

---

# Computer Logic Design Fundamentals

## Chapter 4 – Sequential Circuits

### Part 1 – Storage Elements and Sequential Circuit Analysis

Prof. Yueming Wang

[ymingwang@zju.edu.cn](mailto:ymingwang@zju.edu.cn)

College of Computer Science and Technology,  
Zhejiang University

# Overview

---

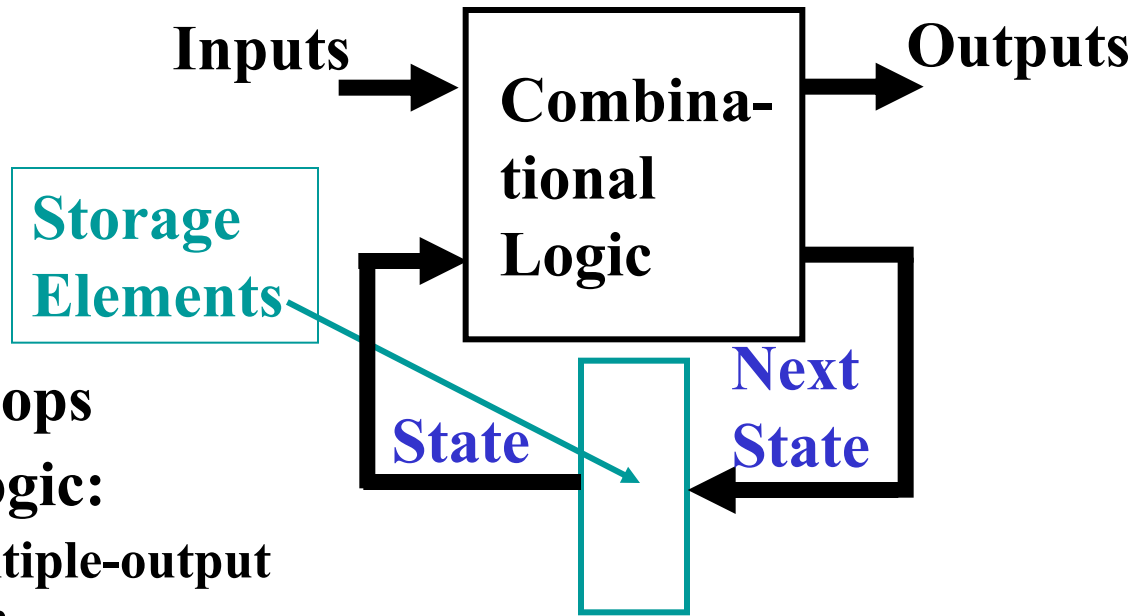
- **Part 1 - Storage Elements and Analysis**
  - Introduction to sequential circuits
  - Types of sequential circuits
  - Storage elements
    - Latches
    - Flip-flops
  - Sequential circuit analysis
    - State tables
    - State diagrams
    - Equivalent states
    - Moore and Mealy Models
- **Part 2 - Sequential Circuit Design**
- **Part 3 – State Machine Design**

# Introduction to Sequential Circuits

---

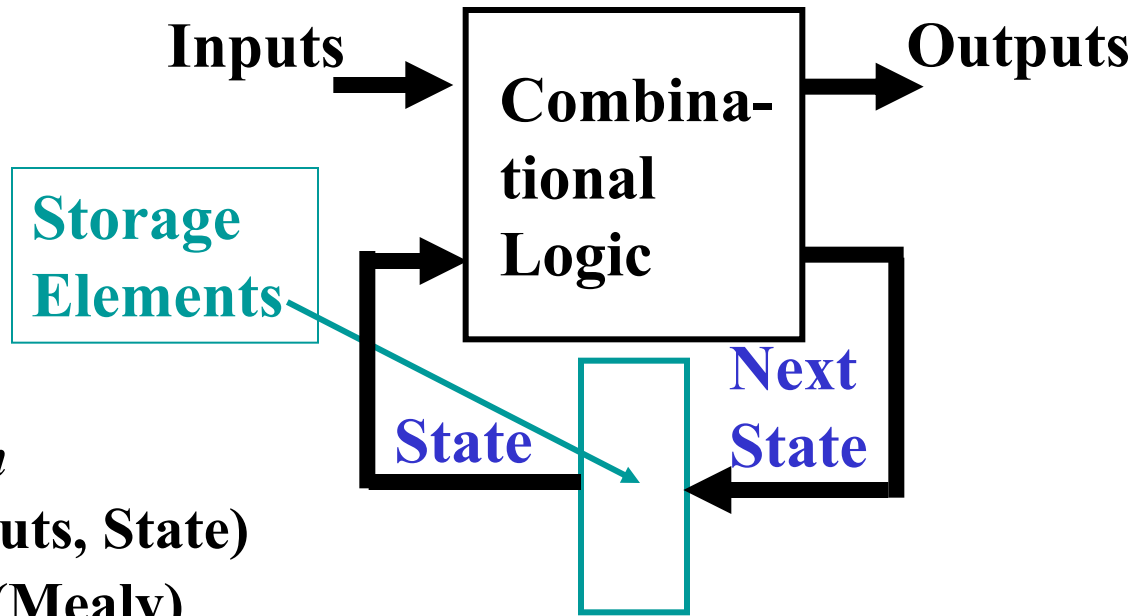
- A Sequential circuit contains:

- Storage elements: Latches or Flip-Flops
- Combinational Logic:
  - Implements a multiple-output switching function
  - Inputs are signals from the outside.
  - Outputs are signals to the outside.
  - Other inputs, State or Present State, are signals from storage elements.
  - The remaining outputs, Next State are inputs to storage elements.



# Introduction to Sequential Circuits

---



- **Combinatorial Logic**
  - *Next state function*  
 $\text{Next State} = f(\text{Inputs}, \text{State})$
  - *Output function (Mealy)*  
 $\text{Outputs} = g(\text{Inputs}, \text{State})$
  - *Output function (Moore)*  
 $\text{Outputs} = h(\text{State})$
- **Output function type depends on specification and affects the design significantly**

# Types of Sequential Circuits

---

- **Depends on the times at which:**
  - storage elements observe their inputs, and
  - storage elements change their state
- **Synchronous**
  - Behavior defined from knowledge of its signals at discrete instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)
- **Asynchronous**
  - Behavior defined from knowledge of inputs at any instant of time and the order in continuous time in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!
  - Nevertheless, the synchronous abstraction makes complex designs tractable!

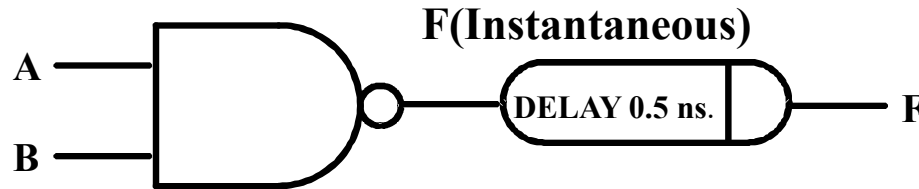
# Discrete Event Simulation

---

- In order to understand the time behavior of a sequential circuit we use discrete event simulation.
- Rules:
  - Gates modeled by an ideal (instantaneous) function and a fixed gate delay
  - Any change in input values is evaluated to see if it causes a change in output value
  - Changes in output values are scheduled for the fixed gate delay after the input change
  - At the time for a scheduled output change, the output value is changed along with any inputs it drives

# Simulated NAND Gate

- **Example: A 2-Input NAND gate with a 0.5 ns. delay:**



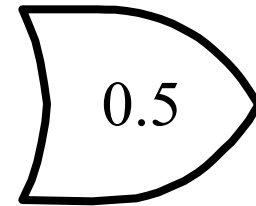
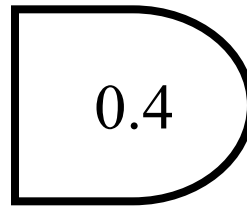
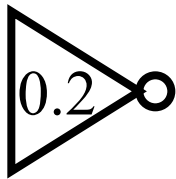
- Assume A and B have been 1 for a long time
- At time  $t=0$ , A changes to a 0 at  $t= 0.8$  ns, back to 1.

t (ns)	A	B	F(I)	F	Comment
$-\infty$	1	1	0	0	A=B=1 for a long time
0	$1 \Rightarrow 0$	1	$1 \Leftarrow 0$	0	F(I) changes to 1
0.5	0	1	1	$1 \Leftarrow 0$	F changes to 1 after a 0.5 ns delay
0.8	$1 \Leftarrow 0$	1	$1 \Rightarrow 0$	1	F(Instantaneous) changes to 0
1.3	1	1	0	$1 \Rightarrow 0$	F changes to 0 after a 0.5 ns delay

# Gate Delay Models

---

- Suppose gates with delay  $n$  ns are represented for  $n = 0.2$  ns,  $n = 0.4$  ns,  $n = 0.5$  ns, respectively:



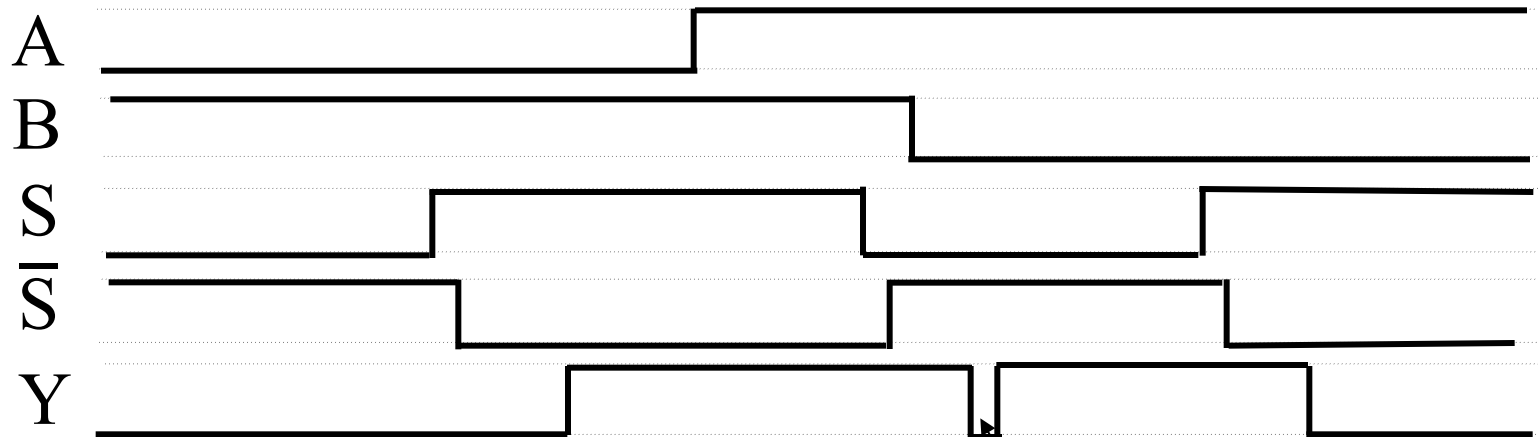
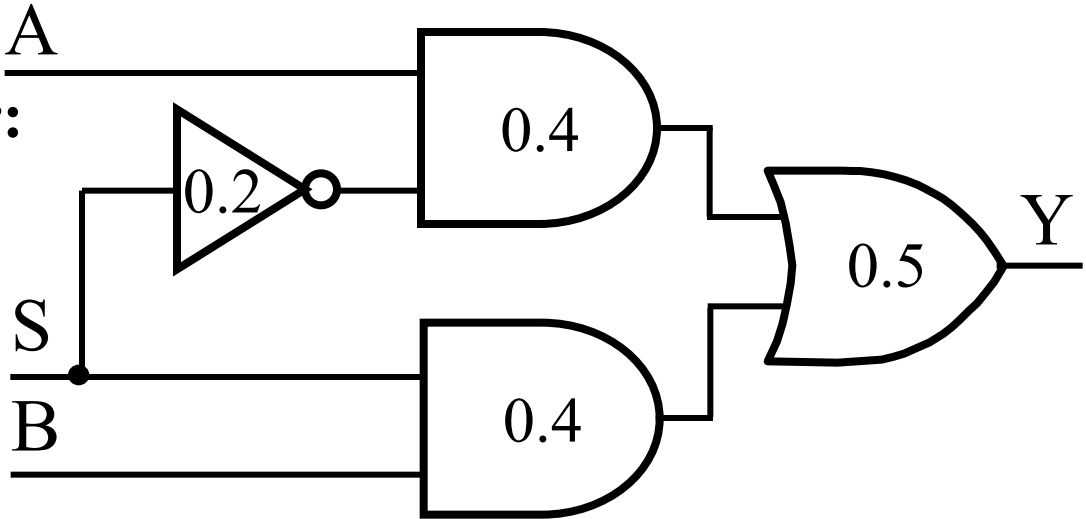


