S}$ $\overline{R}$ Latch with NAND Gates



(a) Logic diagram

$\overline{S}$	$\overline{R}$	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

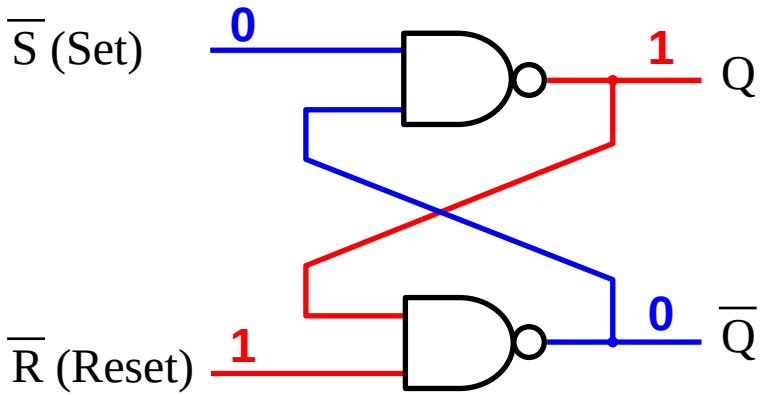
(b) Function table

Known as the $\overline{S}$ $\overline{R}$ Latch

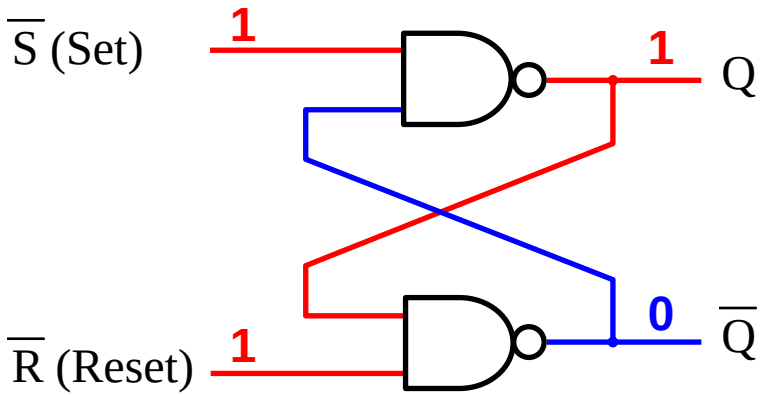
- If $\bar{S} = 0$ and $\bar{R} = 1$ then **Set** ($Q = 1$, $\overline{Q} = 0$)
- If $\bar{S} = 1$ and $\bar{R} = 0$ then **Reset** ($Q = 0$, $\overline{Q} = 1$)
- When $\bar{S} = \bar{R} = 1$, Q and $\overline{Q}$ are **unchanged** (remain the same)
- The latch stores its outputs Q and $\overline{Q}$ as long as $\bar{S} = \bar{R} = 1$
- When $\bar{S} = \bar{R} = 0$, Q and $\overline{Q}$ are **undefined** (should never be used)

$\overline{S}$ $\overline{R}$ Latch Operation

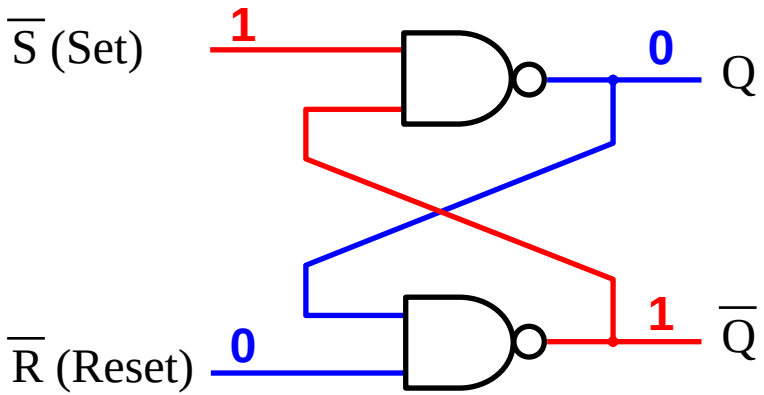
Set Operation



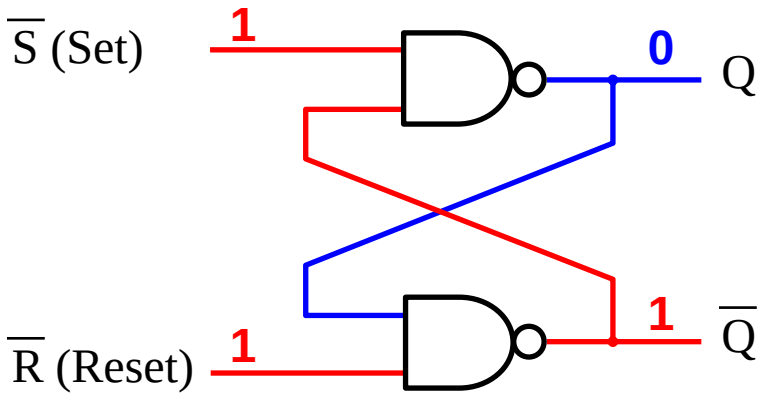
Store Operation



Reset Operation

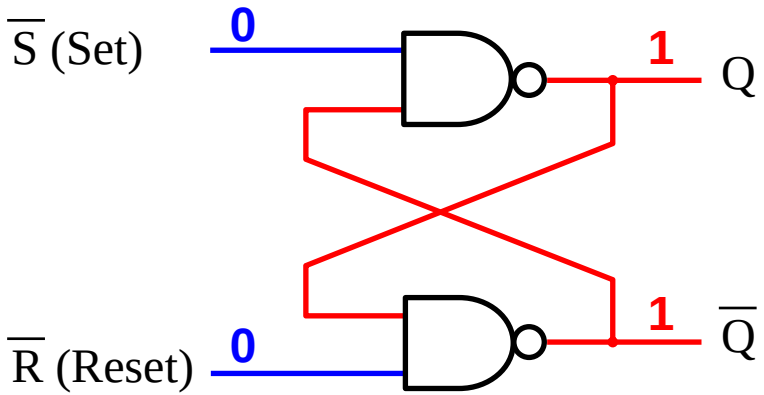


Store Operation

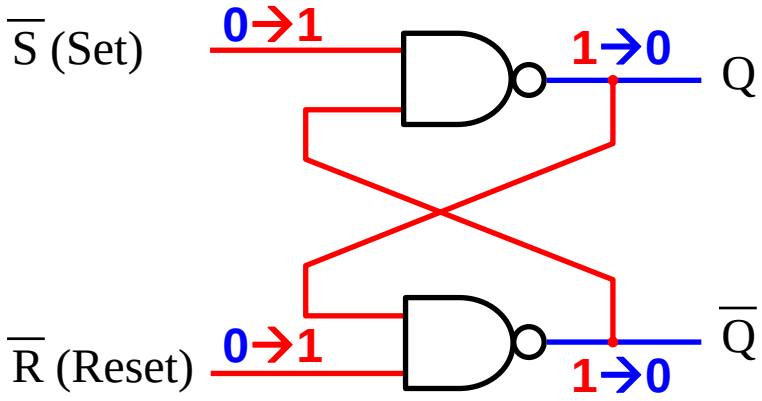


$\overline{S}$ $\overline{R}$ Latch Invalid Operation

Invalid Operation



Race Condition

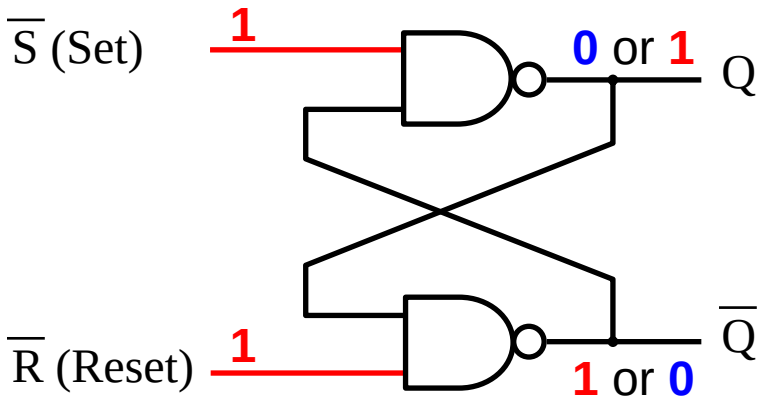


$\overline{S}$ = $\overline{R}$ = 0 should never be used

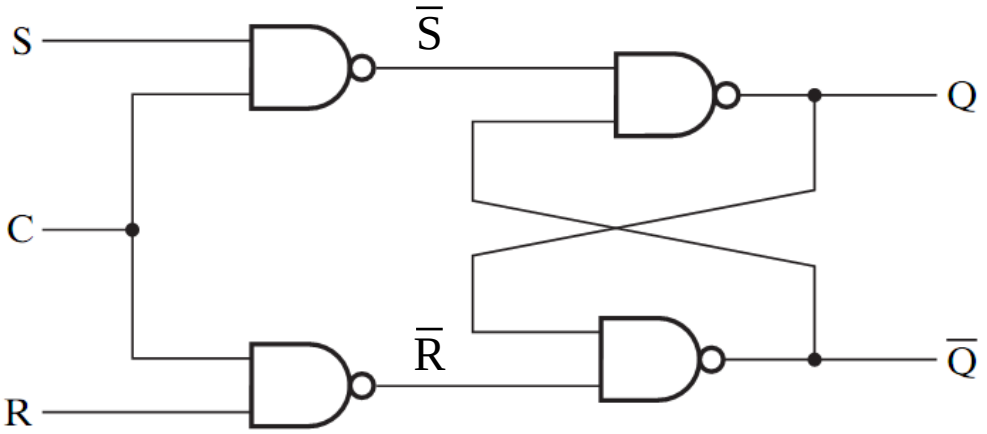
If $\overline{S}$ and $\overline{R}$ change from 0 → 1 simultaneously then race condition (oscillation) occurs

Final Q and $\overline{Q}$ are unknown

Unknown State



Gated SR Latch with Clock Enable



(a) Logic diagram

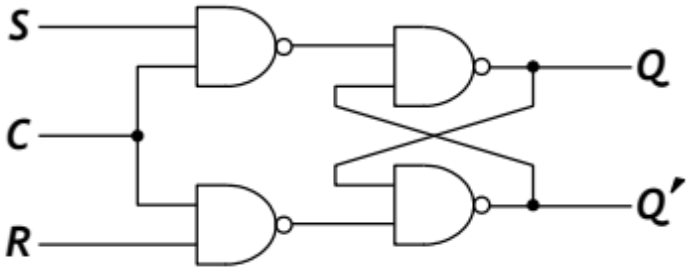
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

- An additional Clock (enable) input signal **C** is used
- Clock controls **when** the state of the latch can be changed
- When **C=0**, the S and R inputs have no effect on the latch
  - The latch remains in the same state, regardless of S and R
- When **C=1**, then normal latch operation

SR Latch with a Clock Input

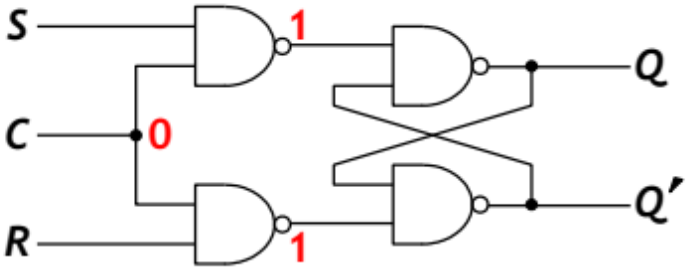
Logic Diagram



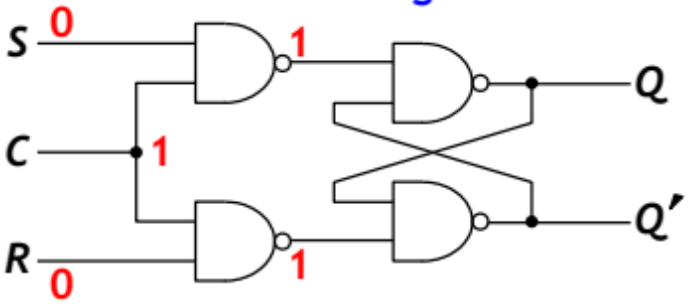
Function Table

C	S	R	Next State
0	X	X	No Change
1	0	0	No Change
1	0	1	Q=0; Reset
1	1	0	Q=1; Set
1	1	1	Indeterminate

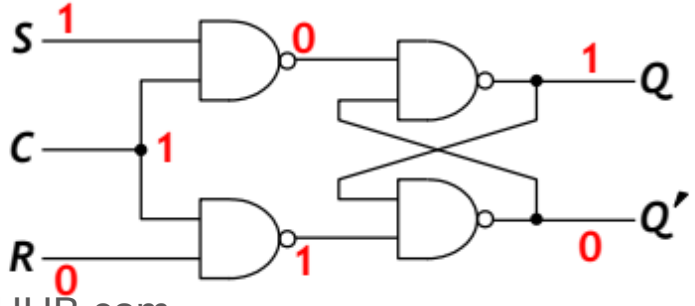
No Change



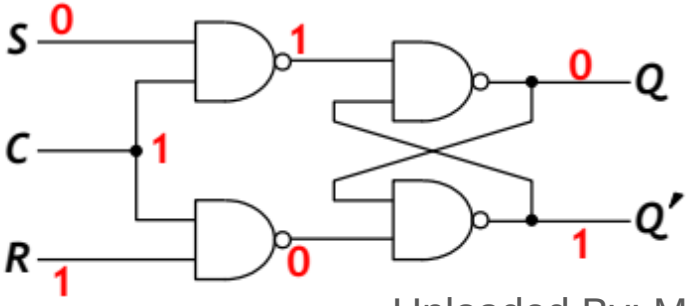
No Change



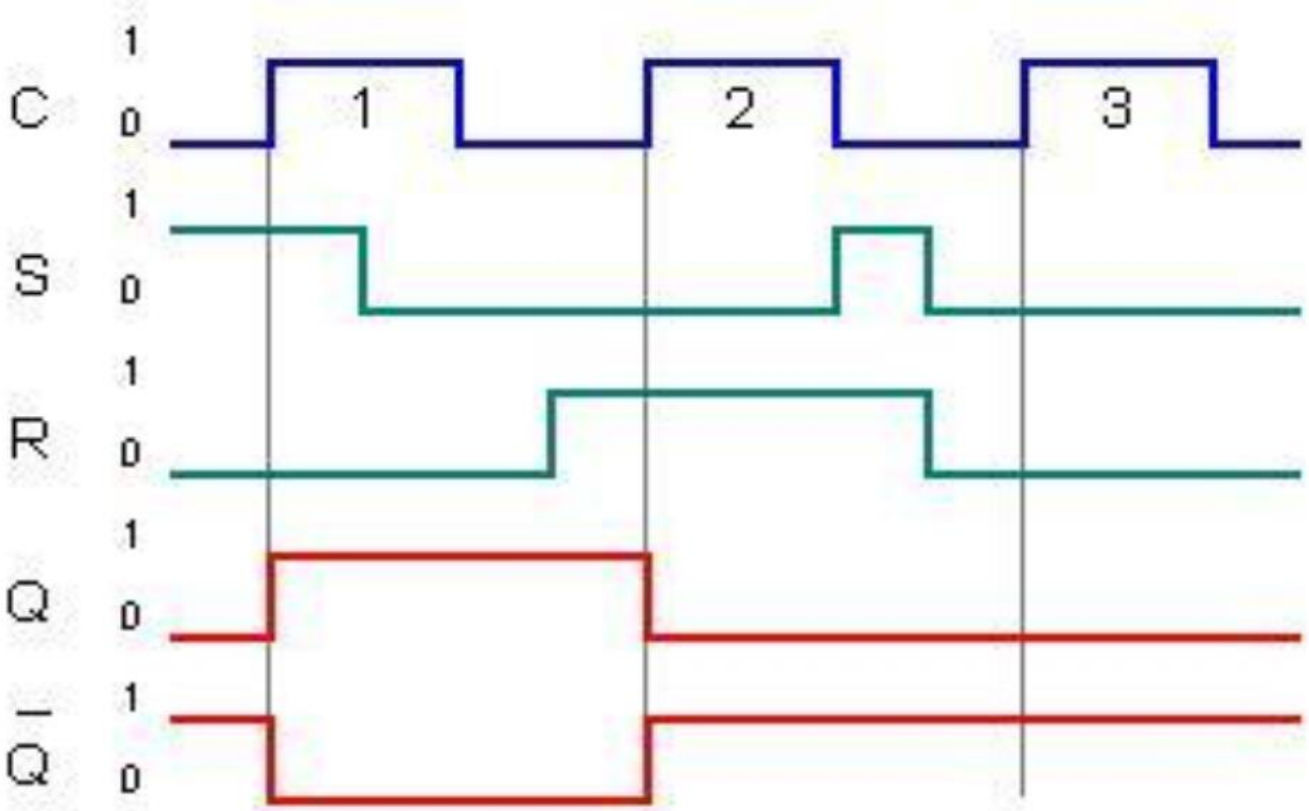
Set



Reset



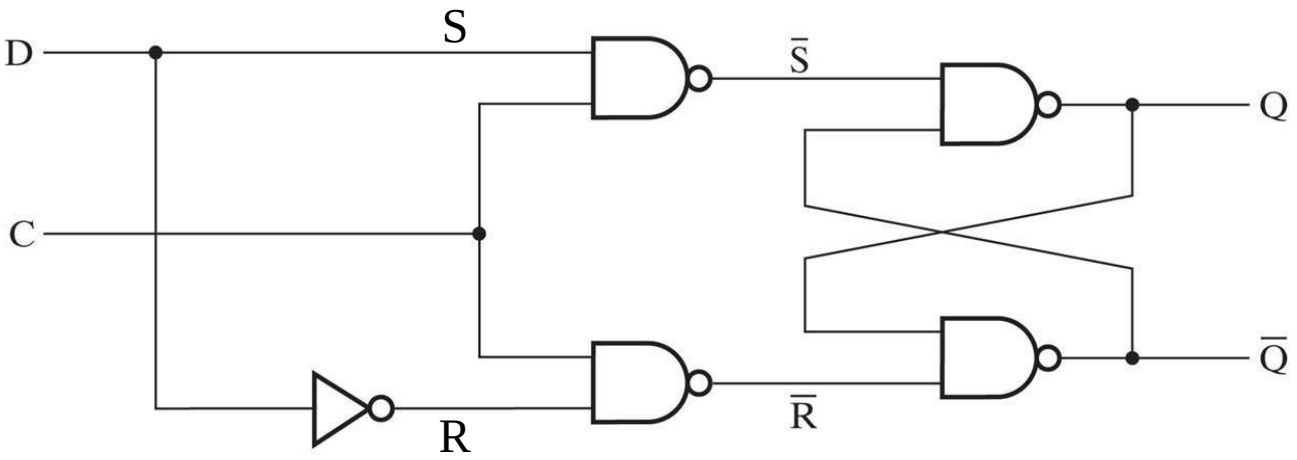
SR Latch with a Clock Input Timing Diagram



Function Table

C	S	R	Next State
0	X	X	No Change
1	0	0	No Change
1	0	1	Q=0; Reset
1	1	0	Q=1; Set
1	1	1	Indeterminate

