Logic and Computer Design Fundamentals

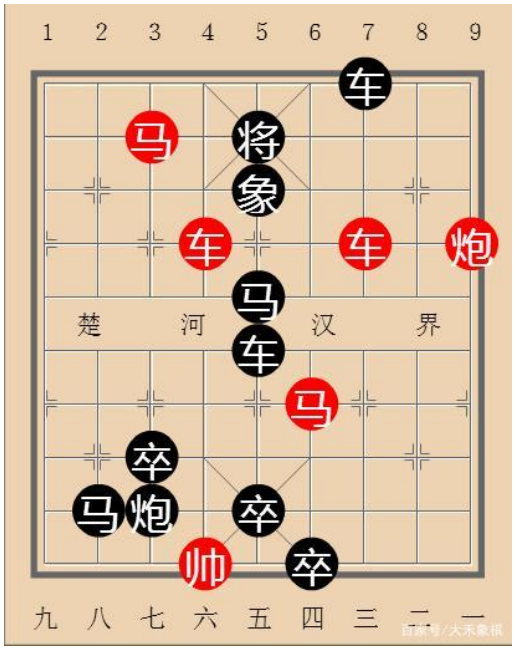
Chapter 4 – Sequential Circuits

Part 1 – Storage Elements and Sequential Circuit Analysis

Ming Cai
cm@zju.edu.cn
College of Computer Science and Technology
Zhejiang University

Overview

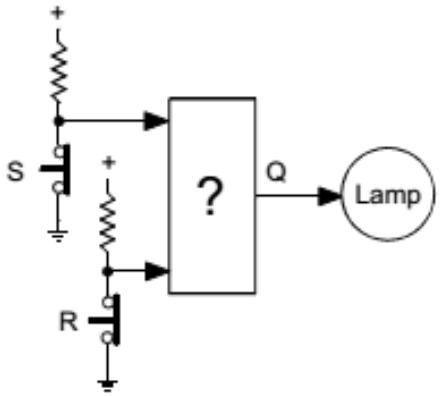
- **Part 1 - Storage Elements and Analysis**
  - **Introduction to sequential circuits**
  - **Types of sequential circuits**
  - **Storage elements**
    - **Latches**
    - **Flip-flops**
  - **Sequential circuit analysis**
    - **State tables**
    - **State diagrams**
    - **Equivalent states**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



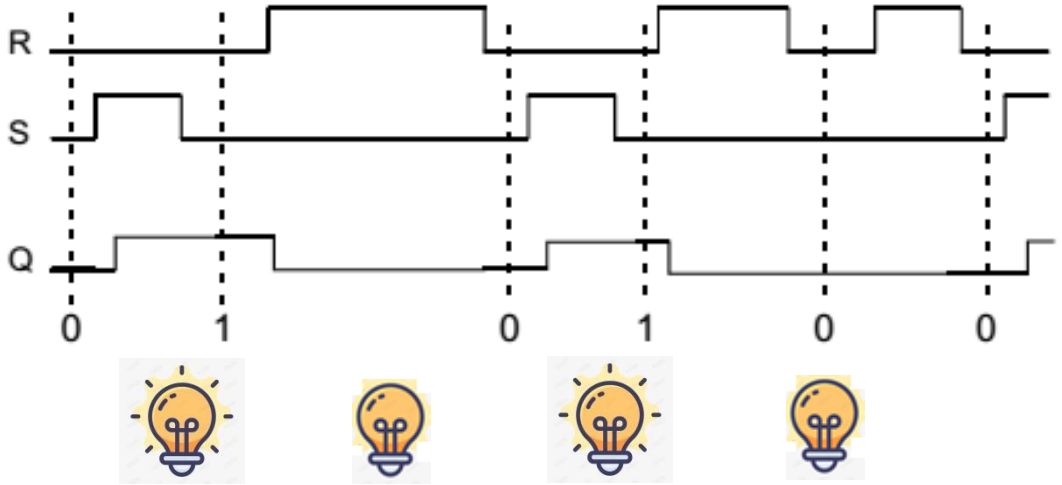
Going Beyond Combinational Logic

- **Problem:** **Design a circuit to control a lamp by two switches S and R. Assume that S and R are not pushed simultaneously.**

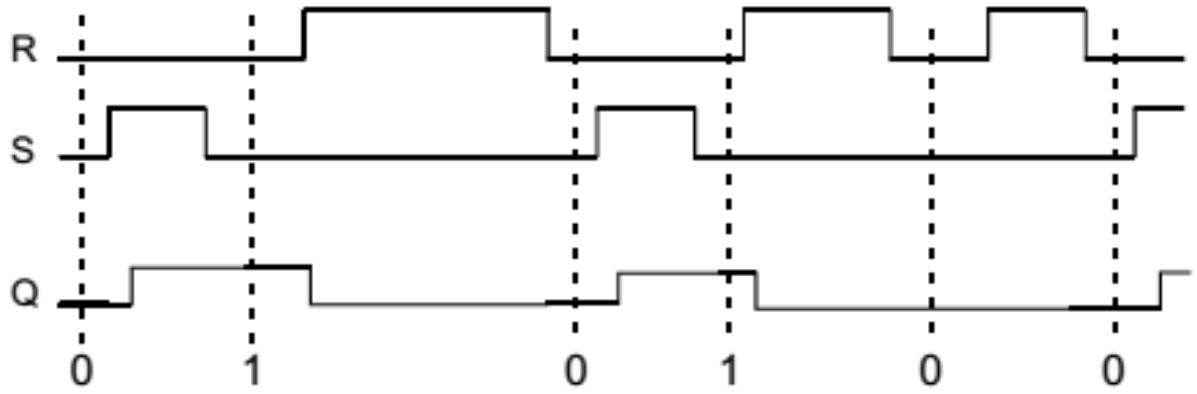
1) **if S = 1, then the lamp should turn on**
2) **if S = 0, then the lamp should stay on**
3) **if R = 1, then the lamp should turn off**
4) **if R = 0, then the lamp should stay off**



delayed switch



Going Beyond Combinational Logic (continued)



Truth Table

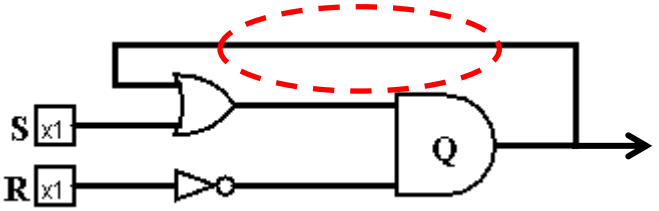
R	S	Q
0	0	?
0	1	1
1	0	0
1	1	?

R	S	Q_{n-1}	Q
0	0		Q_{n-1}
0	1		1
1	0		0
1	1		?

Boolean Function

$$Q = \overline{R}\,\overline{S}\,Q_{n-1} + \overline{R}\,S = \overline{R}\,(\overline{S}Q_{n-1} + S)$$
$$= \overline{R}\,(Q_{n-1} + S)$$

Logic Circuit

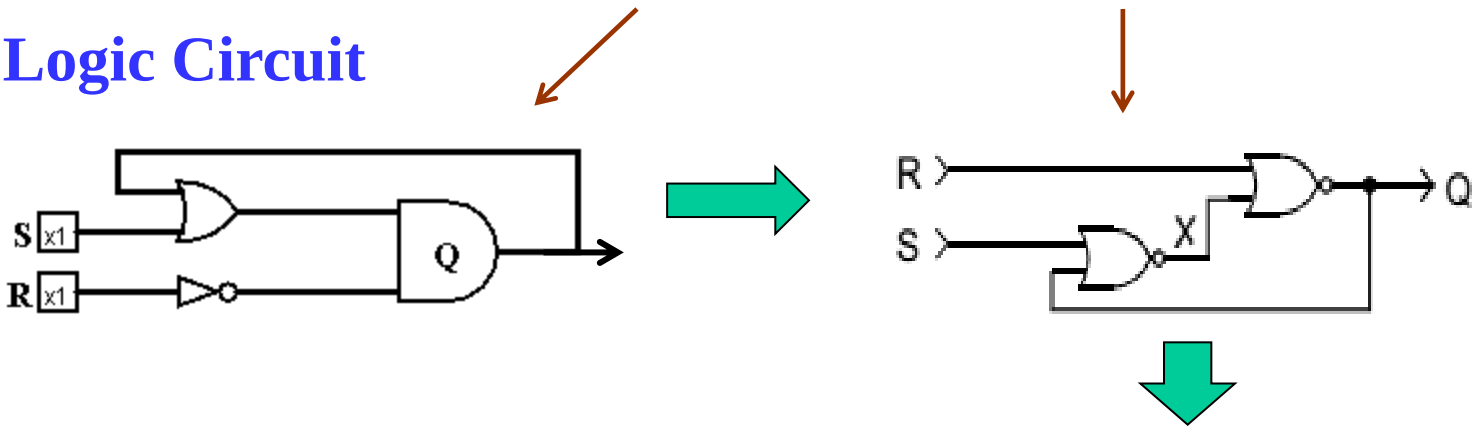


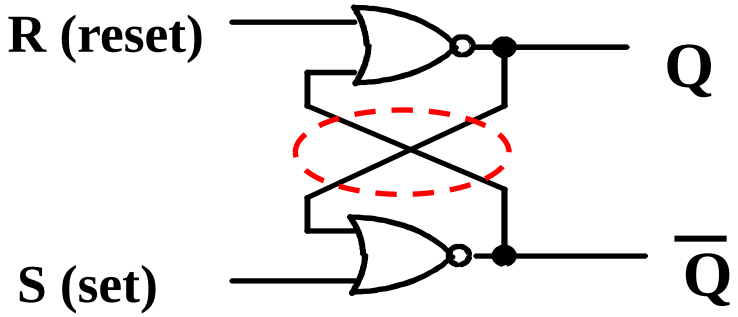
Going Beyond Combinational Logic (continued)

Boolean Function

$$Q = \overline{R}\,\overline{S}\,Q_{n-1} + \overline{R}\,S = \overline{R}\,(Q_{n-1} + S) = \overline{R + (\overline{Q_{n-1} + S})}$$

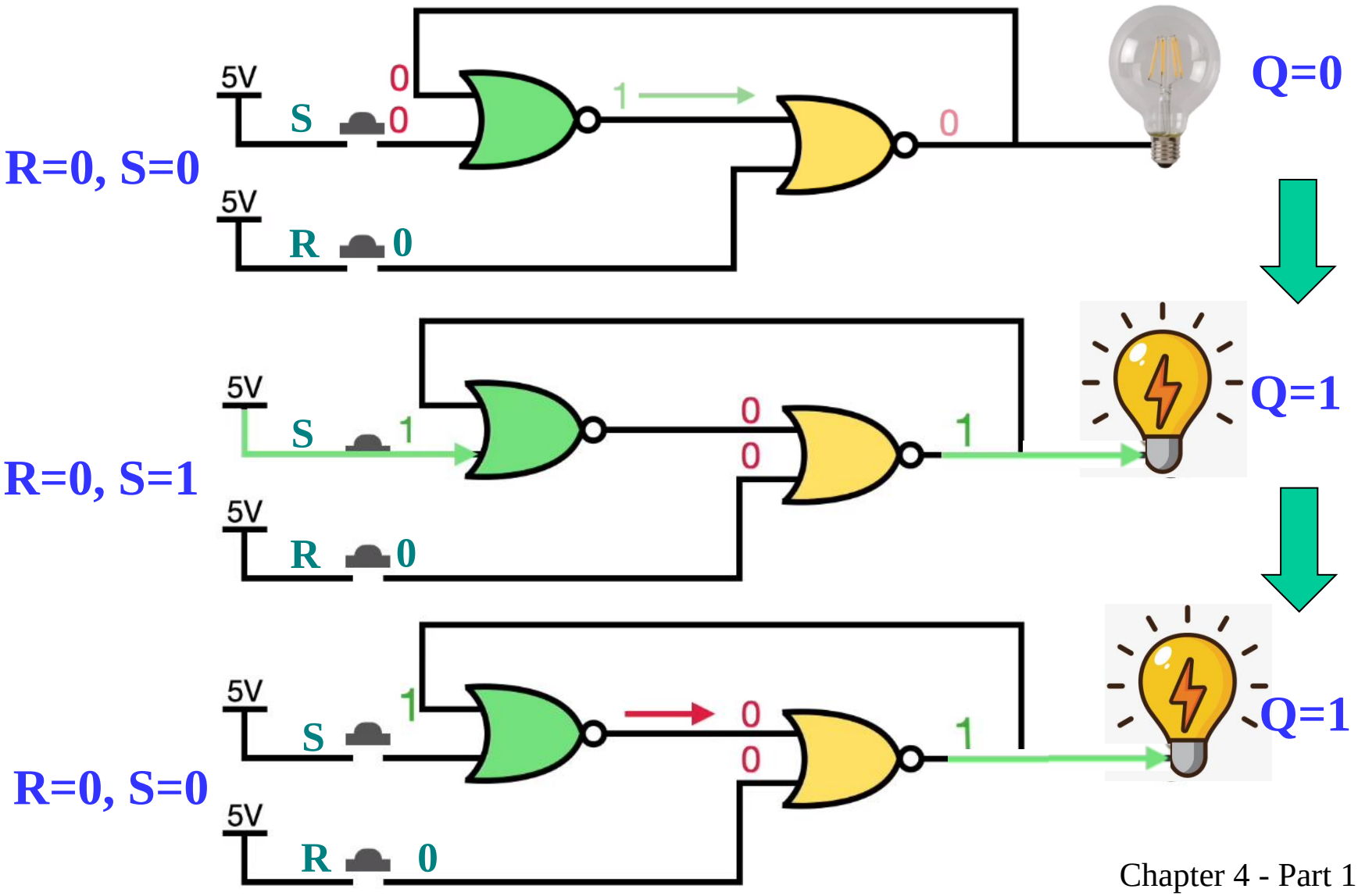
Logic Circuit





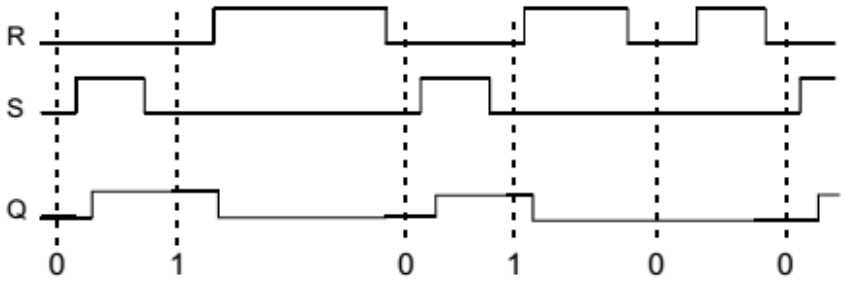
The circuit is an SR Latch that contains two cross-coupled NOR gates.

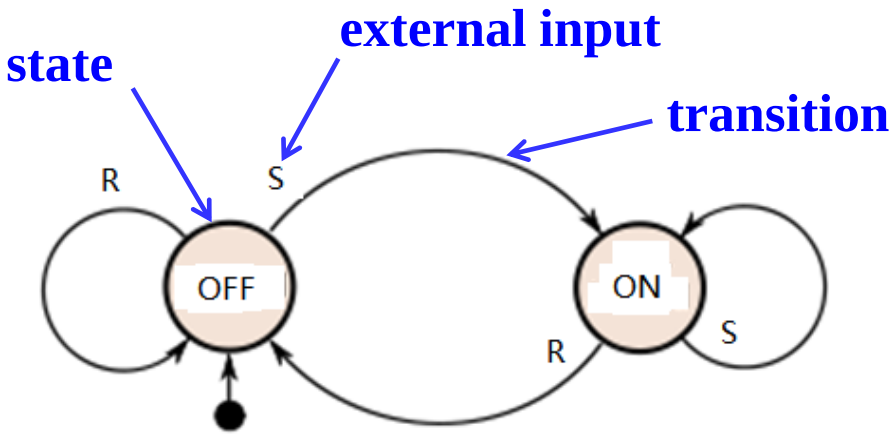
Going Beyond Combinational Logic (continued)



Going Beyond Combinational Logic (continued)

- A circuit whose output depends not only on the present input but also on the history of the input is called a **sequential circuit**.
- A sequential circuit can be described by a state diagram, which consists of a finite number of states at any given time. It can change from one state to another in response to external inputs.