# Circuit Delay Model

- Consider a simple 2-input multiplexer:

- With function:

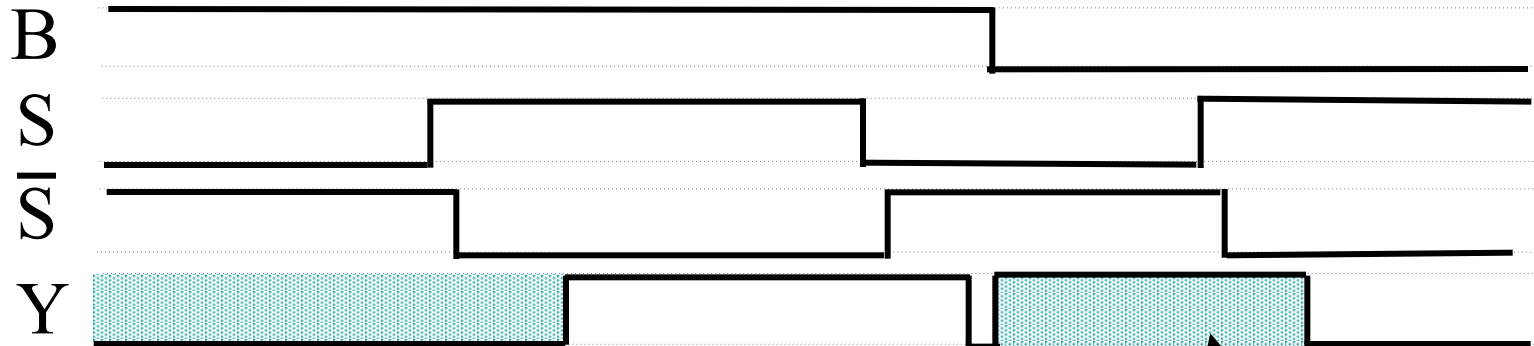
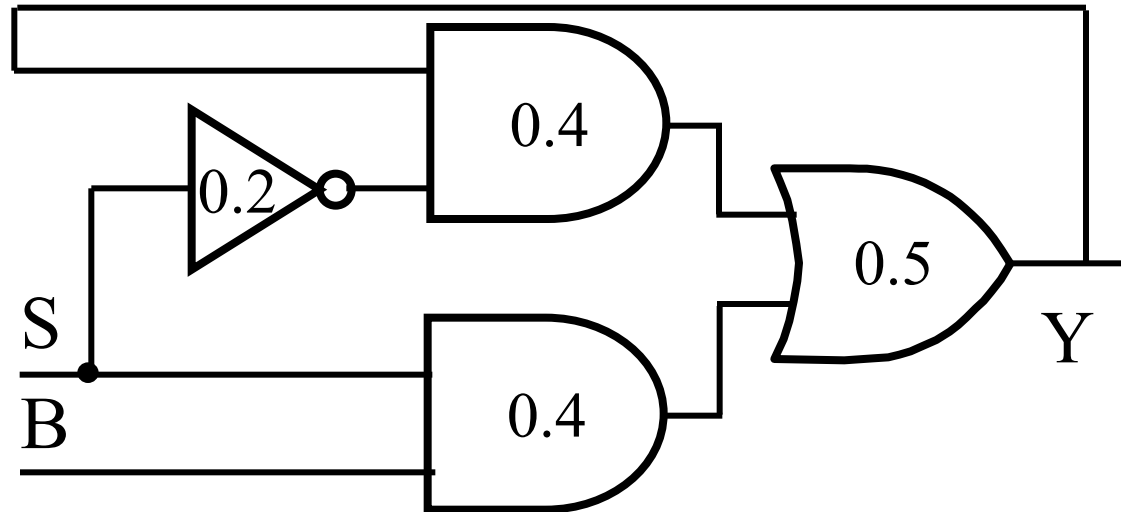
- $Y = A$  for  $S = 1$
- $Y = B$  for  $S = 0$



- “Glitch” is due to delay of inverter

# Storing State

- What if A connected to Y?
- Circuit becomes:
- With function:
  - $Y = B$  for  $S = 1$ , and  $Y(t)$  dependent on  $Y(t - 0.9)$  for  $S = 0$



- The simple combinational circuit has now become a sequential circuit because its output is a function of a time sequence of input signals!

Y is stored value in shaded area

# Storing State (Continued)

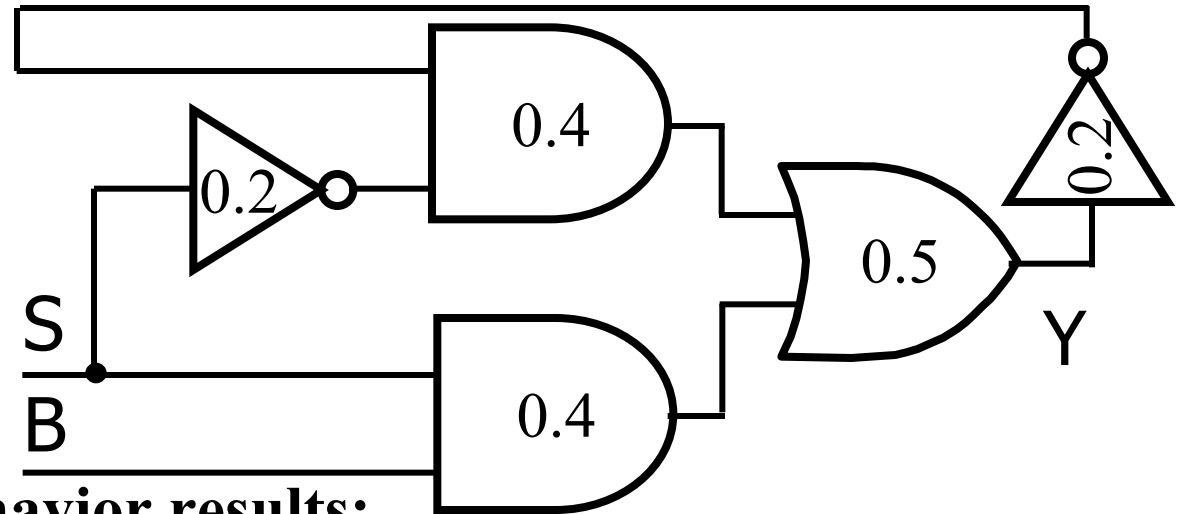
- **Simulation example as input signals change with time. Changes occur every 100 ns, so that the tenths of ns delays are negligible.**

Time ↓	B	S	Y	Comment
	1	0	0	Y “remembers” 0
	1	1	1	Y = B when S = 1
	1	0	1	Now Y “remembers” B = 1 for S = 0
	0	0	1	No change in Y when B changes
	0	1	0	Y = B when S = 1
	0	0	0	Y “remembers” B = 0 for S = 0
	1	0	0	No change in Y when B changes

- **Y represent the state of the circuit, not just an output.**

# Storing State (Continued)

- Suppose we place an inverter in the “feedback path.”



- The following behavior results:

- The circuit is said to be unstable.
- For  $S = 0$ , the circuit has become what is called an *oscillator*. Can be used as crude clock.

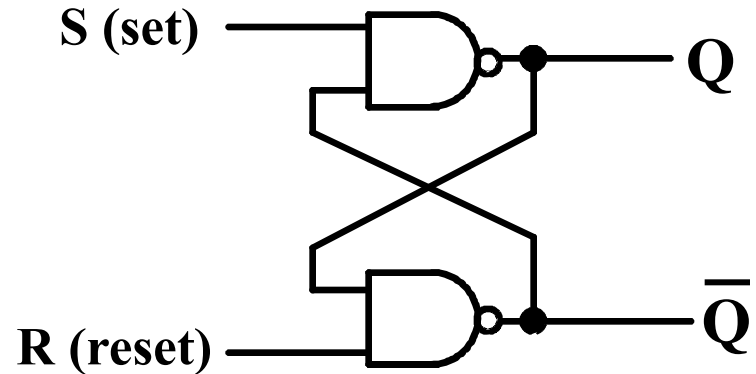
B	S	Y	Comment
0	1	0	Y = B when S = 1
1	1	1	
1	0	1	Now Y “remembers” A
1	0	0	Y, 1.1 ns later
1	0	1	Y, 1.1 ns later
1	0	0	Y, 1.1 ns later

## Basic (NAND) $\bar{S} - \bar{R}$ Latch

- **“Cross-Coupling”**  
two NAND gates gives  
the  $\bar{S}$  - $\bar{R}$  Latch:
  - **Which has the time  
sequence behavior:**
- S = 0, R = 0 is  
forbidden as  
input pattern**

Time

0
1
1
1
0
1
0



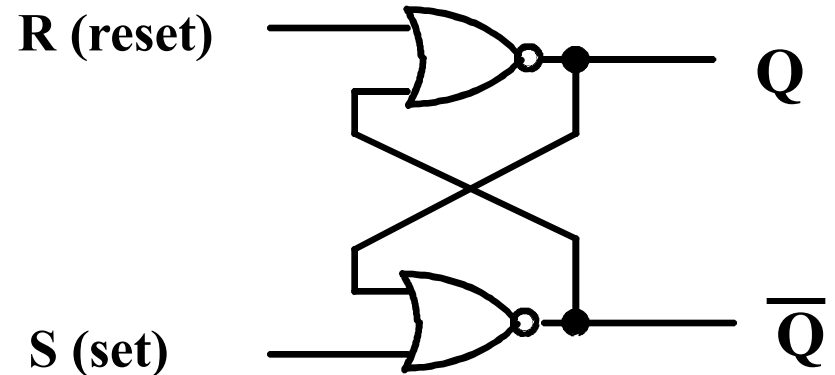
**Q1:**

Time ↓

R	S	Q	$\bar{Q}$	Comment
1	1	?	?	Stored state unknown
1	0	1	0	“Set” Q to 1
1	1	1	0	Now Q “remembers” 1
0	1	0	1	“Reset” Q to 0
1	1	0	1	Now Q “remembers” 0
0	0	1	1	Both go high
1	1	?	?	Unstable!

