

# Sequential Circuit

Randy E. Saputra  
Prodi S1 Sistem Komputer  
Fakultas Teknik Elektro

Credits: David Harris  
Harvey Mudd College

(Some materials copied/taken/adapted from  
Harris' lecture notes)

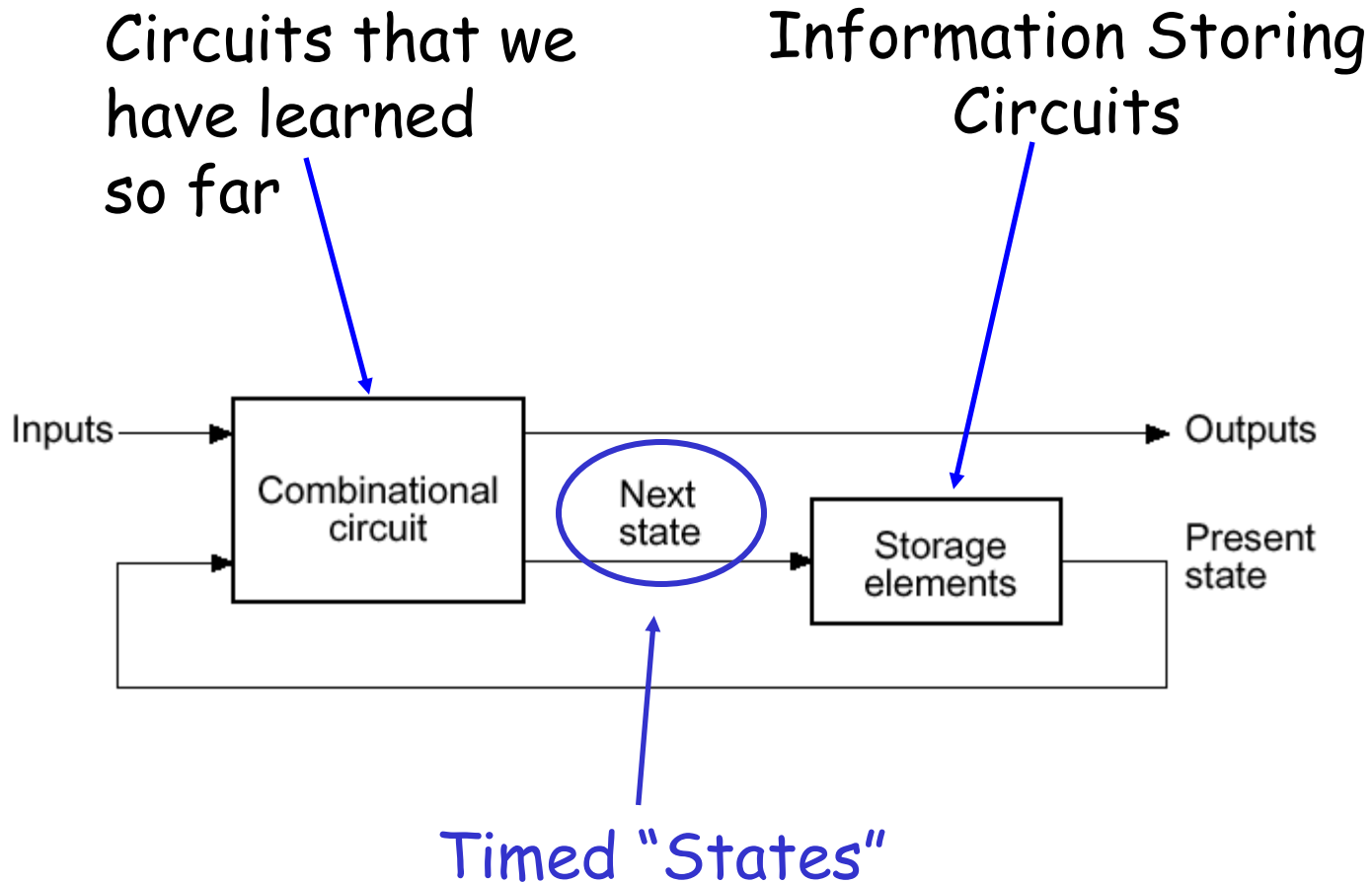
# Combinational Logic

- ❑ 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

# Sequential Circuits



# Sequential Logic: Concept

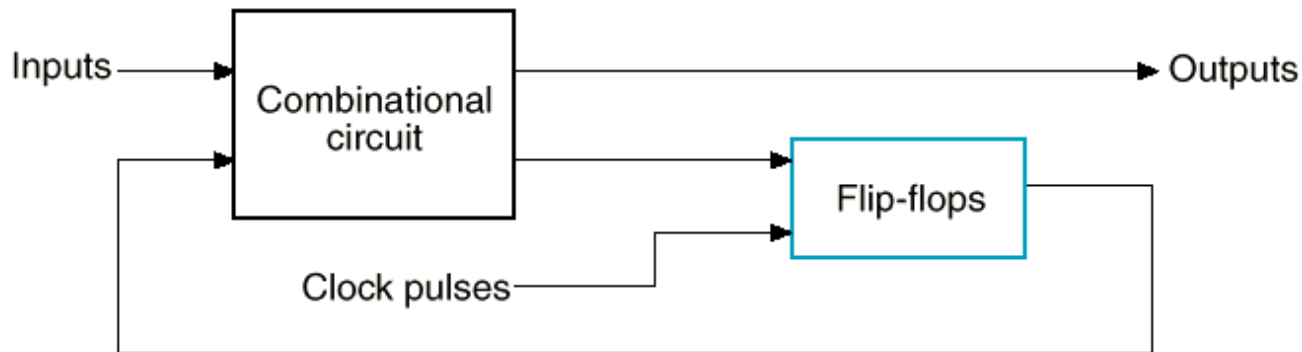
- ❑ Sequential Logic circuits remember past inputs and past circuit state.
- ❑ Outputs from the system are “fed back” as new inputs
  - With gate delay and wire delay
- ❑ The storage elements are circuits that are capable of storing binary information: memory.