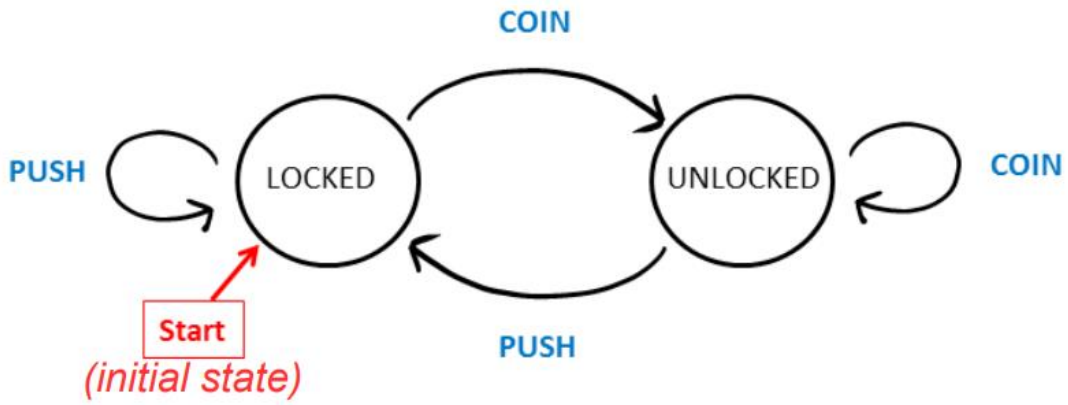


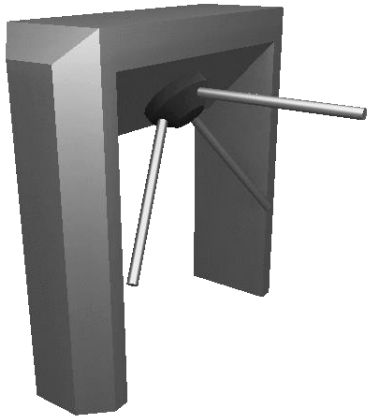


state diagram for a delayed switch

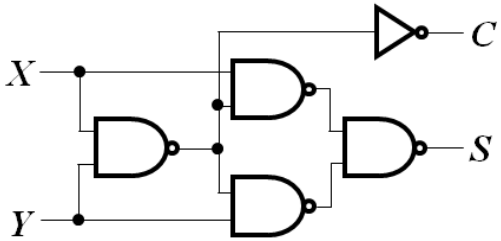
Going Beyond Combinational Logic (continued)

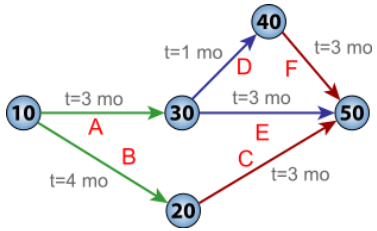
- **Another example: turnstile**



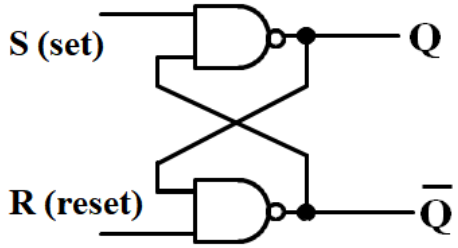


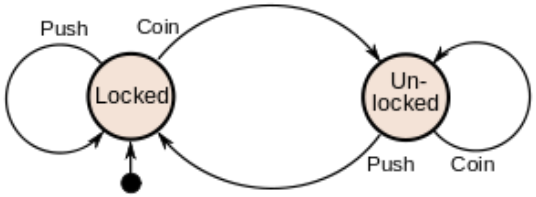
- **Combinational circuit vs. Sequential circuit**





Combinational circuit

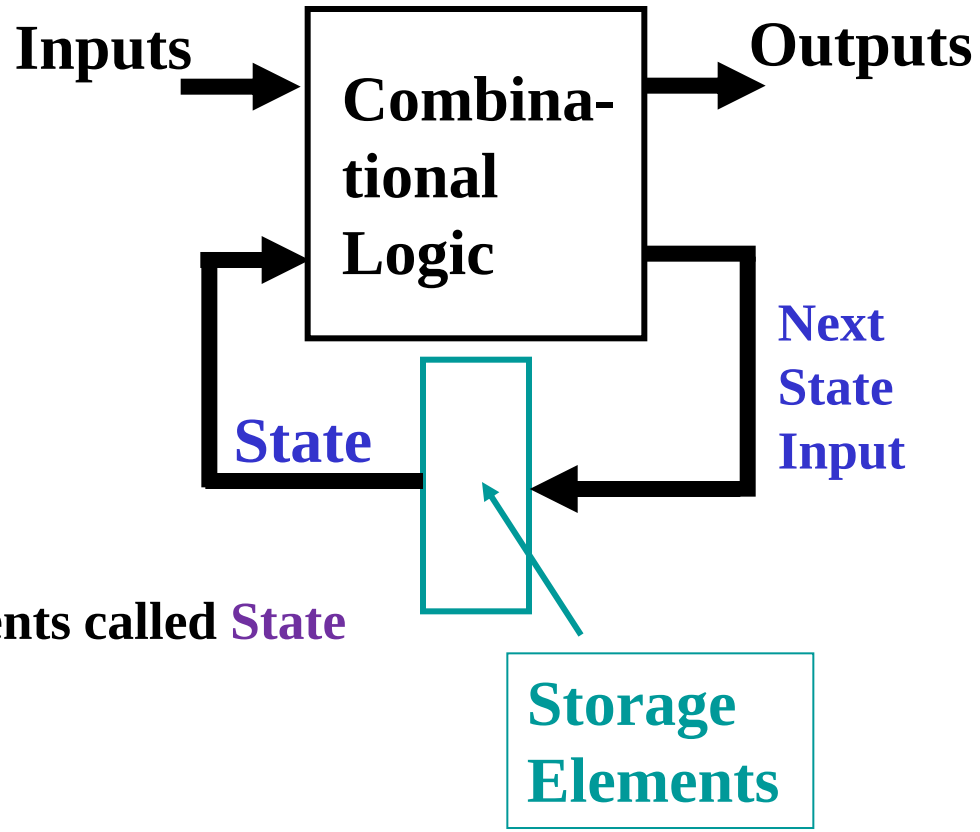




Sequential circuit

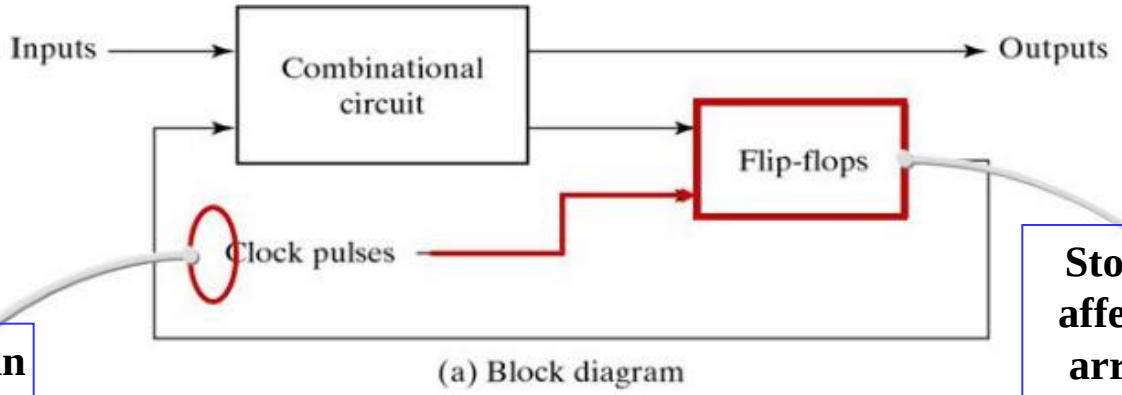
Introduction to Sequential Circuits

- **A sequential circuit contains:**
  - **Storage elements: Latches or Flip-Flops**
  - **Combinational Logic:**
    - **Inputs:**
      - **signals from the outside**
      - **signals from storage elements called State or Present State.**
    - **Outputs:**
      - **signals to the outside**
      - **signals to storage elements called Next State**



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)**



(a) Block diagram

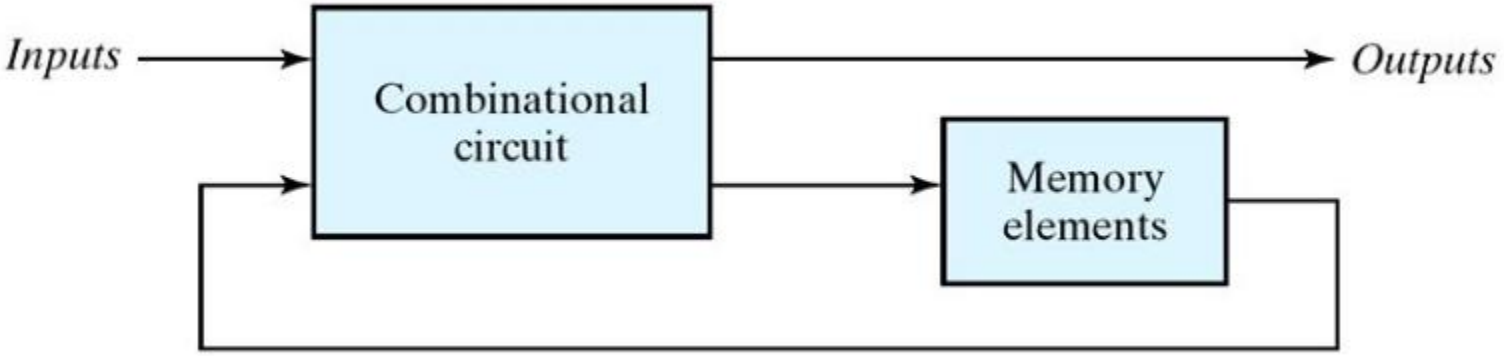
(b) Timing diagram of clock pulses

Use clock pluses in the inputs of storage elements

Storage elements are affected only with the arrival of each pulse

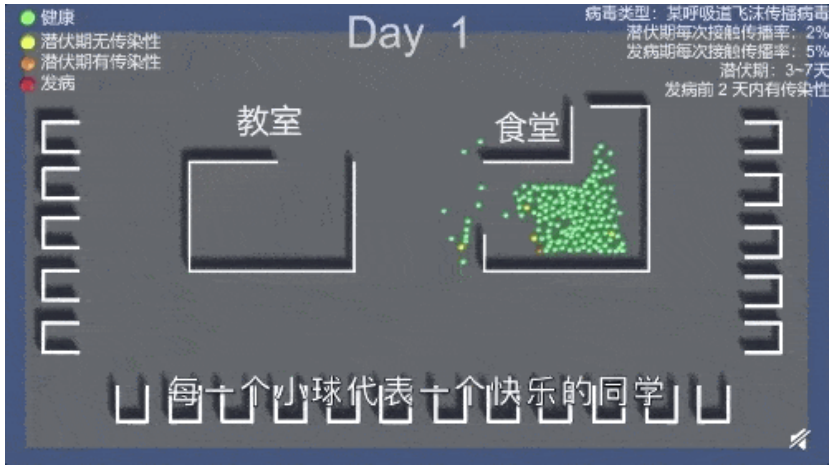
Types of Sequential Circuits (continued)

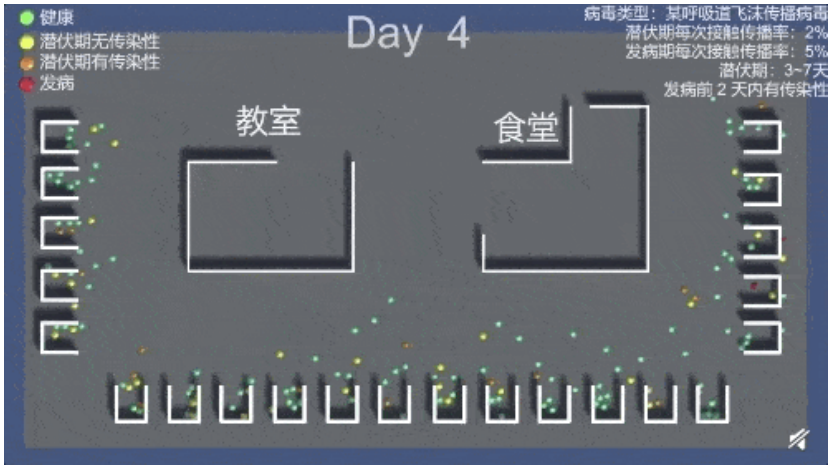
- **Asynchronous**
  - **Behavior defined from knowledge of inputs at any instant of time and the order in continuous time in which inputs change**
  - **If clock just regarded as another input, all circuits are asynchronous!**
  - **Nevertheless, the synchronous abstraction makes complex designs tractable!**



Discrete Event Simulation (1/2)

- **In order to understand the time behavior of a sequential circuit we use discrete event simulation to analyze system dynamics.**

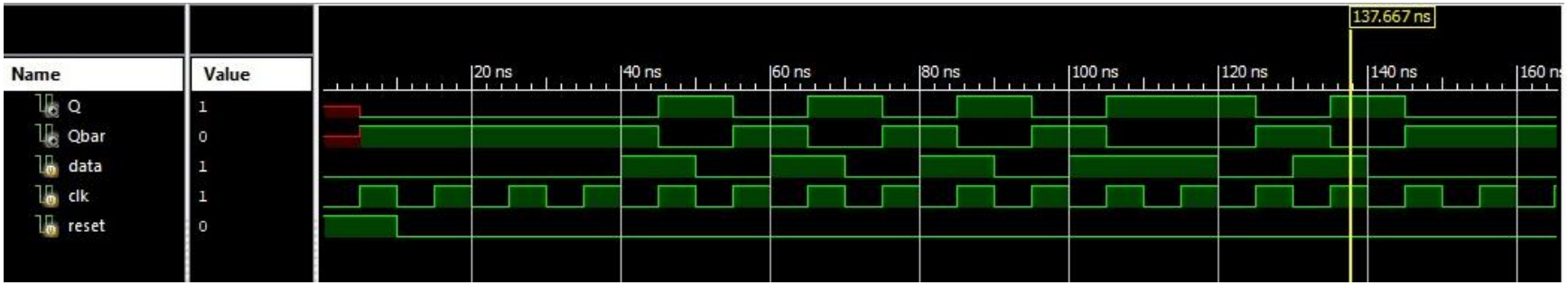




Day 1——Day 3 → Day 4——Day 10

Example: simulation of spread of COVID-19 in the campus

Discrete Event Simulation (2/2)

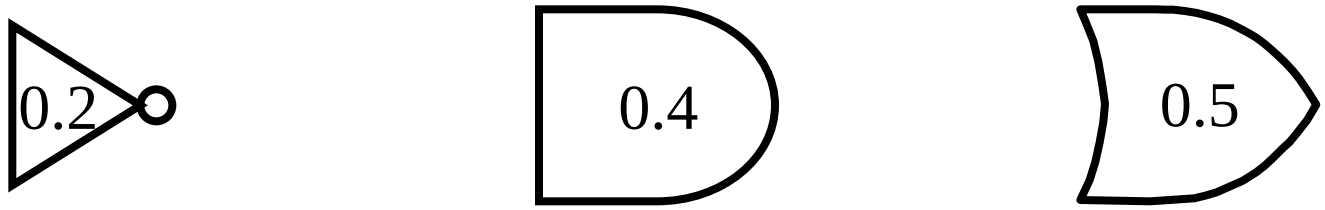


Example: Output waveform of D Flip Flop with reset input

- **Rules of simulation for digital circuits:**
  - **Any change in input values is evaluated to see if it causes a change in output value**
  - **Gates modeled as an ideal circuit with a fixed gate delay**
  - **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**

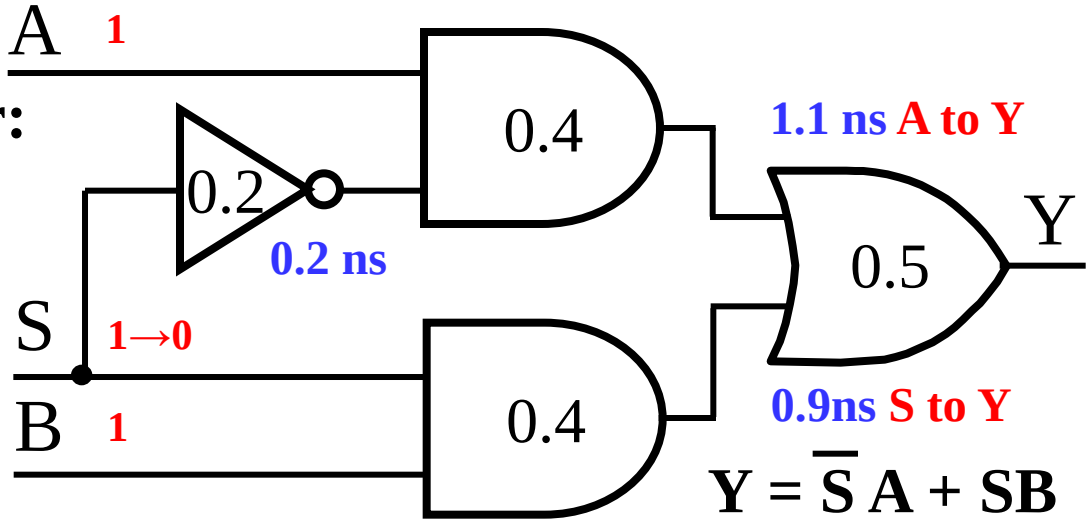
Gate Delay Models

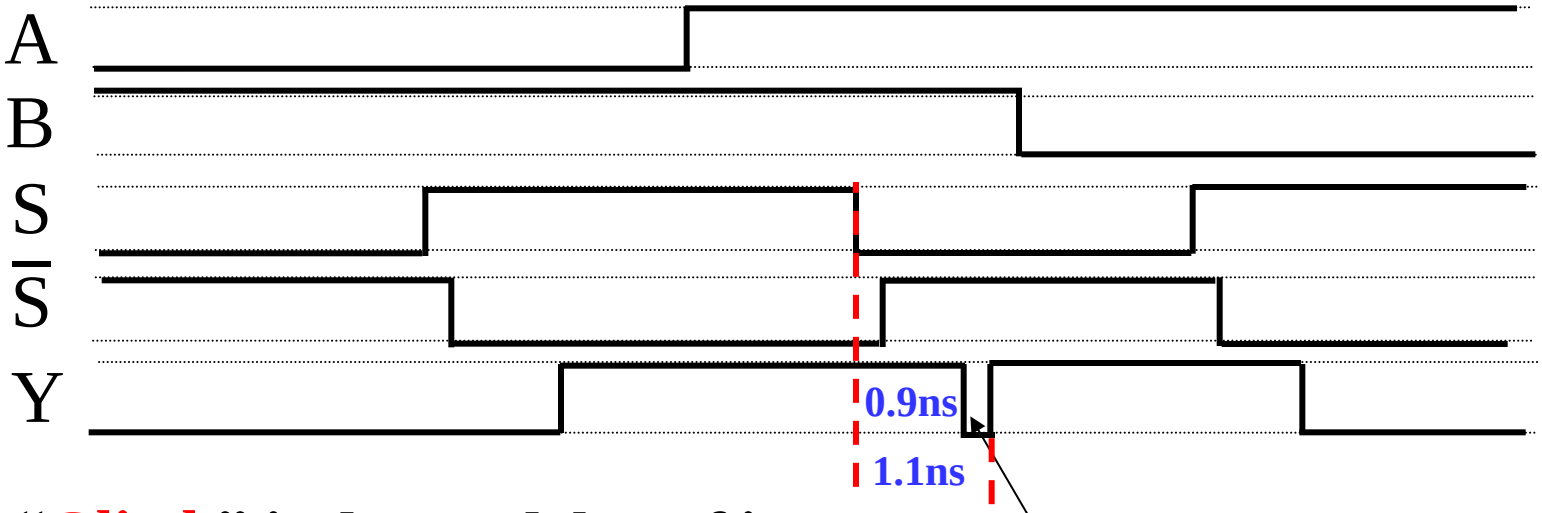
- **Suppose gates with delay *n* ns are represented for *n* = 0.2 ns, *n* = 0.4 ns, *n* = 0.5 ns, respectively:**



Circuit Delay Model

- **Consider a simple 2-input multiplexer:**
- **With function:**
  - **Y = A for S = 0**
  - **Y = B for S = 1**