# Basic (NOR) S – R Latch

- Cross-coupling two NOR gates gives the S – R Latch:



- Which has the time sequence behavior:

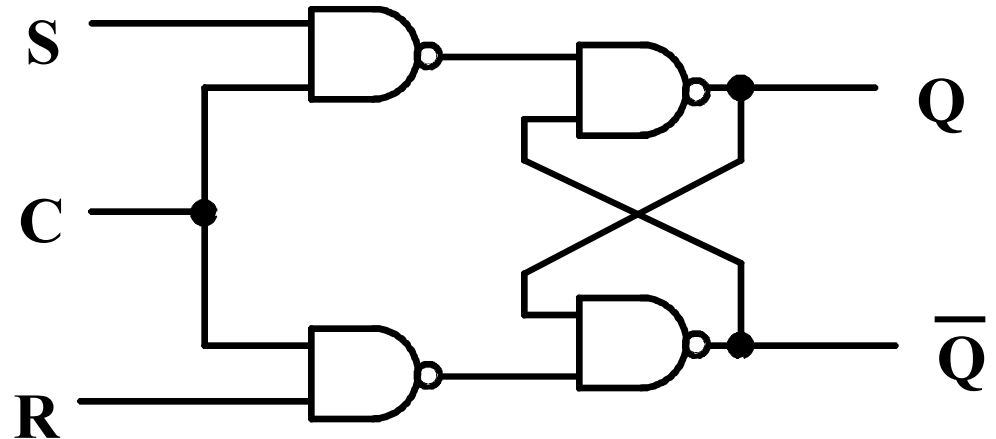
Time



R	S	Q	$\bar{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	“Set” Q to 1
0	0	1	0	Now Q “remembers” 1
1	0	0	1	“Reset” Q to 0
0	0	0	1	Now Q “remembers” 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

# Clocked S - R Latch

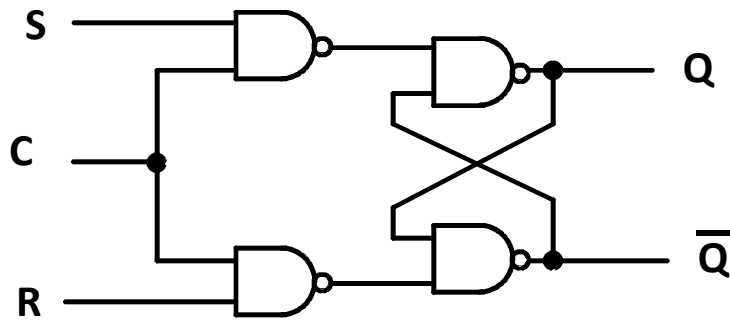
- Adding two NAND gates to the basic  $\overline{S}$  -  $\overline{R}$  NAND latch gives the clocked S - R latch:



- Has a time sequence behavior similar to the basic S-R latch except that the S and R inputs are only observed when the line C is high.
- C means “control” or “clock”.

# Clocked S - R Latch (continued)

- The Clocked S-R Latch can be described by a table:



C	S	R	Q(t + 1)
0	X	X	No change
1	0	0	No change
1	0	1	0: Clear Q
1	1	0	1: Set Q
1	1	1	Indeterminate

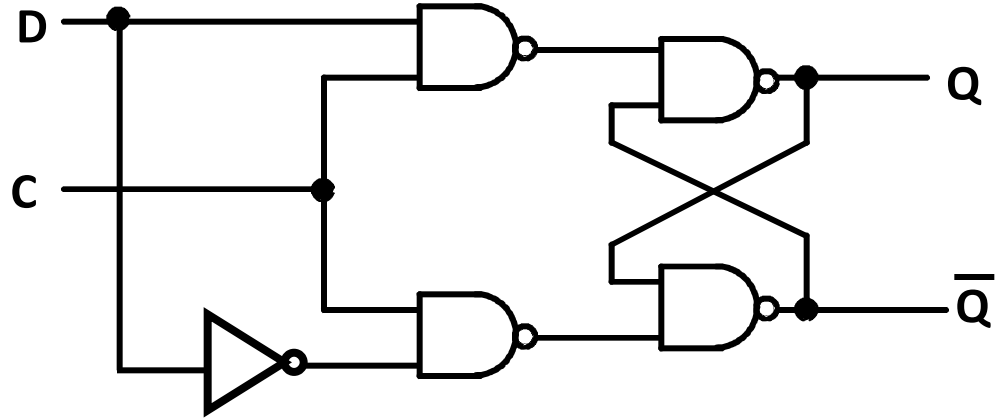
- The table describes what happens after the clock [at time (t+1)] based on:

- current inputs (S,R,C) and
- current state Q(t).

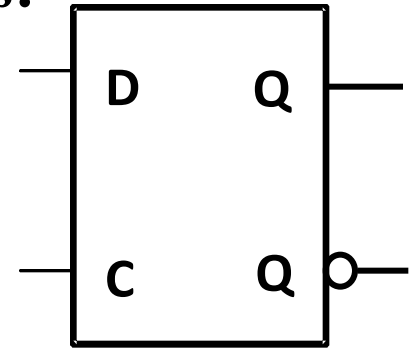


# D Latch

- **Adding an inverter to the S-R Latch, gives the D Latch:**
- **Note that there are no “indeterminate” states!**



**The graphic symbol for a D Latch is: \_\_\_\_\_**



<b>C</b>	<b>D</b>	<b>Q(t + 1)</b>
<b>0</b>	<b>X</b>	<b>No change</b>
<b>1</b>	<b>0</b>	<b>0: Clear Q</b>
<b>1</b>	<b>1</b>	<b>1: Set Q</b>

# Flip-Flops

---

- **The latch timing problem**
- **Master-slave flip-flop**
- **Edge-triggered flip-flop**
- **Standard symbols for storage elements**
- **Direct inputs to flip-flops**

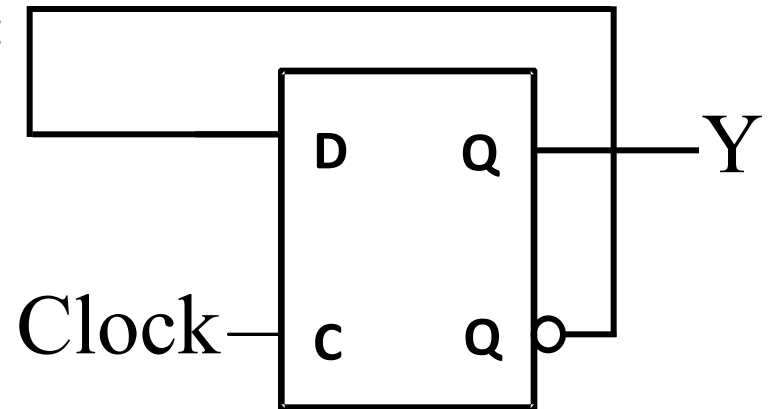
# The Latch Timing Problem

---

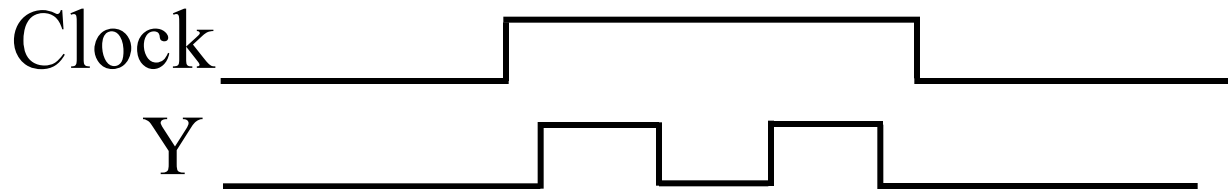
- **In a sequential circuit, paths may exist through combinational logic:**
  - From one storage element to another
  - From a storage element back to the same storage element
- **The combinational logic between a latch output and a latch input may be as simple as an interconnect**
- **For a clocked D-latch, the output  $Q$  depends on the input  $D$  whenever the clock input  $C$  has value 1**

# The Latch Timing Problem (continued)

- Consider the following circuit:



- Suppose that initially  $Y = 0$ .



- As long as  $C = 1$ , the value of  $Y$  continues to change!
- The changes are based on the delay present on the loop through the connection from  $Y$  back to  $Y$ .
- This behavior is clearly unacceptable.
- Desired behavior:  $Y$  changes only once per clock pulse

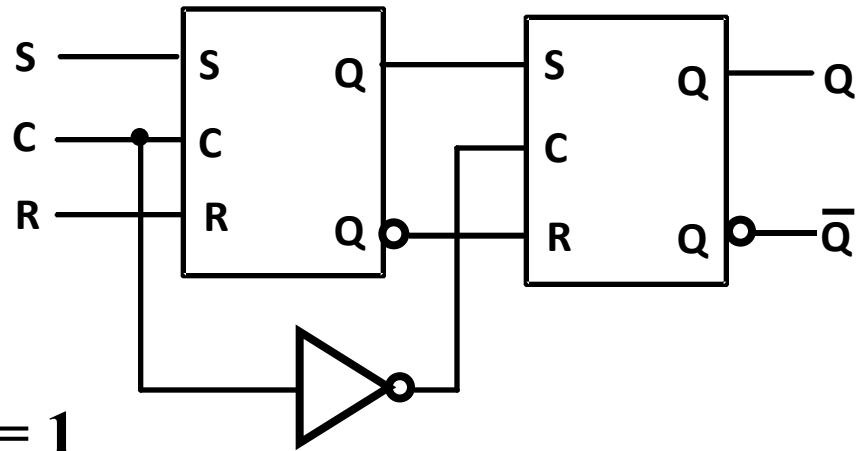
# The Latch Timing Problem (continued)

---

- A solution to the latch timing problem is to break the closed path from Y to Y within the storage element
- The commonly-used, path-breaking solutions replace the clocked D-latch with:
  - a master-slave flip-flop
  - an edge-triggered flip-flop

# S-R Master-Slave Flip-Flop

- Consists of two clocked S-R latches in series with the clock on the second latch inverted
- The input is observed by the first latch with  $C = 1$
- The output is changed by the second latch with  $C = 0$
- The path from input to output is broken by the difference in clocking values ( $C = 1$  and  $C = 0$ ).
- The behavior demonstrated by the example with D driven by Y given previously is prevented since the clock must change from 1 to 0 before a change in Y based on D can occur.



# Flip-Flop Problem

---

- The change in the flip-flop output is delayed by the pulse width which makes the circuit slower or
- S and/or R are permitted to change while  $C = 1$ 
  - Suppose  $Q = 0$  and S goes to 1 and then back to 0 with R remaining at 0
    - The master latch sets to 1
    - A 1 is transferred to the slave
  - Suppose  $Q = 0$  and S goes to 1 and back to 0 and R goes to 1 and back to 0
    - The master latch sets and then resets
    - A 0 is transferred to the slave
  - This behavior is called *1s catching*

# Flip-Flop Solution

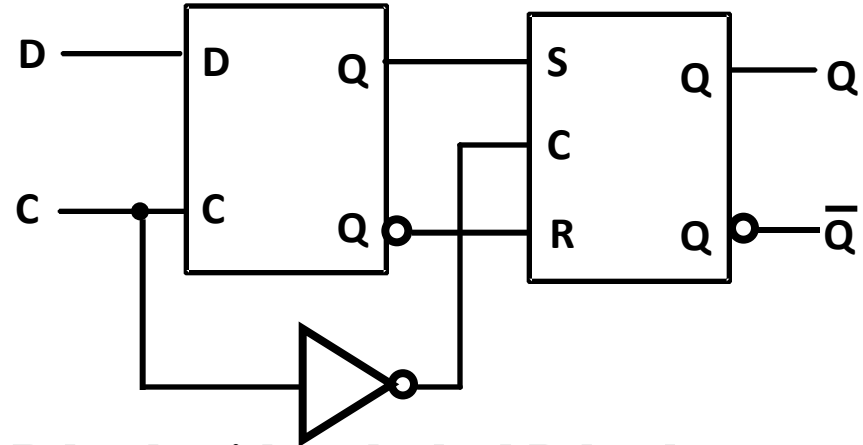
---

- Use edge-triggering instead of master-slave
- An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a transition of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level, or
- A master-slave D flip-flop which also exhibits edge-triggered behavior can be used.