# Synchronous vs. Asynchronous

There are two types of sequential circuits:

- ❑ **Synchronous** sequential circuit: circuit output changes only at some discrete instants of time. This type of circuits achieves synchronization by using a timing signal called the *clock*.
- ❑ **Asynchronous** sequential circuit: circuit output can change at **any** time (clockless).

# Flip flops as state memory



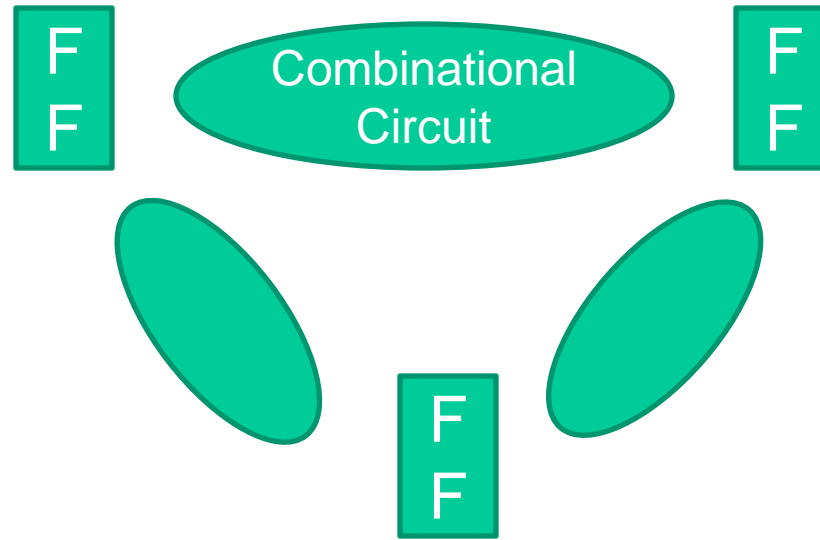
(a) Block diagram



(b) Timing diagram of clock pulses

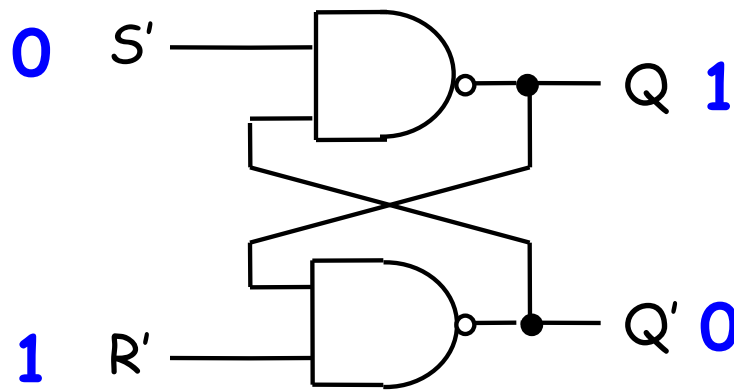
The flip-flops receive their inputs from the combinational circuit and also from a clock signal with pulses that occur at fixed intervals of time, as shown in the timing diagram.

# Clock Period



Smallest clock period = largest combinational circuit delay between any two directly connected FF, subjected to impact of FF setup time.

# SR Latch (NAND version)

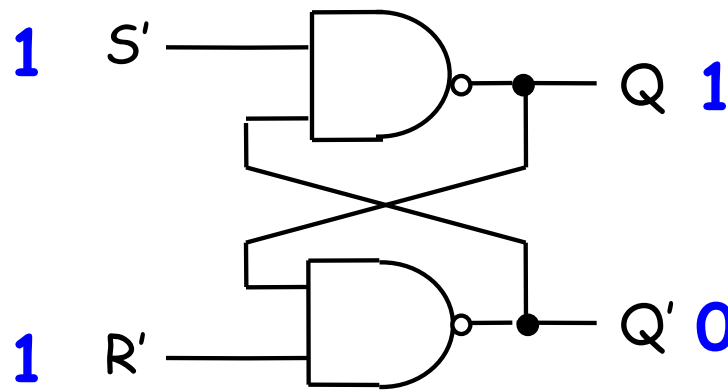


$S'$	$R'$	$Q$	$Q'$
0	0		
0	1	1	0
1	0		
1	1		

**Set**

$X$	$Y$	NAND
0	0	1
0	1	1
1	0	1
1	1	0

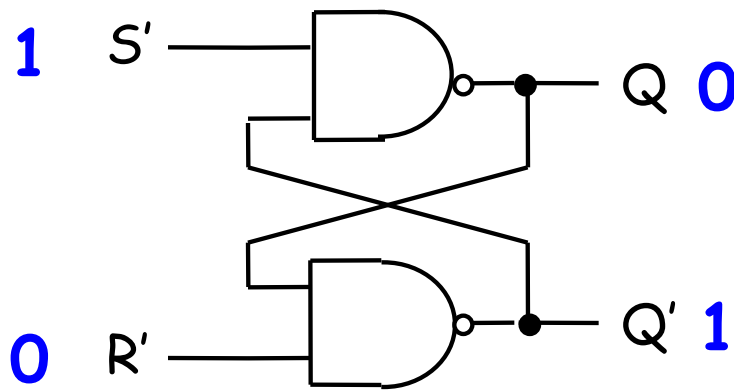
# SR Latch (NAND version)