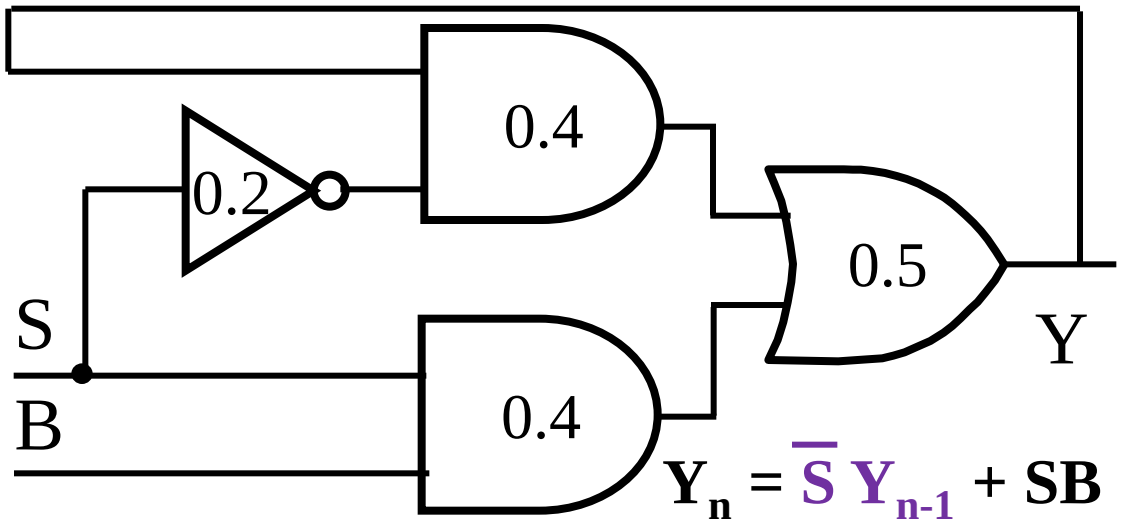




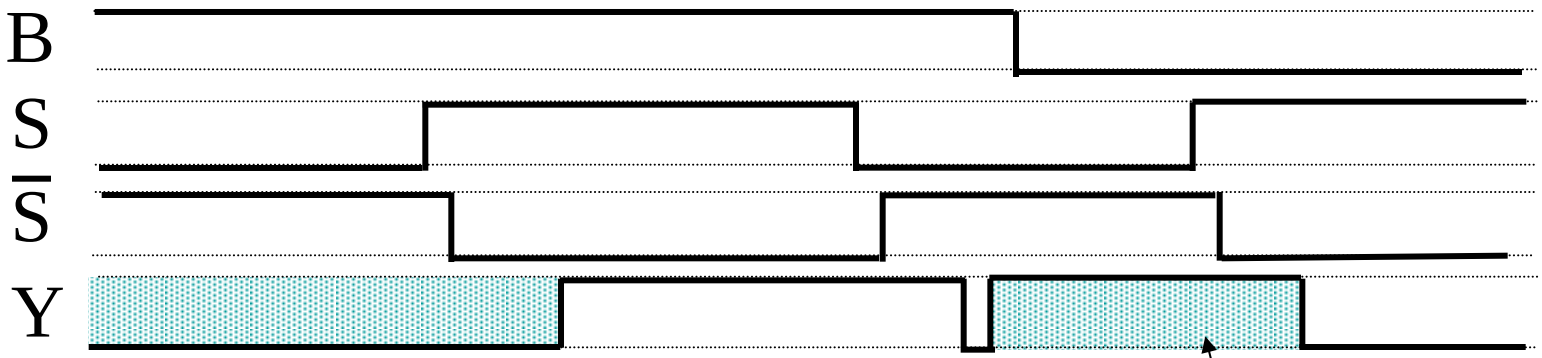
- **"Glitch" is due to delay of inverter**

Storing State

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



$$Y_n = \overline{S}\, Y_{n-1} + SB$$



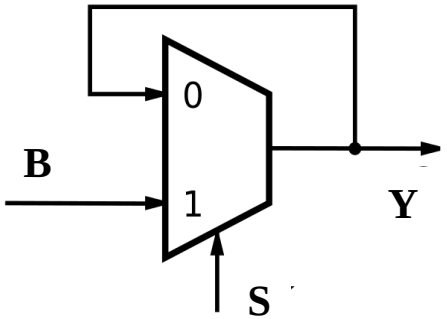
- **The simple combinational circuit has now become a sequential circuit because its output is a function of a time sequence of input signals!**

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.**

$$Y_n = \overline{S}\, Y_{n-1} + SB$$



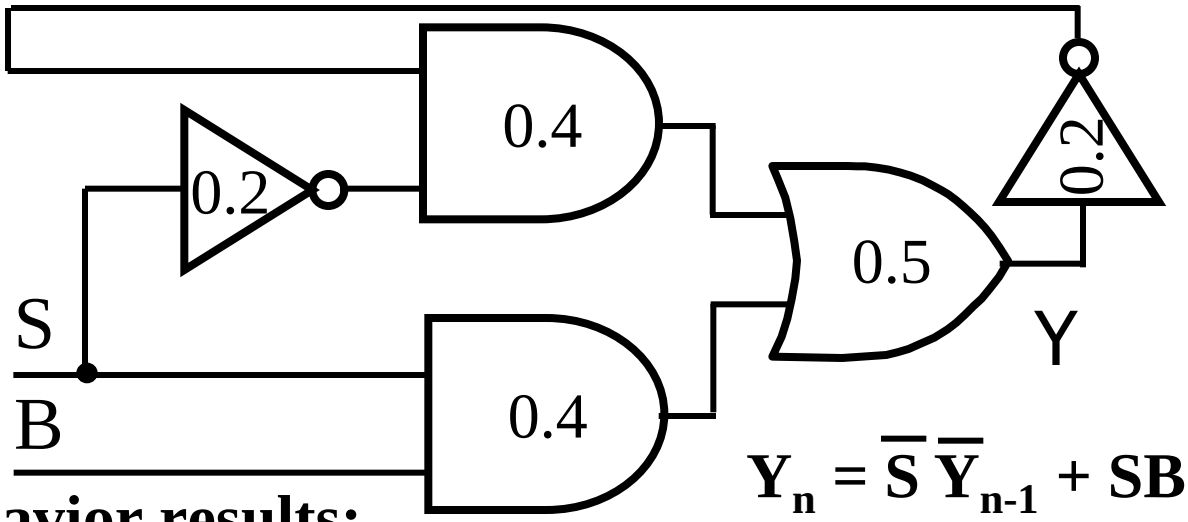
Time

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 represents the state of the circuit, not just an output.**

Storing State (Continued)

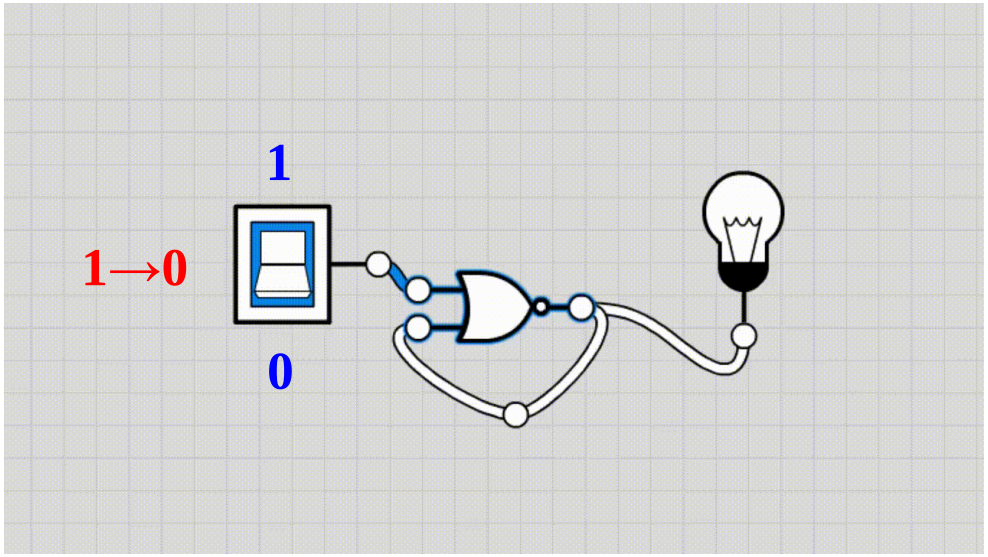
- **Suppose we place an inverter in the "feedback path."**



- **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

Oscillation Error

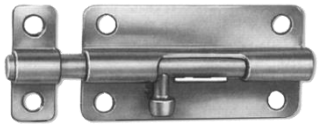


- **Oscillation errors are common problems when designing digital circuits with feedback loops (if you are not designing an oscillator).**
- **Digital circuits will become unstable when oscillations occur.**

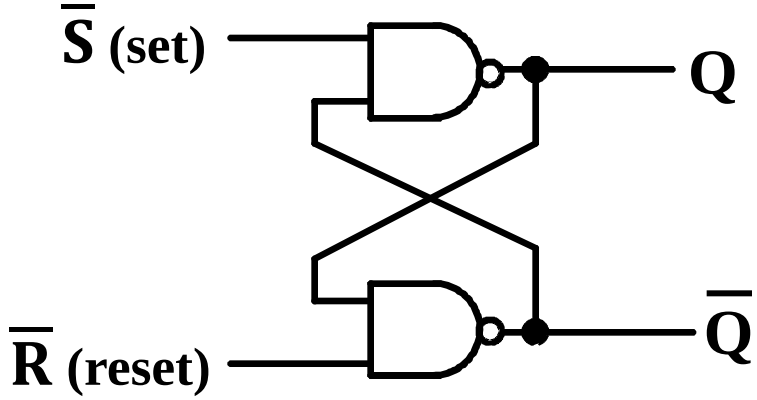
Latches

- **Many components to store historical state**
  - **Capacitors, Inductors, Memories, etc.**
  - **Latches, Triggers**
- **Satisfy the following three conditions can be referred to as latches**
  - **There are two stable states, i.e., "0", "1";**
  - **Long term maintaining a given stable state;**
  - **Under certain conditions, it can change state, such as setting “1” or resetting “0”.**
- **The simplest latches are RS latch and D latch**

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



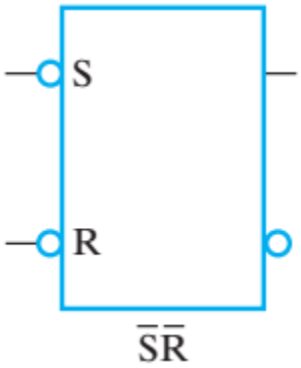
- "**Cross-Coupling**" two NAND gates gives the $\bar{S}$ -$\bar{R}$ Latch:
- Which has the time sequence behavior:
- S = 0, R = 0 is forbidden as input pattern



Time

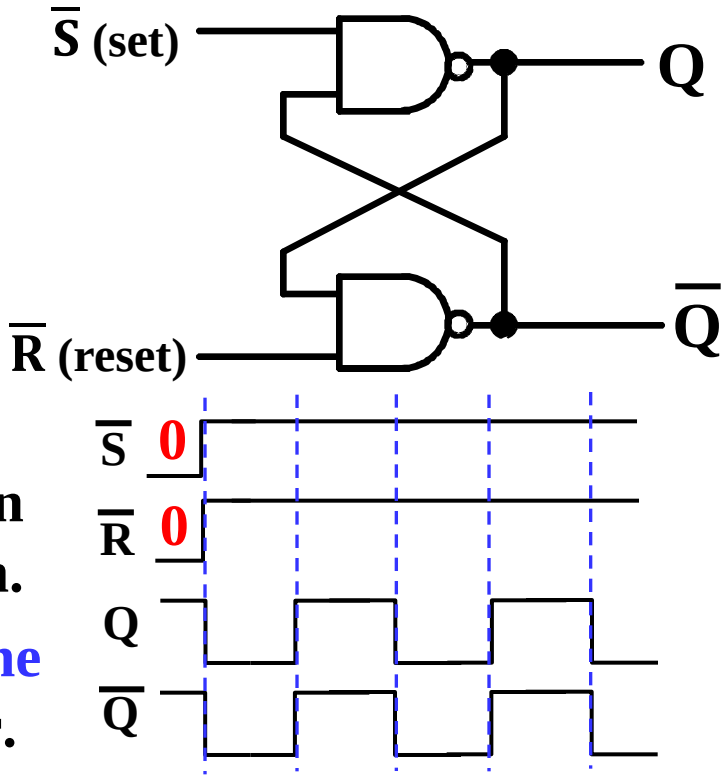
R	S	Q	$\bar{Q}$	Comment
1	1	?	?	Stored state unknown
1	0	1	0	"Set" Q to 1
1	1	1	0	Now Q "remembers" 1
0	1	0	1	"Reset" Q to 0
1	1	0	1	Now Q "remembers" 0
0	0	1	1	Both go high
1	1	?	?	Unstable!

graphics symbol for $\bar{S}\bar{R}$ Latch



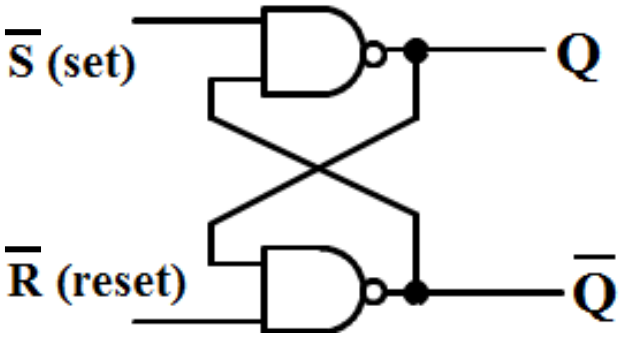
Unstable Latch Behavior (Oscillation)

- **Why both inputs of the latch to 0 are forbidden?**
  - **If both gates have the same delay then they will both output a 0 at the same time. Feeding 0 back to the input will produce 1, again at exactly the same time, which again will produce a 0, and so on and on.**
  - **This oscillating behavior, called the critical race, will continue forever.**

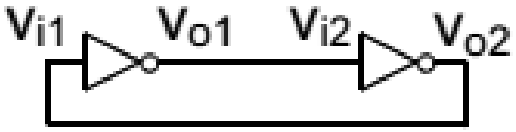


  - **If the two gates do not have the same delay then the latch will go into one state or the other. However, we do not know which state the latch will go into. Thus, the latch's next state is undefined.**

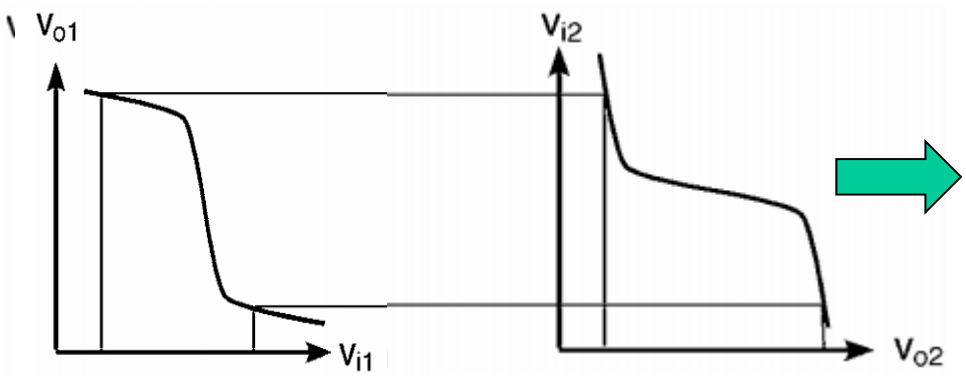
Unstable Latch Behavior (Metastable State)

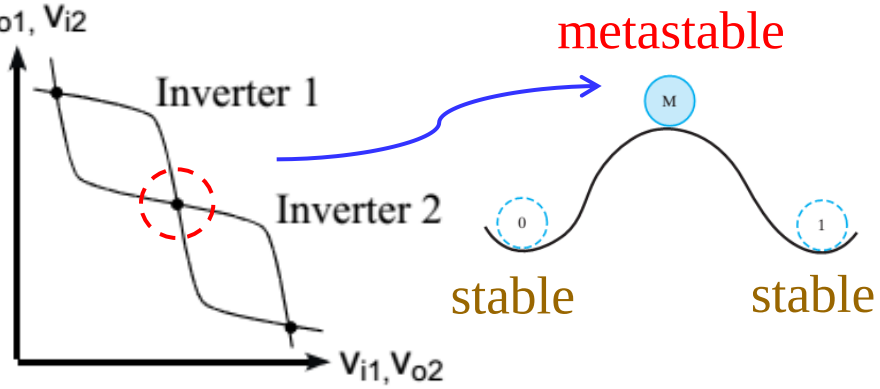


- **Equivalent circuit for the latch when R = S = 1**



- **Consider the transfer characteristics of two inverters**



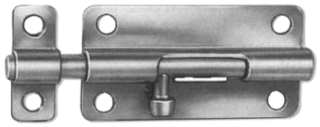


- **The dot in the middle represents a metastable state.**
- **Small changes in any of the signals are amplified and the circuit leaves the metastable state.**

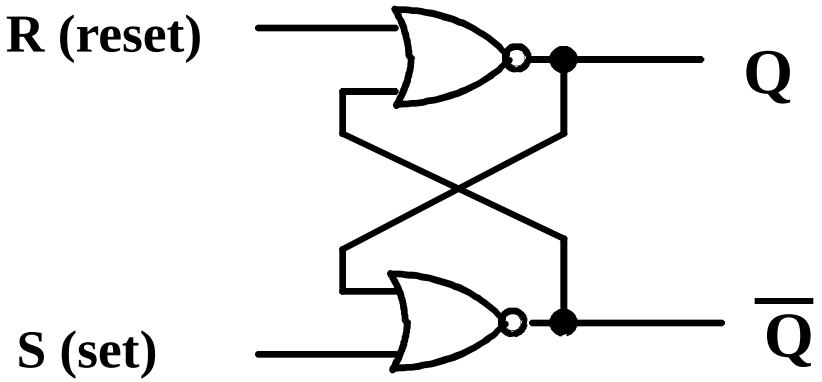
Avoiding Unstable Behavior of Latches

- **Since both the oscillation and the metastable state are undesirable behaviors, we should try to avoid them.**
- **Do not change R and S from 0 to 1 at the same time.**
  - **This is necessary to avoid the oscillation behavior.**
  - **One way to guarantee that is to never allow them to both be 0 at the same time.**
- **Once you change an input, do not change it again until the circuit has had time to complete all its signal transitions and reach a stable state.**
  - **This is necessary to avoid the metastable behavior.**