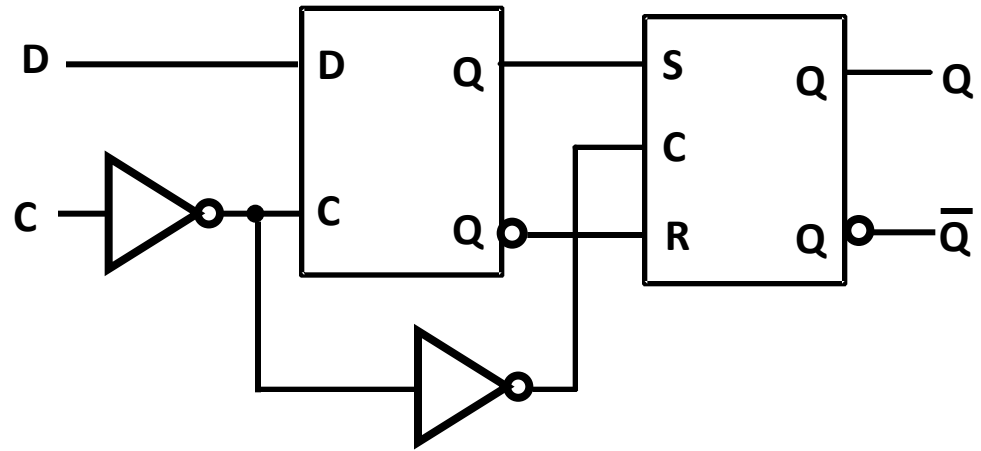
# Edge-Triggered D Flip-Flop

- The function of edge-triggered D flip-flop is same as the master-slave D flip-flop
- It can be formed by:
  - Replacing the first clocked S-R latch with a clocked D latch or
  - Adding a D input and inverter to a master-slave S-R flip-flop
- The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs
- The change of the D flip-flop output is associated with the negative edge at the end of the pulse
- It is called a *negative-edge triggered* flip-flop



# Positive-Edge Triggered D Flip-Flop

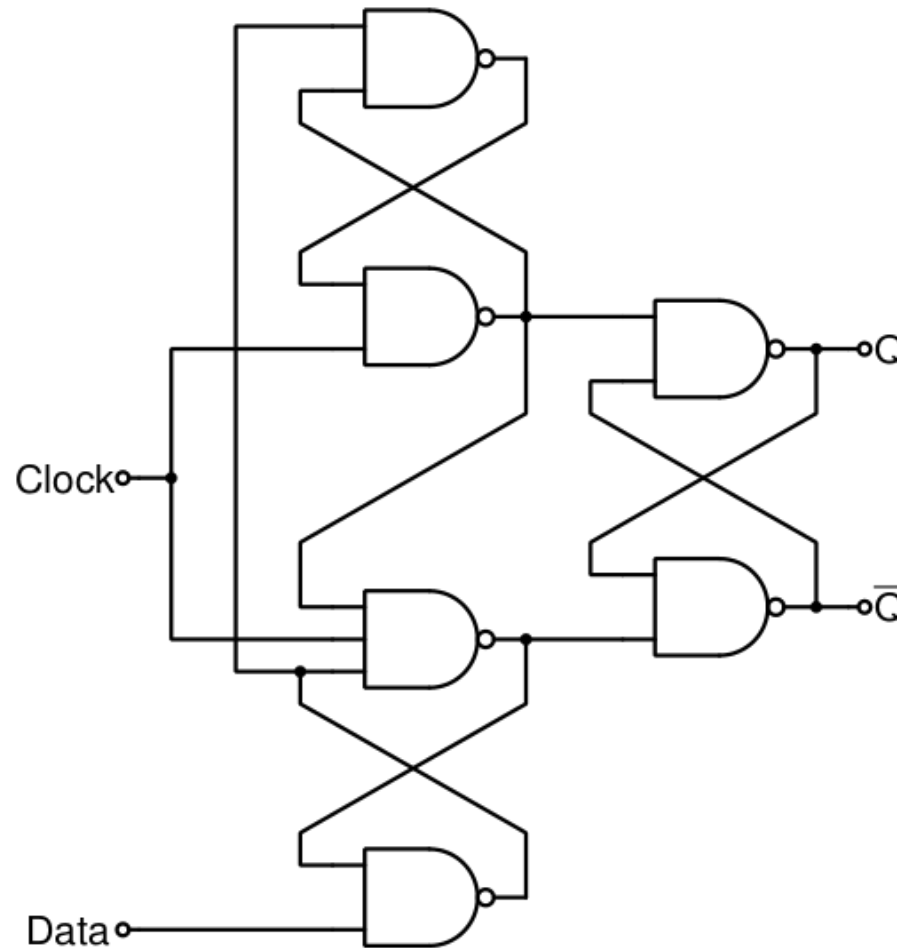
- Formed by adding inverter to clock input



- Q changes to the value on D applied at the positive clock edge within timing constraints to be specified
- Our choice as the standard flip-flop for most sequential circuits

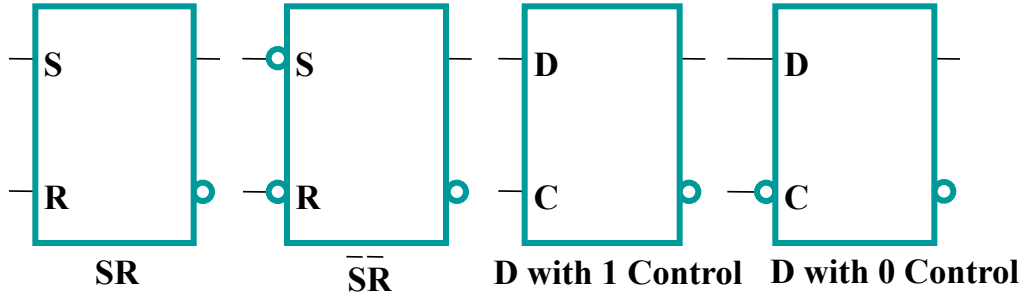
# Actual Circuit of Edge-Triggered D Flip-Flop

---



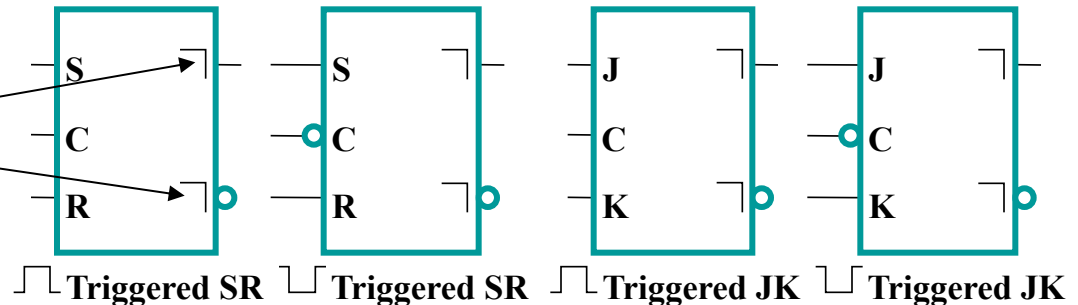
**Complicated to be analyzed**

# Standard Symbols for Storage Elements



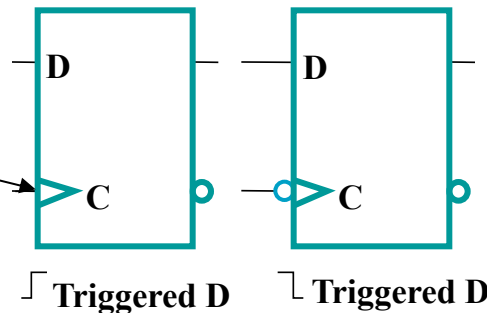
(a) Latches

- **Master-Slave:**  
**Postponed output**  
**indicators**



(b) Master-Slave Flip-Flops

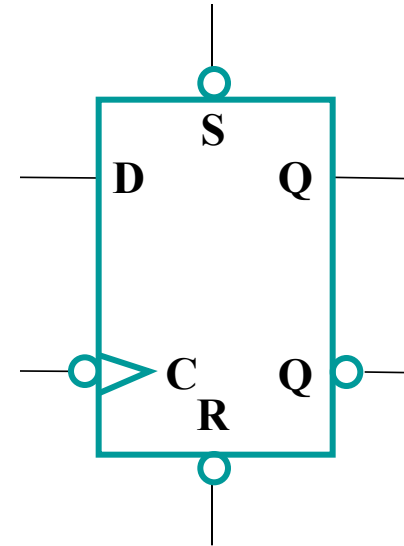
- **Edge-Triggered:**  
**Dynamic**  
**indicator**



(c) Edge-Triggered Flip-Flops

# Direct Inputs

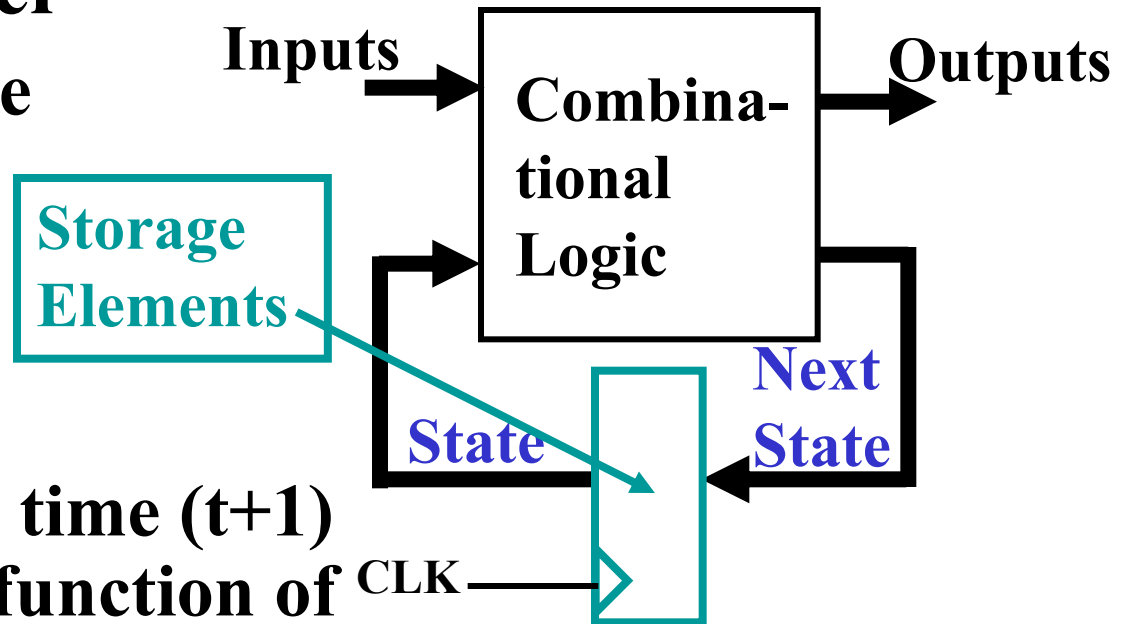
- At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation
- This initialization is often done outside of the clocked behavior of the circuit, i.e., asynchronously.
- Direct R and/or S inputs that control the state of the latches within the flip-flops are used for this initialization.
- For the example flip-flop shown
  - 0 applied to  $\overline{R}$  resets the flip-flop to the 0 state
  - 0 applied to  $\overline{S}$  sets the flip-flop to the 1 state