$S'$	$R'$	$Q$	$Q'$	
0	0			
0	1	1	0	Set
1	0			
1	1	1	0	Hold

$X$	$Y$	NAND
0	0	1
0	1	1
1	0	1
1	1	0

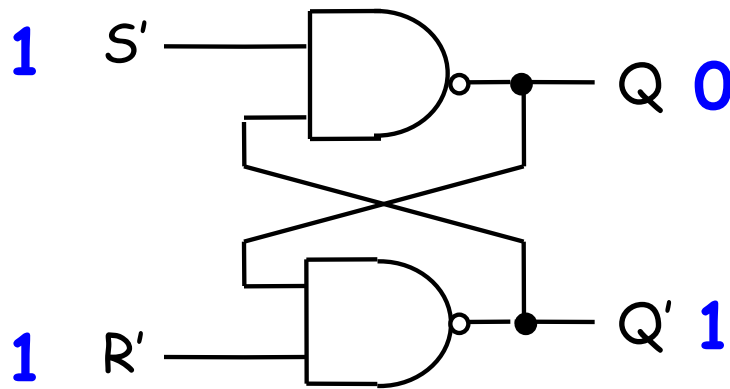
# SR Latch (NAND version)



$S'$	$R'$	$Q$	$Q'$	
0	0			
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold

$X$	$Y$	NAND
0	0	1
0	1	1
1	0	1
1	1	0

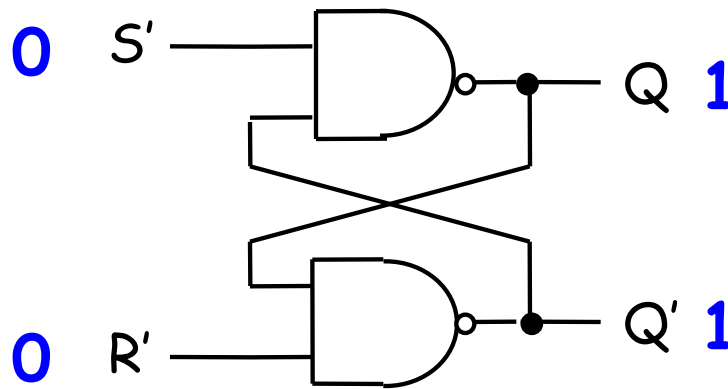
# SR Latch (NAND version)



$S'$	$R'$	$Q$	$Q'$	
0	0			
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold
		0	1	Hold

$X$	$Y$	NAND
0	0	1
0	1	1
1	0	1
1	1	0

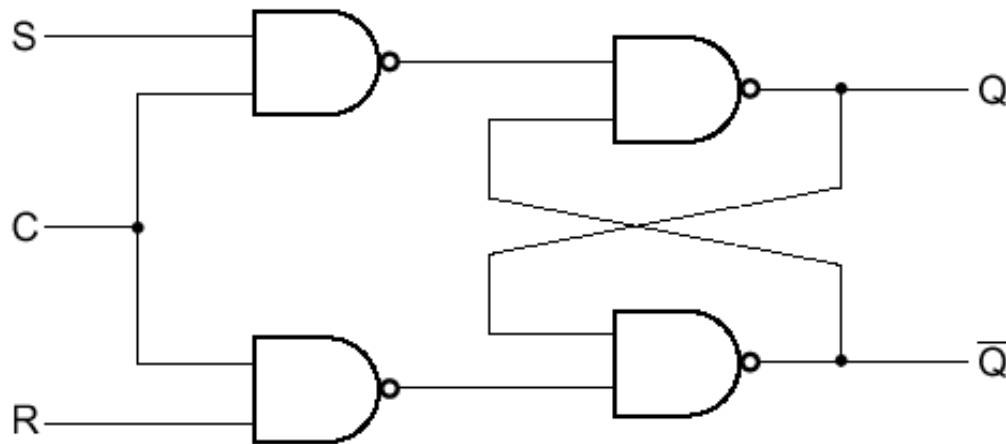
# SR Latch (NAND version)



$S'$	$R'$	$Q$	$Q'$	
0	0	1	1	Disallowed
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold
		0	1	Hold