Basic (NOR) S – R Latch



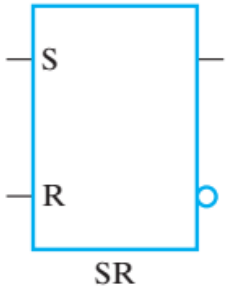
- Cross-coupling two NOR gates gives the S – R Latch:
- Which has the time sequence behavior:
- S = 1, R = 1 is forbidden as input pattern



Time

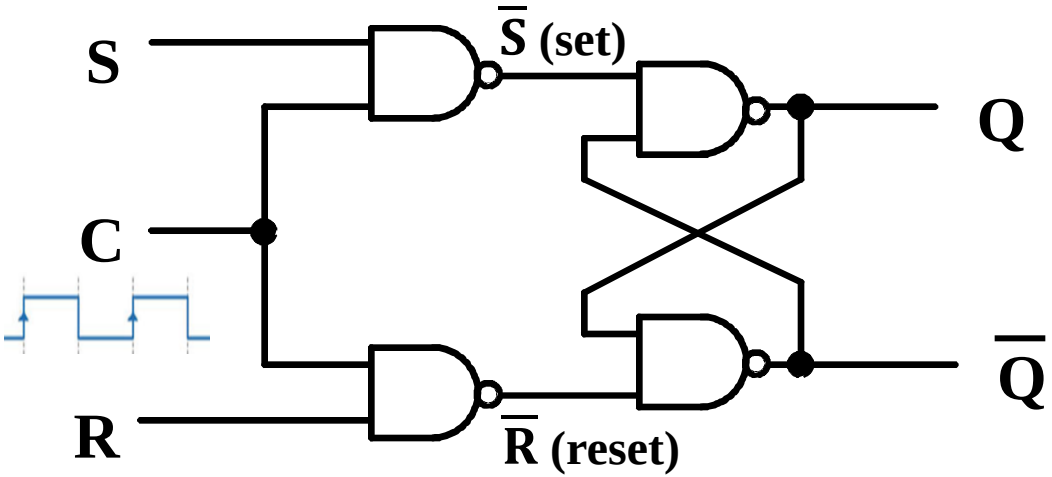
R	S	Q	$\overline{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	"Set" Q to 1
0	0	1	0	Now Q "remembers" 1
1	0	0	1	"Reset" Q to 0
0	0	0	1	Now Q "remembers" 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

graphics symbol for SR Latch

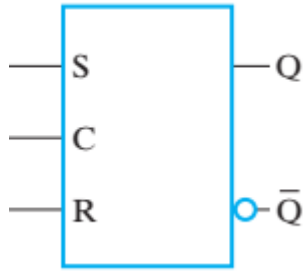


Clocked S – R Latch: Synchronous Circuit

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



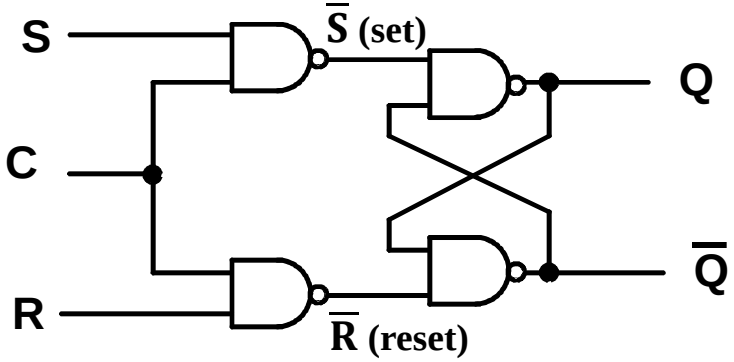
- Has a time sequence behavior similar to the basic S-R latch except that the S and R inputs are only observed when the line C is high.
- C means "control" or "clock".
- From asynchronous to synchronous
  - Asynchronous Sequential Circuits: Basic $\overline{S}$ – $\overline{R}$ Latch
  - Synchronous Sequential Circuits: Clocked S - R Latch



SR Latch with Control Input

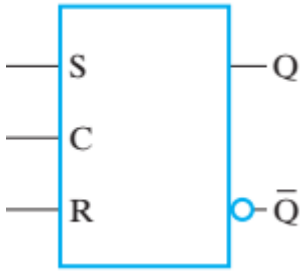
Clocked S - R Latch (continued)

- **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

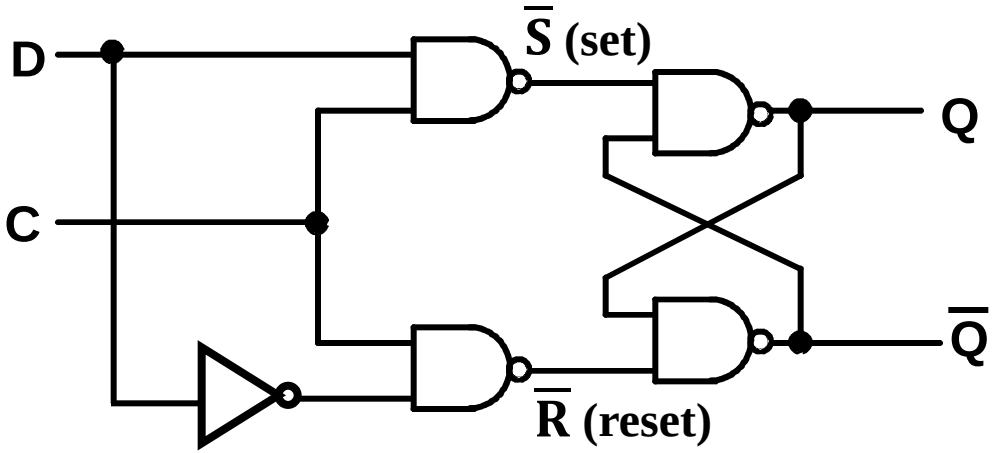
- **The table describes what happens after the clock [at time (t+1)] based on:**
  - **current inputs (S,R,C) and**
  - **current state Q(t).**



SR Latch with Control Input

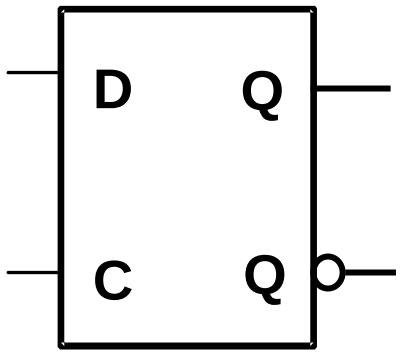
D Latch

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



C	D	Q(t + 1)
0	X	No change
1	0	0: Clear Q
1	1	1: Set Q

The graphic symbol for a D Latch is:



Flip-Flops

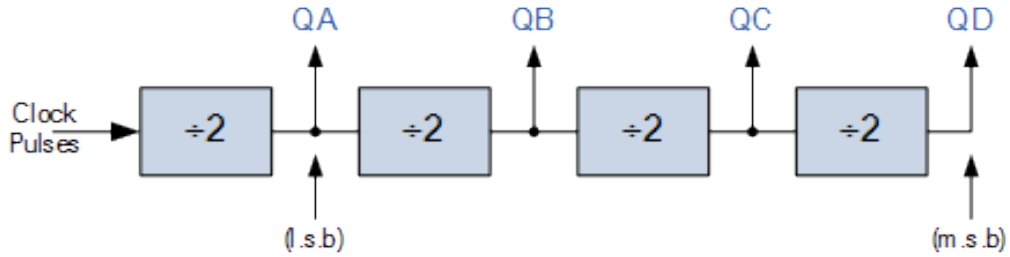
- The latch timing problem
- Master-slave flip-flop
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops

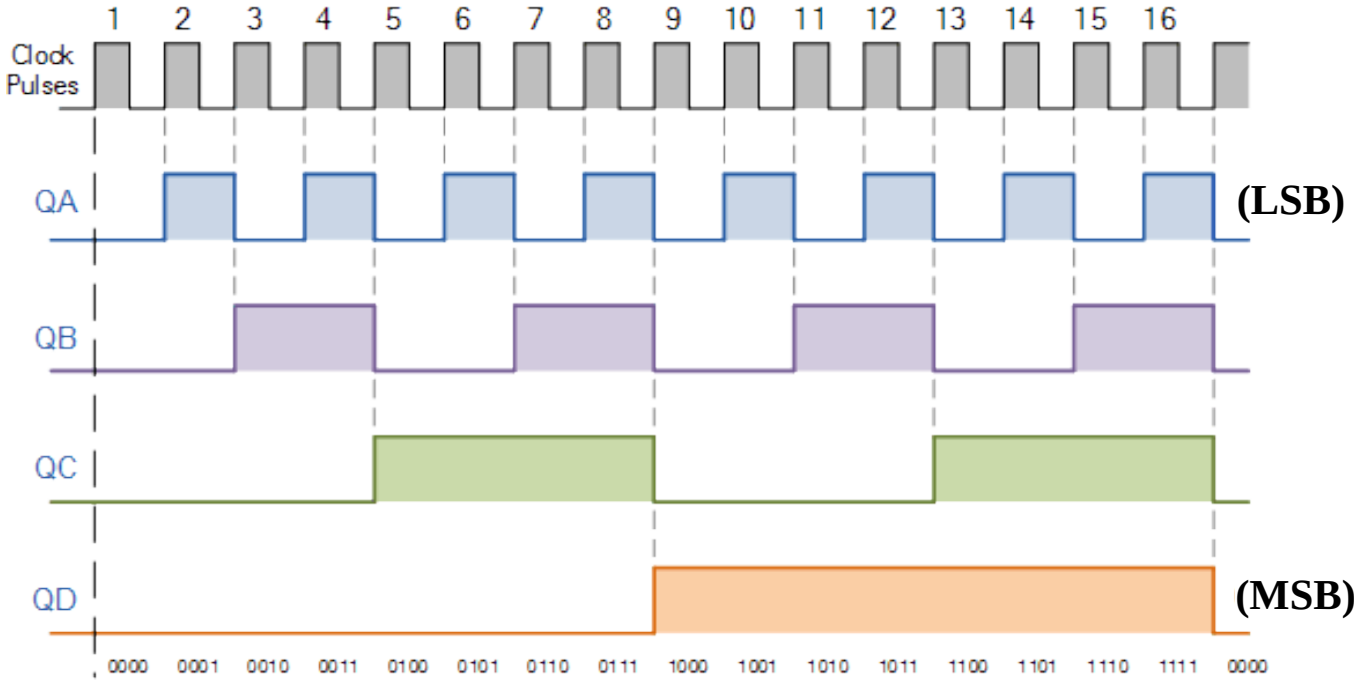
The Latch Timing Problem

- **In a latch circuit, paths may exist through combinational logic:**
  - **From one storage element to another storage element**
  - **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 has value 1**

The Latch Timing Problem (continued)

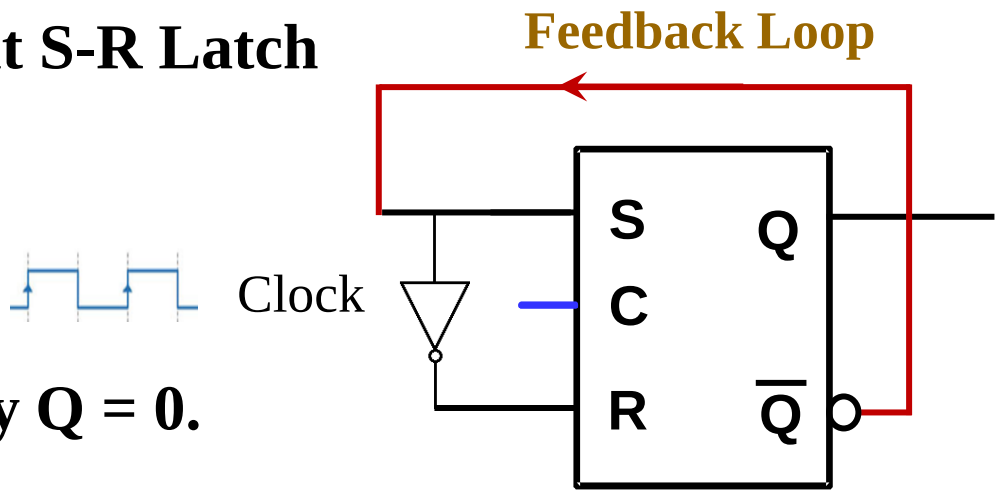
- **Consider the following circuit known as a binary counter**



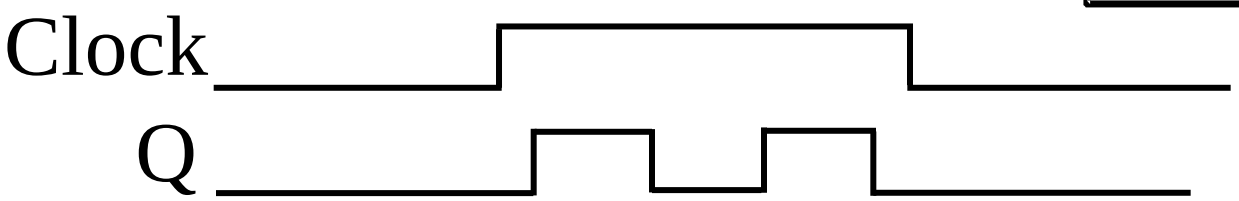


The Latch Timing Problem (continued)

- **Consider the LSB-bit S-R Latch**



- **Suppose that initially Q = 0.**



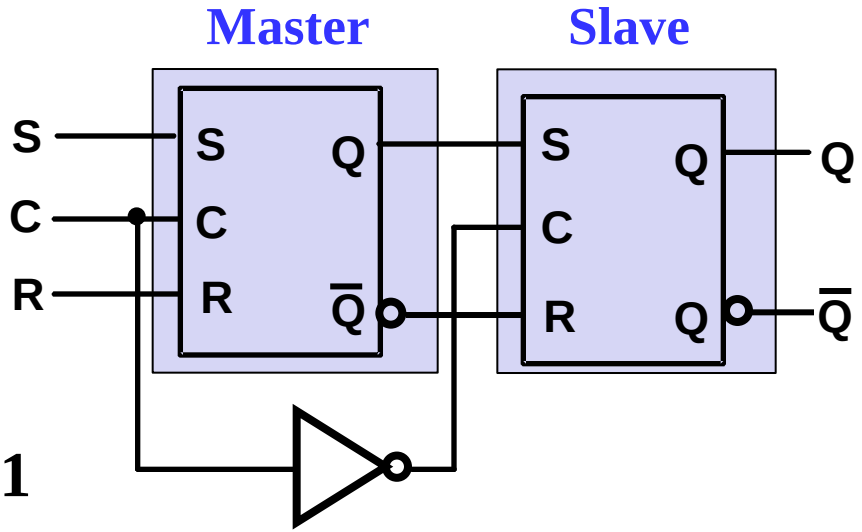
- **As long as C = 1, the value of Q continues to change!**
- **The changes are based on the delay present on the loop through the connection from Q back to Q.**
- **Desired behavior: Q changes only once per clock pulse**
- **This behavior is clearly unacceptable !**

The Latch Timing Problem (continued)

- **A solution to the latch timing problem is to break the closed path from Q to Q within the storage element**
- **The commonly-used, path-breaking solutions replace the S-R latch with:**
  - **Master-slave flip-flops (level-triggered FF)**
  - **Edge-triggered flip-flops**

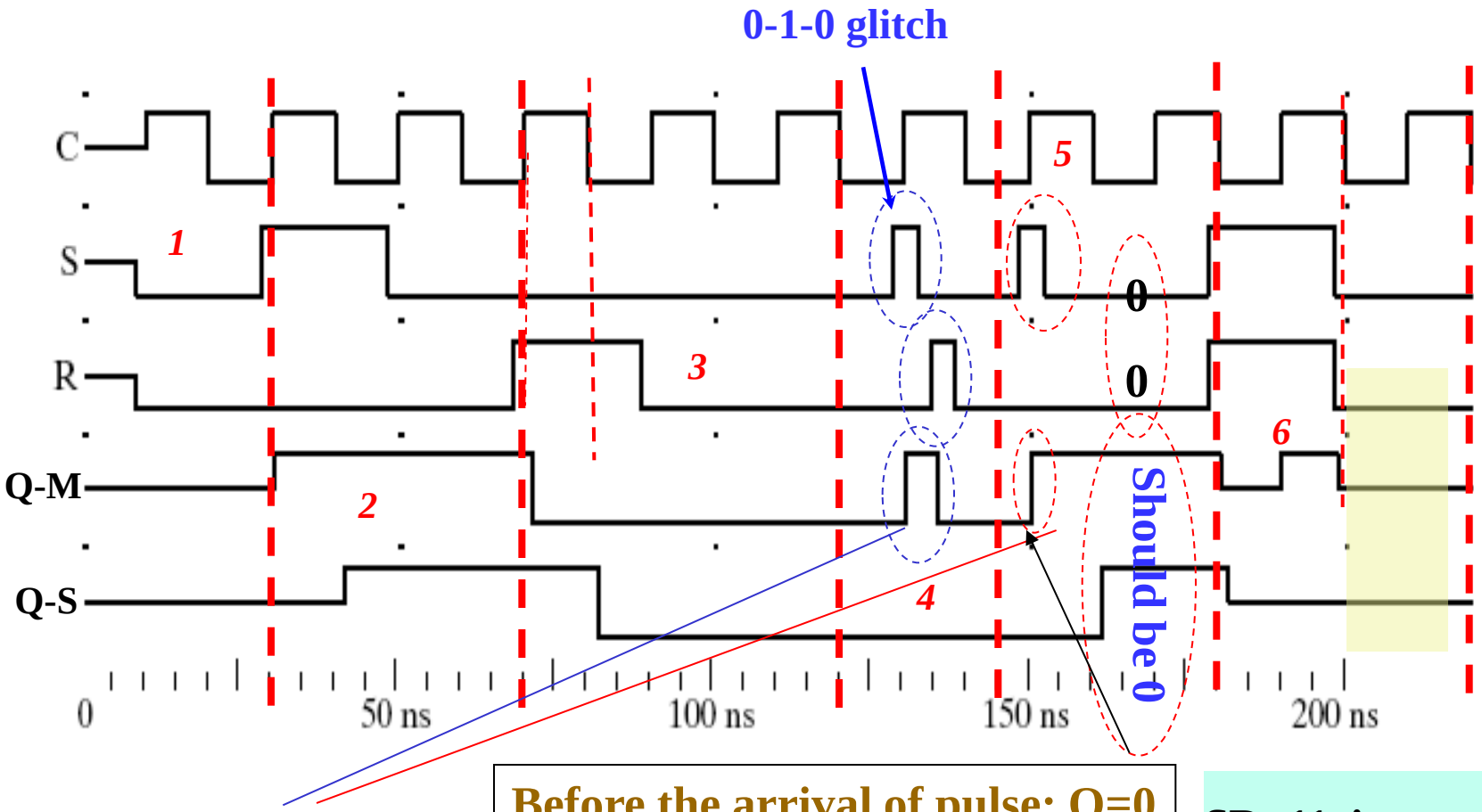
S-R Master-Slave Flip-Flop

- **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**
  - **Master: read input at first half of clock cycle**
  - **Slave: change the output at second half of clock cycle**
- **The path from input to output is broken by the difference in clocking values (alternating clocks).**