D-Latch with Clock Enable

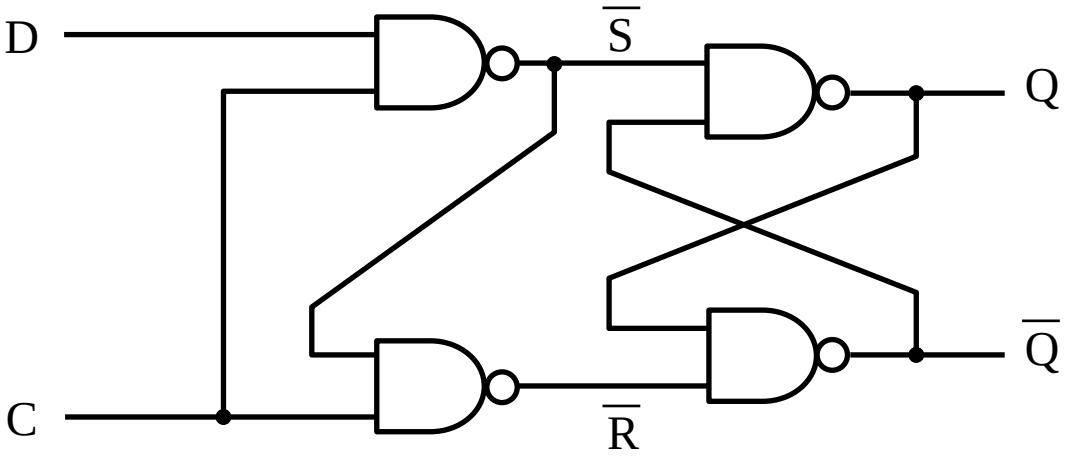


(a) Logic diagram

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

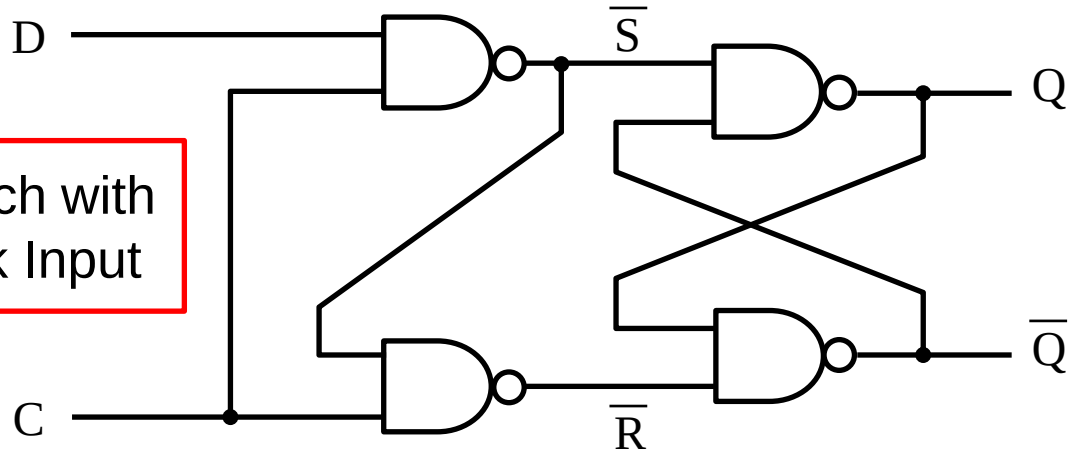
(b) Function table

- One data input D
- $S = D$ and $R = \overline{D}$
- No undefined state
- Inverter can be removed
- When $C = 1$, $R = \bar{S} = \overline{D}$



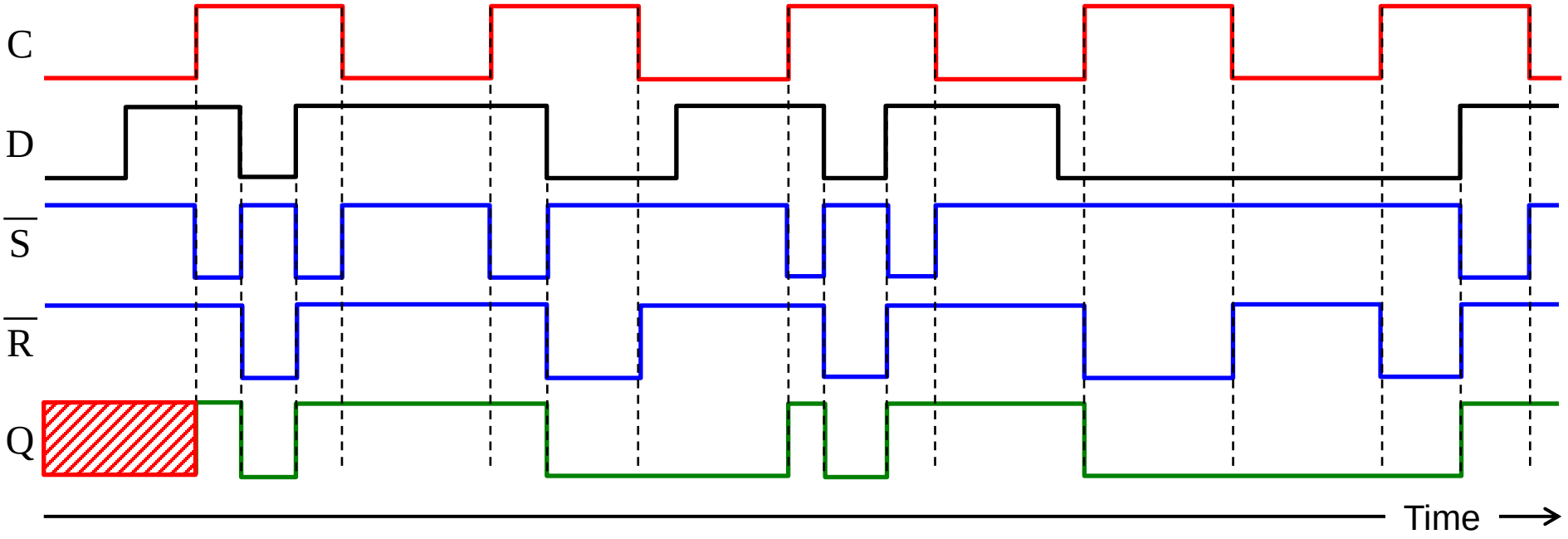
Timing of a D-Latch with Clock Enable

D-Latch with Clock Input



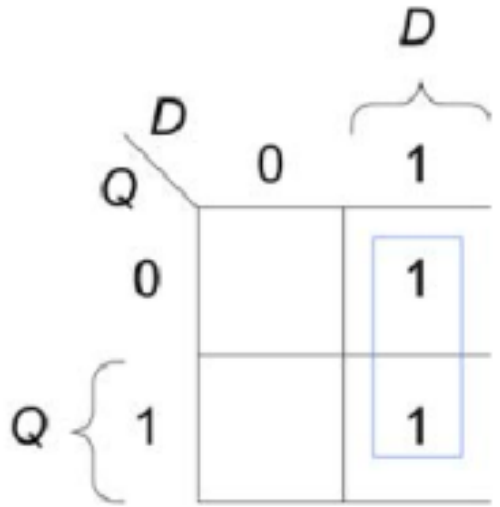
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

Function table



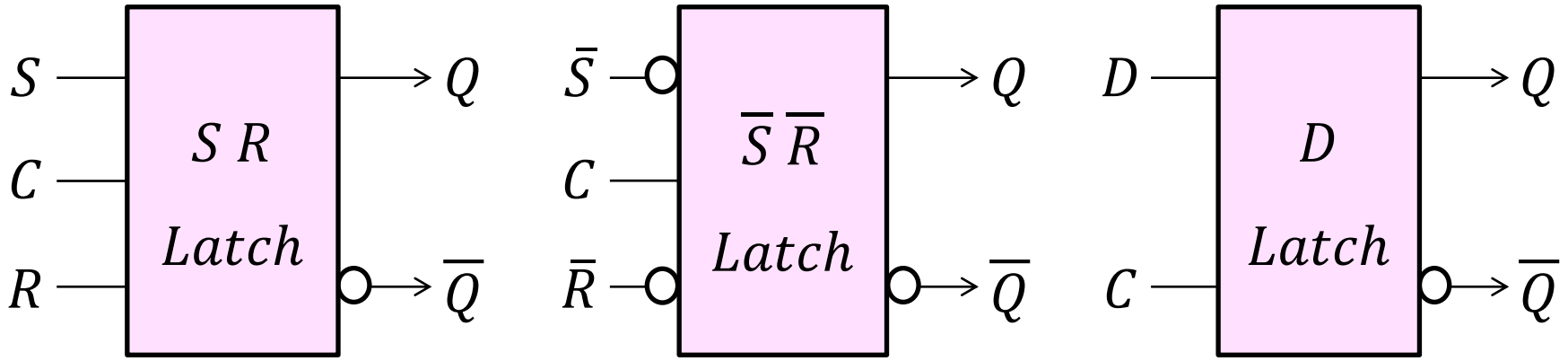
Characteristic Equation of the D-Latch

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



$Q(t+1) = D$

Graphic Symbols for Latches



- A bubble appears at the complemented output $\overline{Q}$

  Indicates that $\overline{Q}$ is the complement of Q

- A bubble also appears at the inputs of an $\overline{S}$ $\overline{R}$ latch

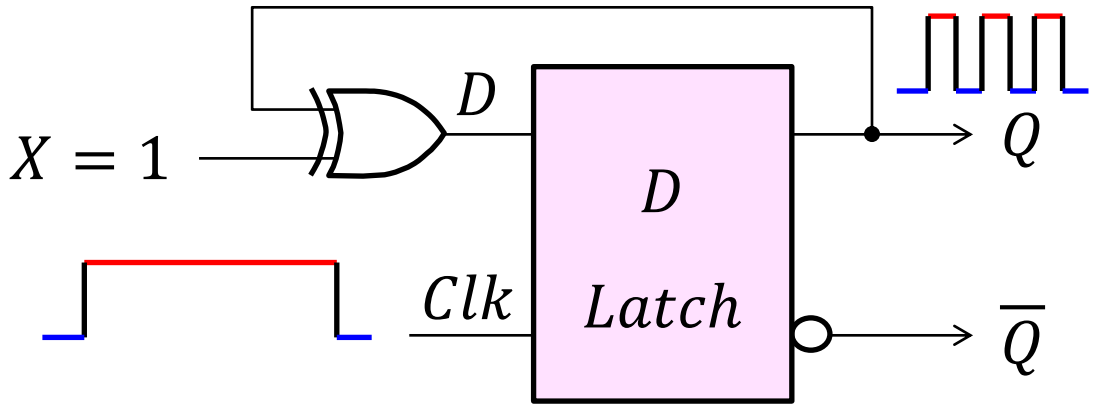
  Indicates that **logic-0** is used (not logic-1) to set (or reset) the latch (as in the NAND latch implementation)

Problem with Latches

- A latch is **level-sensitive** (sensitive to the level of the clock)
- As long as the clock signal is **high** …

  Any change in the value of input D appears in the output Q
- Output Q keeps changing its value during a clock cycle
- Final value of output Q is uncertain

Due to this uncertainty, latches are NOT used as memory elements in synchronous circuits

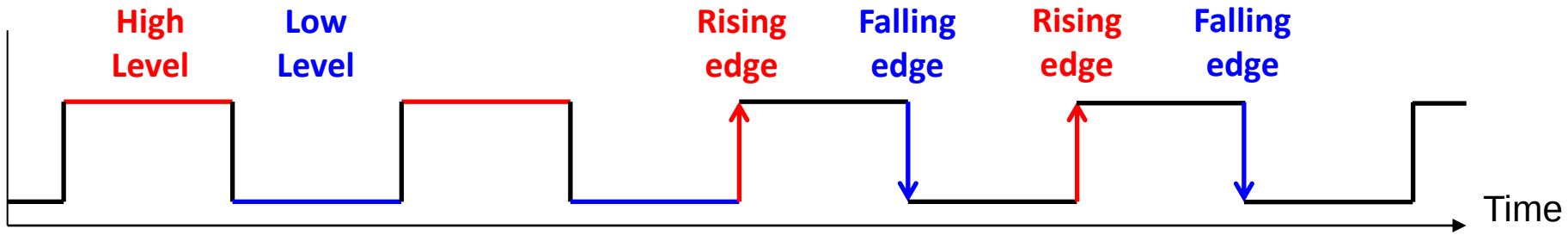


Next . . .

- Introduction to Sequential Circuits
  - Combinational versus Sequential Circuits
  - Synchronous versus Asynchronous Sequential Circuits
- **Storage Elements**
  - Latches
  - **Flip-Flops**
- Analysis of Clocked Sequential circuits
  - State and Output Equations
  - State Table
  - State Diagram
- Mealy versus Moore Sequential Circuits

Flip-Flops

- A **Flip-Flop** is a better memory element for synchronous circuits
- Solves the problem of latches in synchronous sequential circuits
- A **latch** is sensitive to the **level** of the clock
- However, a **flip-flop** is sensitive to the **edge** of the clock
- A flip-flop is called an **edge-triggered** memory element
- It changes it output value at the **edge** of the clock

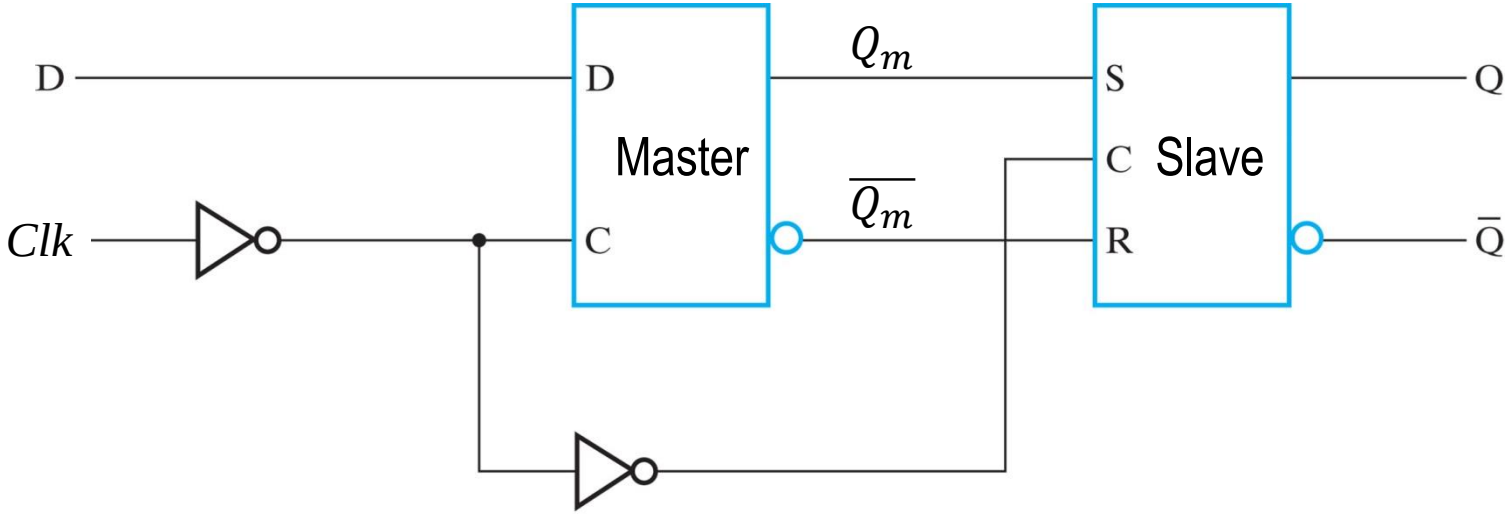


Positive Edge-Triggered D Flip-Flop

- Built using two latches in a **master-slave** configuration
- A master latch (D-type) receives external inputs
- A slave latch (SR-type) receives inputs from the master latch
- Only one latch is enabled at any given time

  When **Clk=0**, the master is enabled and the D input is latched (slave disabled)

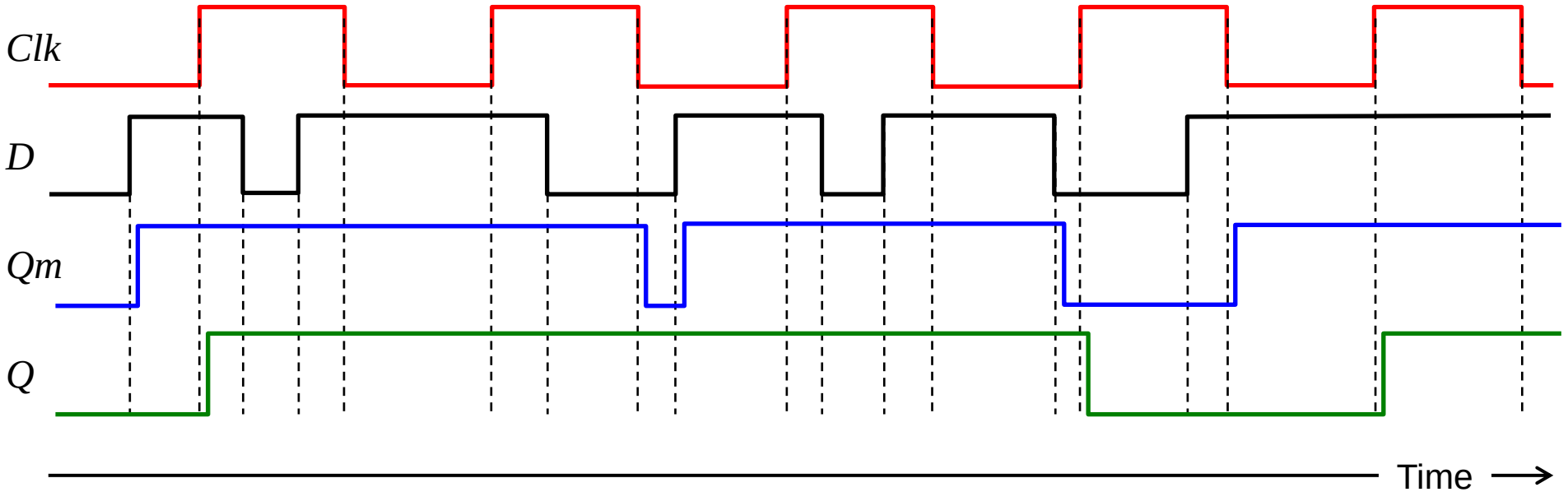
  When **Clk=1**, the slave is enabled to generate the outputs (master is disabled)



Outputs change when *Clk* changes **from 0 to 1**

D Flip-Flop Timing Diagram

- The diagram shows the timing of a positive-edge D Flip-Flop
- The master latch changes its output *Qm* when the clock *C* is 0
- The rising edge of the clock triggers the D Flip-Flop
- Notice the slight delay in the output *Q* after the rising edge

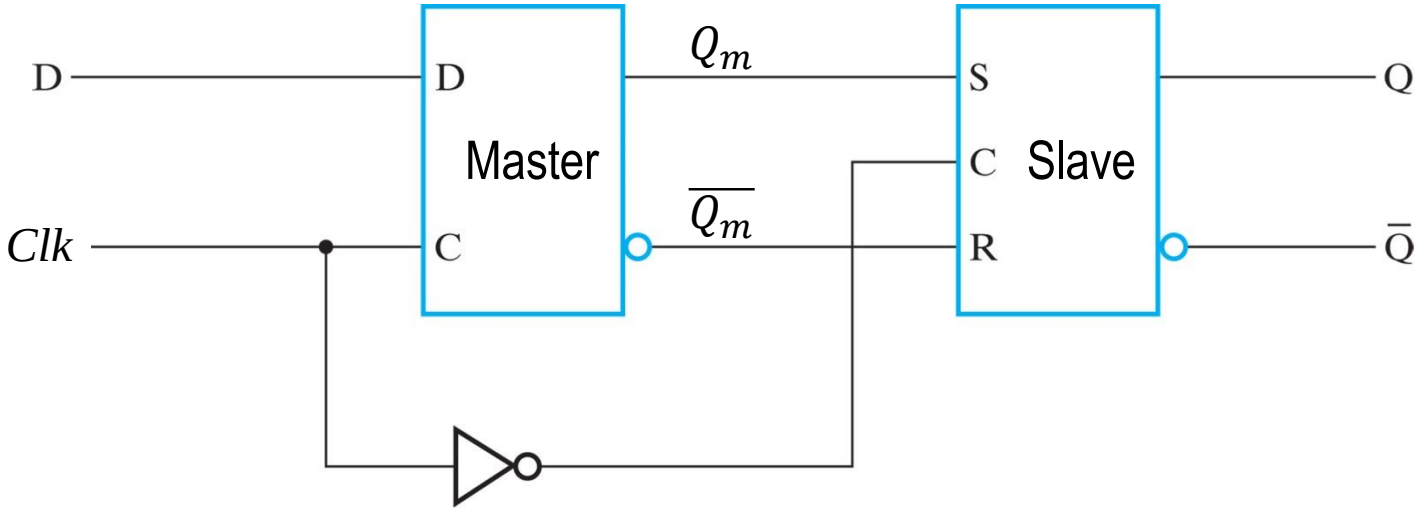


Negative Edge-Triggered D Flip-Flop

- Similar to positive edge-triggered flip-flop
- The first inverter at the Master C input is removed
- Only one latch is enabled at any given time