$X$	$Y$	NAND
0	0	1
0	1	1
1	0	1
1	1	0

# SR Latch with Clock Signal



(a) Logic diagram

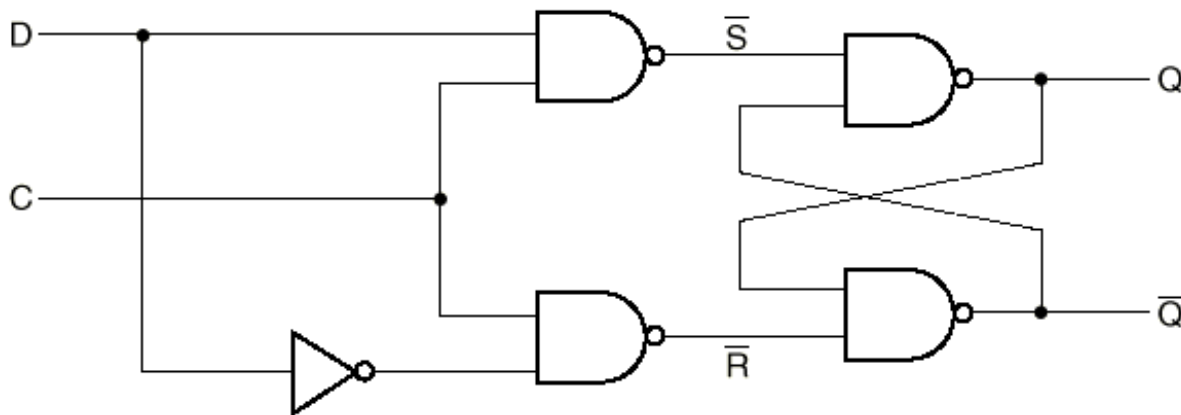
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

Latch is sensitive to input changes ONLY when C=1

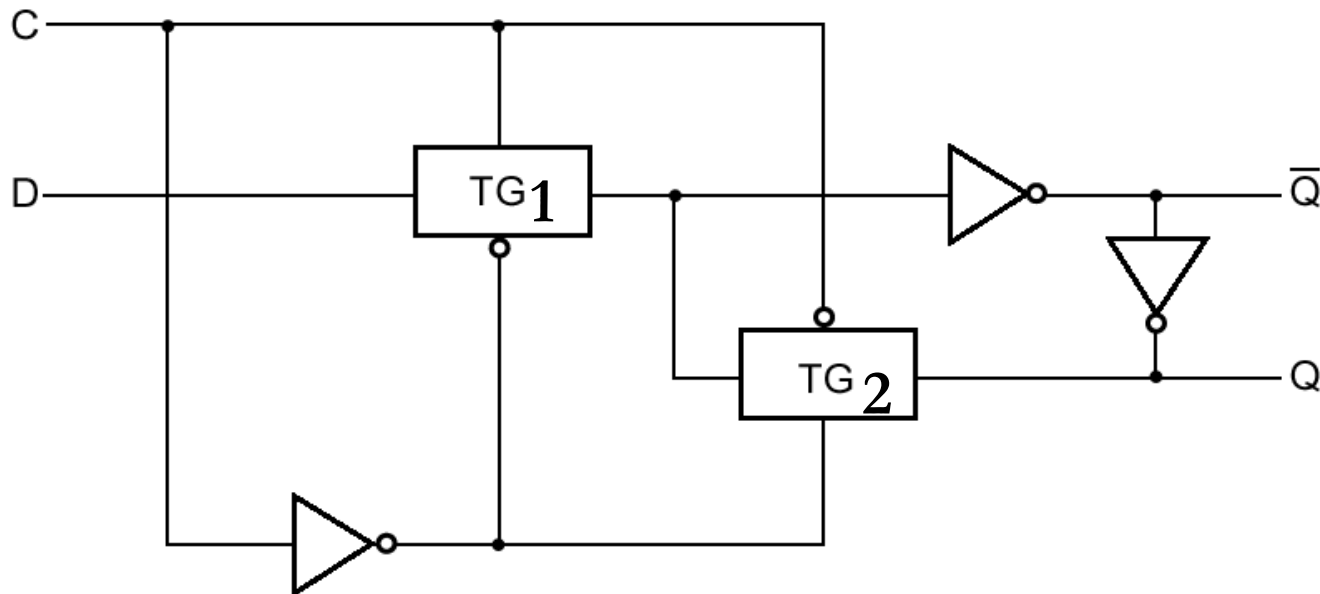
# D Latch

- One way to eliminate the undesirable indeterminate state in the RS flip flop is to ensure that inputs S and R are never 1 simultaneously. This is done in 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