# Sequential Circuit Analysis

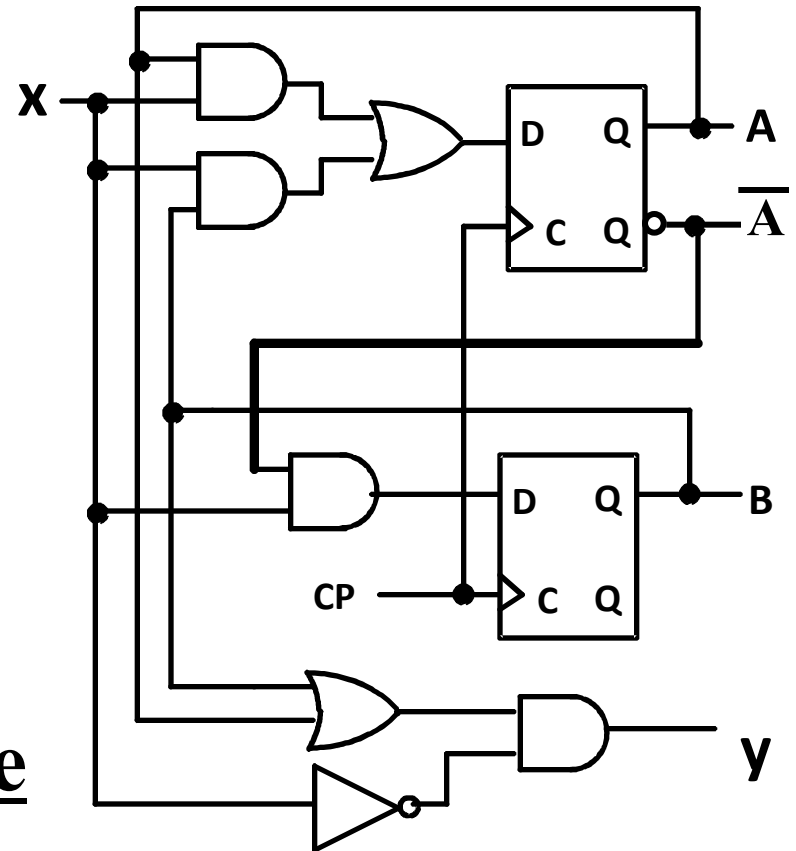
## ■ General Model

- Current State at time (t) is stored in an array of flip-flops.
- Next State at time (t+1) is a Boolean function of State and Inputs.
- Outputs at time (t) are a Boolean function of State (t) and (sometimes) Inputs (t).



# Example 1 (from Fig. 4-13)

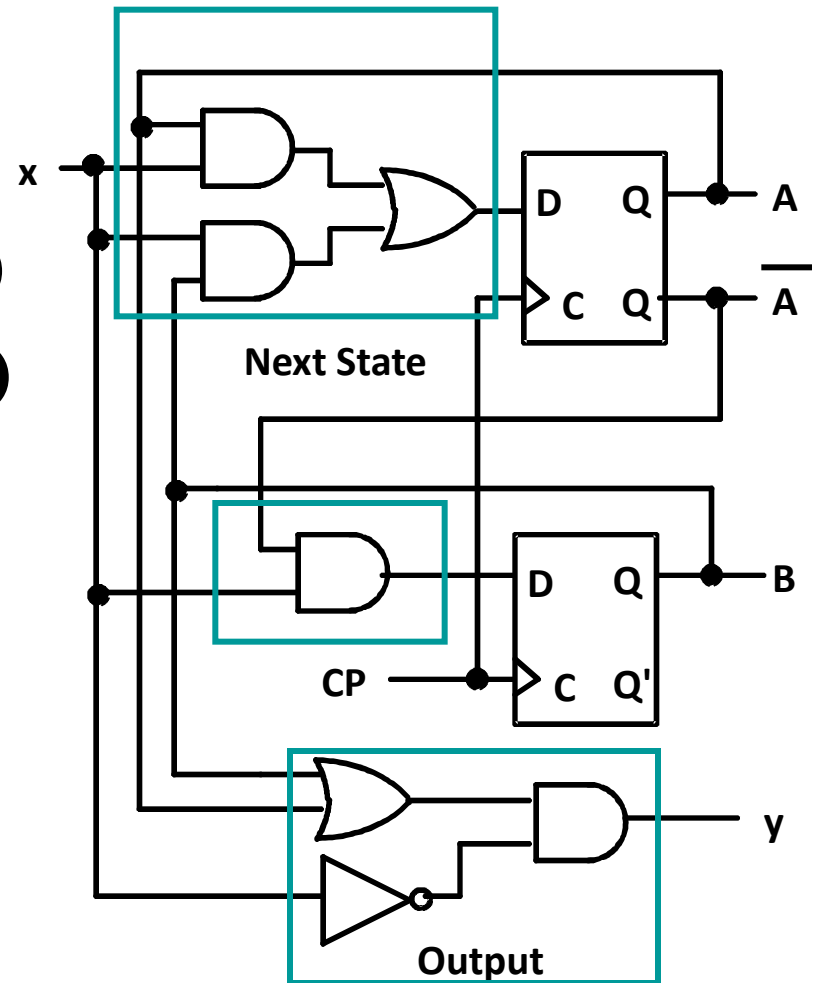
- Input:  $x(t)$
- Output:  $y(t)$
- State:  $(A(t), B(t))$
- What is the Output Function?
- What is the Next State Function?



# Example 1 (from Fig. 4-13) (continued)

## ■ Boolean equations for the functions:

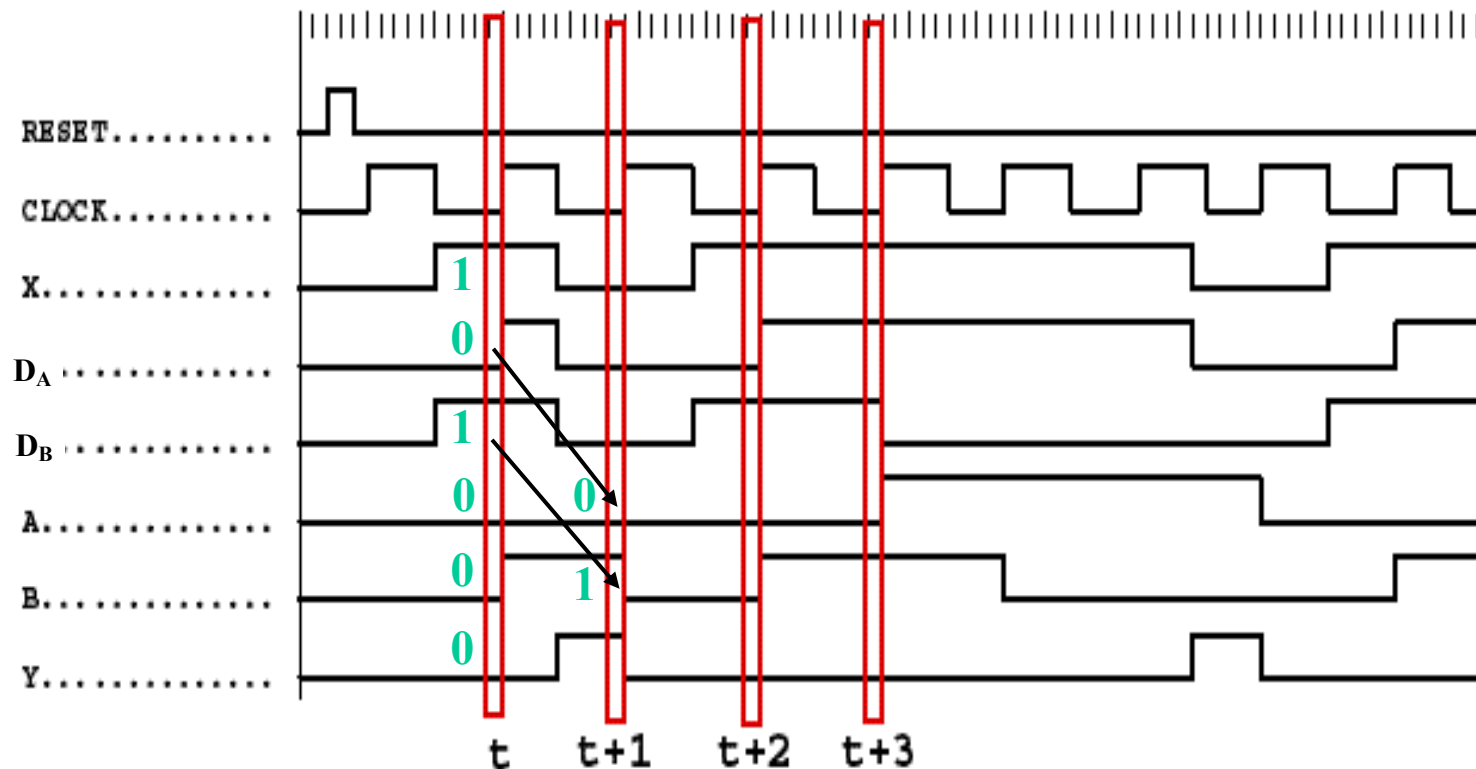
- $D_A(t) = A(t)x(t) + B(t)x(t)$
- $A(t+1) = A(t)x(t) + B(t)x(t)$
- $D_B(t) = \bar{A}(t)x(t)$
- $B(t+1) = \bar{A}(t)x(t)$
- $y(t) = \bar{x}(t)(B(t) + A(t))$





# Example 1(from Fig. 4-13) (continued)

- Where in time are inputs, outputs and states defined?



# 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 1: State Table (from Fig. 4-13)

---

- 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

# Example 1: Alternate State Table

- 2-dimensional table that matches well to a K-map.  
Present state rows and input columns in Gray code order.