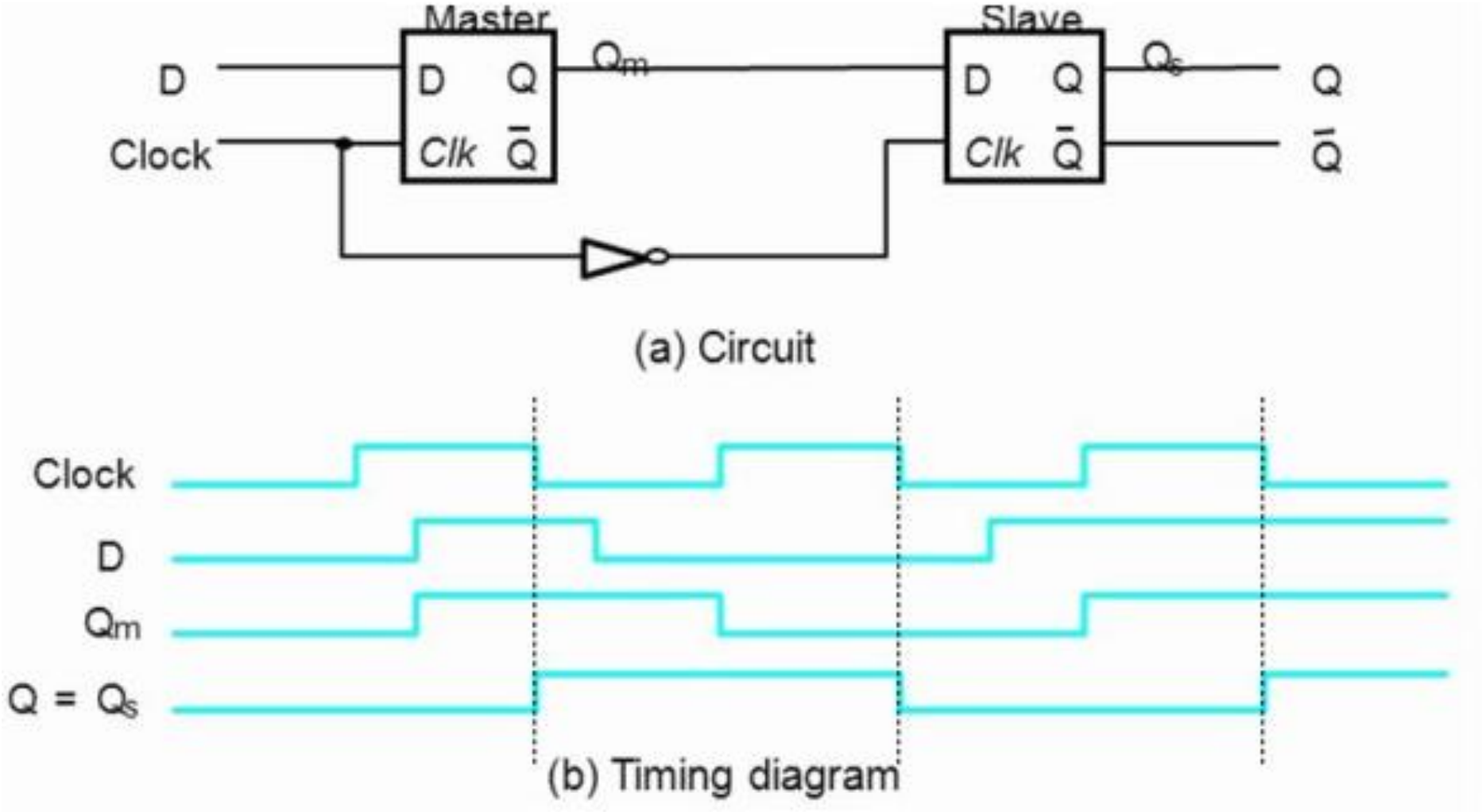
When **Clk=1**, the master is enabled and the D input is latched (slave disabled)

When **Clk=0**, the slave is enabled to generate the outputs (master is disabled)

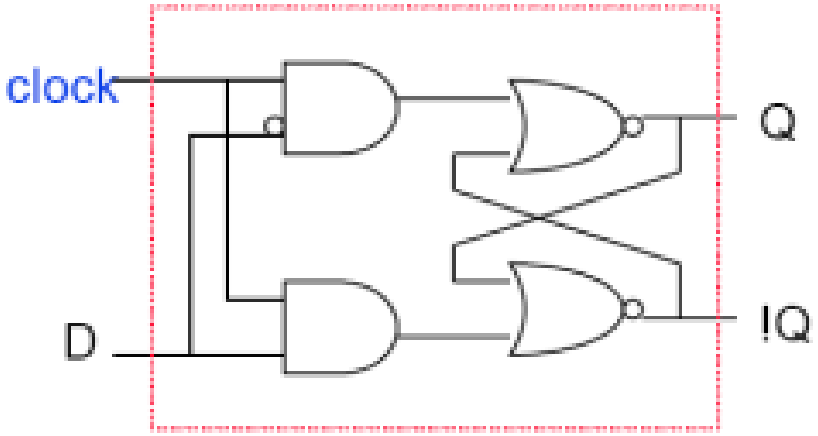


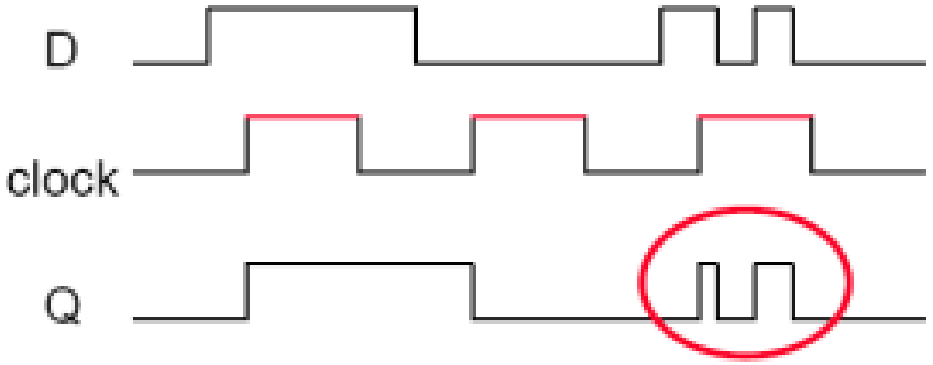
Outputs change when *Clk* changes **from 1 to 0**

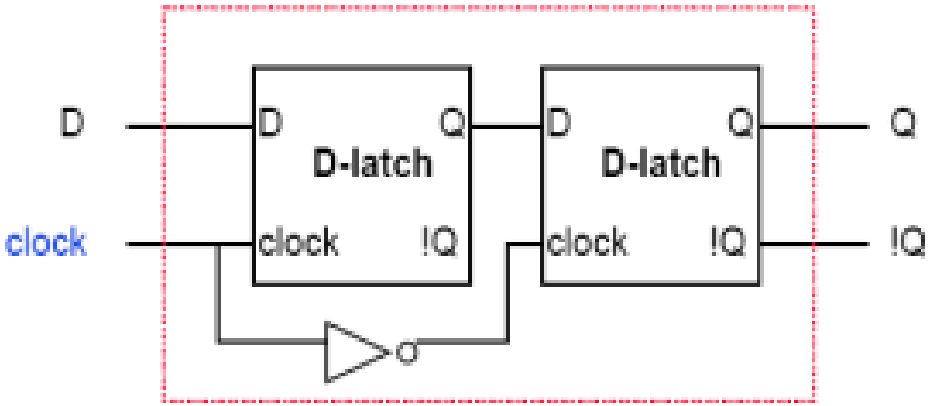
Negative-Edge D Flip-Flop Timing Diagram

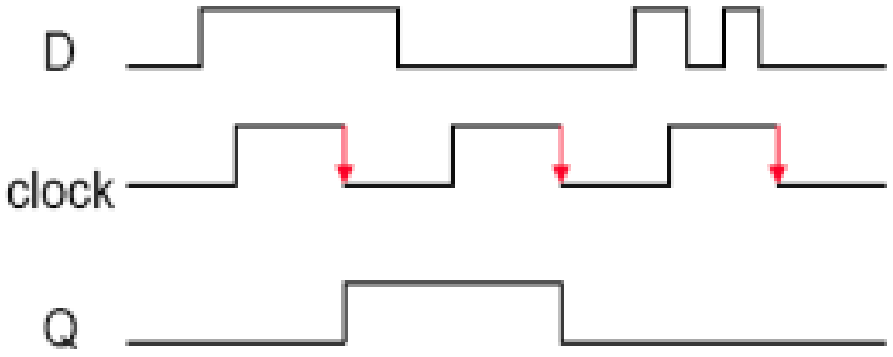


D-Latch VS Edge-Triggered D Flip-Flop

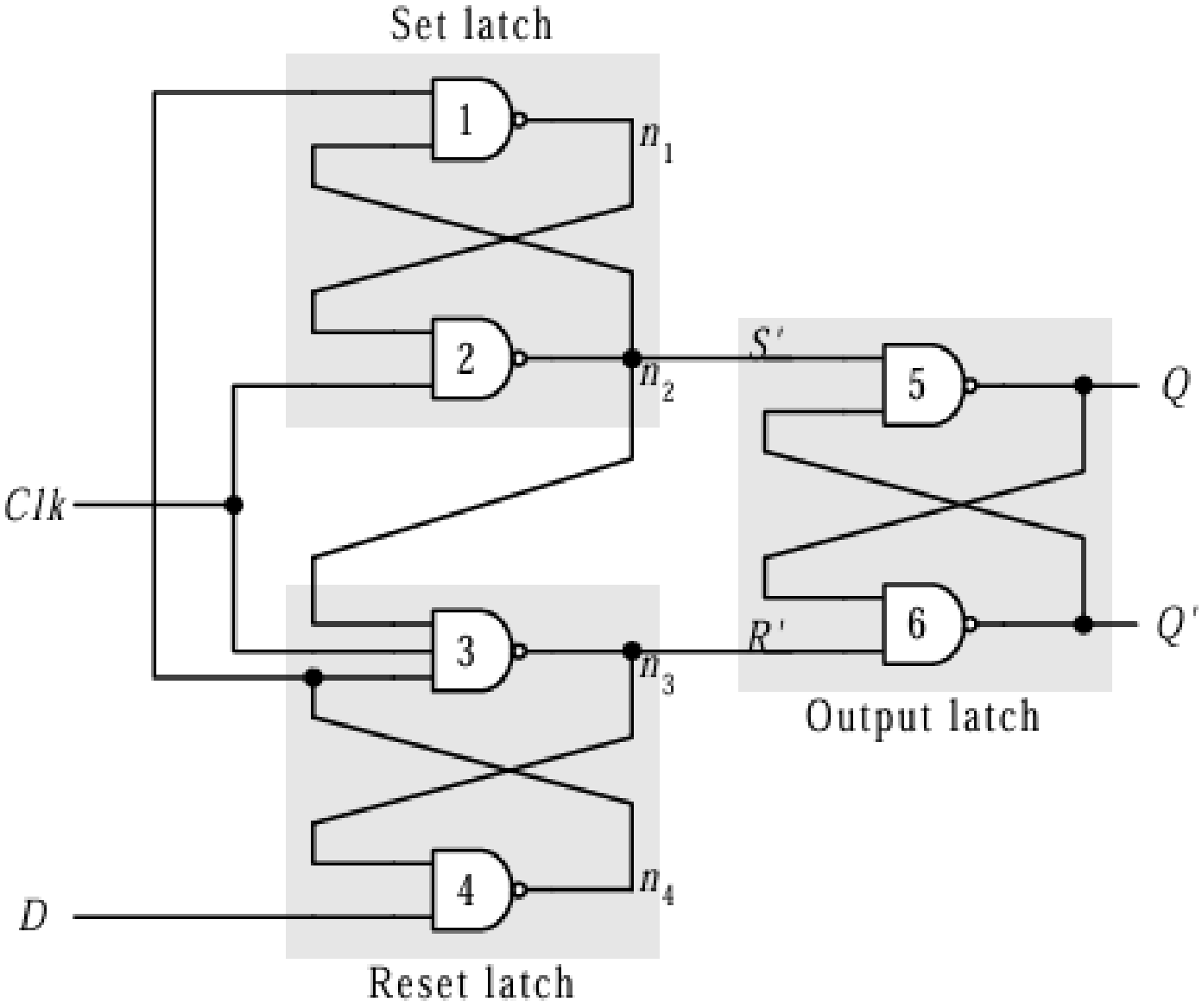




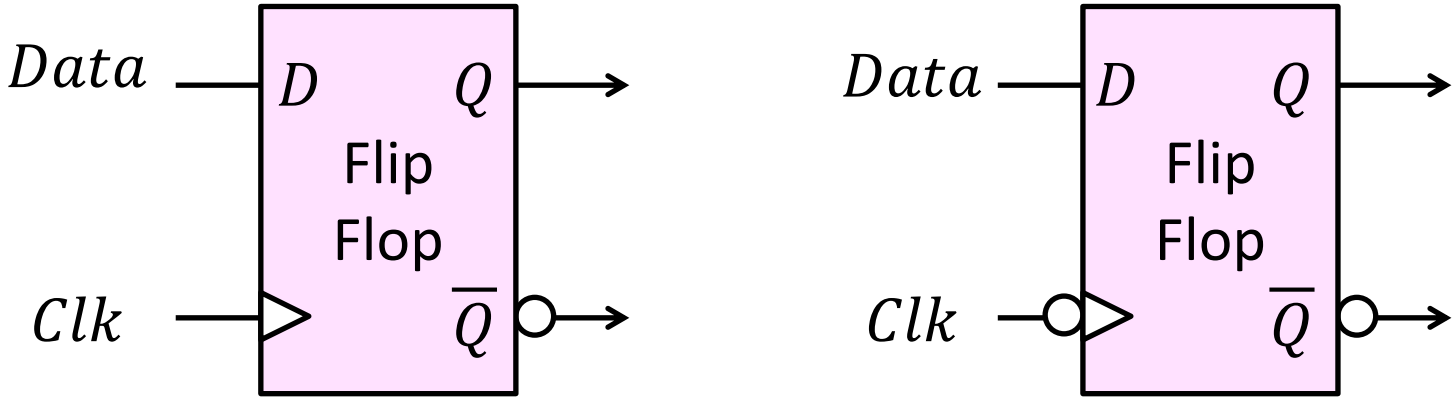




Another Construction - Positive Edge D-FF



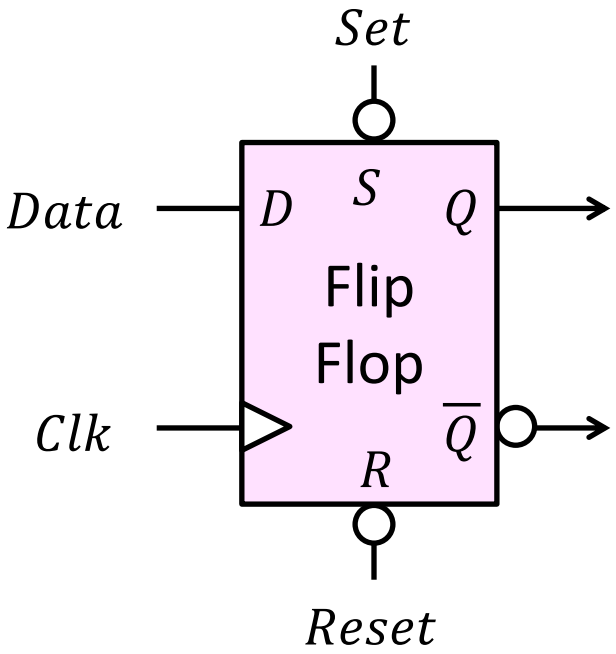
Graphic Symbols for Flip-Flops



- A Flip-Flop has a similar symbol to a Latch
- The difference is the arrowhead at the clock input
- The arrowhead indicates sensitivity to the edge of the clock
- A circle at the *Clk* input indicates negative edge-triggered FF

D-FF with Asynchronous Set and Reset

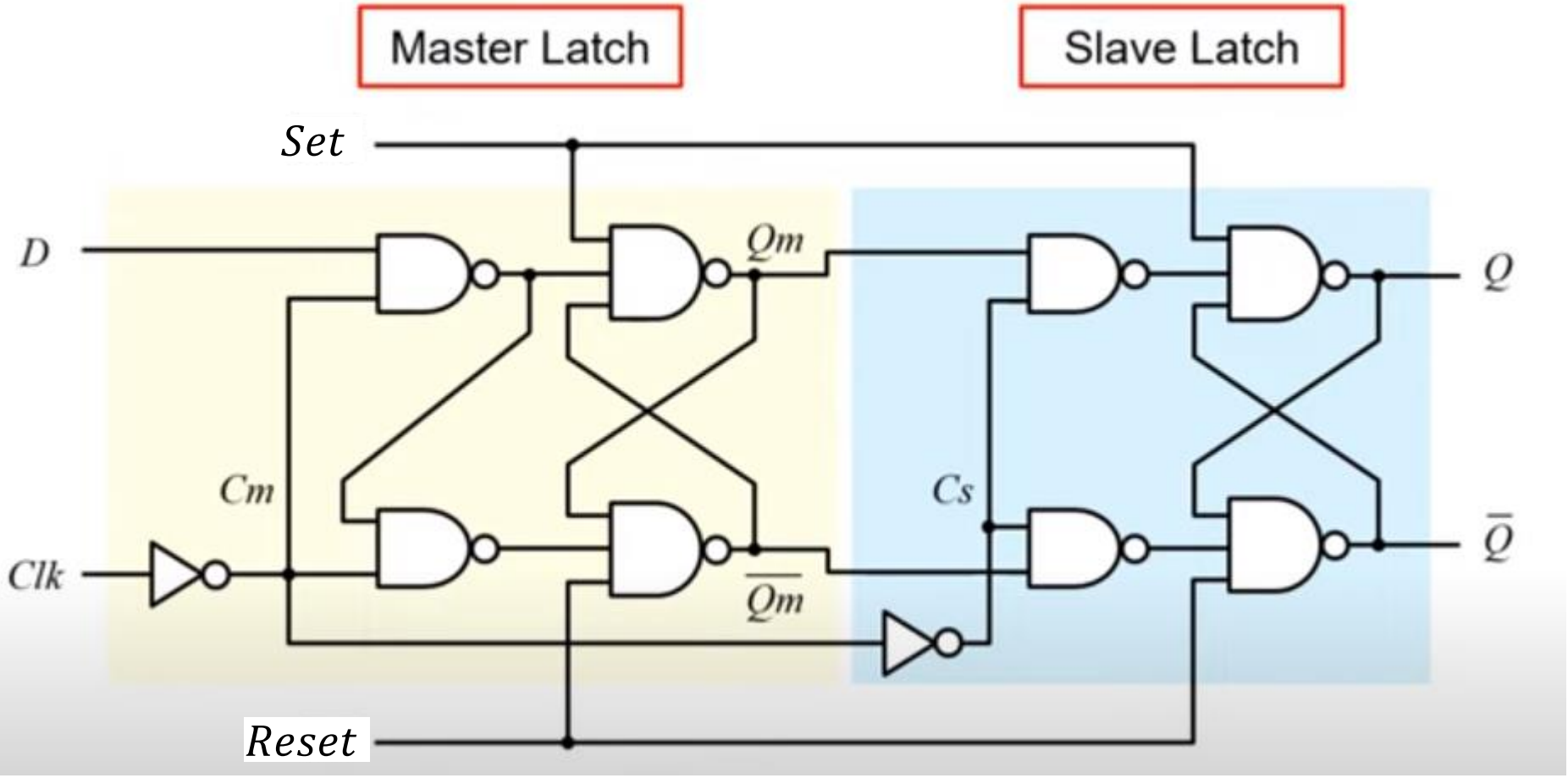
- When Flip-Flops are powered, their initial state is **unknown**
- Some flip-flops have an **asynchronous Set** and/or **Reset** inputs
- Set forces *Q* to become **1**, independently of the clock
- Reset forces *Q* to become **0**, independently of the clock