# D Latch with Transmission Gate

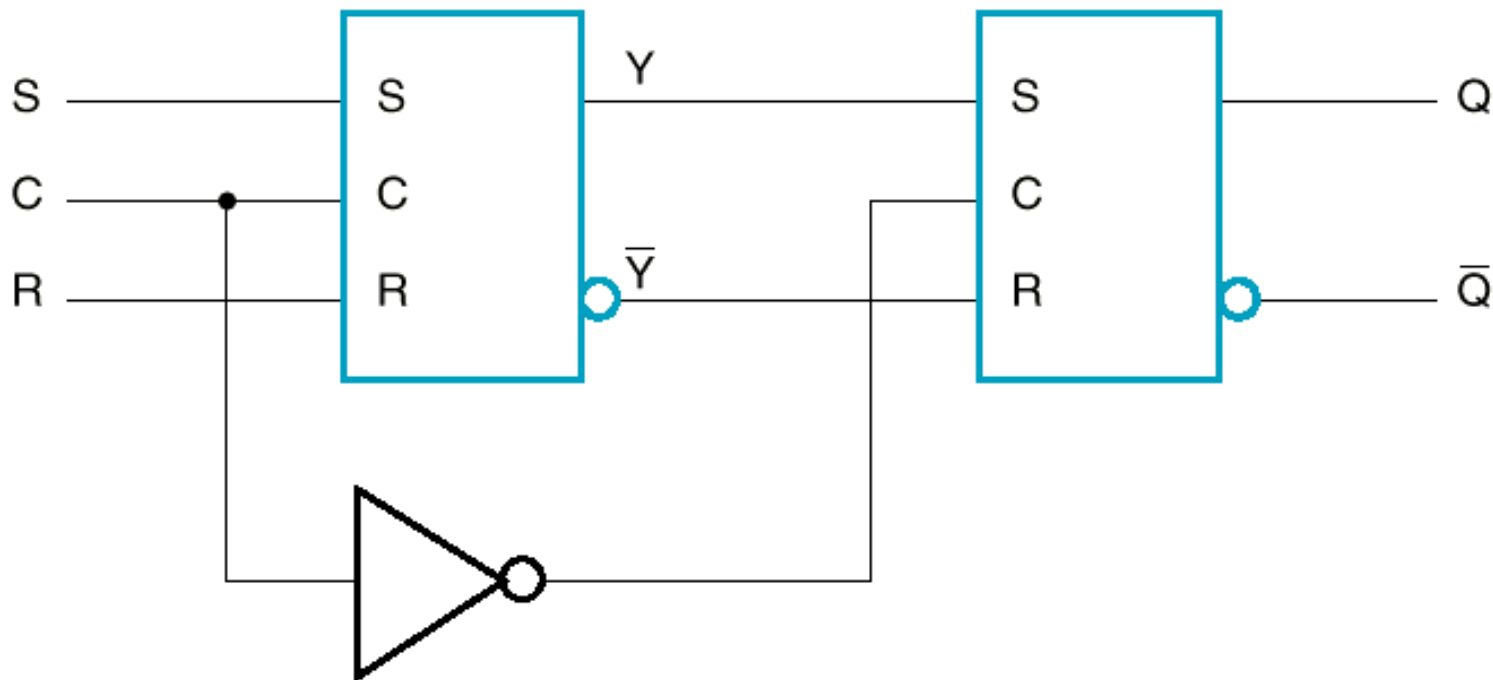


- $C=1 \rightarrow$  TG1 closes and TG2 opens  $\rightarrow Q'=D'$  and  $Q=D$
- $C=0 \rightarrow$  TG1 opens and TG2 closes  $\rightarrow$  Hold Q and  $Q'$

# Flip-Flop

- ❑ Latches are “transparent” (= any change on the inputs is seen at the outputs immediately when  $C=1$ ).
- ❑ This causes synchronization problems.
- ❑ Solution: use latches to create flip-flops that can respond (update) only on specific times (instead of any time).
- ❑ Types: RS flip-flop and D flip-flop

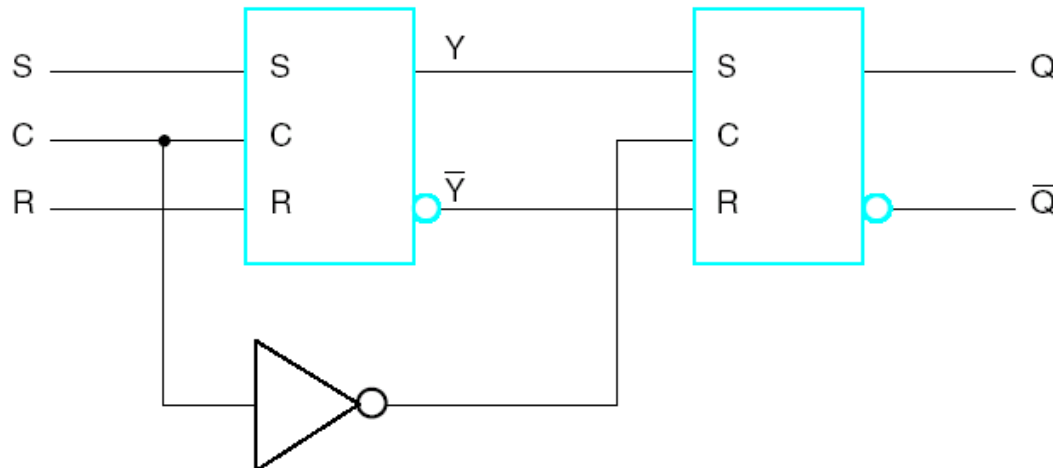
# Master-Slave FF configuration using SR latches