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

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

# State Diagrams

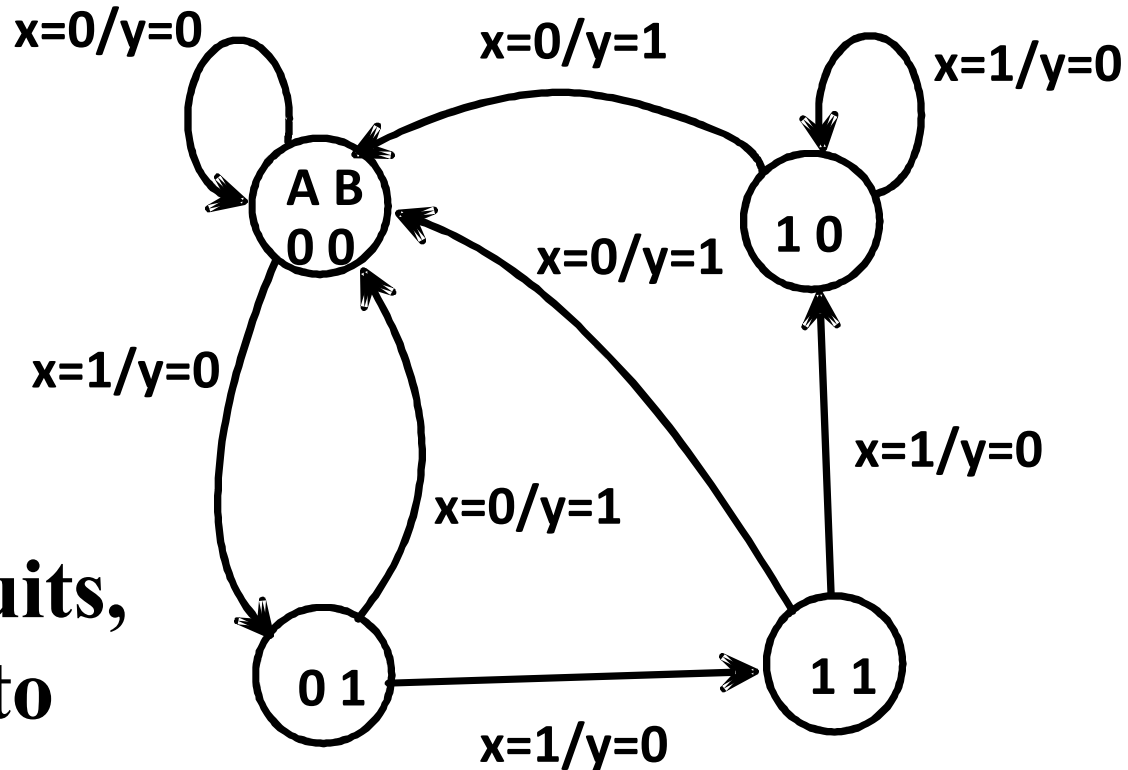
---

- **Label form:**
  - **On circle with output included:**
    - **state/output**
    - **Moore type output depends only on state**
  - **On directed arc with the output included:**
    - **input/output**
    - **Mealy type output depends on state and input**

# Example 1: State Diagram

---

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



# Equivalent State Definitions

---

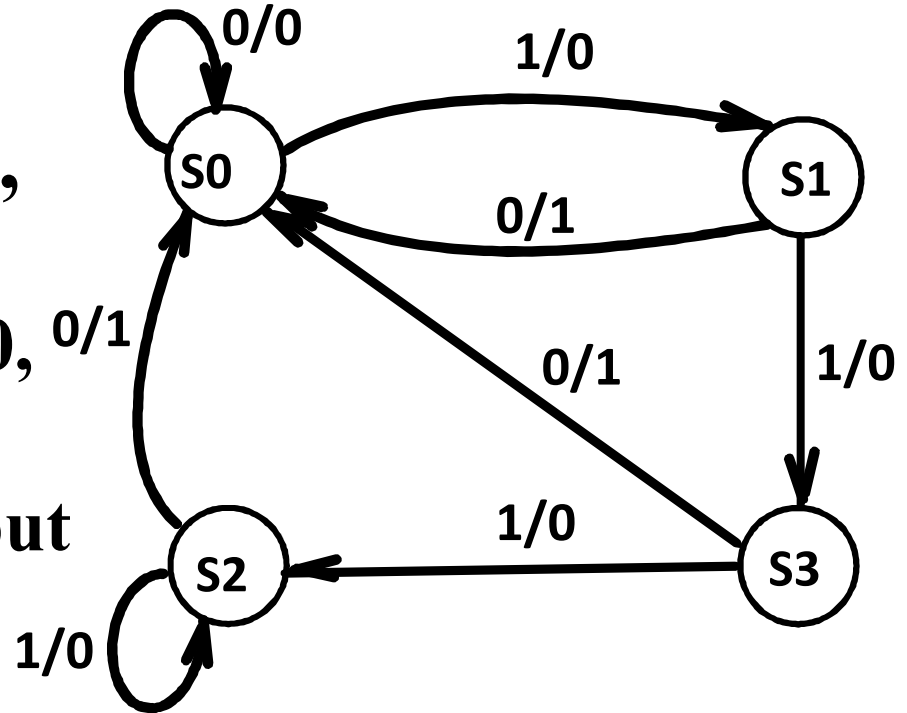
- Two states are *equivalent* if their response for each possible input sequence is an identical output sequence.
- Alternatively, two states are *equivalent* if their outputs produced for each input symbol is identical and their next states for each input symbol are the same or equivalent.



# Equivalent State Example

- Text Figure 4-15(a):
- For states S3 and S2,

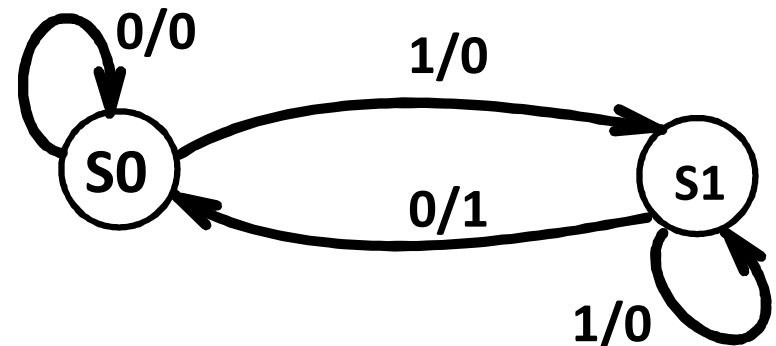
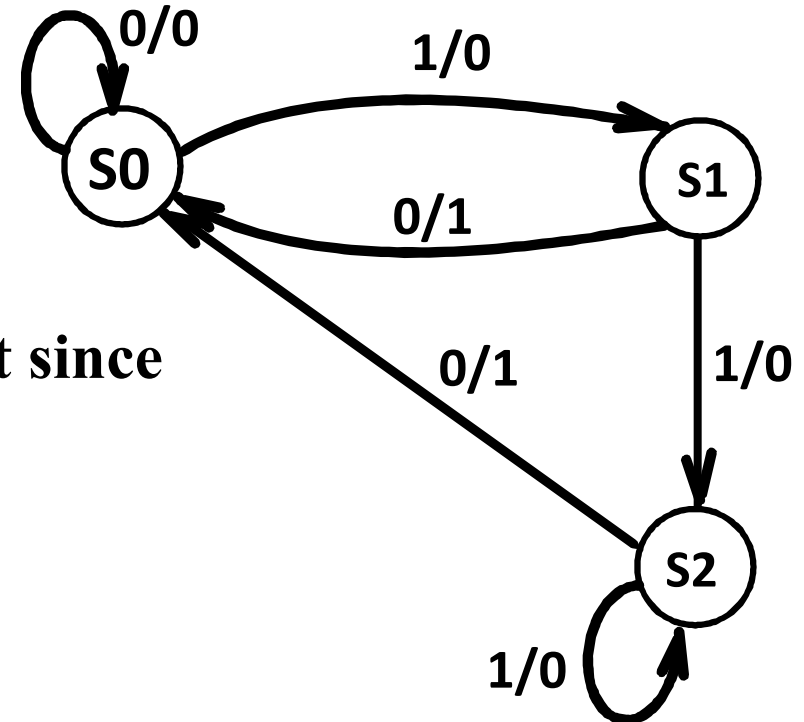
- the output for input 0 is 1 and input 1 is 0, and
- the next state for input 0 is S0 and for input 1 is S2.



- By the alternative definition, states S3 and S2 are equivalent.

# Equivalent State Example

- Replacing S3 and S2 by a single state gives state diagram:
- Examining the new diagram, states S1 and S2 are equivalent since
  - their outputs for input 0 is 1 and input 1 is 0, and
  - their next state for input 0 is S0 and for input 1 is S2,
- Replacing S1 and S2 by a single state gives state diagram:



# Moore and Mealy Models

---

- **Sequential Circuits or Sequential Machines are also called *Finite State Machines* (FSMs). Two formal models exist:**

- **Moore Model**

- **Named after E.F. Moore**
- **Outputs are a function ONLY of states**
- **Usually specified on the states.**

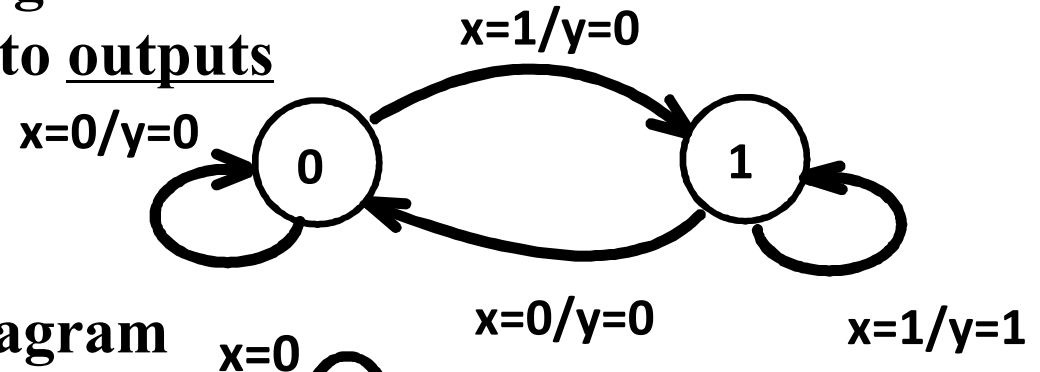
- **Mealy Model**

- **Named after G. Mealy**
- **Outputs are a function of inputs AND states**
- **Usually specified on the state transition arcs.**

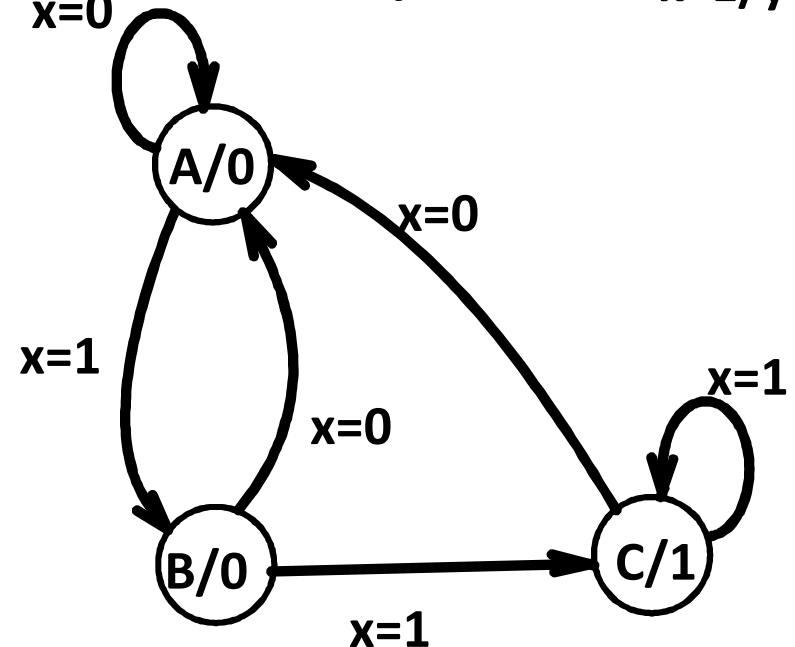
# Moore and Mealy Example Diagrams

---

- **Mealy Model State Diagram**  
maps inputs and state to outputs



- **Moore Model State Diagram**  
maps states to outputs



# Moore and Mealy Example Tables

---

- **Moore Model state table maps state to outputs**

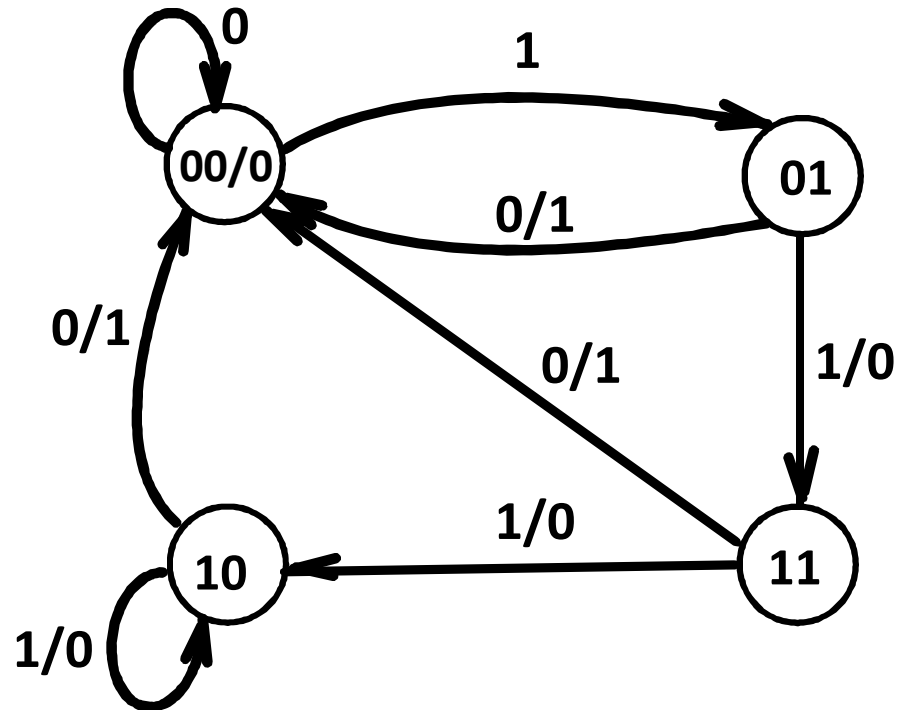
Present State	Next State		Output
	x=0	x=1	
A	A	B	0
B	A	C	0
C	A	C	1