Inputs				Outputs	
Set	*Reset*	*Data*	*Clk*	*Q*	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
1	1	0	↑	0	1
1	1	1	↑	1	0

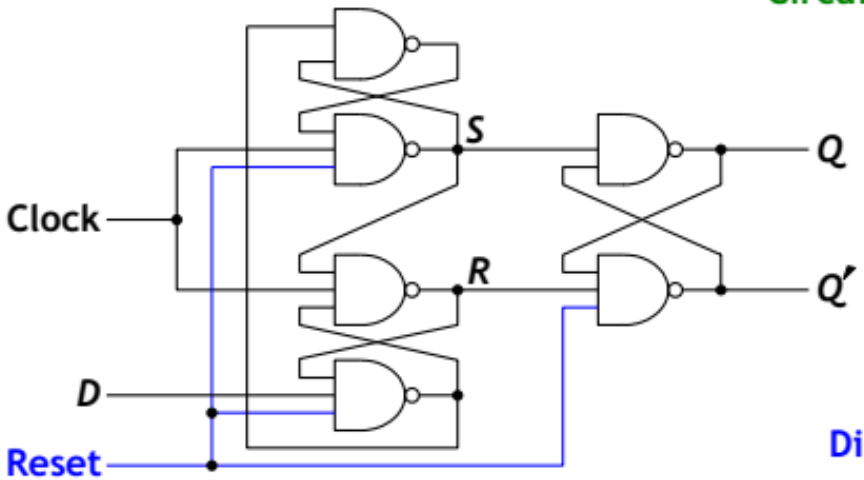
Function Table

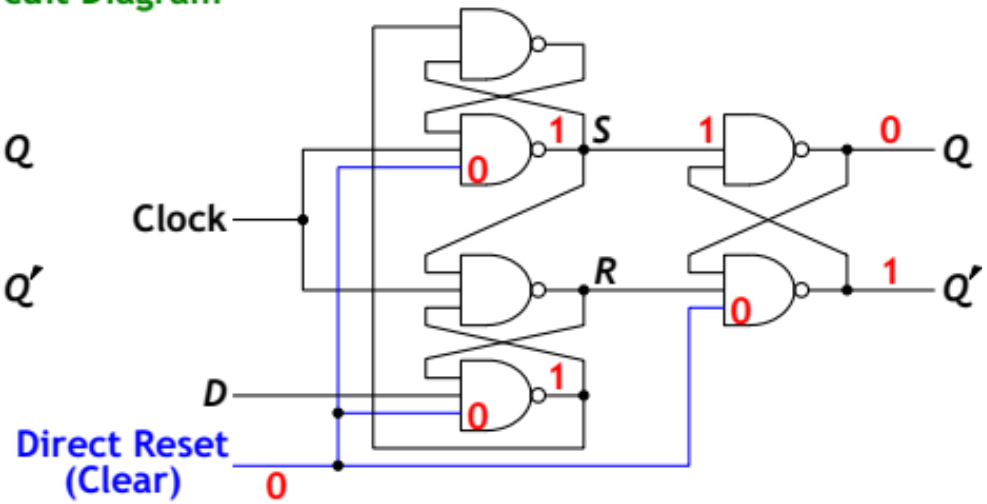
D-FF with Asynchronous Set and Reset



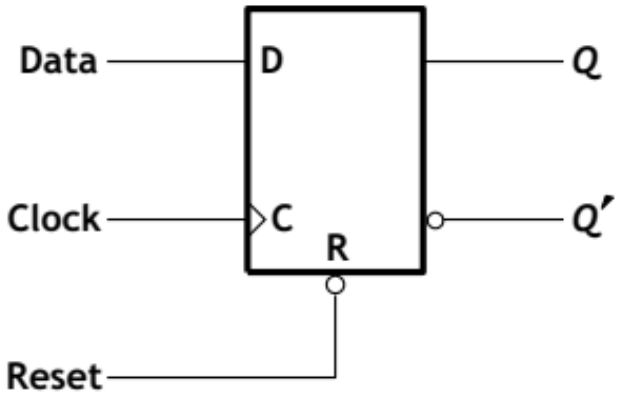
D Flip-Flop with Asynchronous Reset

Circuit Diagram





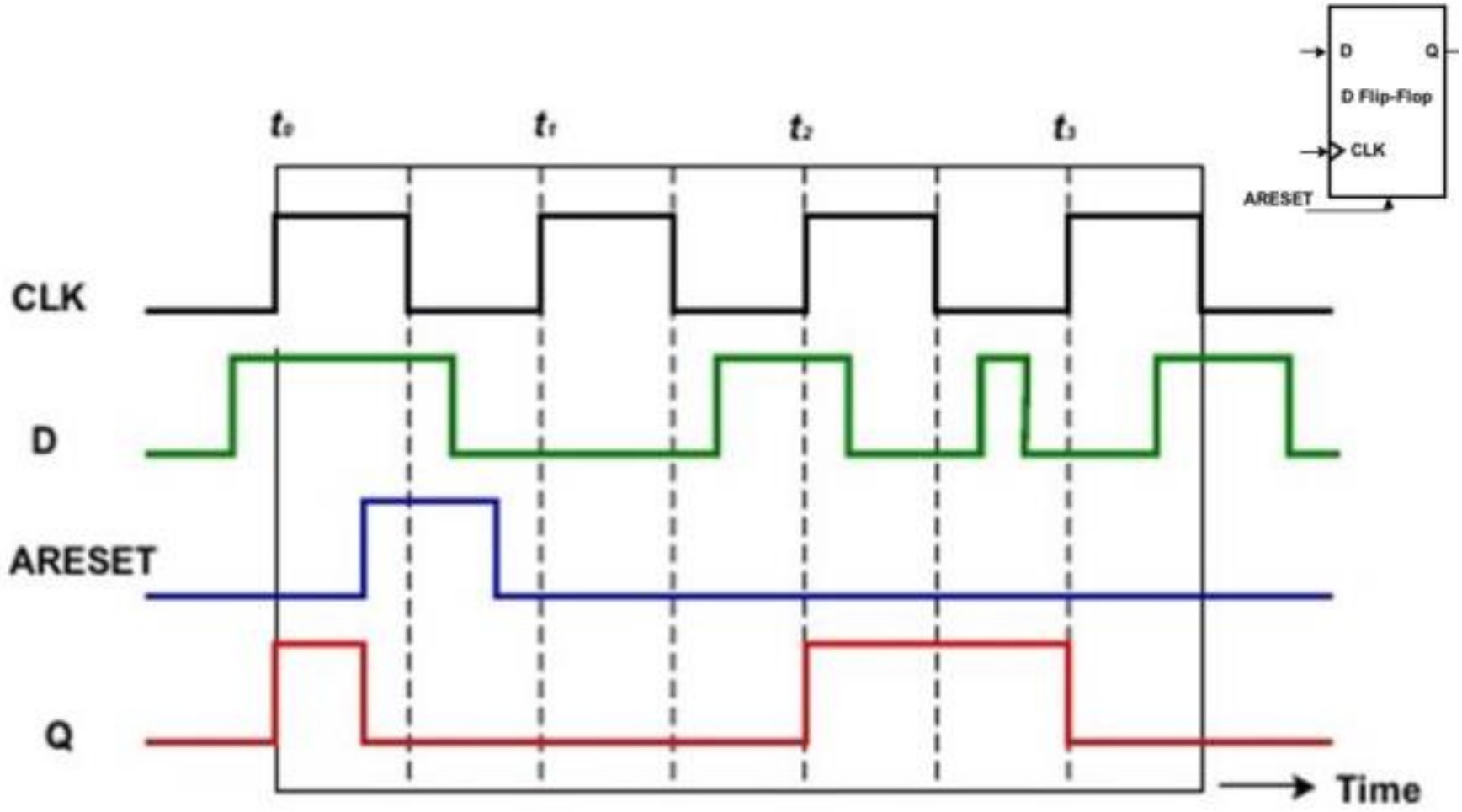
Graphic Symbol



Function Table

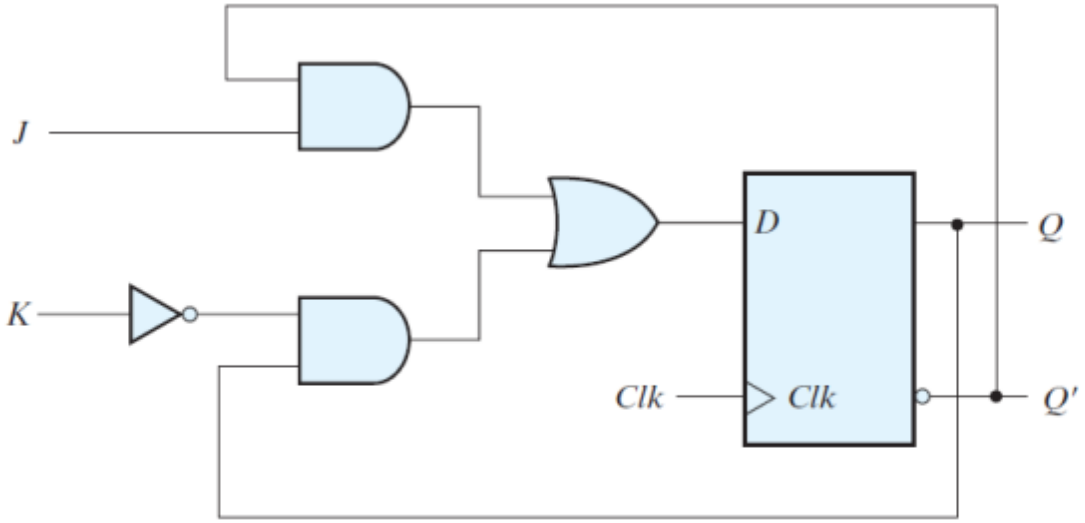
R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

D Flip-Flop with Asynchronous Reset

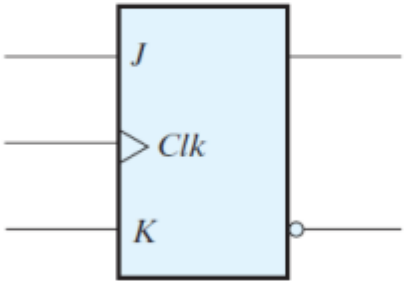


JK Flip-Flop

- The *D* Flip-Flop is the most commonly used type
- The *JK* is another type of Flip-Flop with inputs: *J, K*, and *Clk*
- When *JK* = 10 ➔ Set, When *JK* = 01 ➔ Reset
- When *JK* = 00 ➔ No change, When *JK* = 11 ➔ Invert outputs
- *JK* can be implemented using D-FF and gates



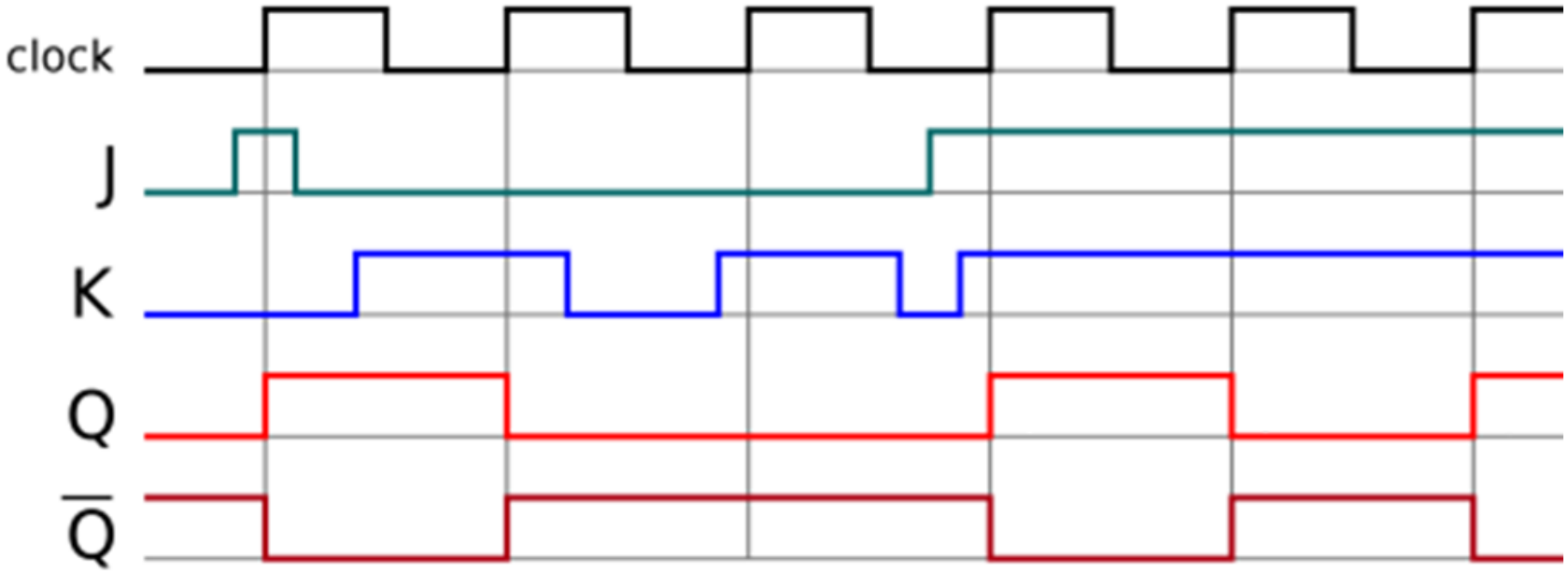
(a) Circuit diagram



(b) Graphic symbol

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q}_t$

JK Flip-Flop Timing Diagram

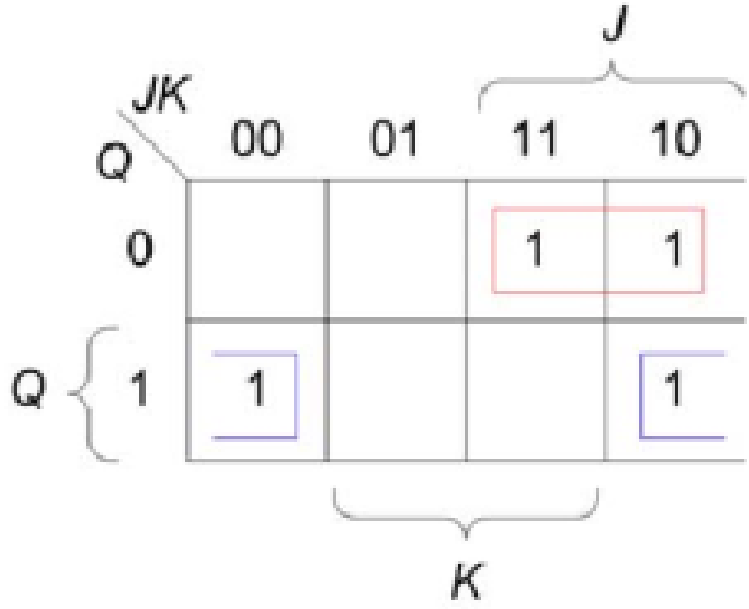


J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q}_t$

Characteristic Equation of the JK Flip-Flop

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q}_t$

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



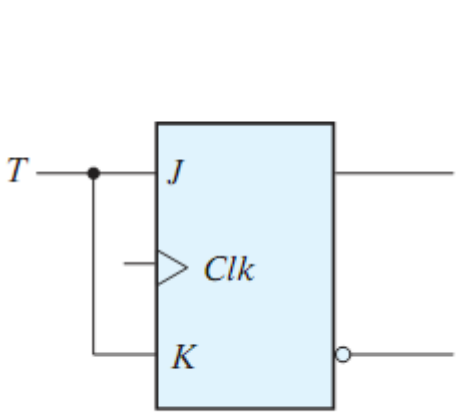
$Q(t+1) = JQ' + K'Q$

T Flip-Flop

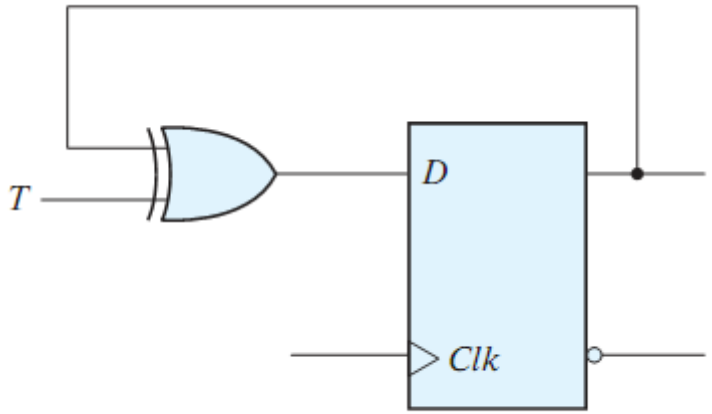
- The *T* (Toggle) flip-flop has inputs: *T* and *Clk*
- When *T* = 0 ➔ No change,
- When *T* = 1 ➔ Invert outputs

T	Q_{t+1}
0	Q_t
1	$\overline{Q}_t$

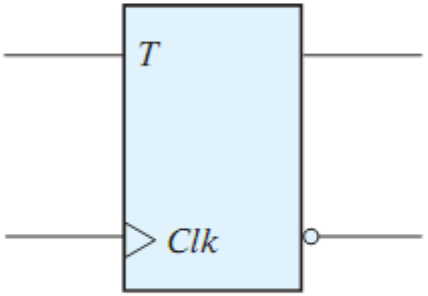
- The *T* flip-flop can be implemented using a *JK* flip-flop
- It can also be implemented using a *D* flip-flop and a XOR gate