S-R Master-Slave Flip-Flop Timing



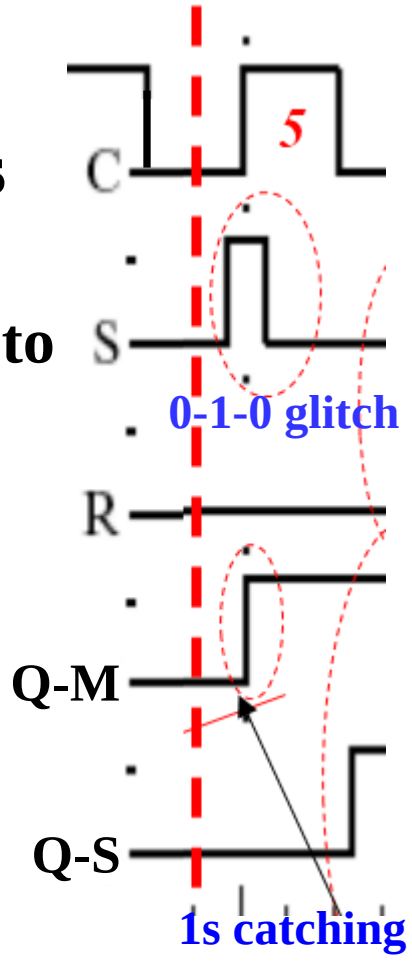
Disposable sampling
(1s catching)

Before the arrival of pulse: Q=0
Before the end of the pulse:
RS=00, Q should keep "0"

SR=11, input state is illegal

S-R Master-Slave Flip-Flop Problem

- **The change in the flip-flop output is delayed by the pulse width which makes the circuit slower.**
- **0-1-0 glitch on either R or S while clock is high will be "caught" by master stage:**
  - **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.**



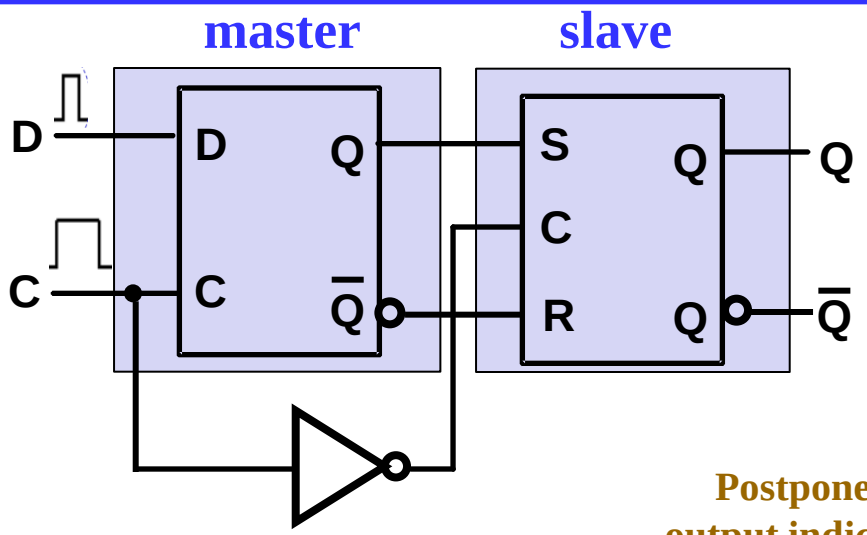
Flip-Flop Solution

(b) Positive-edge response

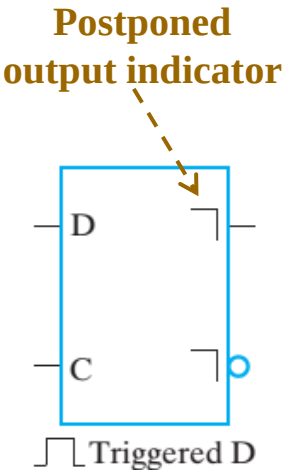
- **Two solutions for avoiding 1s catching:**
  - **reducing inputs: D master-slave FF**
  - **reducing sampling window: Edge-triggered FF**
- **An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a transition of the clock signal**
- **Edge-triggered flip-flops can be built directly at the electronic circuit level**

D Master-Slave Flip-Flop

- A D master-slave flip-flop is triggered by high level or low level.



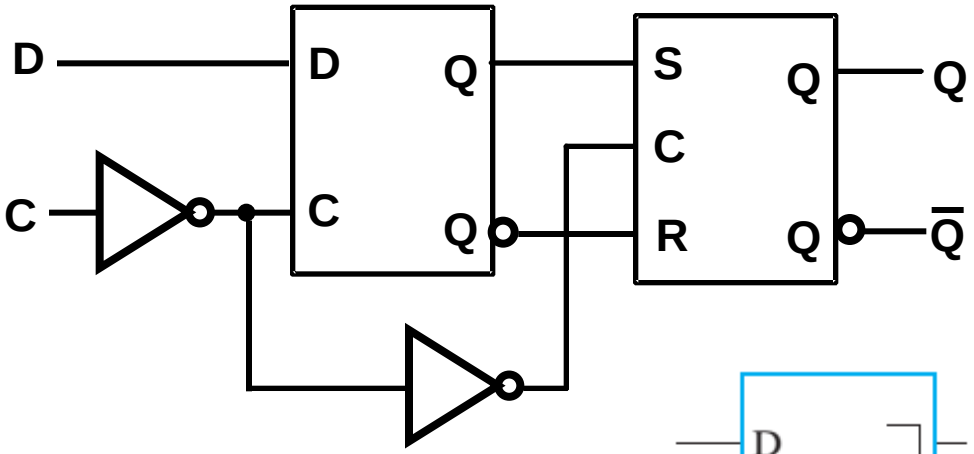
- It can be formed by:
  - Replacing the first S-R latch with a clocked D latch
- The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs.
- The change of the D flip-flop output is associated with the negative edge at the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

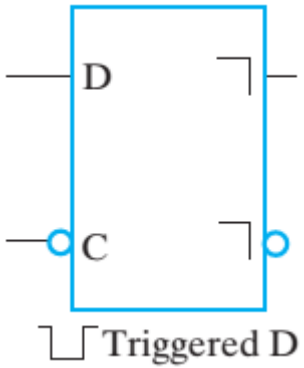


D Master-Slave FF

Positive-Level Triggered D Flip-Flop

- **Formed by adding inverter to clock input**

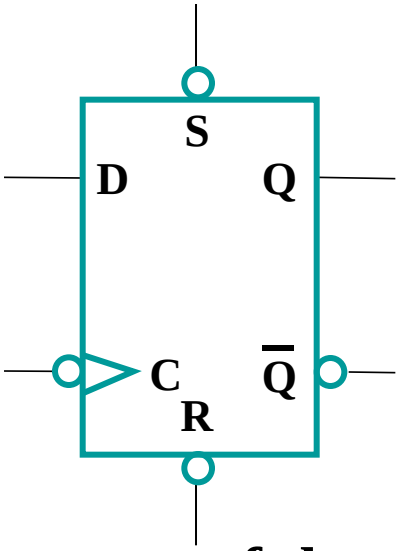




- **Q changes to the value on D applied at the positive clock edge within timing constraints to be specified**
- **Our choice as the standard flip-flop for most sequential circuits**

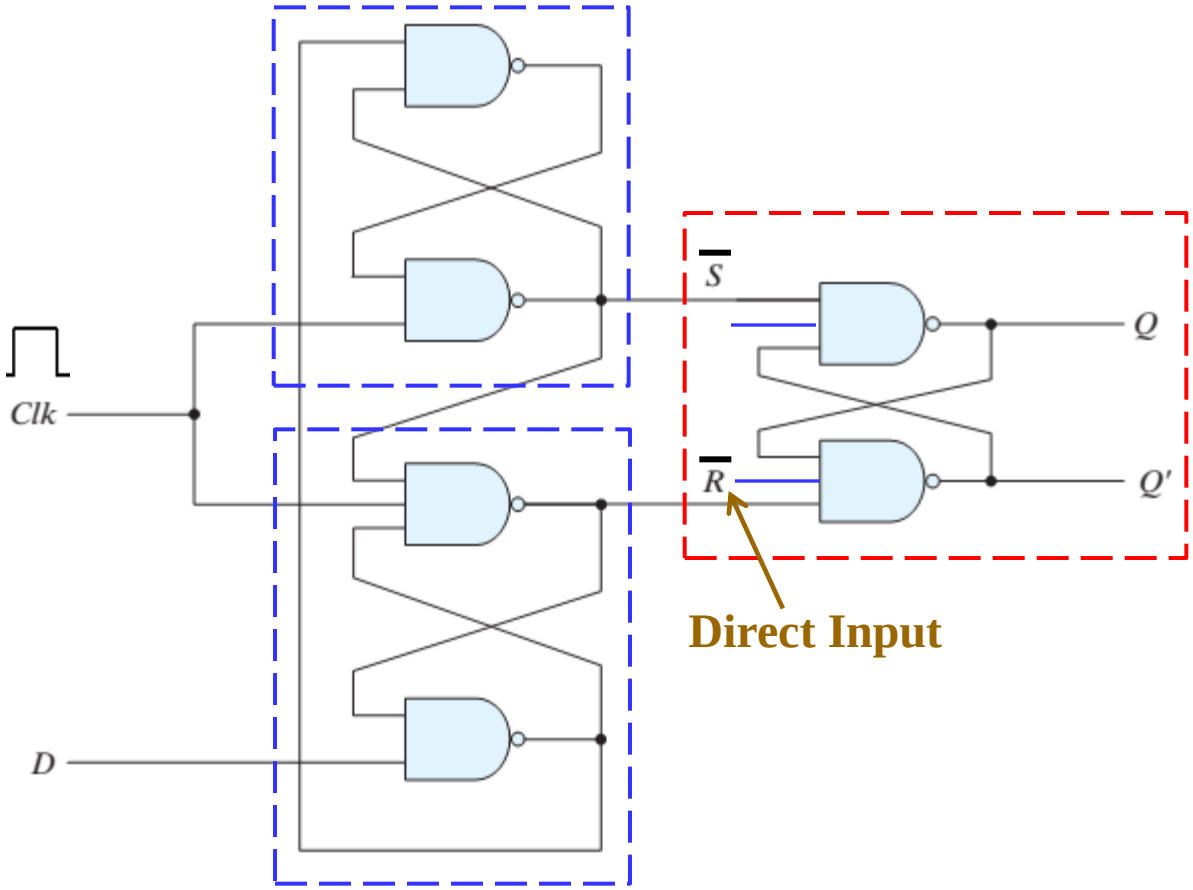
Direct Inputs

- 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
  - When R is 0, resets the flip-flop to the 0 state
  - When S is 0, sets the flip-flop to the 1 state
  - When R and S are both 1, flip-flop works normally
  - State undefined when R and S are both set to 0

Edge-Triggered D Flip-Flop

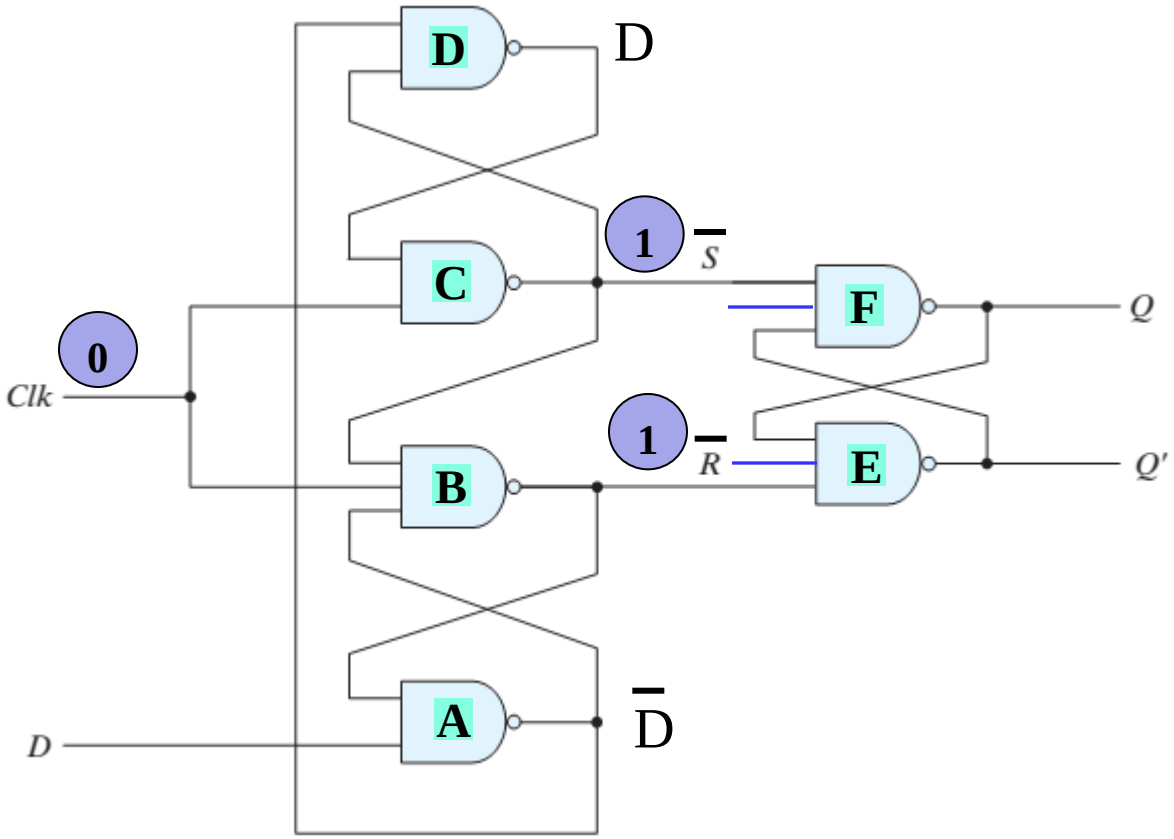


Function Table

Asynchronous		Positive-Edge-Triggered			
$\overline{R}$	$\overline{S}$	Cp	D	Q	$\overline{Q}$
0	1	X	X	0	1
1	0	X	X	1	0
1	1	↑	0	0	1
1	1	↑	1	1	0

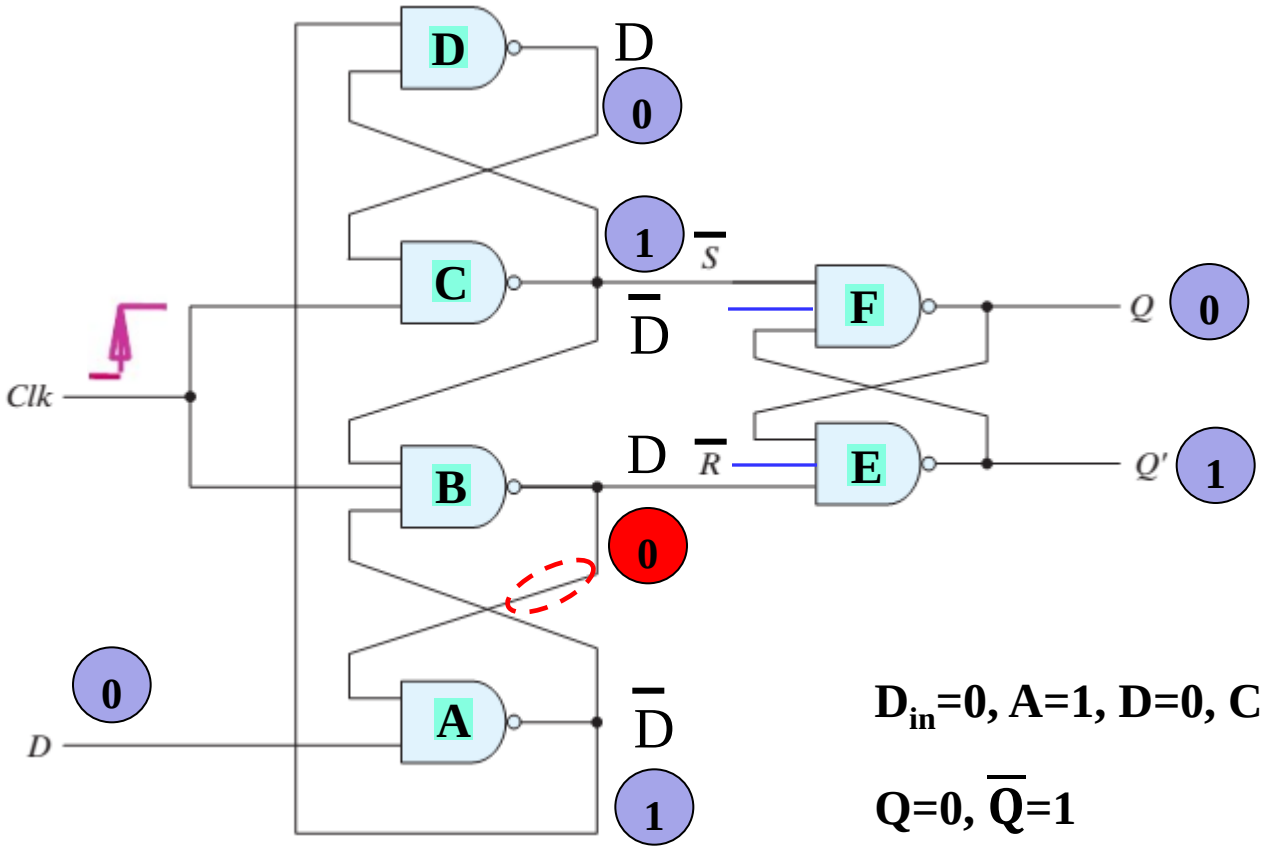
Positive-Edge Triggered D Flip-Flop

Edge-Triggered D Flip-Flop (continued)