- **Mealy Model state table maps inputs and state to outputs**

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

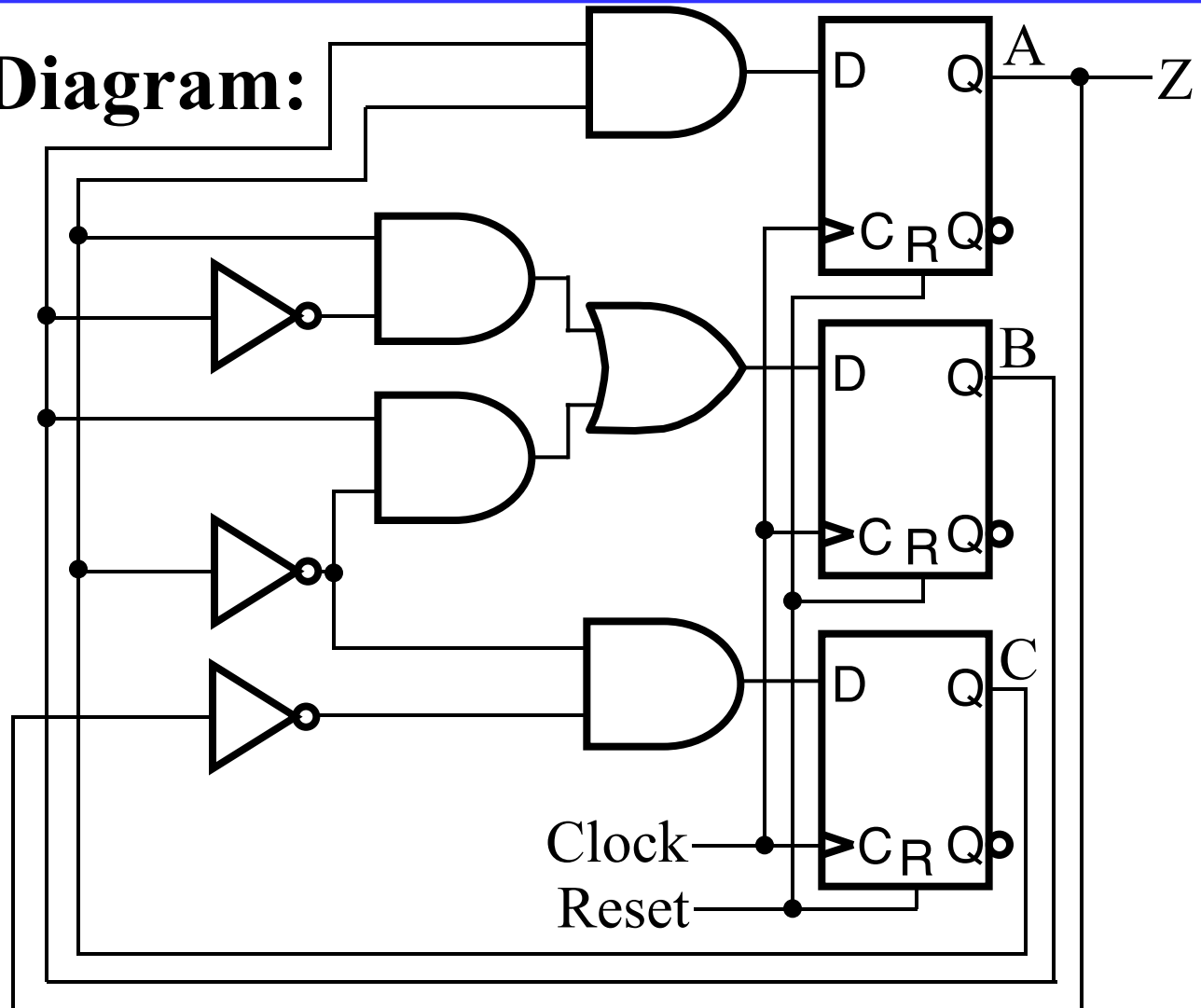
# Mixed Moore and Mealy Outputs

- In real designs, some outputs may be Moore type and other outputs may be Mealy type.
- Example: Figure 4-15(a) can be modified to illustrate this
  - State 00: Moore
  - States 01, 10, and 11: Mealy
- Simplifies output specification



# Example 2: Sequential Circuit Analysis

- Logic Diagram:



# Example 2: Flip-Flop Input Equations

---

## ■ Variables

- Inputs: None
- Outputs:  $Z$
- State Variables:  $A, B, C$

## ■ Initialization: Reset to (0,0,0)

## ■ Equations

- $A(t+1) =$   $Z =$
- $B(t+1) =$
- $C(t+1) =$



# Example 2: State Table

---

$X' = X(t+1)$

A B C	A'B'C'	Z
0 0 0		
0 0 1		
0 1 0		
0 1 1		
1 0 0		
1 0 1		
1 1 0		
1 1 1		

# Example 2: State Table

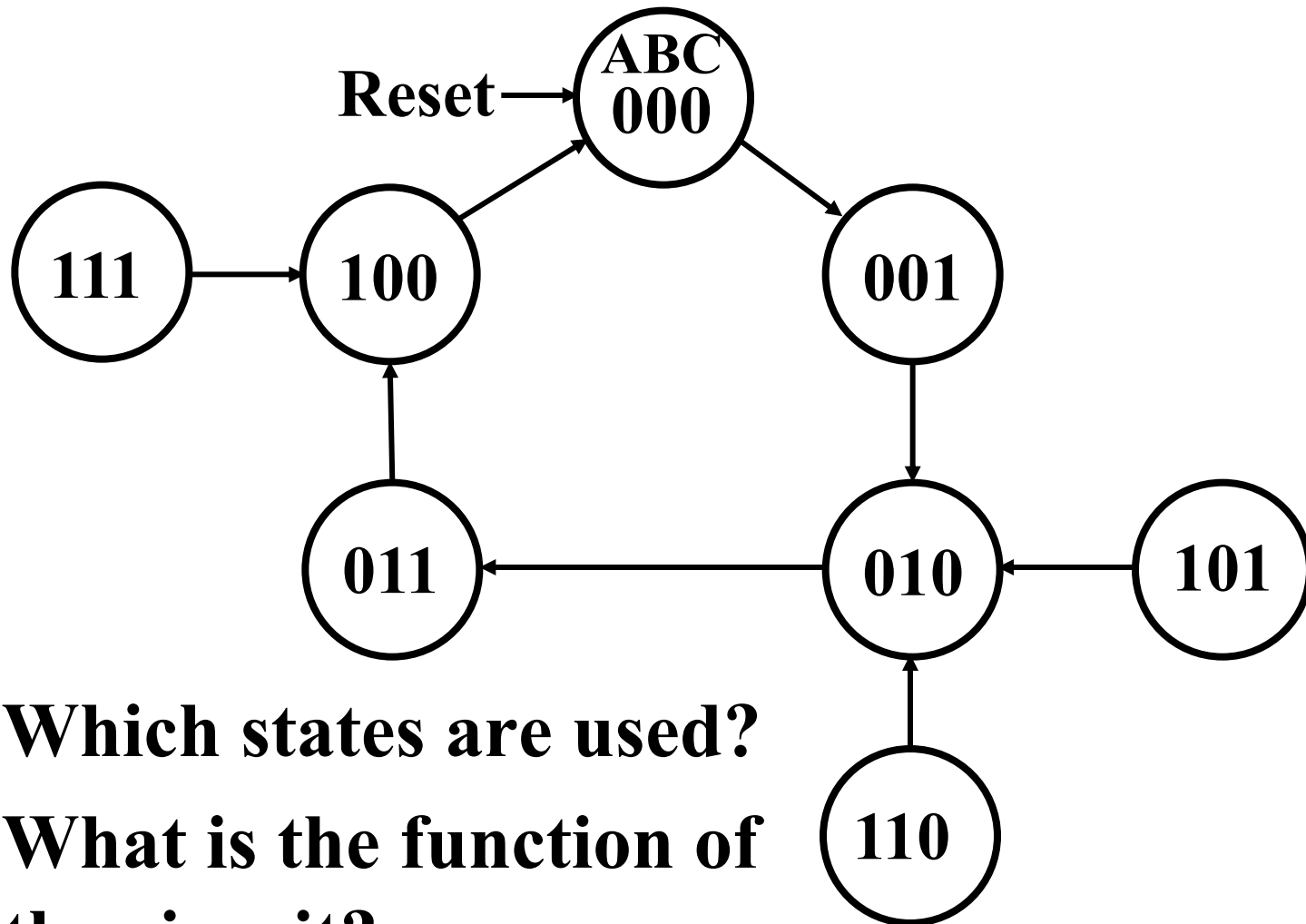
---

$X' = X(t+1)$

A B C	A'B'C'	Z
0 0 0	0 0 1	0
0 0 1	0 1 0	0
0 1 0	0 1 1	0
0 1 1	1 0 0	0
1 0 0	0 0 0	1
1 0 1	0 1 0	1
1 1 0	0 1 0	1
1 1 1	1 0 0	1