(a) From *JK* flip-flop

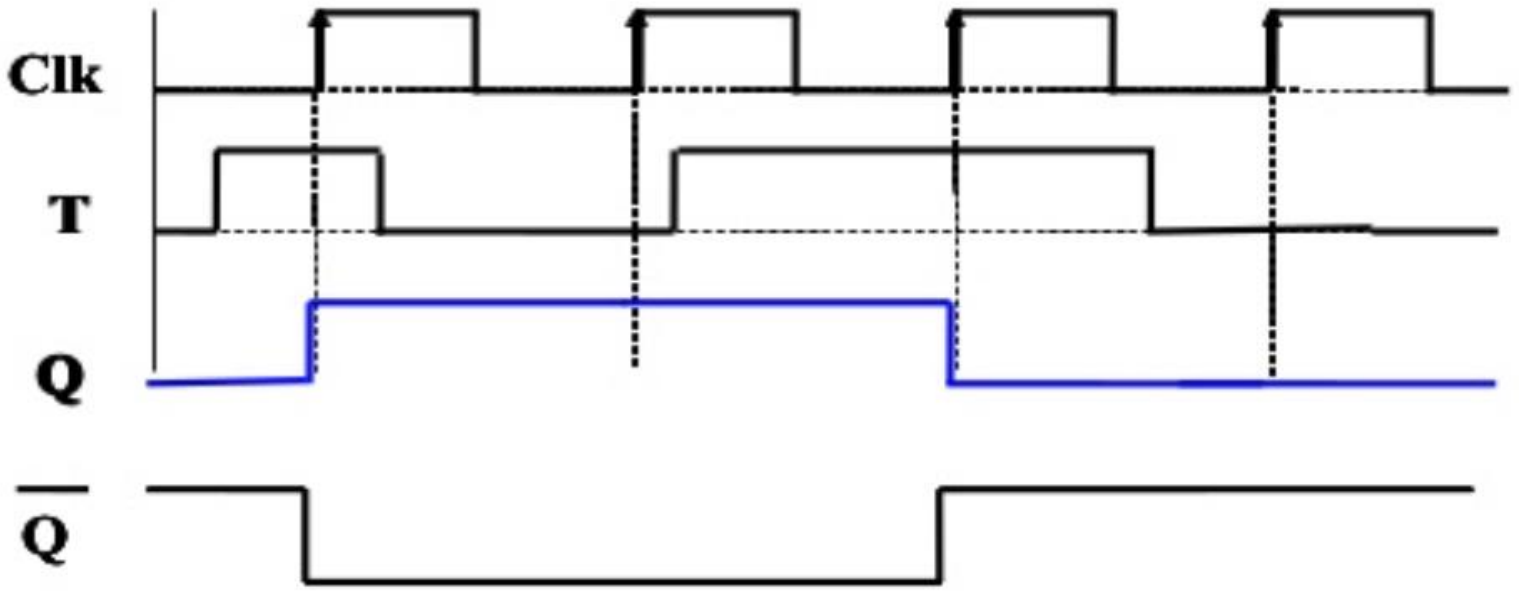


(b) From *D* flip-flop



(c) Graphic symbol

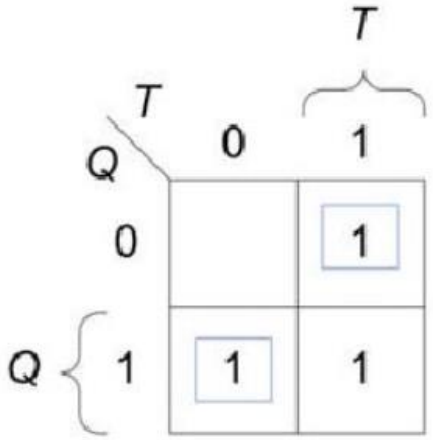
T Flip-Flop Timing Diagram



T	Q_{t+1}
0	Q_t
1	$\overline{Q}_t$

Characteristic Equation of the T-Flip Flop

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



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

T	Q_{t+1}
0	Q_t
1	$\overline{Q}_t$

Flip-Flop Characteristic Table

- Defines 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 edge arrives

  $Q(t+1)$ refers to next state **after** the clock edge arrives

D Flip-Flop	
D	$Q(t+1)$
0	0 Reset
1	1 Set

JK Flip-Flop	
J K	$Q(t+1)$
0 0	Q(t) No change
0 1	0 Reset
1 0	1 Set
1 1	Q'(t) Complement

T Flip-Flop	
T	$Q(t+1)$
0	Q(t) No change
1	Q'(t) Complement

Flip-Flop Characteristic Equation

- The characteristic equation defines the operation of a flip-flop
- For D Flip-Flop: $Q(t+1) = D$
- For JK Flip-Flop: $Q(t+1) = J\,Q'(t) + K'\,Q(t)$
- For T Flip-Flop: $Q(t+1) = T \oplus Q(t)$
- Clearly, the D Flip-Flop is the simplest among the three

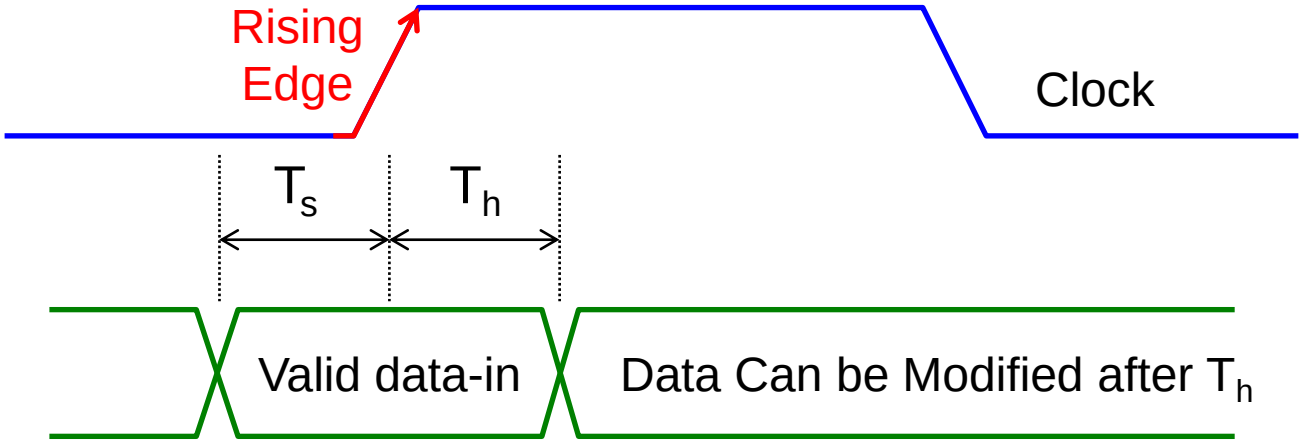
D Flip-Flop	
D	$Q(t+1)$
0	0 Reset
1	1 Set

JK Flip-Flop	
J K	$Q(t+1)$
0 0	Q(t) No change
0 1	0 Reset
1 0	1 Set
1 1	Q'(t) Complement

T Flip-Flop	
T	$Q(t+1)$
0	Q(t) No change
1	Q'(t) Complement

Timing Considerations for Flip-Flops

- **Setup Time (T_s):** Time duration for which the data input must be valid and stable before the arrival of the clock edge.
- **Hold Time (T_h):** Time duration for which the data input must not be changed after the clock transition occurs.
- T_s and T_h must be ensured for the proper operation of flip-flops



Summary

- In a sequential circuit there is internal memory
  - Output is a function of current inputs and present state
  - The stored memory value defines the present state
  - Similarly, the next state depends on current inputs and present state
- Two types of sequential circuits:
  - Synchronous sequential circuits are clocked (easier to implement)
  - Asynchronous sequential circuits are not clocked
- Two types of Memory elements: Latches and Flip-Flops
- Latches are level-sensitive, flip-flops are edge-triggered
- Flip-flops are better memory elements for synchronous circuits
- A flip-flop is described using a characteristic table and equation

Next . . .

- Introduction to Sequential Circuits
  - Combinational versus Sequential Circuits
  - Synchronous versus Asynchronous Sequential Circuits
- Storage Elements
  - Latches
  - Flip-Flops
- **Analysis of Clocked Sequential circuits**
  - **State and Output Equations**
  - **State Table**
  - **State Diagram**
- Mealy versus Moore Sequential Circuits

Analysis of Clocked Sequential Circuits

Analysis is describing what a given circuit will do

The output of a clocked sequential circuit is determined by

1. Inputs
2. State of the Flip-Flops

Analysis Procedure:

1. Obtain the equations at the inputs of the Flip-Flops
2. Obtain the next state and the output equations
3. Fill the state table for all possible input and state values
4. Draw the state diagram

Analysis Example

- Is this a clocked sequential circuit?

  YES!

- What type of Memory?

  D Flip-Flops

- How many state variables?

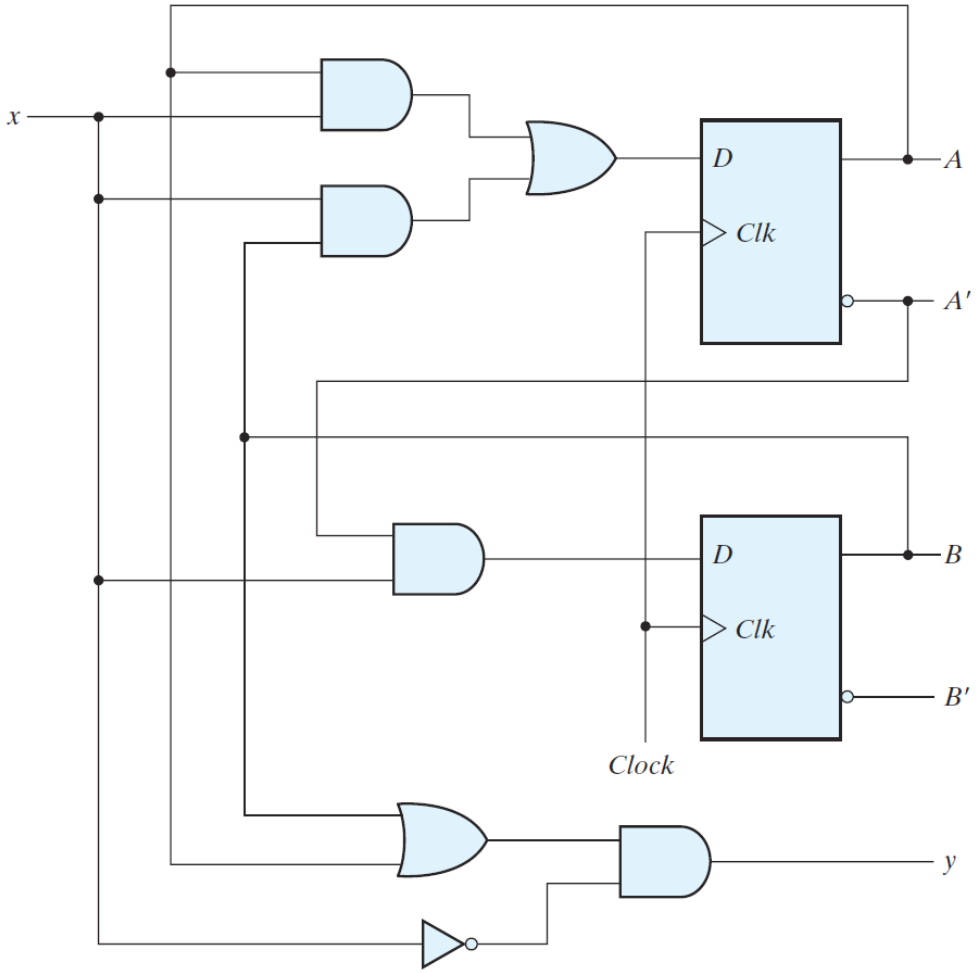
  Two state variables: A and B

- What are the Inputs?

  One Input: x

- What are the Outputs?

  One Output: y



Flip-Flop Input Equations

- What are the equations on the D inputs of the flip-flops?

  $D_A = A\,x + B\,x$

  $D_B = A'\,x$

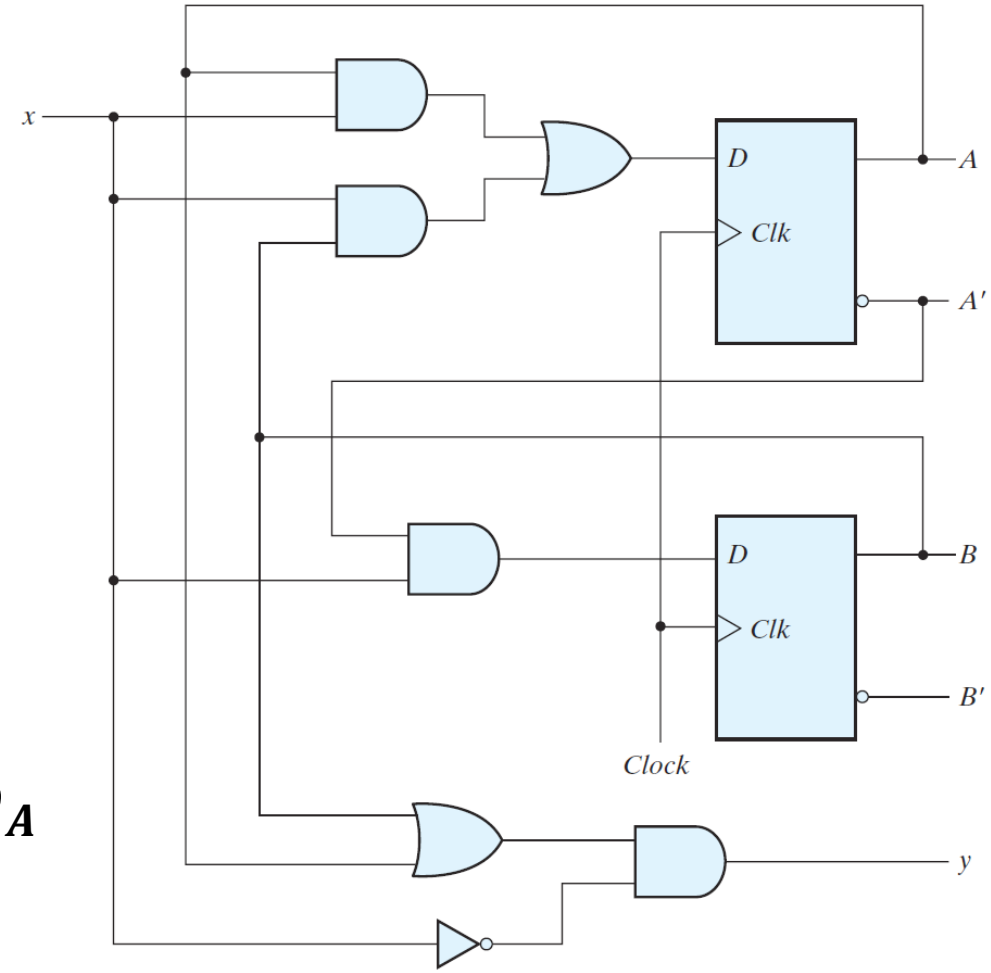
- A and B are the **current state**

  $A(t) = A, \quad B(t) = B$

- D_A and D_B are the **next state**

  $A(t+1) = D_A, \quad B(t+1) = D_B$

- The values of A and B will be D_A and D_B at the next clock edge



Next State and Output Equations

- The next state equations define the **next state**

  At the **inputs** of the Flip-Flops

- Next state equations?

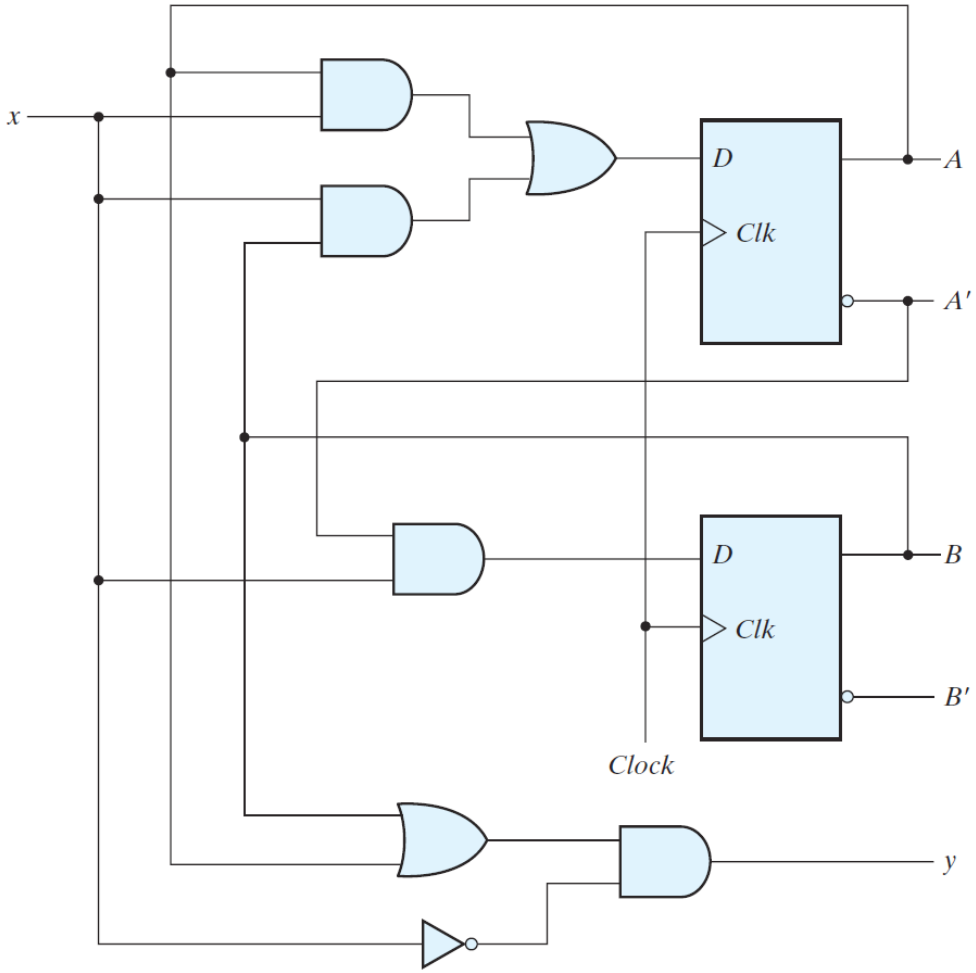
$$A(t+1) = D_A = A\,x + B\,x$$

$$B(t+1) = D_B = A'\,x$$

- There is only one output y

- What is the output equation?

$$y = (A + B)\,x'$$



State Table

- State table shows the Next State and Output in a tabular form
- Next State Equations: $A(t+1) = A\,x + B\,x$ and $B(t+1) = A'\,x$
- Output Equation: $y = (A + B)\,x'$

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

Another form of the state table

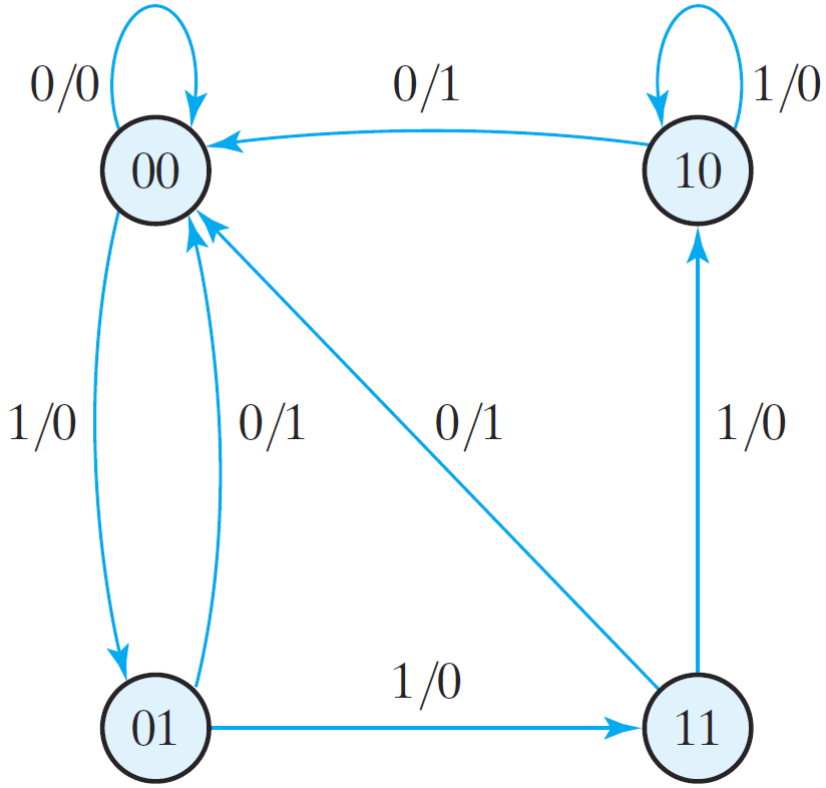
Present State		Next State				Output	
		x = 0		x = 1		x = 0	x = 1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