Positive-Edge Triggered D Flip-Flop

Edge-Triggered D Flip-Flop (continued)

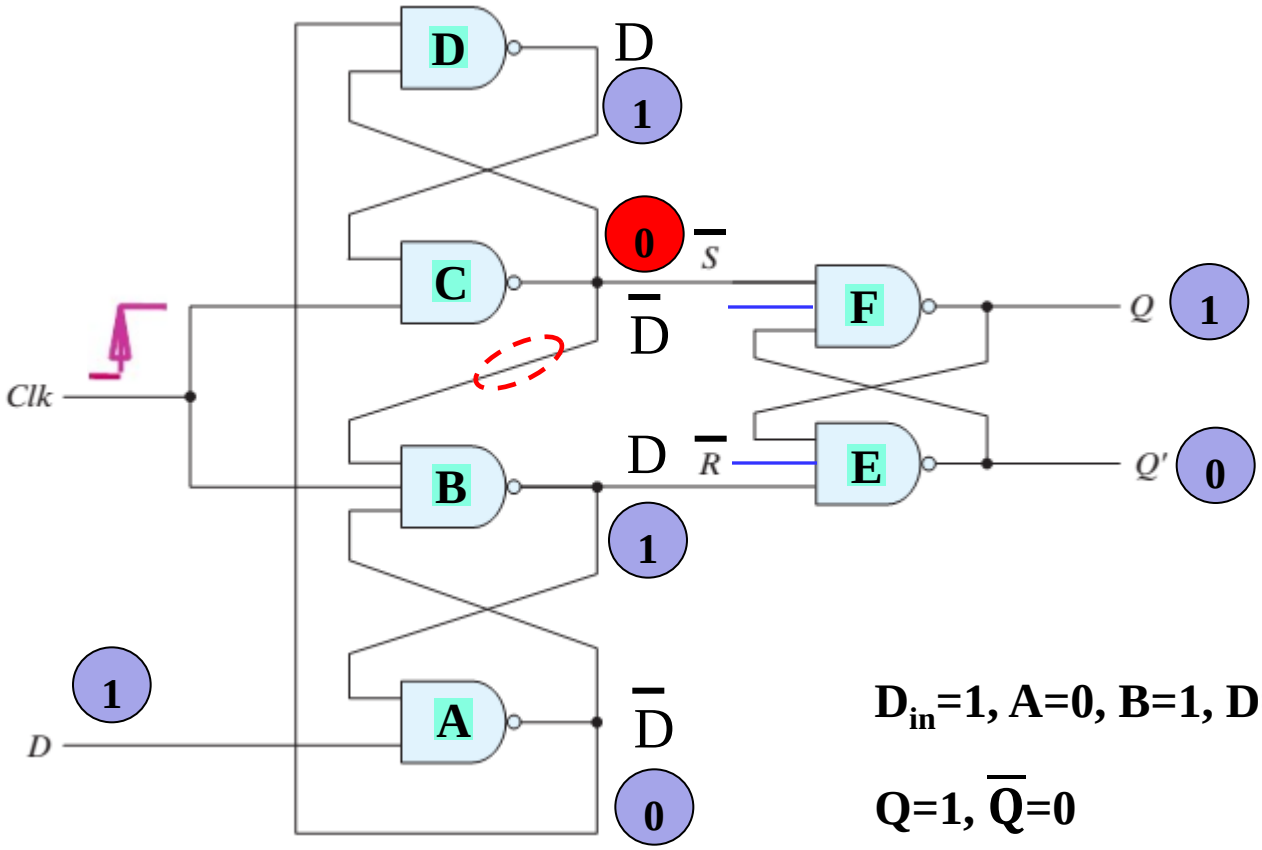


D_{in}=0, A=1, D=0, C=1, **B=0**

Q=0, $\overline{Q}$=1

Positive-Edge Triggered D Flip-Flop

Edge-Triggered D Flip-Flop (continued)

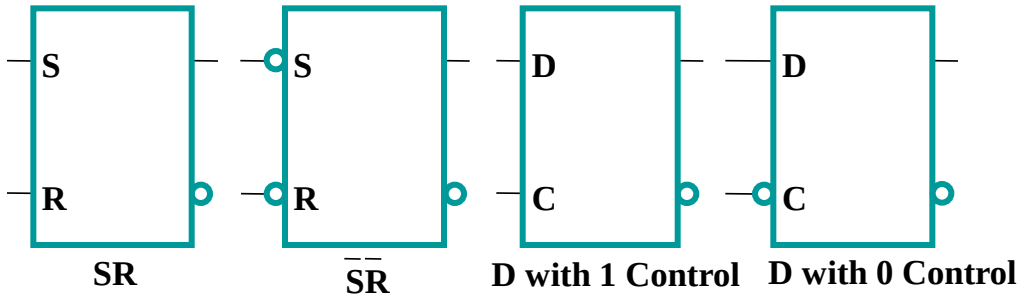


D_{in}=1, A=0, B=1, D=1, C=0

Q=1, $\overline{Q}$=0

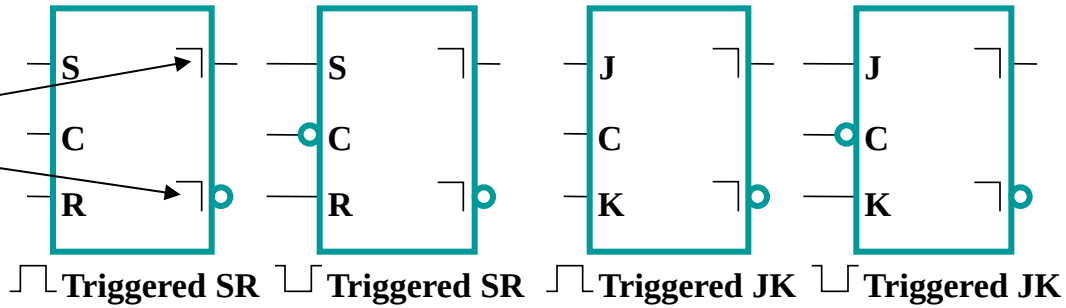
Positive-Edge Triggered D Flip-Flop

Standard Symbols for Storage Elements



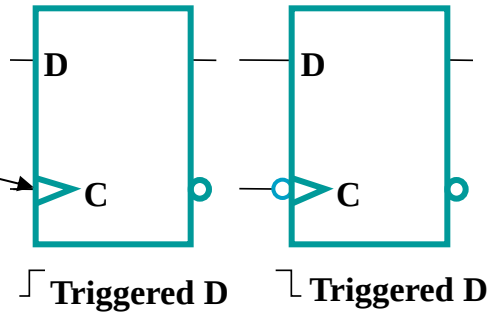
(a) Latches

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



(b) Master-Slave Flip-Flops

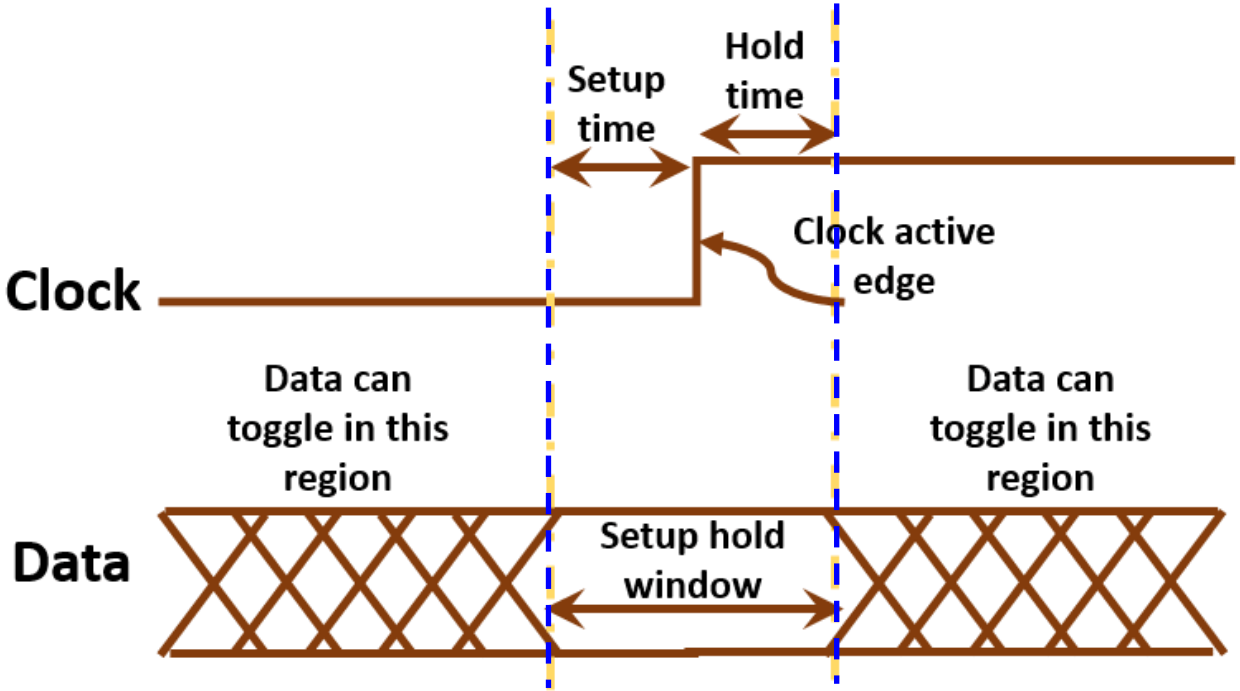
- **Edge-Triggered: Dynamic indicator**



(c) Edge-Triggered Flip-Flops

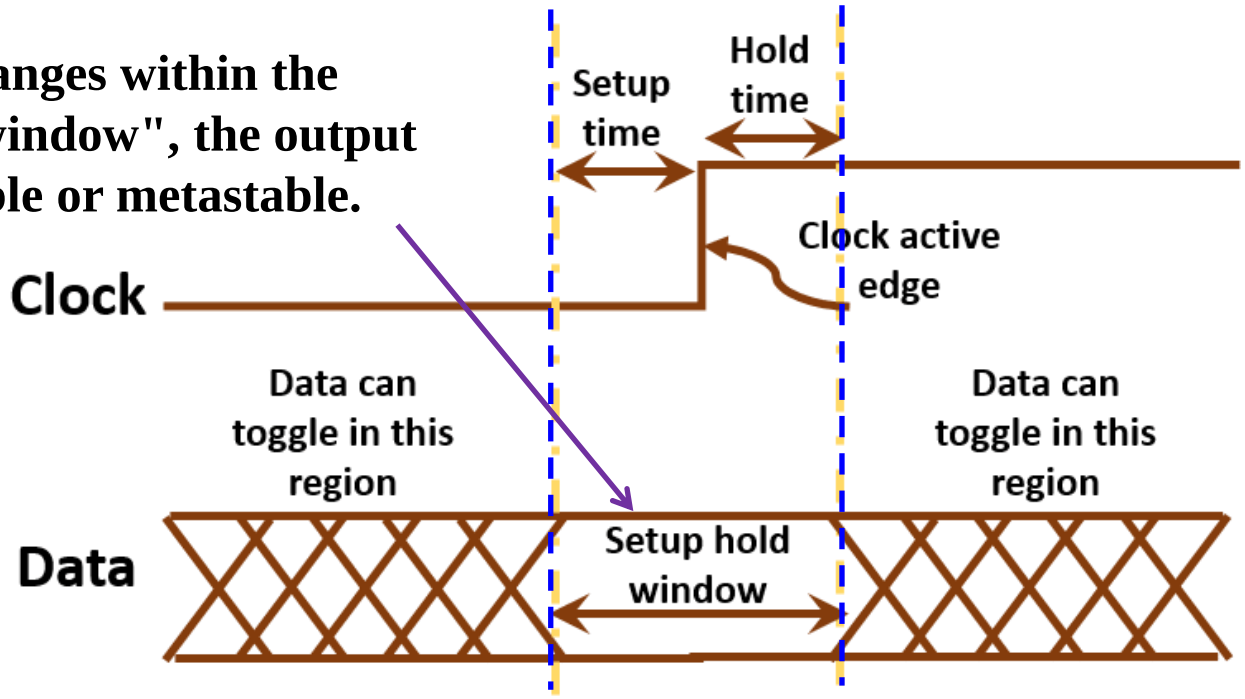
Flip-Flop Timing Parameters

- **Setup time: The minimum amount of time data input should be held steady before the clock event. This is so that the data can be stored successfully in the flip-flop.**
- **Hold time: The minimum amount of time data input should be held steady after the clock event so that data is reliably sampled by the clock.**



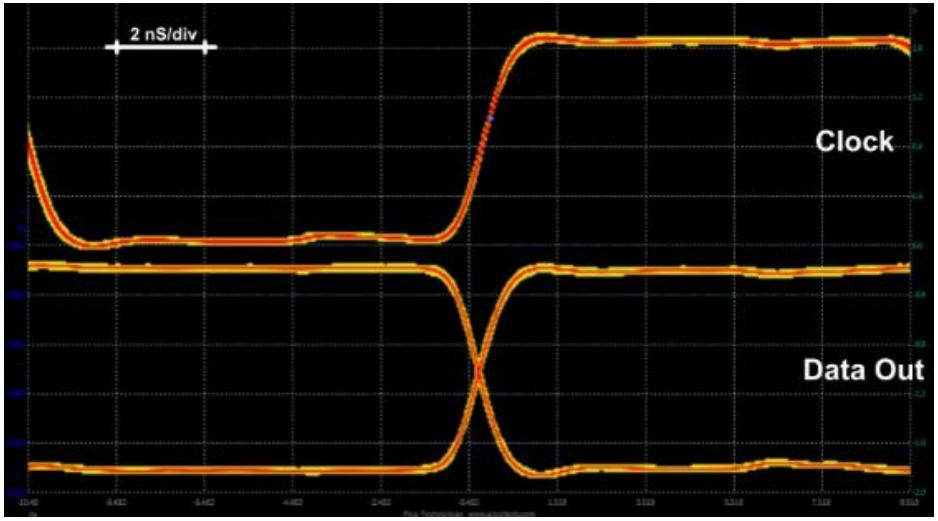
Flip-Flop Timing Parameters (continued)

If the data changes within the "setup-hold-window", the output is unpredictable or metastable.

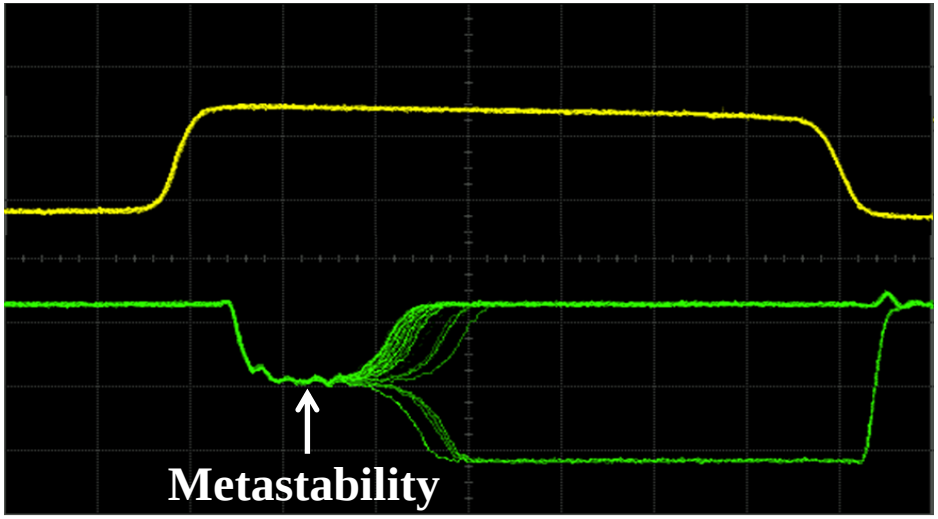


- E.g., if the setup time is 20 ns, it means that data has to be stable at least 20ns before the capturing clock-edge.
- Setup time and hole time together define a "setup-hold-window", in which data has to remain stable.

Flip-Flop Timing Parameters (continued)



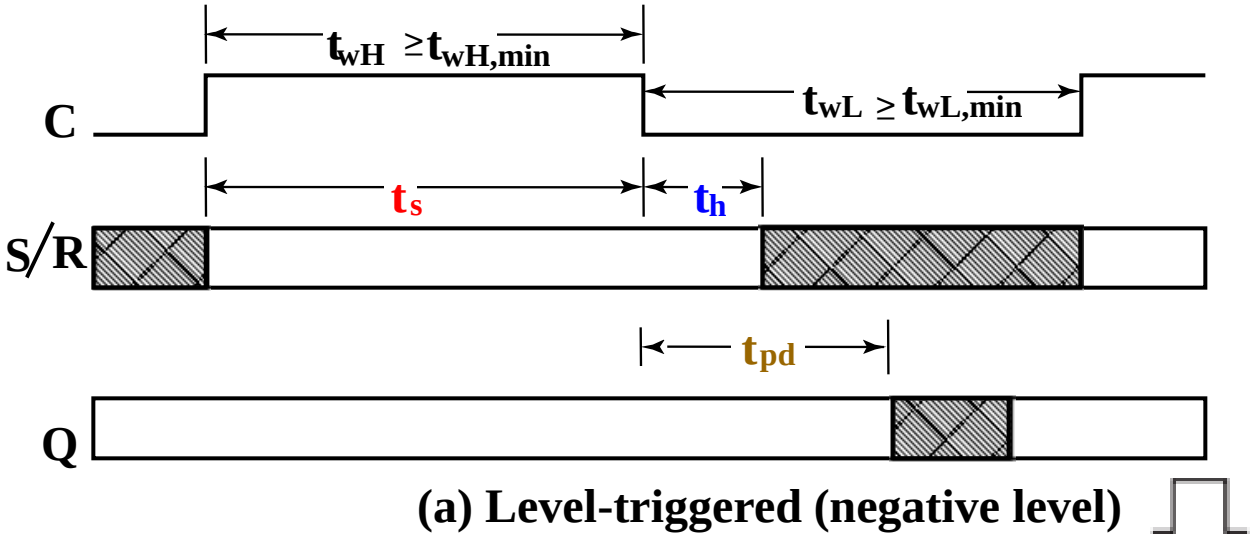
- **No setup time violation**
  - Input data arrives earlier than setup time before the rising edge of Clock.
  - Data out changes cleanly to either 0 or 1.