# Example 2: State Diagram

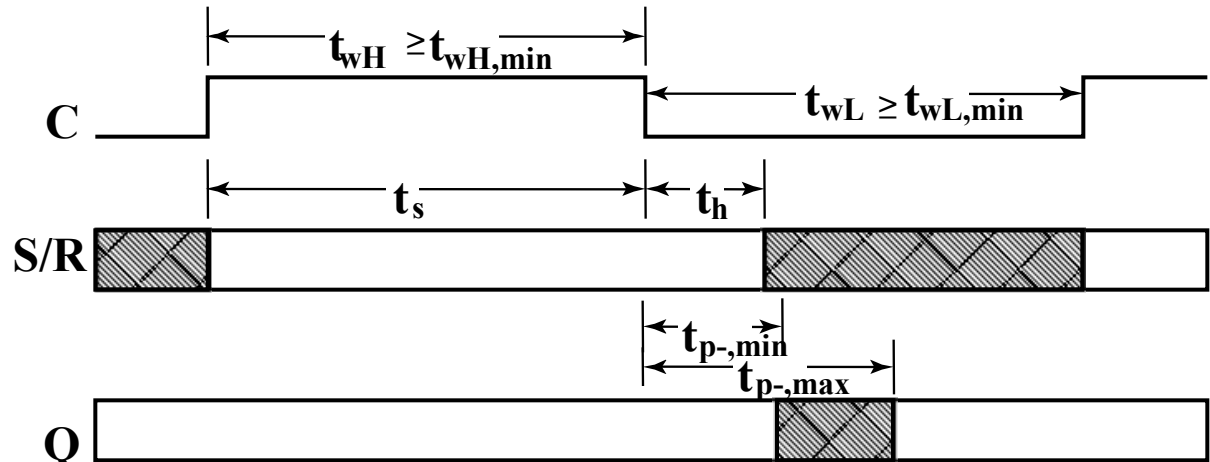
---



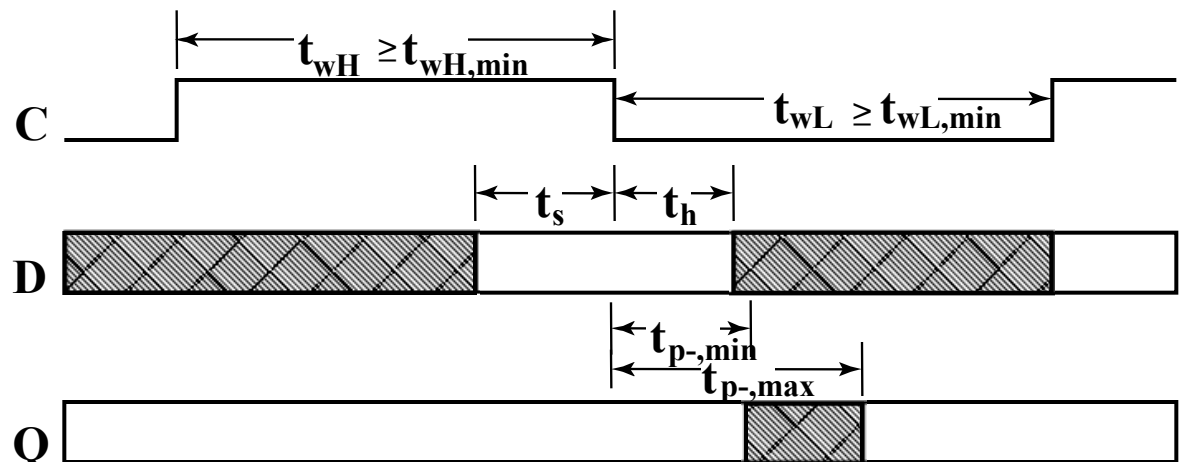
- Which states are used?
- What is the function of the circuit?

# Flip-Flop Timing Parameters

- $t_s$  - setup time
- $t_h$  - hold time
- $t_w$  - clock pulse width
- $t_{px}$  - propagation delay
  - $t_{PHL}$  - High-to-Low
  - $t_{PLH}$  - Low-to-High
  - $t_{pd} - \max(t_{PHL}, t_{PLH})$



(a) Pulse-triggered (positive pulse)



(b) Edge-triggered (negative edge)

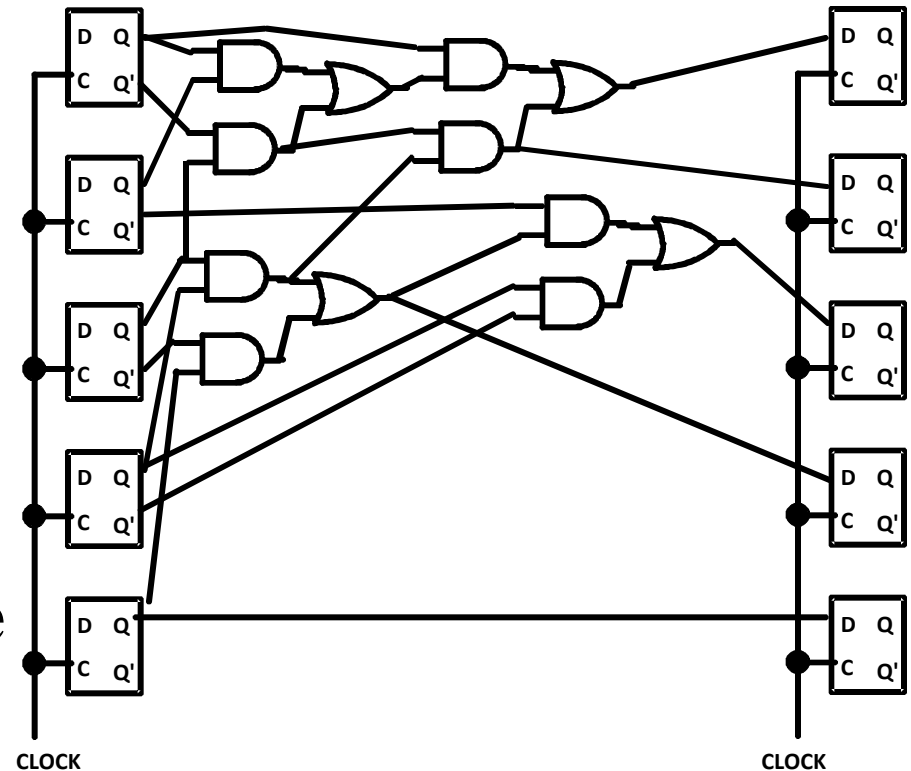
# Flip-Flop Timing Parameters

---

- $t_s$  - setup time
  - Master-slave - Equal to the width of the triggering pulse
  - Edge-triggered - Equal to a time interval that is generally much less than the width of the the triggering pulse
- $t_h$  - hold time - Often equal to zero
- $t_{px}$  - propagation delay
  - Same parameters as for gates except
  - Measured from clock edge that triggers the output change to the output change

# Circuit and System Level Timing

- Consider a system comprised of ranks of flip-flops connected by logic:
- If the clock period is too short, some data changes will not propagate through the circuit to flip-flop inputs before the setup time interval begins



# Circuit and System Level Timing

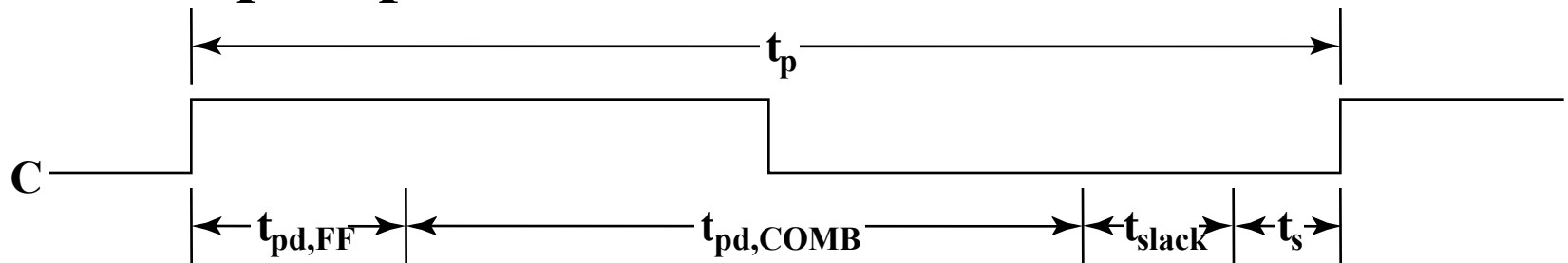
---

## ■ New Timing Components

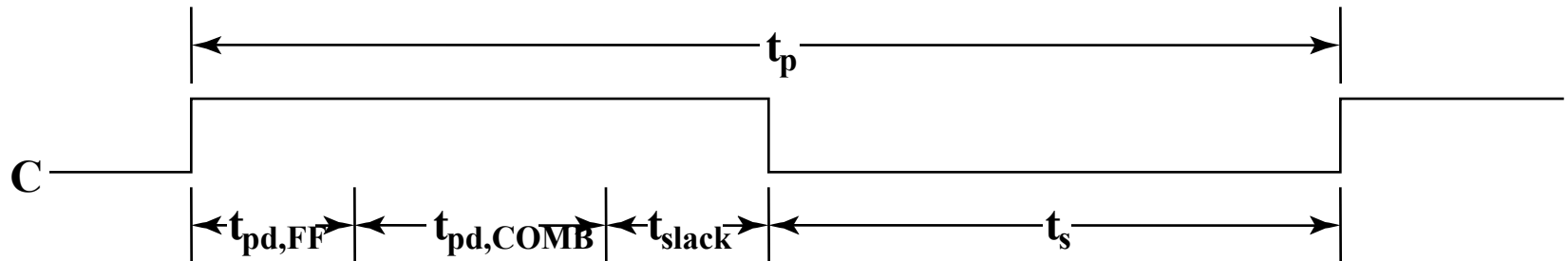
- $t_p$  - clock period - The interval between occurrences of a specific clock edge in a periodic clock
- $t_{pd,COMB}$  - total delay of combinational logic along the path from flip-flop output to flip-flop input
- $t_{slack}$  - extra time in the clock period in addition to the sum of the delays and setup time on a path
  - Can be either positive or negative
  - Must be greater than or equal to zero on all paths for correct operation

# Circuit and System Level Timing

- Timing components along a path from flip-flop to flip-flop



(a) Edge-triggered (positive edge)



(b) Pulse-triggered (negative pulse)



# Circuit and System Level Timing

---

- **Timing Equations**

$$t_p = t_{\text{slack}} + (t_{\text{pd,FF}} + t_{\text{pd,COMB}} + t_s)$$

- For  $t_{\text{slack}}$  greater than or equal to zero,

$$t_p \geq \max (t_{\text{pd,FF}} + t_{\text{pd,COMB}} + t_s)$$

for all paths from flip-flop output to flip-flop input

- **Can be calculated more precisely by using  $t_{\text{PHL}}$  and  $t_{\text{PLH}}$  values instead of  $t_{\text{pd}}$  values, but requires consideration of inversions on paths**

# Calculation of Allowable $t_{pd,COMB}$

---

- **Compare the allowable combinational delay for a specific circuit:**
  - a) Using edge-triggered flip-flops
  - b) Using master-slave flip-flops
- **Parameters**
  - $t_{pd,FF}(\max) = 1.0 \text{ ns}$
  - $t_s(\max) = 0.3 \text{ ns}$  for edge-triggered flip-flops
  - $t_s = t_{wH} = 2.0 \text{ ns}$  for master-slave flip-flops
  - Clock frequency = 250 MHz

# Calculation of Allowable $t_{pd,COMB}$

---

- **Calculations:**  $t_p = 1/\text{clock frequency} = 4.0 \text{ ns}$ 
  - **Edge-triggered:**  $4.0 \geq 1.0 + t_{pd,COMB} + 0.3$ ,  $t_{pd,COMB} \leq 2.7 \text{ ns}$
  - **Master-slave:**  $4.0 \geq 1.0 + t_{pd,COMB} + 2.0$ ,  $t_{pd,COMB} \leq 1.0 \text{ ns}$
- **Comparison:** Suppose that for a gate, average  $t_{pd} = 0.3 \text{ ns}$ 
  - **Edge-triggered:** Approximately 9 gates allowed on a path
  - **Master-slave:** Approximately 3 gates allowed on a path

# Assignments

---

- 4-2、 4-4、 4-6、 4-7、 4-8、 4-9、 4-11、 4-13、  
4-58、 4-59