State Diagram

- State diagram is a graphical representation of a state table
- The circles are the states
- Two state variable ➔ Four states (ALL values of A and B)
- Arcs are the state transitions

  Labeled with: Input x / Output y

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



Combinational versus Sequential Analysis

Analysis of Combinational Circuits

- Obtain the Boolean Equations
- Fill the Truth Table

Output is a function of input only

Analysis of Sequential Circuits

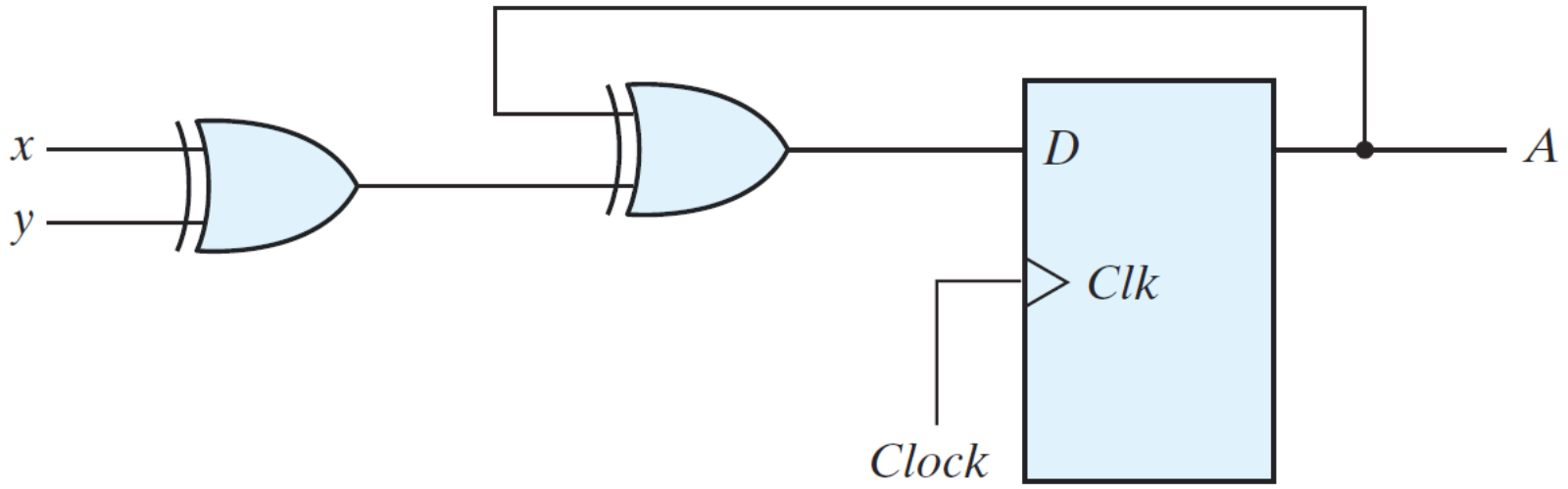
- Obtain the Next State Equations
- Obtain the Output Equations
- Fill the State Table
- Draw the State Diagram

Next state is a function of input and current state

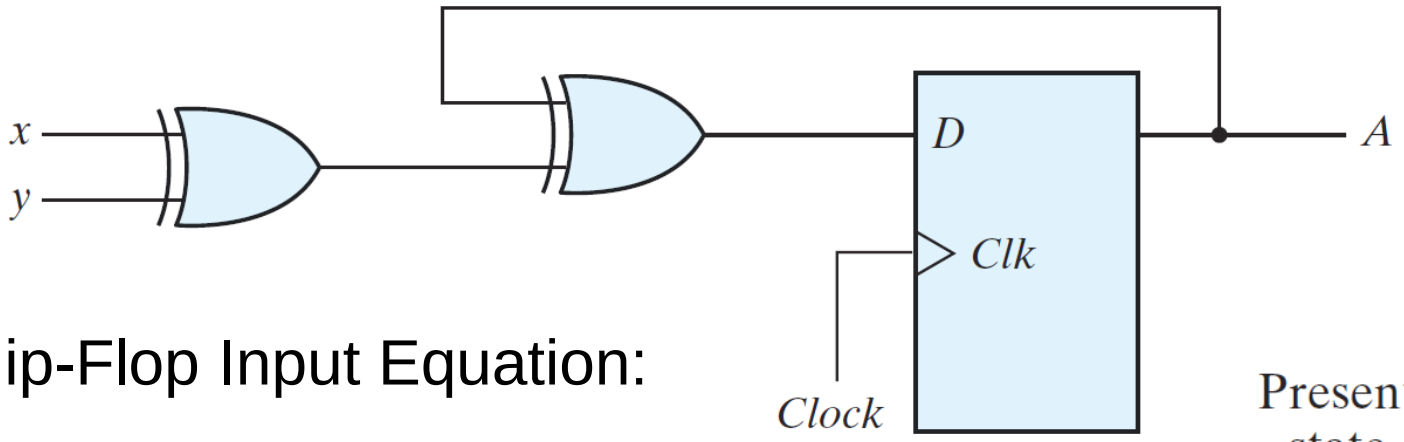
Output is a function of input and current state

Example with Output = Current State

- Analyze the sequential circuit shown below
- Two inputs: x and y
- One state variable A
- No separate output ➔ Output = current state A
- Obtain the next state equation, state table, and state diagram



Example with Output = Current State



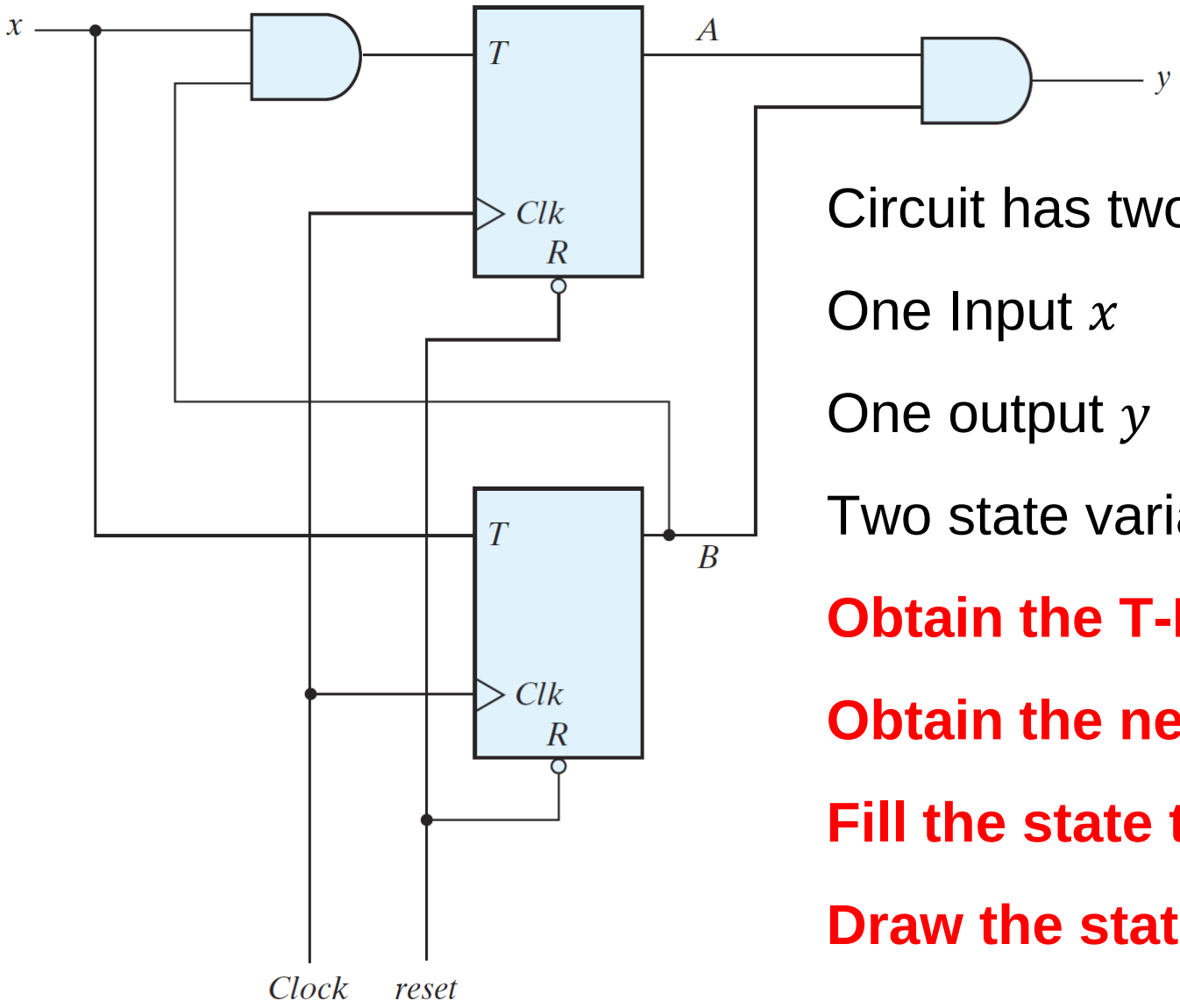
- Flip-Flop Input Equation:

$$D_A = A \oplus x \oplus y$$

- Next State Equation: $A(t+1) = A \oplus x \oplus y$

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

Sequential Circuit with T Flip-Flops



Circuit has two T Flip-Flops

One Input x

One output y

Two state variables: A and B

Obtain the T-FF input equations

Obtain the next state equations

Fill the state table

Draw the state diagram

Recall: Flip-Flop Characteristic Equation

- For D Flip-Flop: $Q(t+1) = D$
- For T Flip-Flop: $Q(t+1) = T \oplus Q(t)$
- For JK Flip-Flop: $Q(t+1) = J\,Q'(t) + K'\,Q(t)$

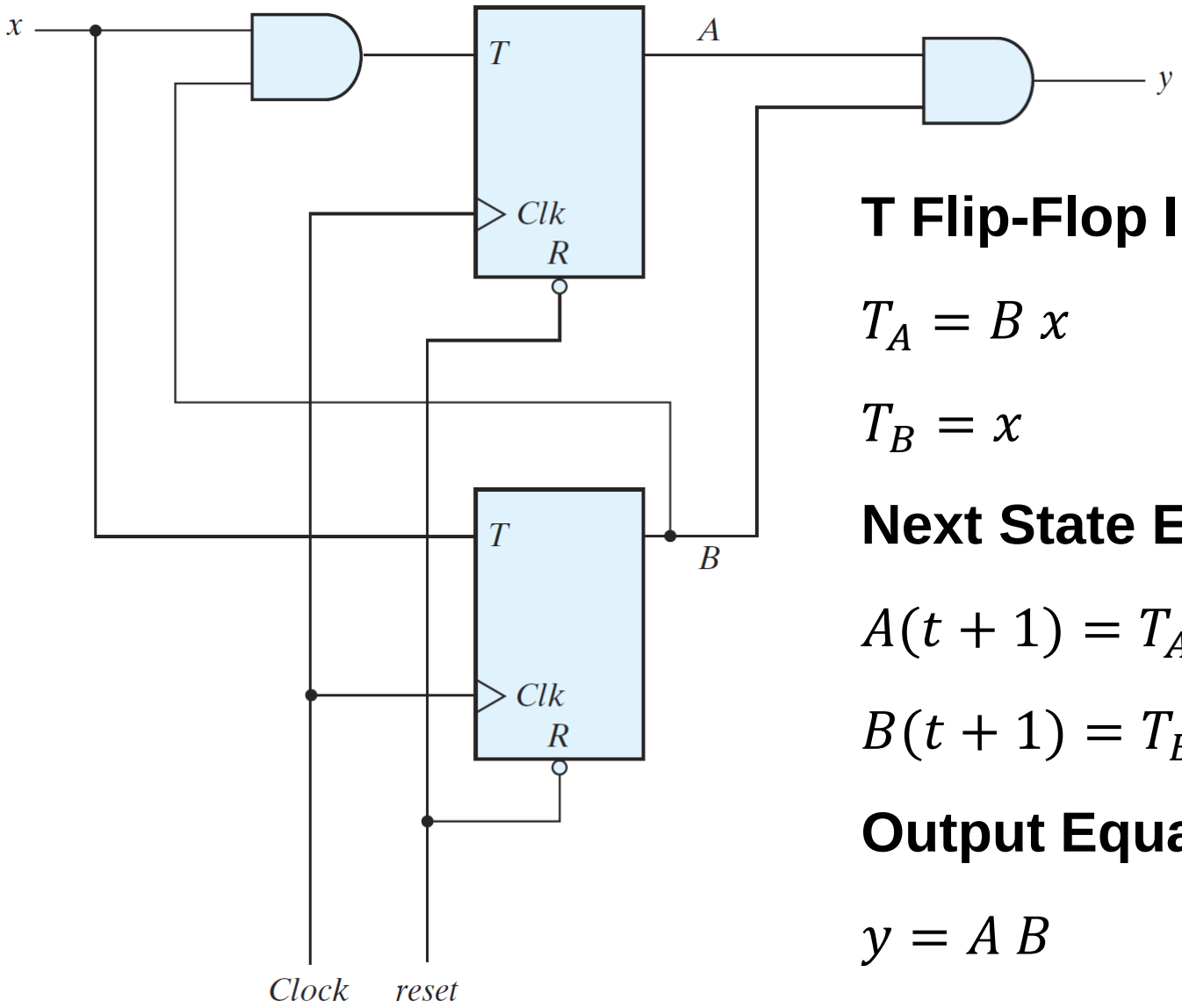
These equations define the Next State

D Flip-Flop		
D	$Q(t+1)$	
0	0	Reset
1	1	Set

T Flip-Flop		
T	$Q(t+1)$	
0	Q(t)	No change
1	Q'(t)	Complement

JK Flip-Flop		
J K	$Q(t+1)$	
0 0	Q(t)	No change
0 1	0	Reset
1 0	1	Set
1 1	Q'(t)	Complement

Sequential Circuit with T Flip-Flops



T Flip-Flop Input Equations:

$$T_A = B\,x$$

$$T_B = x$$

Next State Equations:

$$A(t+1) = T_A \oplus A = (B\,x) \oplus A$$

$$B(t+1) = T_B \oplus B = x \oplus B$$

Output Equation:

$$y = A\,B$$

From Next State Equations to State Table

T Flip-Flop Input Equations:

$T_A = B\,x$

$T_B = x$

Next State Equations:

$A(t+1) = (B\,x) \oplus A$

$B(t+1) = x \oplus B$

Output Equation:

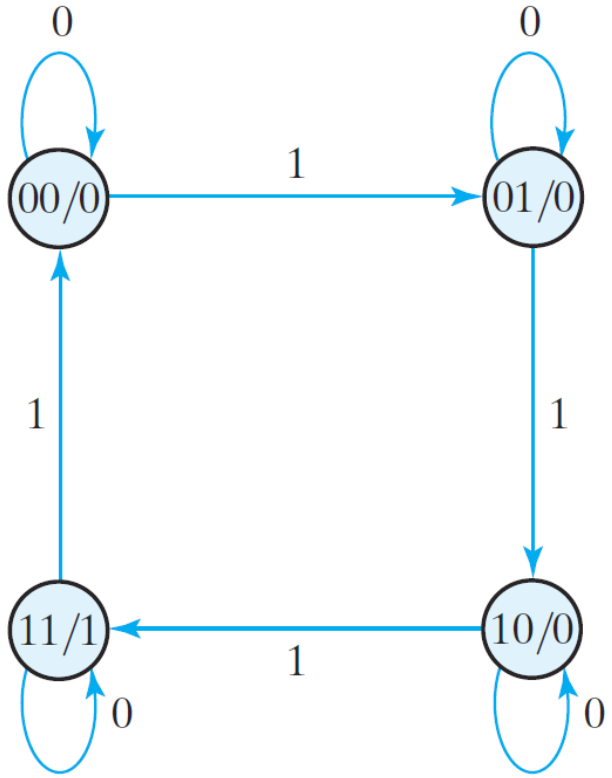
$y = A\,B$

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

Notice that the output is a function of the present state only.
It does **NOT** depend on the input x

From State Table to State Diagram

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



- Four States: $AB = 00,\ 01,\ 10,\ 11$ (drawn as circles)
- Output Equation: $y = A\,B$ (does not depend on input x)
- Output y is shown inside the state circle (AB/y)

Sequential Circuit with a JK Flip-Flops

One Input x and two state variables: A and B (outputs of Flip-Flops)

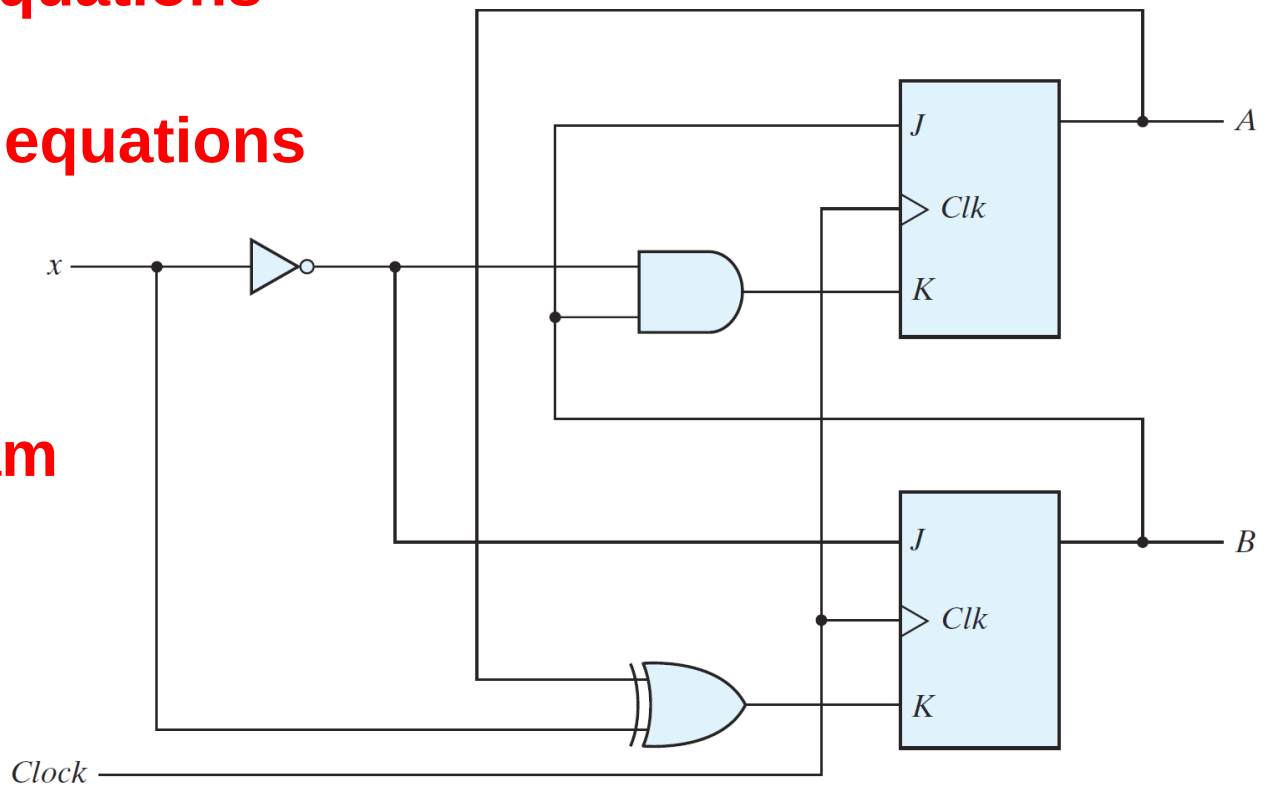
No separate output ➔ Output = Current state $A\ B$