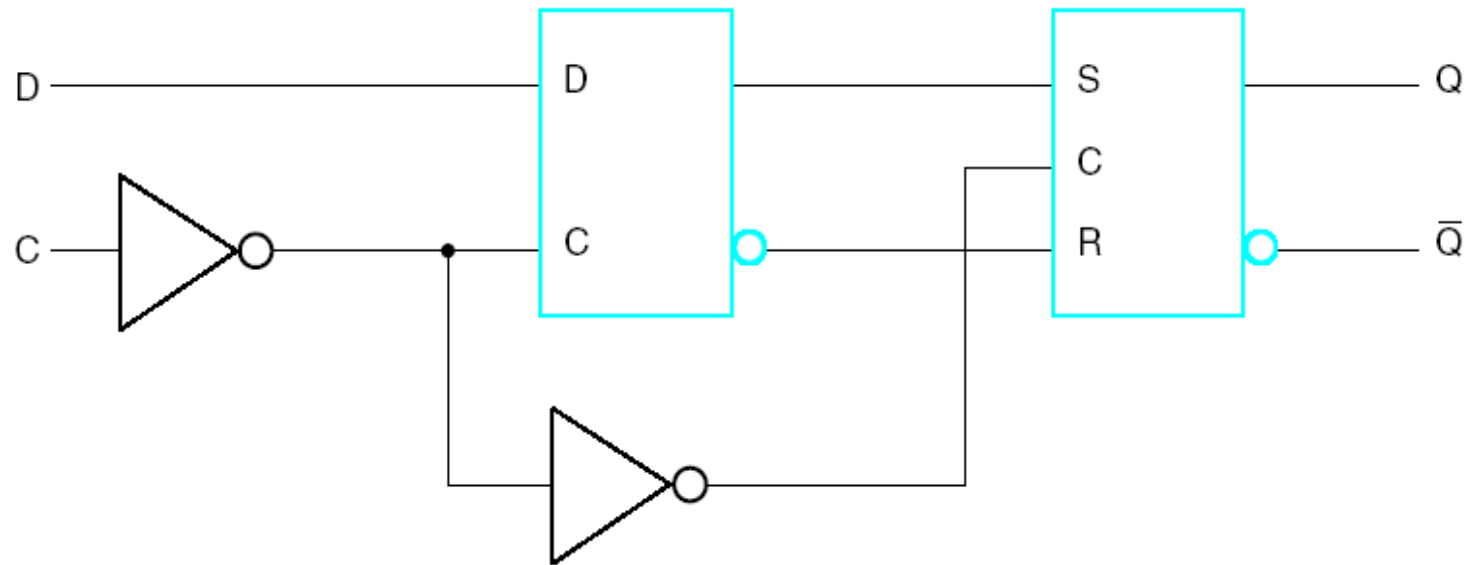
# Master-Slave FF configuration using SR latches

S	R	Clk	Q	Q'	
0	0	1	$Q_0$	$Q_0'$	Store
0	1	1	0	1	Reset
1	0	1	1	0	Set
1	1	1	1	1	Disallowed
X	X	0	$Q_0$	$Q_0'$	Store

- When  $C=1$ , master is enabled and stores *new* data, slave stores *old* data.
- When  $C=0$ , master's state passes to enabled slave, master not sensitive to new data (disabled).



# D Flip-Flop



# Characteristic Tables

- ❑ Defines the logical properties of a flip-flop (such as a truth table does for a logic gate).
- ❑  $Q(t)$  – present state at time  $t$
- ❑  $Q(t+1)$  – next state at time  $t+1$

# Characteristic Tables

## SR Flip-Flop

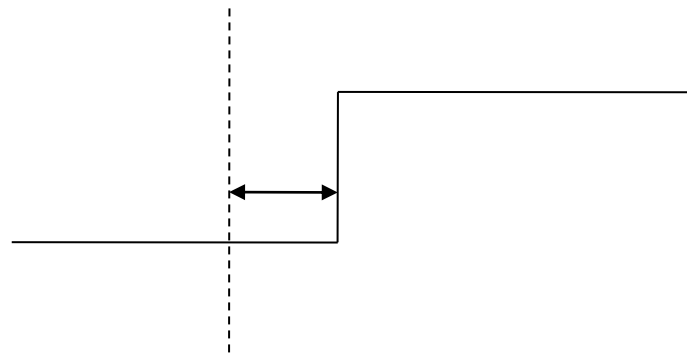
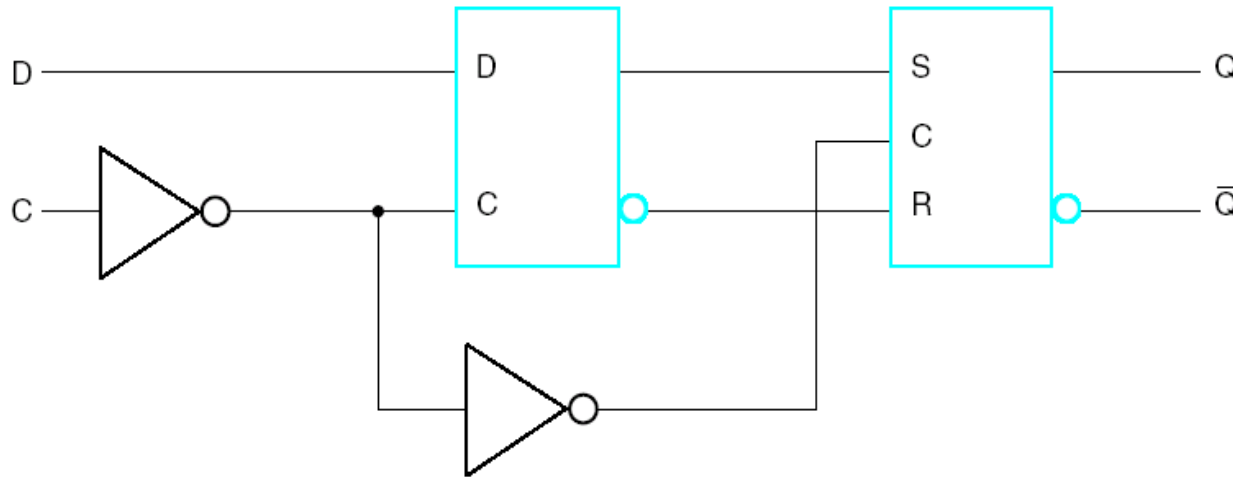
S	R	Q(t+1)	Operation
0	0	Q(t)	No change/Hold
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined/Invalid