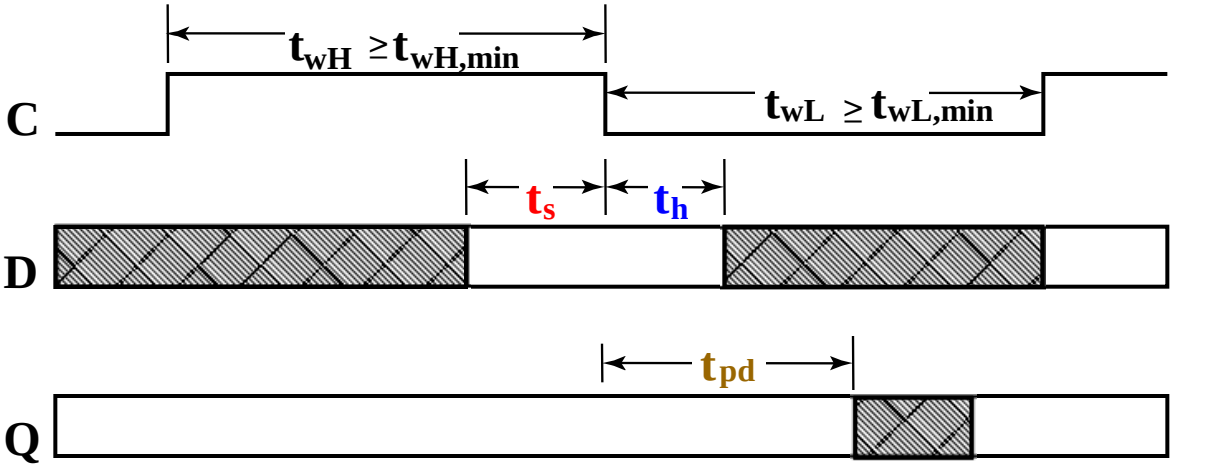
- **Setup time violation**
  - Input data arrives within the setup time window.
  - Data out becomes undefined (0 or 1 or somewhere in between) for a random period time before settling down to either 0 or 1.

Flip-Flop Timing Parameters (continued)

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



(a) Level-triggered (negative level)



(b) Edge-triggered (negative edge)

Flip-Flop Timing Parameters (continued)

- **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 <u>except</u>**
  - **Measured from clock edge that triggers the output change to the output change**

Big Picture of Latch and Flip-Flop

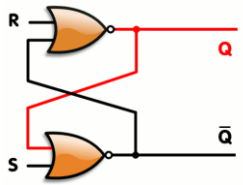
Traditional logic design

Delayed Switch

Problem: can't retain past inputs

Solution

cross-coupled feedback connection



Latch or Clocked Latch

Problem: transparent path

Solution

alternating clocks

Master-Slave Flip-Flop

Problem: catching glitches (0s or 1s)

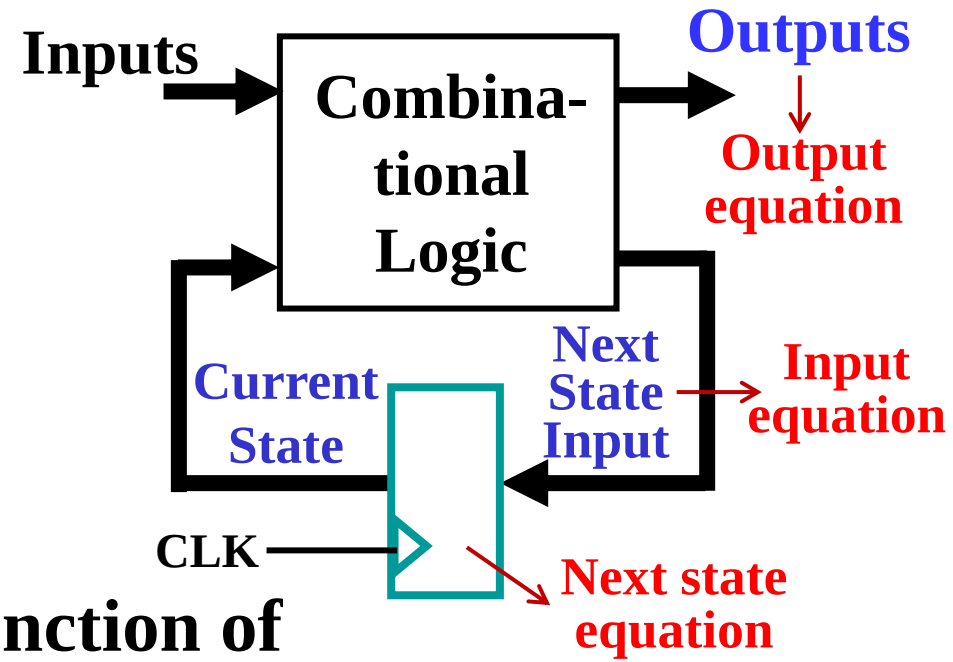
Solution

reducing inputs / sampling window

D Master-Slave Flip-Flops

Edge-Triggered Flip-Flops

Sequential Circuit Analysis



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

Sequential Circuit Analysis (continued)

- **Analysis of sequential circuits including:**
  - **Generating the functionality of the sequential circuit using state table, state diagram, and input/output Boolean equations.**
  - **Determining the timing constraints that a sequential circuit must be satisfied in order to prevent metastability, which allows the circuit to be used without error.**

Sequential Circuit Analysis (continued)

- **Sequential Circuit Analysis is an procedure of specifying the logic diagram of a given sequential circuit.**
- **A state table and state diagram are presented to describe the behavior of the circuit, demonstrating the time sequence of inputs, outputs and states, and illustrating the functionality of the given circuits.**

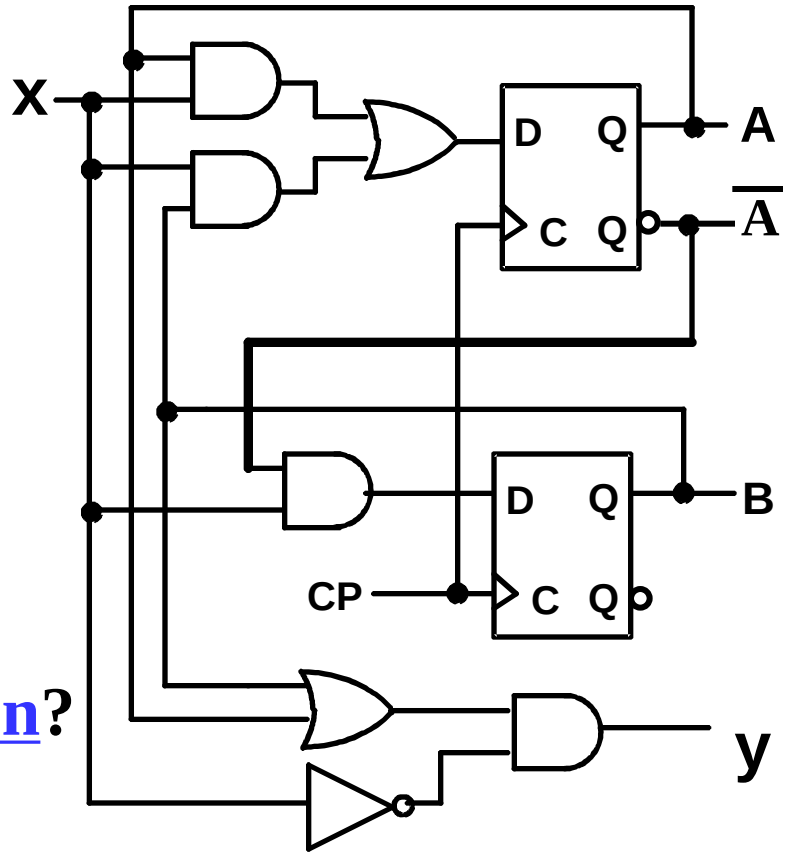
Sequential Circuit Analysis Procedure

1. **Derive the input equations, next state equations and output equations**
2. **Derive the state table (truth table with state):**
   - **Inputs: inputs of circuit, present state of the circuit**
   - **Outputs: outputs of circuit, next state of all flip-fops**
3. **List the next state of the sequential circuit**
4. **Obtain a state diagram**
5. **Analyze the performance of the circuit**
6. **Verify the correctness of the circuit, check the self-recovery capability and draw the timing parameters**

Example 1 (from Fig. 4-13)

- Input: x(t)
- Output: y(t)
- State: (A(t), B(t))

- What is the Output Equation?
  - y =
- What is the Input Equation?
  - D_A =
  - D_B =
- What is the Next State Equation?
  - A(t+1) =
  - B(t+1) =



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

- **Output equation:**

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

- **Input equations:**

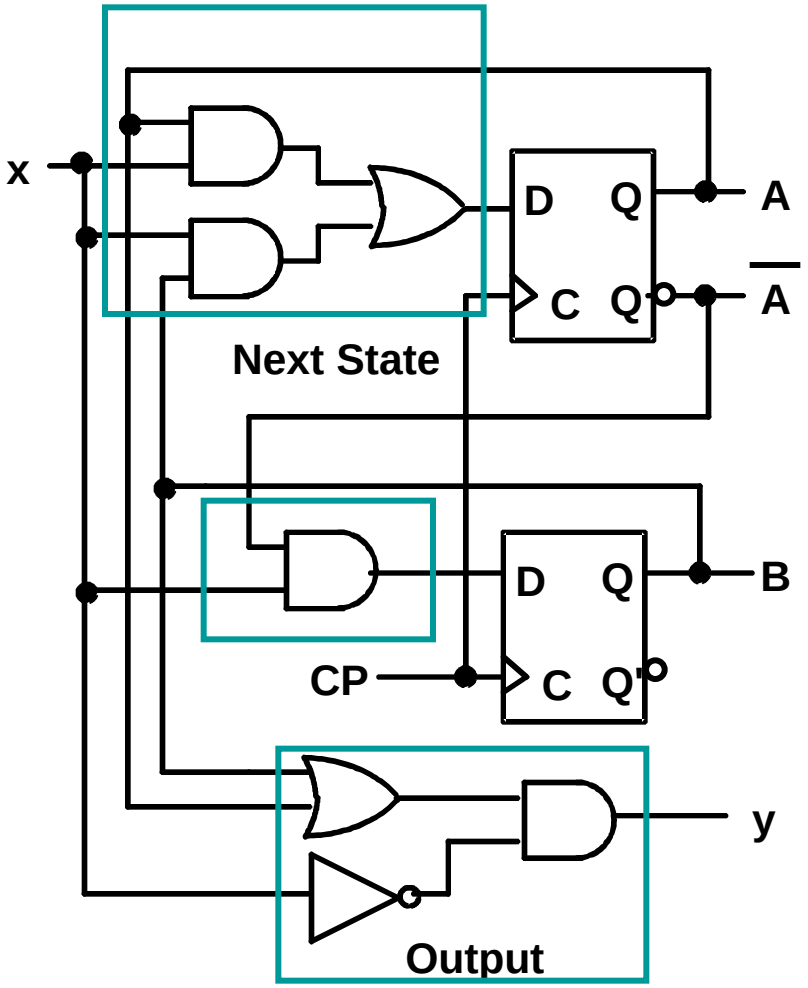
  $D_A = A(t)x(t) + B(t)x(t)$

  $D_B = \bar{A}(t)x(t)$

- **Next State equations:**

  $A(t+1) = D_A$

  $B(t+1) = D_B$



State Table Characteristics

- **State table – a multiple variable table with the following four sections:**
  - ***Input* – the input combinations allowed.**
  - ***Present State* – the values of the state variables for each allowed state.**
  - ***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**
  - **the outputs are Output, Next State**

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

- **The state table can be filled in using the input, next state and output equations:**
- **Input : $D_A = A(t)x(t) + B(t)x(t)$** **Next state: $A(t+1) = D_A$**

  $D_B = \overline{A}(t)x(t)$ **$B(t+1) = D_B$**
- **Output: $y(t) = \overline{x}(t)(B(t) + A(t))$**

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

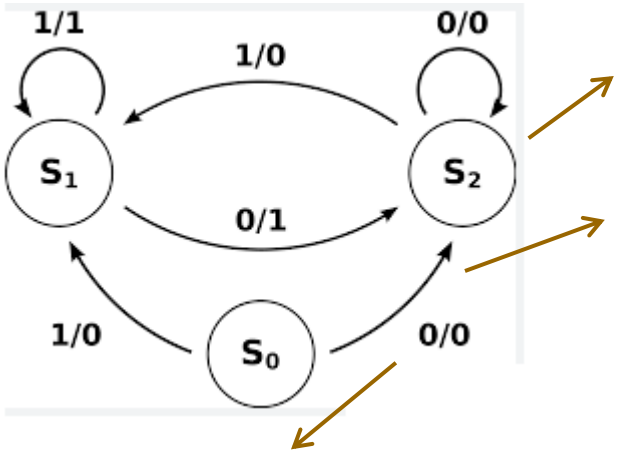
Example 1: Alternate State Table

- **2-dimensional table that matches well to a K-map. Present state rows and input columns in Gray code order.**
  - **$A(t+1) = A(t)x(t) + B(t)x(t)$**
  - **$B(t+1) = \overline{A}(t)x(t)$**
  - **$y(t) = \overline{x}(t)(B(t) + A(t))$**

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

State Diagrams

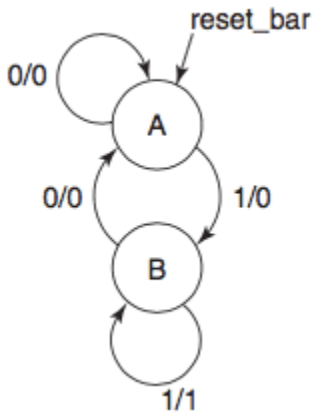
- **The sequential circuit function can be represented as a state diagram with the following components:**

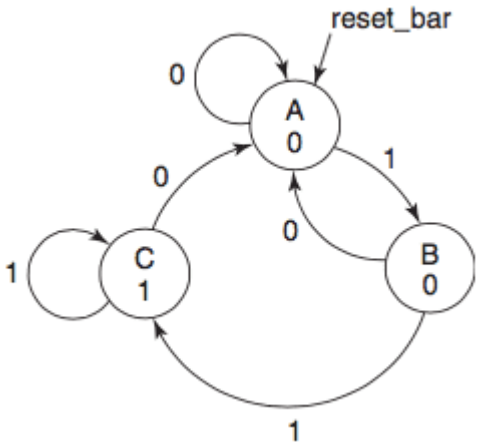


- **A circle with the state name in it for each state**
- **A directed arc from the present state to the next state for each state transition**
- **A label on each directed arc with the input values which causes the state transition, and**
- **A label for output:**
  - **On each directed arc with the output value, or**
  - **On each circle with the output value**

State Diagrams

Mealy type machine



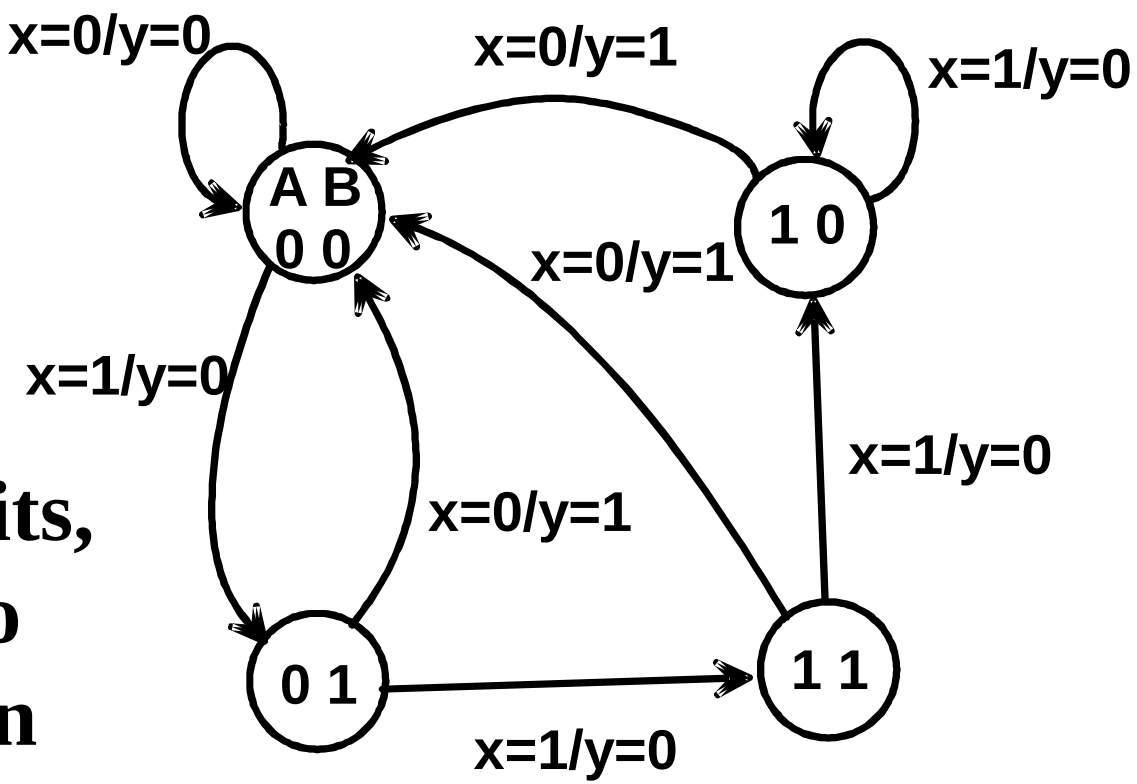


Moore type machine

- **Output label form:**
  - **On directed arc with the output value (Mealy type):**
    - **input/output**
    - **output depends on state and input**
  - **On circle with output value (Moore type):**
    - **state/output**
    - **output depends only on state**