Obtain the JK input equations

Obtain the next state equations

Fill the state table

Draw the state diagram

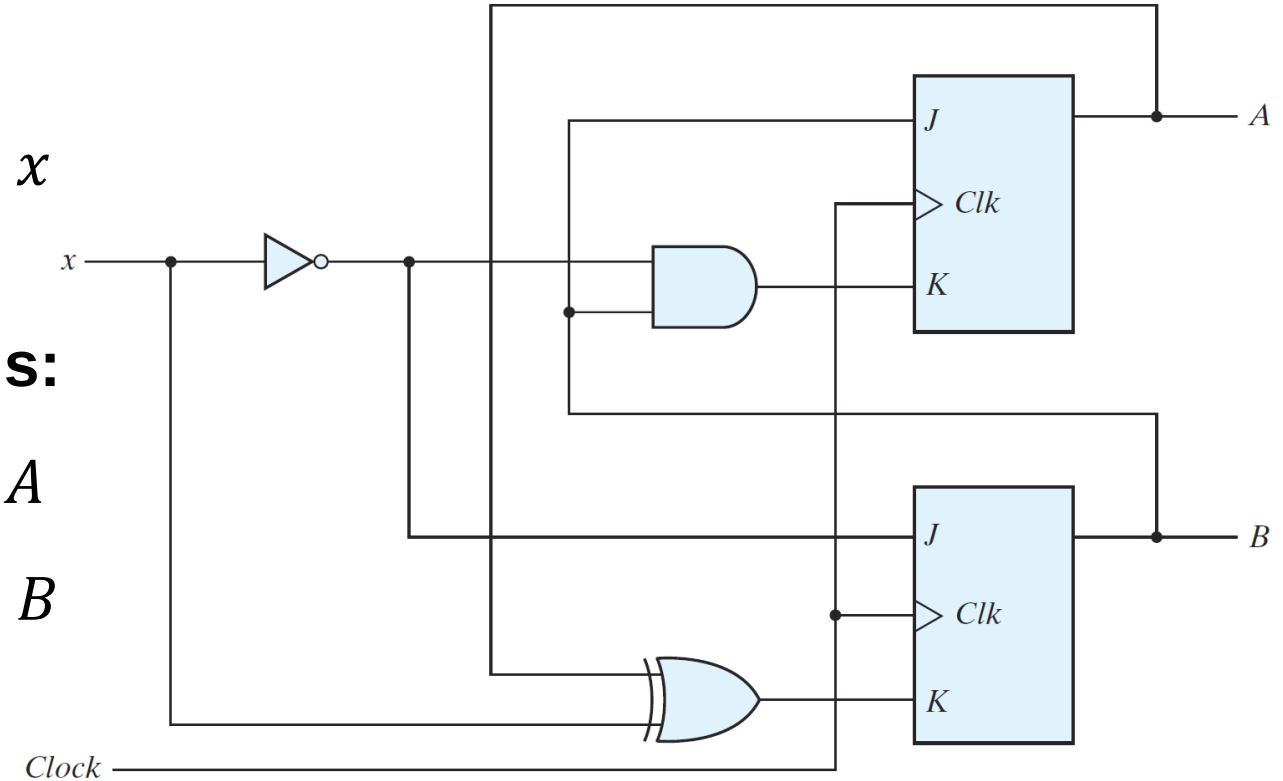


JK Input and Next State Equations

JK Flip-Flop Input Equations:

$J_A = B$ and $K_A = B\,x'$

$J_B = x'$ and $K_B = A \oplus x$



Next State Equations:

$A(t+1) = J_A\,A' + K_A'\,A$

$B(t+1) = J_B\,B' + K_B'\,B$

Substituting:

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

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

From JK Input Equations to State Table

JK Input Equations: $J_A = B$, $K_A = B\,x'$, $J_B = x'$ and $K_B = A \oplus x$

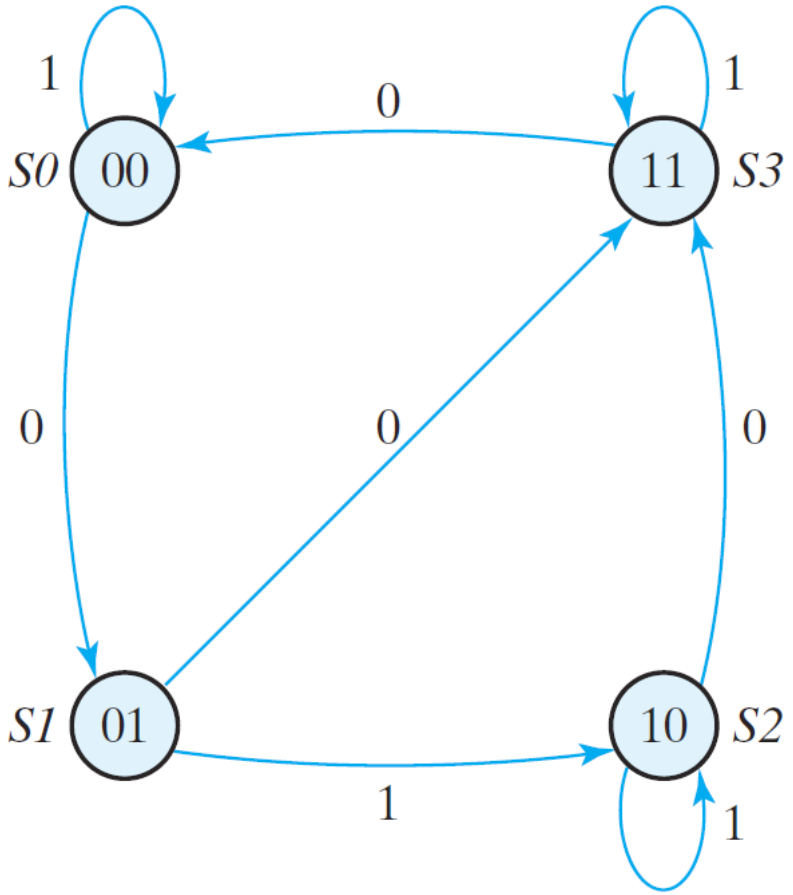
Present State		Input	Next State		Flip-Flop Inputs			
A	**B**	**x**	**A**	**B**	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

From State Table to State Diagram

Four states: $A\ B = 00, 01, 10, and\ 11$ (drawn as circles)

Arcs show the input value x on the state transition

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



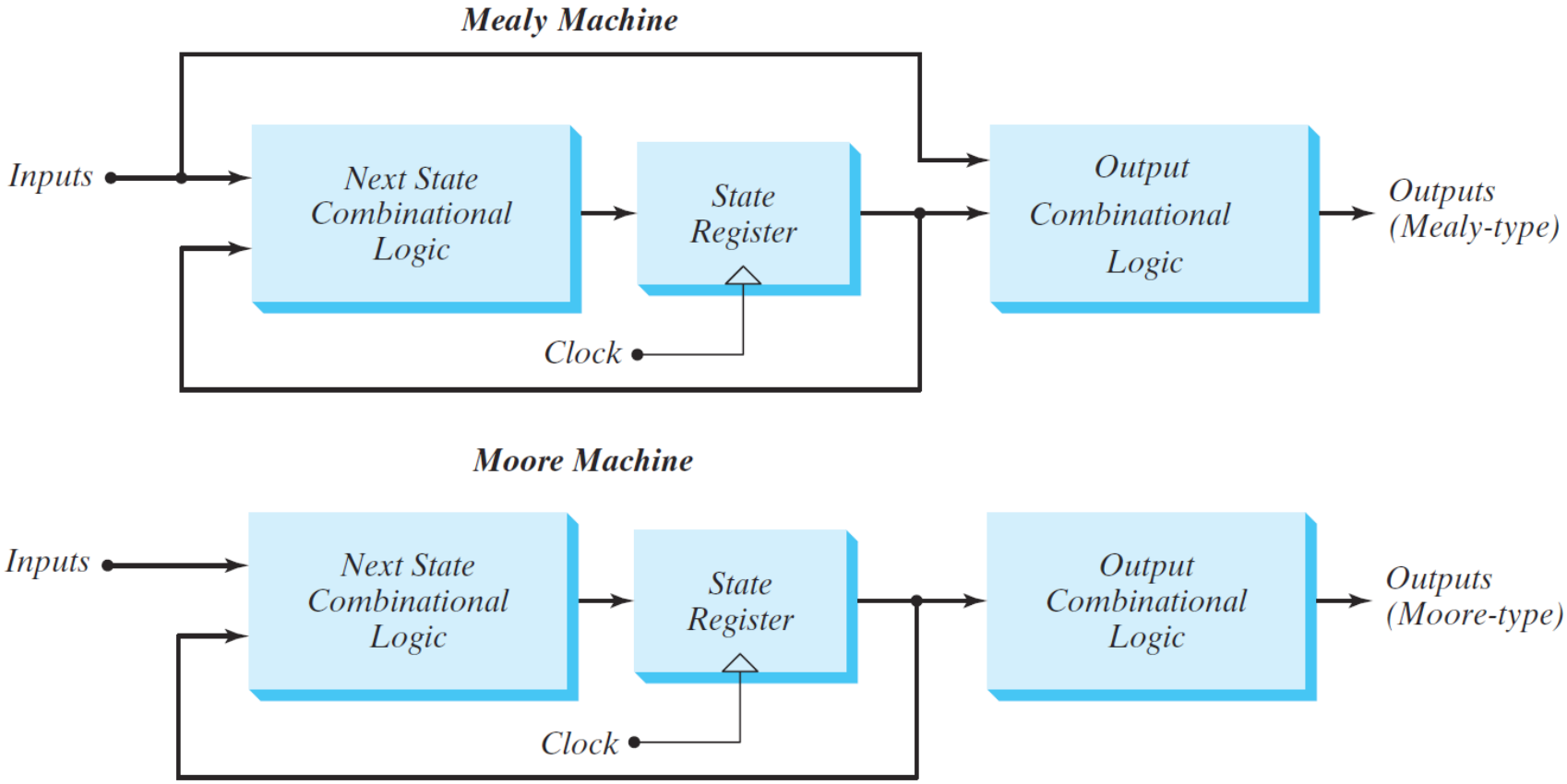
Next . . .

- Introduction to Sequential Circuits
  - Combinational versus Sequential Circuits
  - Synchronous versus Asynchronous Sequential Circuits
- Storage Elements
  - Latches
  - Flip-Flops
- Analysis of Clocked Sequential circuits
  - State and Output Equations
  - State Table
  - State Diagram
- **Mealy versus Moore Sequential Circuits**

Mealy versus Moore Sequential Circuits

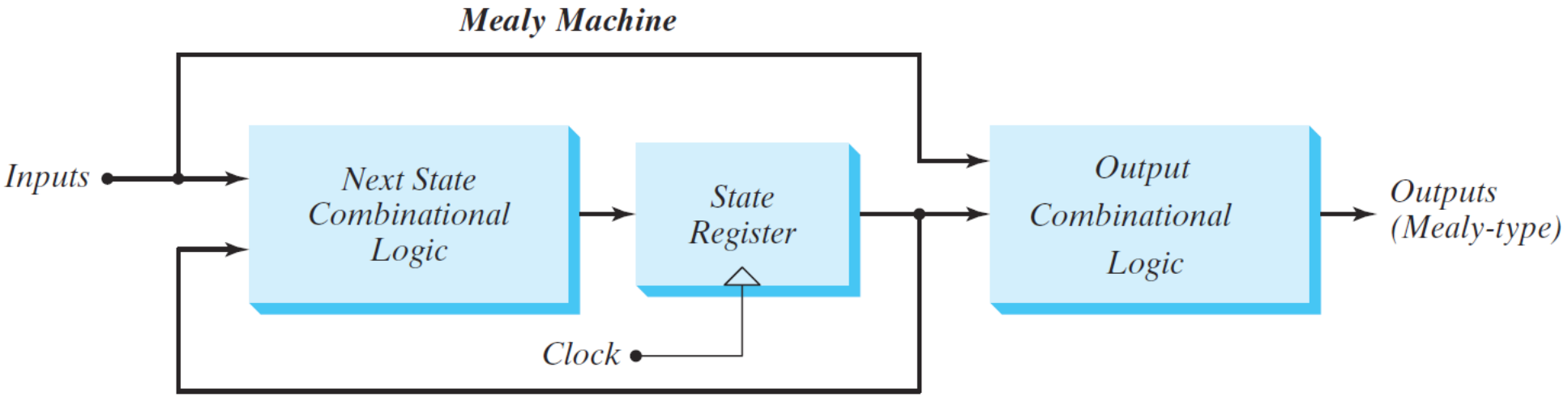
There are two ways to design a clocked sequential circuit:

1. **Mealy Machine:** Outputs depend on present state and inputs
2. **Moore Machine:** Outputs depend on present state only

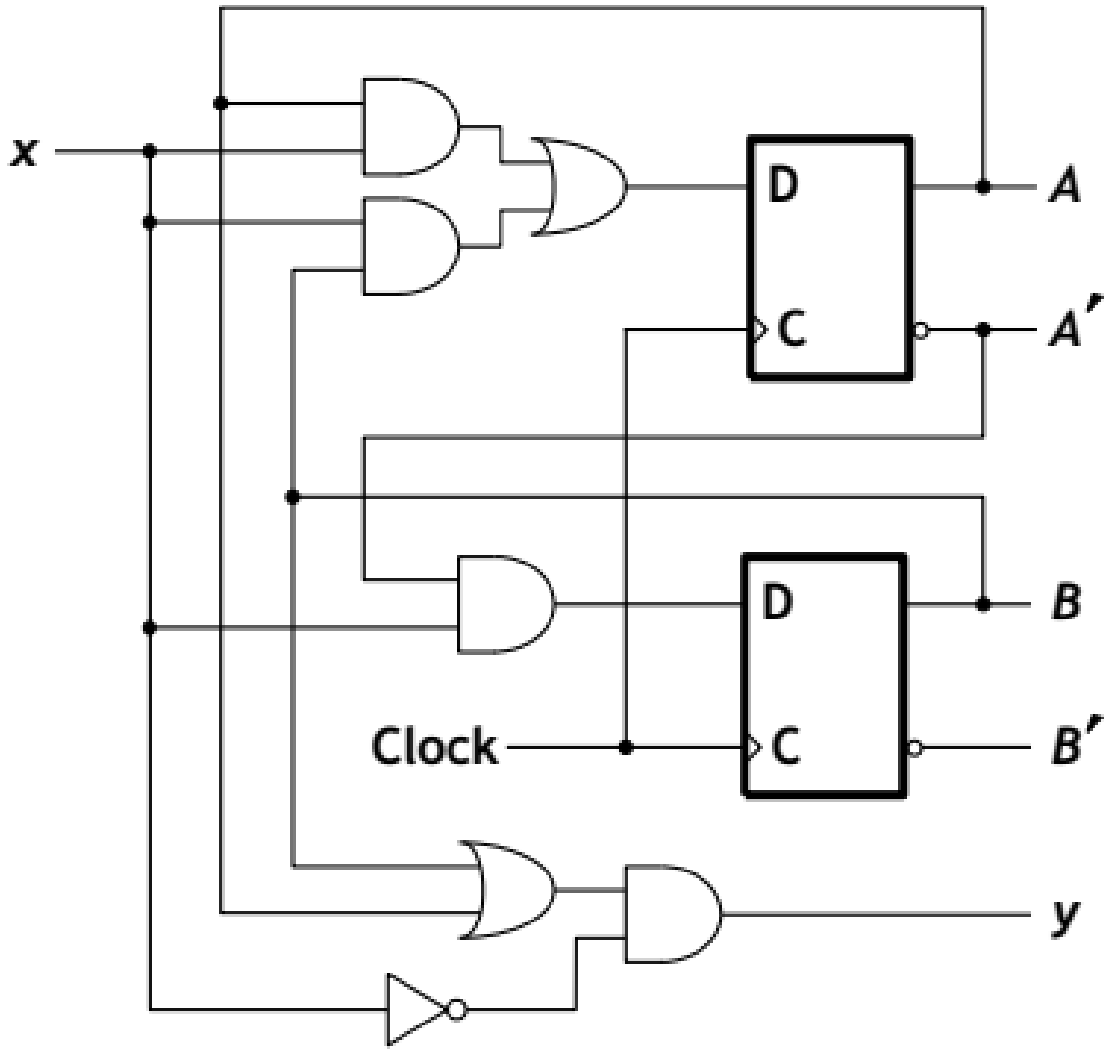


Mealy Machine

- The outputs are a function of the present state and Inputs
- The outputs are **NOT** synchronized with the clock
- The outputs may change if inputs change during the clock cycle
- The outputs may have momentary false values (called glitches)
- The correct outputs are present just before the edge of the clock

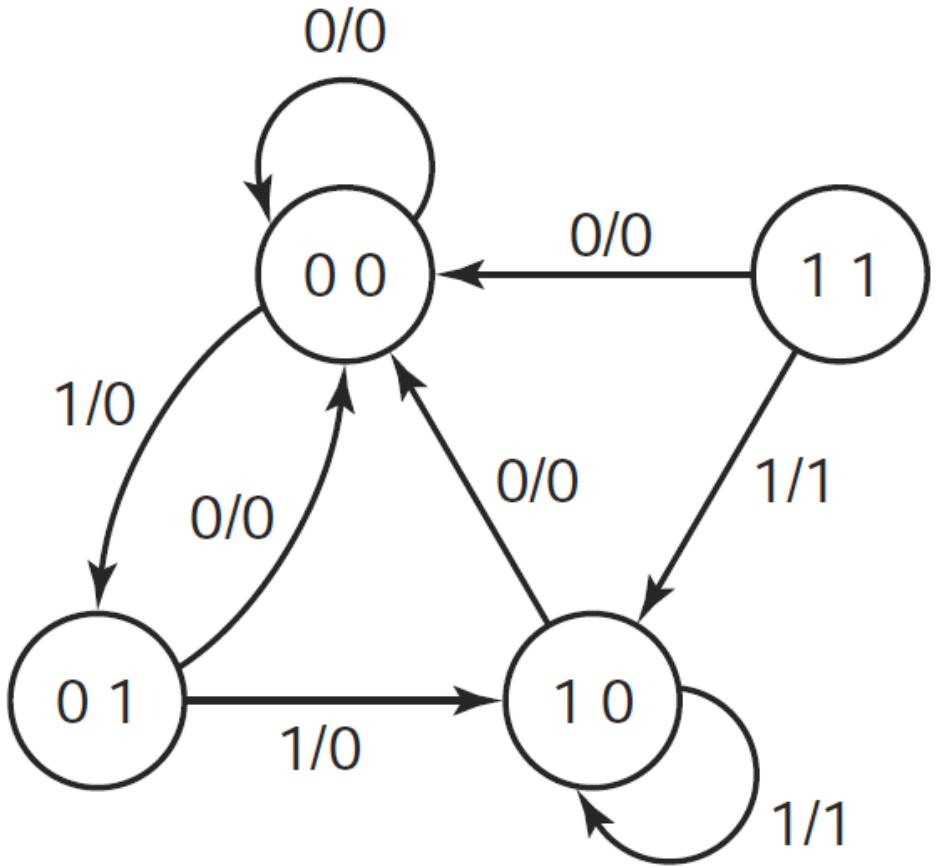


Example of Mealy Model



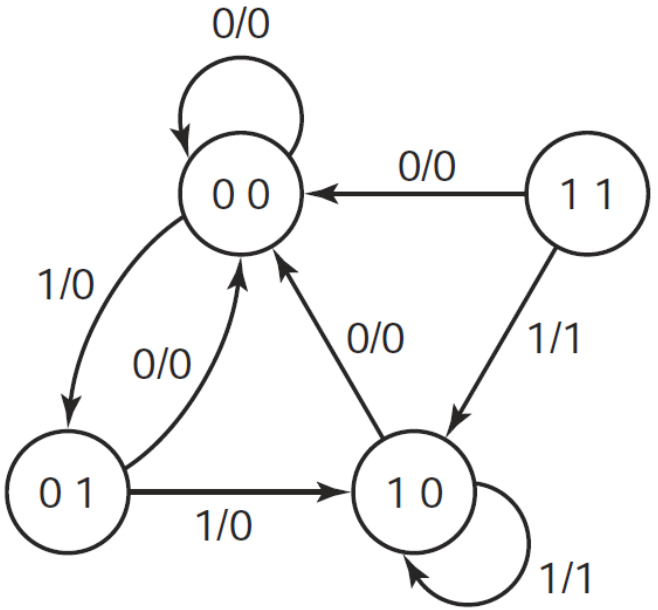
Mealy State Diagram

- An example of a Mealy state diagram is shown on the right
- Each arc is labeled with: **Input / Output**
- The output is shown on the arcs of the state diagram
- The output depends on the current state and input
- Notice that State 11 cannot be reached from the other states