# Characteristic Tables

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

Characteristic Equation:  $Q(t+1) = D(t)$

# D Flip-Flop Timing Parameters



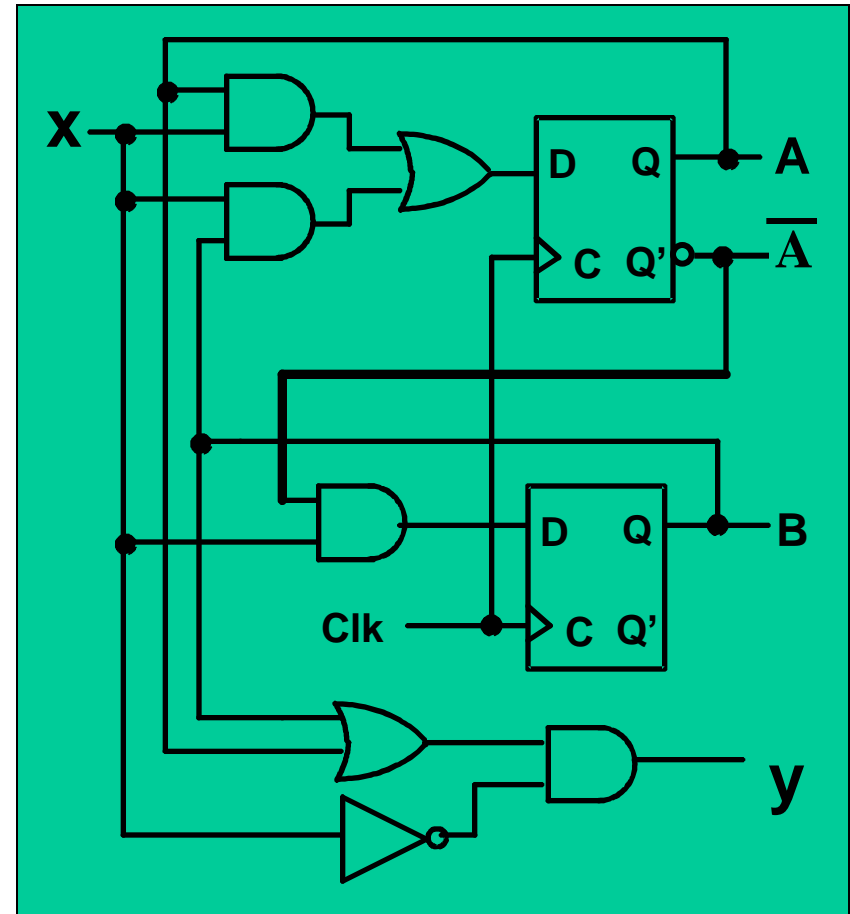
Setup time

# Sequential Circuit Analysis

- ❑ ***Analysis:*** Consists of obtaining a suitable description that demonstrates the time sequence of inputs, outputs, and states.
- ❑ Logic diagram: Boolean gates, flip-flops (of any kind), and appropriate interconnections.
- ❑ The logic diagram is derived from any of the following:
  - Boolean Equations (FF-Inputs, Outputs)
  - State Table
  - State Diagram

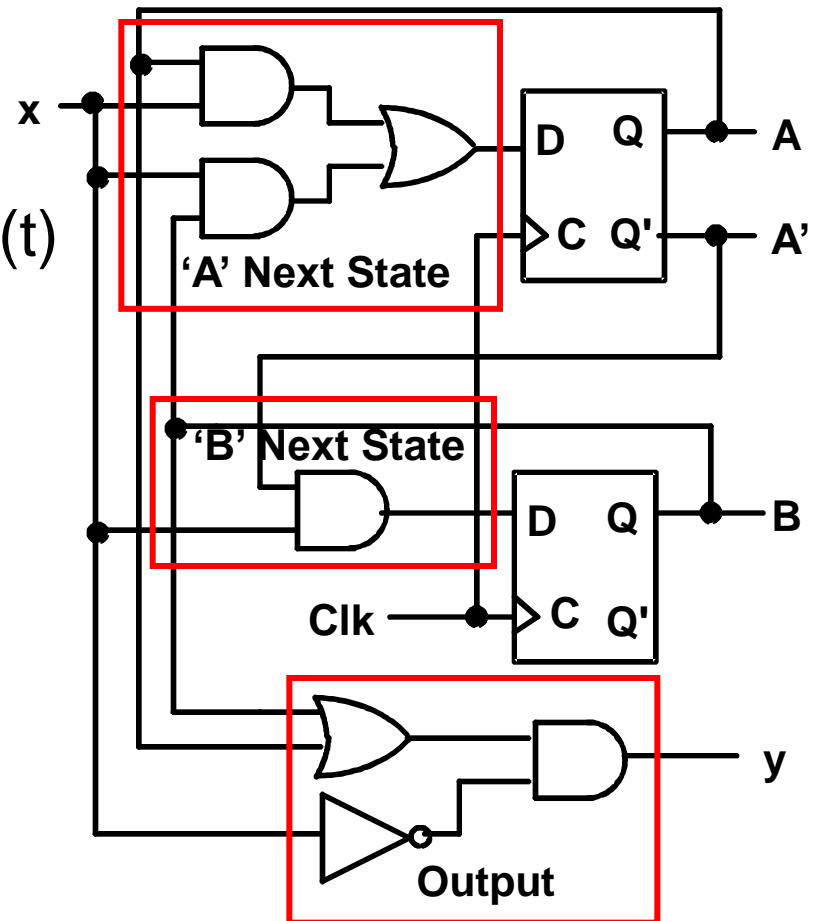
# Example

- ❑ Input:  $x(t)$
- ❑ Output:  $y(t)$
- ❑ State:  $(A(t), B(t))$
- ❑ What is the Output?
- ❑ What is the Next State Function?



# Example

- Boolean equations :
- $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = A'(t)x(t)$
  - $y(t) = x'(t)(B(t) + A(t))$



# State Table Characteristics

- ❑ *State table* – a multiple variable table with the following four sections:
  - *Present State* – the values of the state variables for each allowed state.
  - *Input* – the input combinations allowed.
  - *Next-state* – the value of the state at time  $(t+1)$  based on the present state and the input.
  - *Output* – the value of the output as a function of the present state and (sometimes) the input.
- ❑ From the viewpoint of a truth table:
  - the inputs are Input, Present State
  - and the outputs are Output, Next State

# Example – State Table

□ The state table can be filled in using the next state and output equations:

- $A(t+1) = A(t)x(t) + B(t)x(t)$
- $B(t+1) = \bar{A}(t)x(t);$
- $y(t) = \bar{x}(t)(B(t) + A(t))$

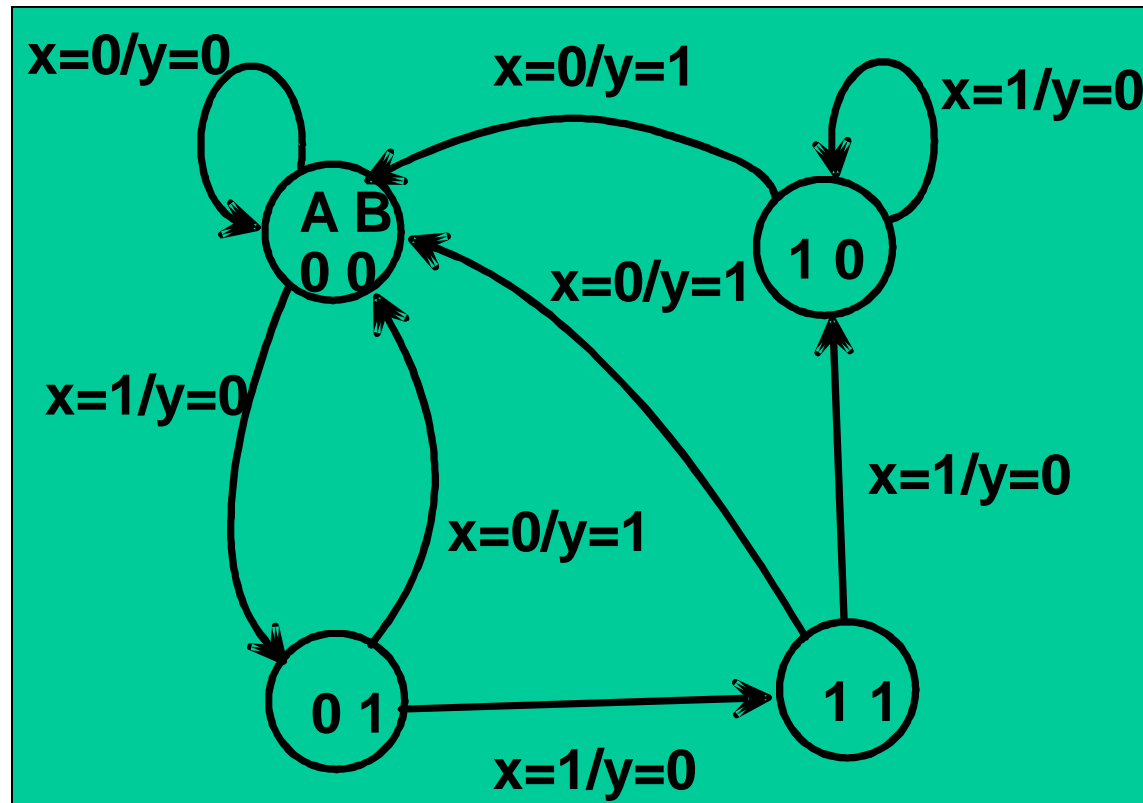
Present State		Input	Next State		Output
A(t)	B(t)	x(t)	A(t+1)	B(t+1)	y(t)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

# State Diagrams

- ❑ The sequential circuit function can be represented in graphical form as a state diagram with the following components:
  - A circle with the state name in it for each state
  - A directed arc from the Present State to the Next State for each state transition
  - A label on each directed arc with the Input values which causes the state transition, and
  - A label:
    - On each circle with the output value produced, or
    - On each directed arc with the output value produced.

# Example – State Diagrams

- ❑ Diagram gets confusing for large circuits
- ❑ For small circuits, usually easier to understand than the state table



# Summary

- ❑ Sequential circuit timing analysis
- ❑ Flip-Flop
  - Transmission gate based flip-flop design
  - Setup time
- ❑ State table and diagrams