Tracing a Mealy State Diagram

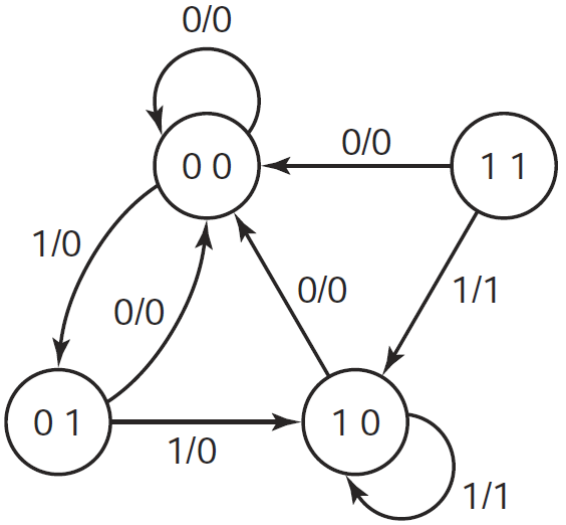
Cycle	0	1	2	3	4	5	6	7	8
Input *x*	0	1	1	0	1	1	1	1	0
Present State *A*	?	0	0	1	0	0	1	1	1
B	?	0	1	0	0	1	0	0	0
Output *z*	0	0	0	0	0	0	1	1	0

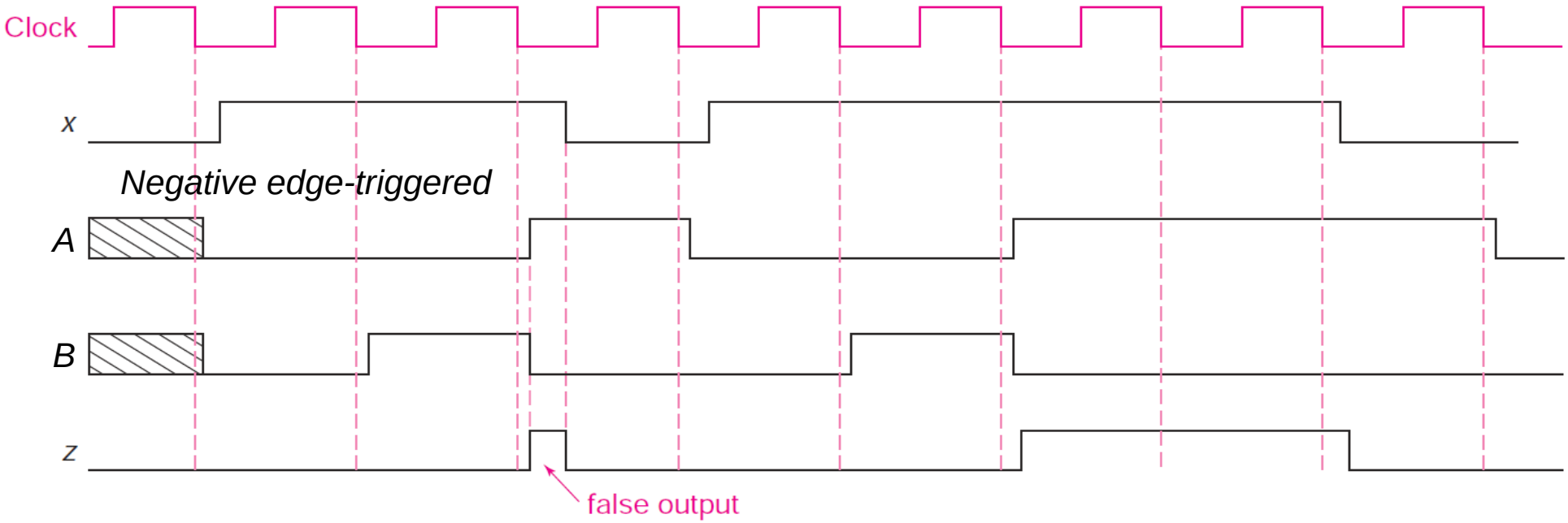


- When the circuit is powered, the initial state (*AB*) is unknown
- Even though the initial state is unknown, the input *x* = 0 forces a transition to state *AB* = 00, regardless of the present state
- Sometimes, a reset input is used to initialize the state to 00

False Output in the Timing Diagram

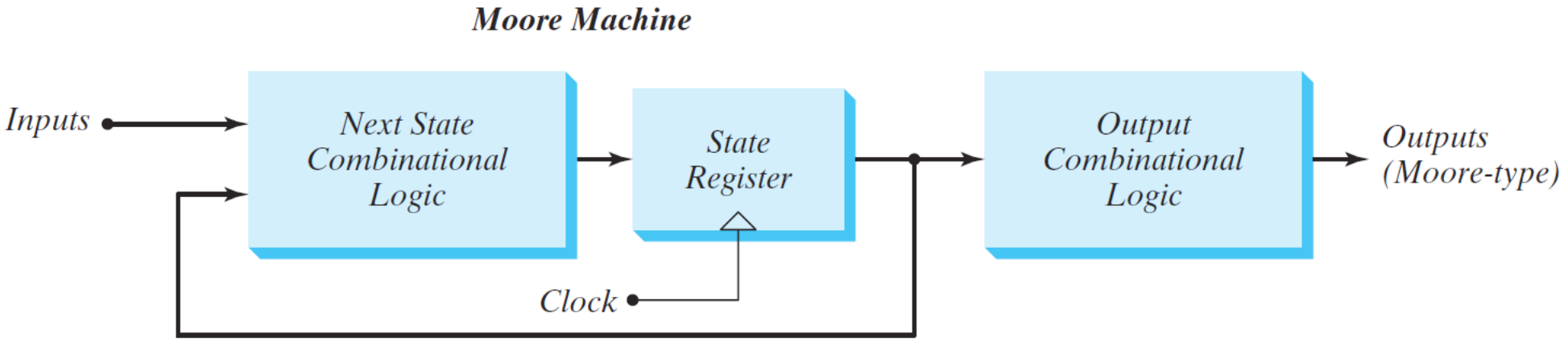
Cycle	0	1	2	3	4	5	6	7	8
Input *x*	0	1	1	0	1	1	1	1	0
Present State *A B*	?	0	0	1	0	0	1	1	1
	?	0	1	0	0	1	0	0	0
Output *z*	0	0	0	0	0	0	1	1	0





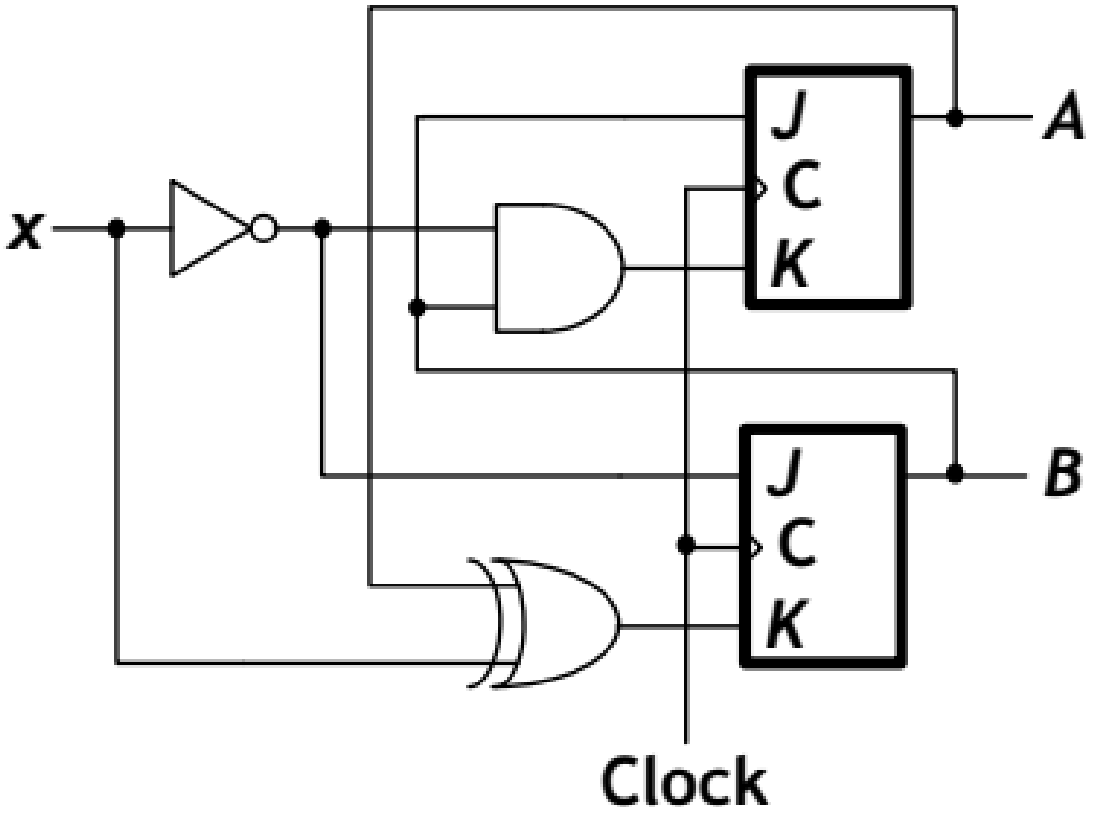
Moore Machine

- The outputs are a function of the Flip-Flop outputs only
- The outputs depend on the current state only
- The outputs are synchronized with the clock
- Glitches cannot appear in the outputs (even if inputs change)
- A given design might mix between Mealy and Moore



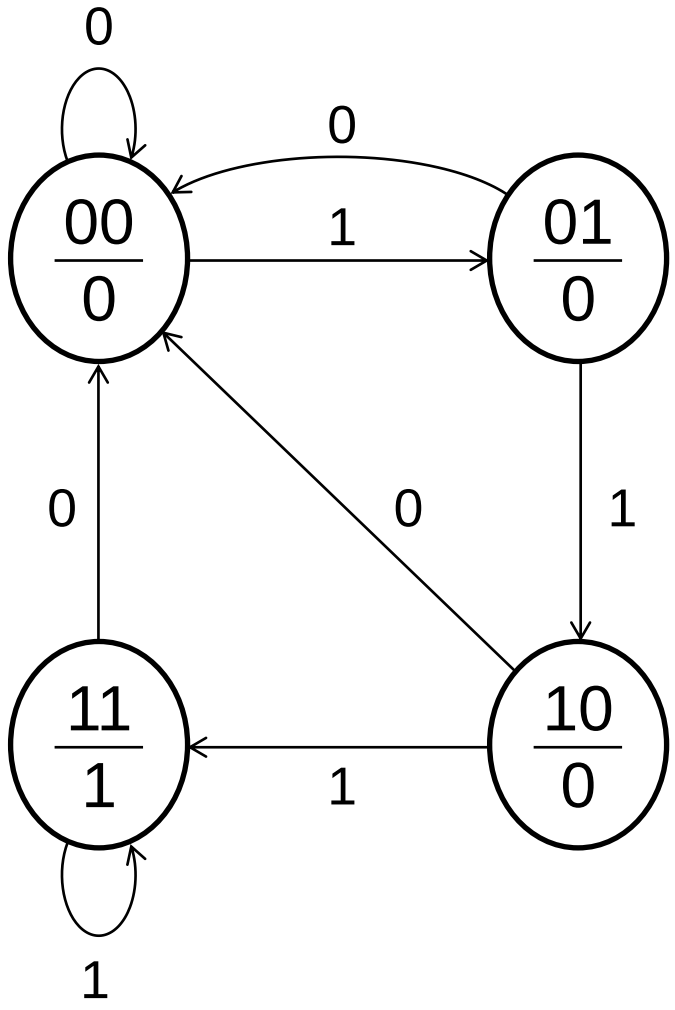
Example of Moore Model

Sequential Circuit with *JK* Flip-Flop



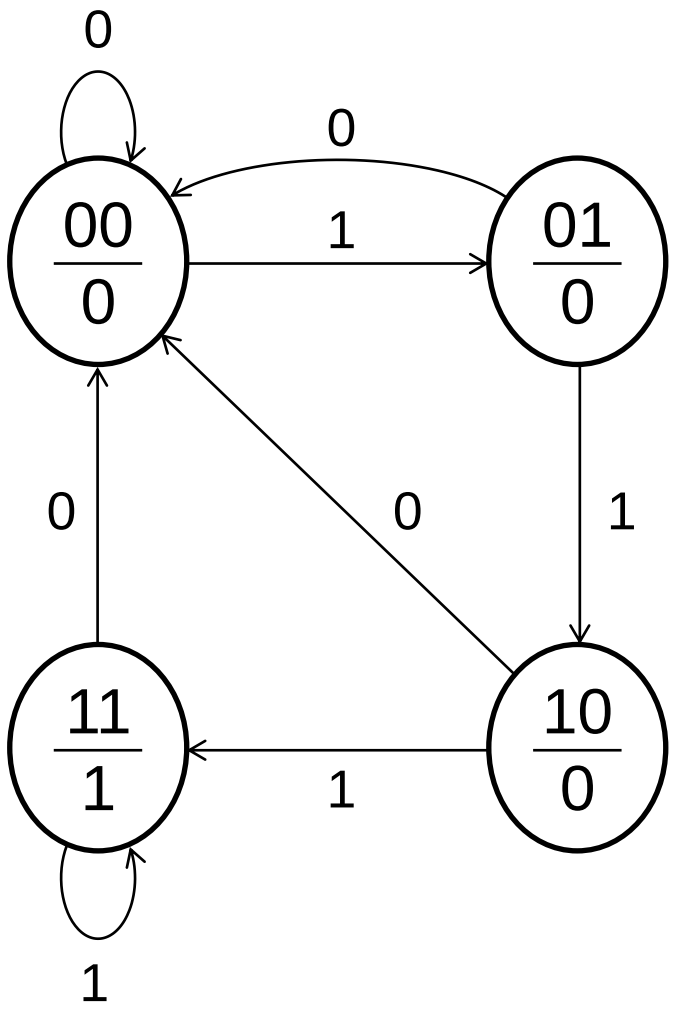
Moore State Diagram

- An example of a Moore state diagram is shown on the right
- Arcs are labeled with input only
- The output is shown inside the state: (State / Output)
- The output depends on the current state only



Tracing a Moore State Diagram

- When the circuit is powered, the initial state (*AB*) and output are unknown
- Input *x* = 0 resets the state *AB* to 00. Can also be done with a reset signal.

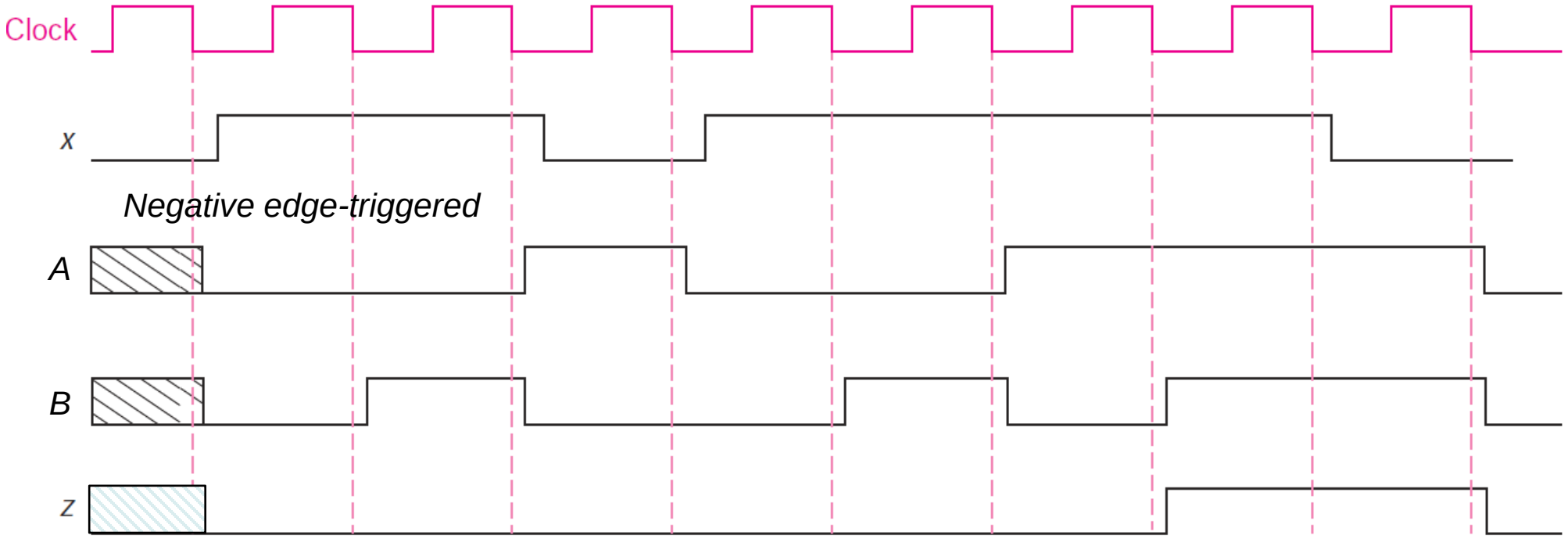


Cycle	0	1	2	3	4	5	6	7	8
Input *x*	0	1	1	0	1	1	1	1	0
Present State *A*	?	0	0	1	0	0	1	1	1
Present State *B*	?	0	1	0	0	1	0	1	1
Output *z*	?	0	0	0	0	0	0	1	1

Timing Diagram of a Moore Machine

Cycle	0	1	2	3	4	5	6	7	8
Input *x*	0	1	1	0	1	1	1	1	0
Present State *A*	?	0	0	1	0	0	1	1	1
B	?	0	1	0	0	1	0	1	1
Output *z*	?	0	0	0	0	0	0	1	1

The output is synchronized with the clock. No false output (or glitch) can appear.



Summary

- To analyze a clocked sequential circuit:

1. Obtain the equations at the **Inputs** of the flip-flops
2. Obtain the **Next State** equations
   - For a D Flip-Flop, the Next State = D input equation
   - For T and JK, use the characteristic equation of the Flip-Flop
3. Obtain the **Output** equations
4. Fill the **State Table**
   - Put all the combinations of current state and input
   - Fill the next state and output columns
5. Draw the **State Diagram**

- Two types of clocked sequential circuits: **Mealy** versus **Moore**