Example 1: State Diagram

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



output: $y(t) = \bar{x}(t)(B(t) + A(t))$

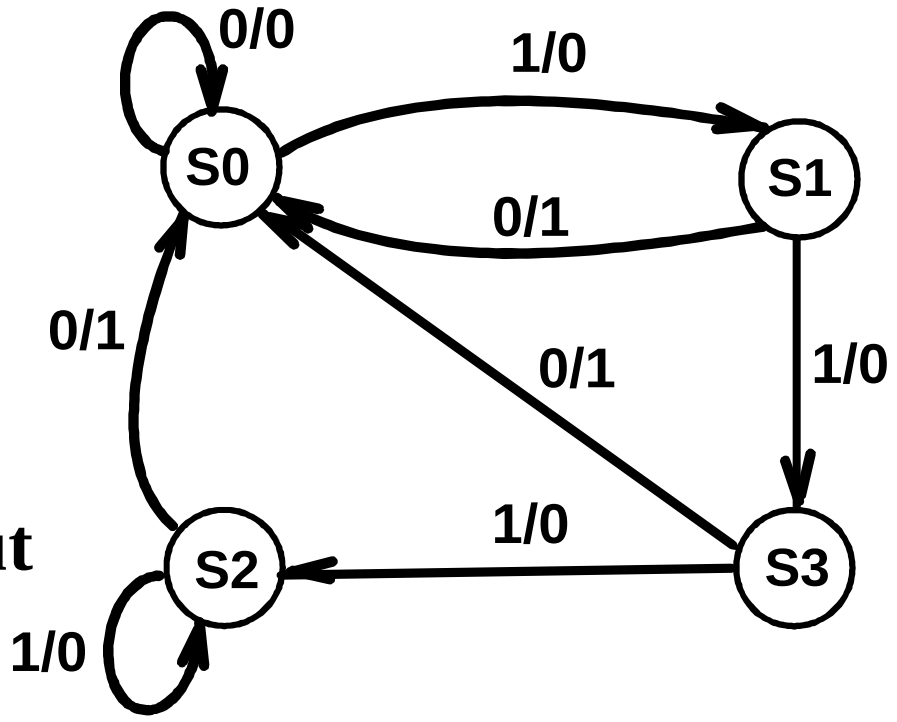
Present State A(t) B(t)	Next State x(t)=0 A(t+1)B(t+1)	Next State x(t)=1 A(t+1)B(t+1)	Output x(t)=0 y(t)	Output x(t)=1 y(t)
0 0	0 0	0 1	0	0
0 1	0 0	1 1	1	0
1 1	0 0	1 0	1	0
1 0	0 0	1 0	1	0

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.**

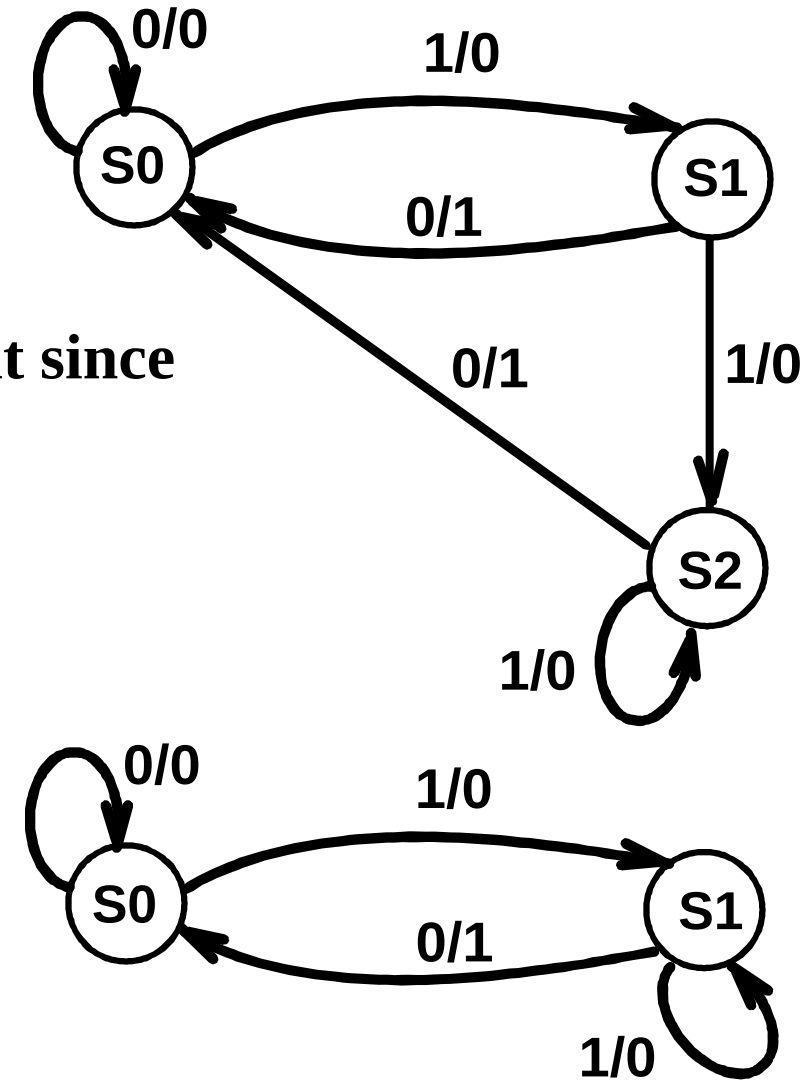
Equivalent State Example

- **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.**



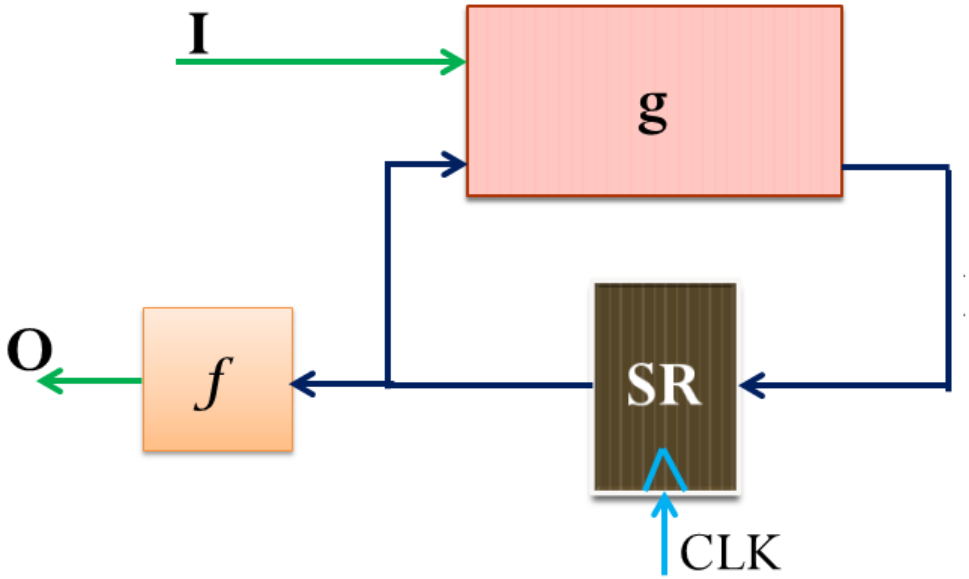
Equivalent State Example

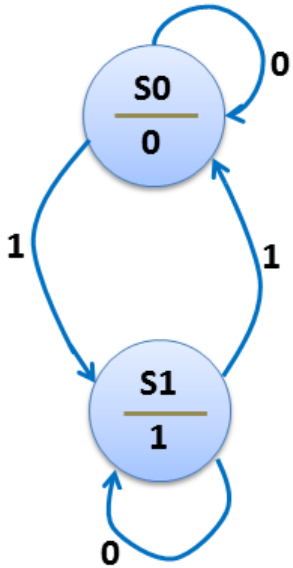
- **Replacing S3 and S2 by a single state gives state diagram:**
- **Examining the new diagram, states S1 and S2 are equivalent since**
  - **their outputs for input 0 is 1 and input 1 is 0, and**
  - **their next state for input 0 is S0 and for input 1 is S2,**
- **Replacing S1 and S2 by a single state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



Moore and Mealy Models

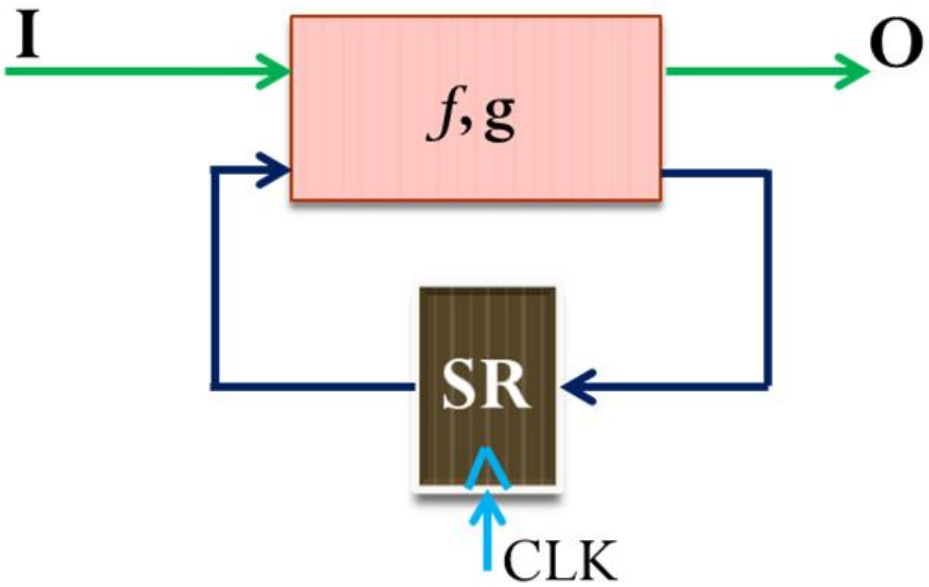
- **Sequential circuits are implemented in two different ways: Moore model and Mealy model.**
- **Moore Model - named after E.F. Moore**
  - **Outputs are a function ONLY of states**
  - **Usually specified on the states.**

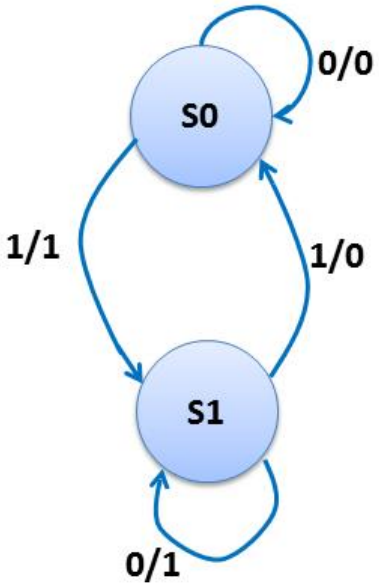




Moore and Mealy Models (continued)

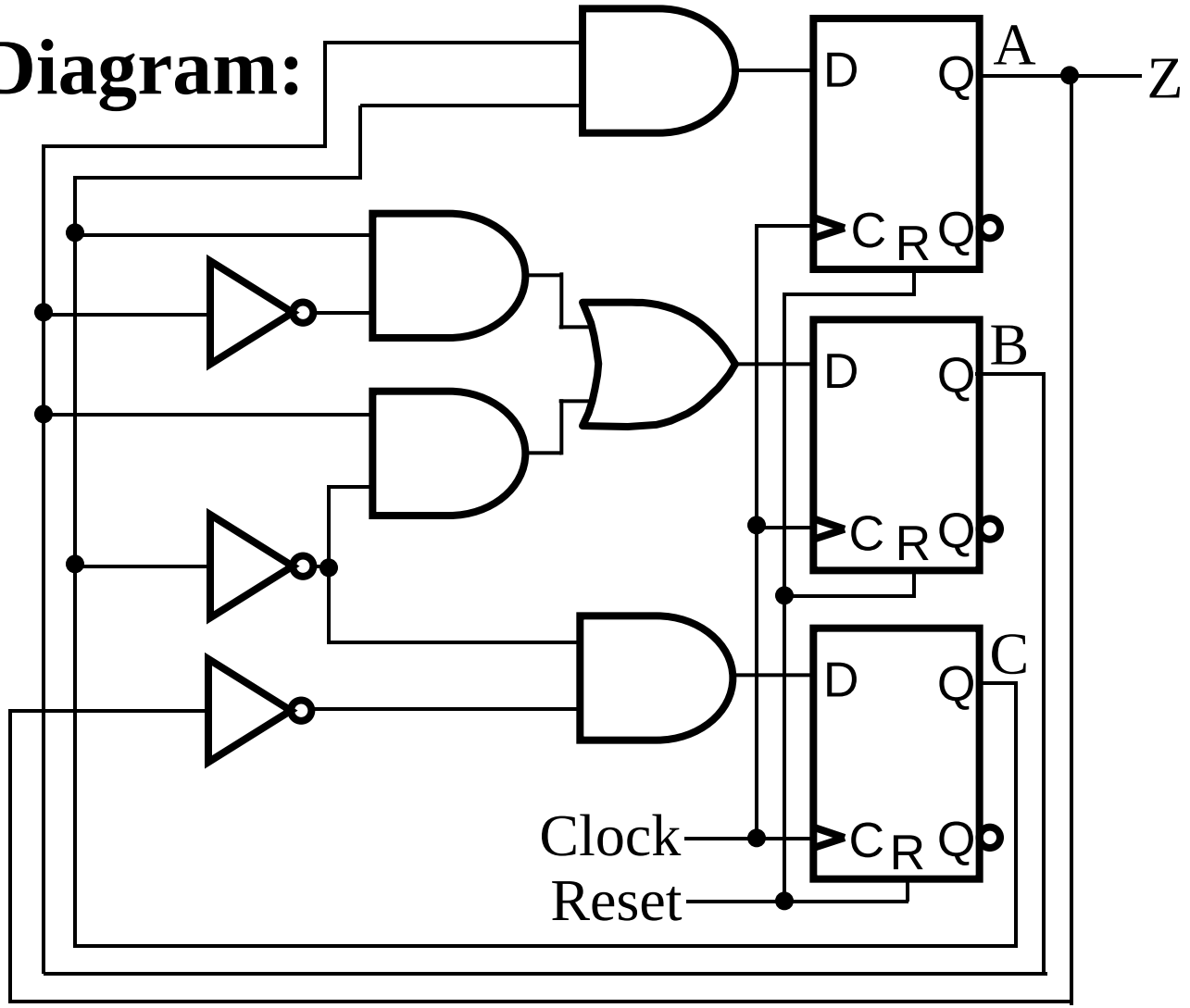
- **Sequential circuits are implemented in two different ways: Moore model and Mealy model.**
- **Mealy Model - named after G. Mealy**
  - **Outputs are a function of inputs AND states**
  - **Usually specified on the state transition arcs.**





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)**
- **Input Equations:**

$$D_A = B(t)C(t)$$

$$D_B = \overline{B}(t)C(t) + B(t)\overline{C}(t)$$

$$D_C = \overline{A}(t)\overline{C}(t)$$

- **Next State Equations:**

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

$$B(t+1) = D_B$$

$$C(t+1) = D_C$$

- **Output Equation:**

$$Z = A(t)$$

Example 2: State Table

A' = A(t+1)

B' = B(t+1)

C' = C(t+1)

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

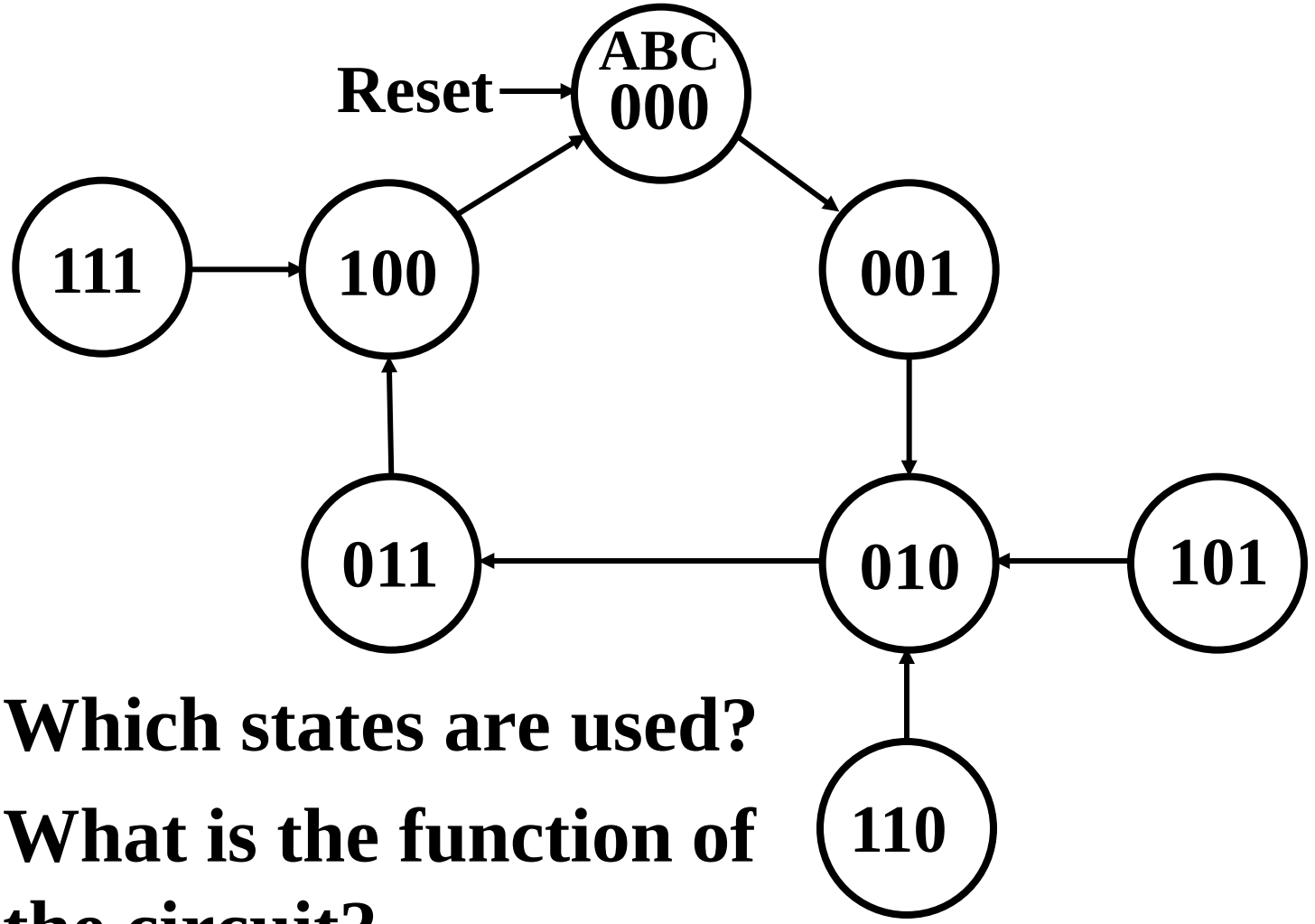
$$B(t+1) = \overline{B}(t)C(t) + B(t)\overline{C}(t)$$

$$C(t+1) = \overline{A}(t)\overline{C}(t)$$

$$Z = A(t)$$

present state	next state	output
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

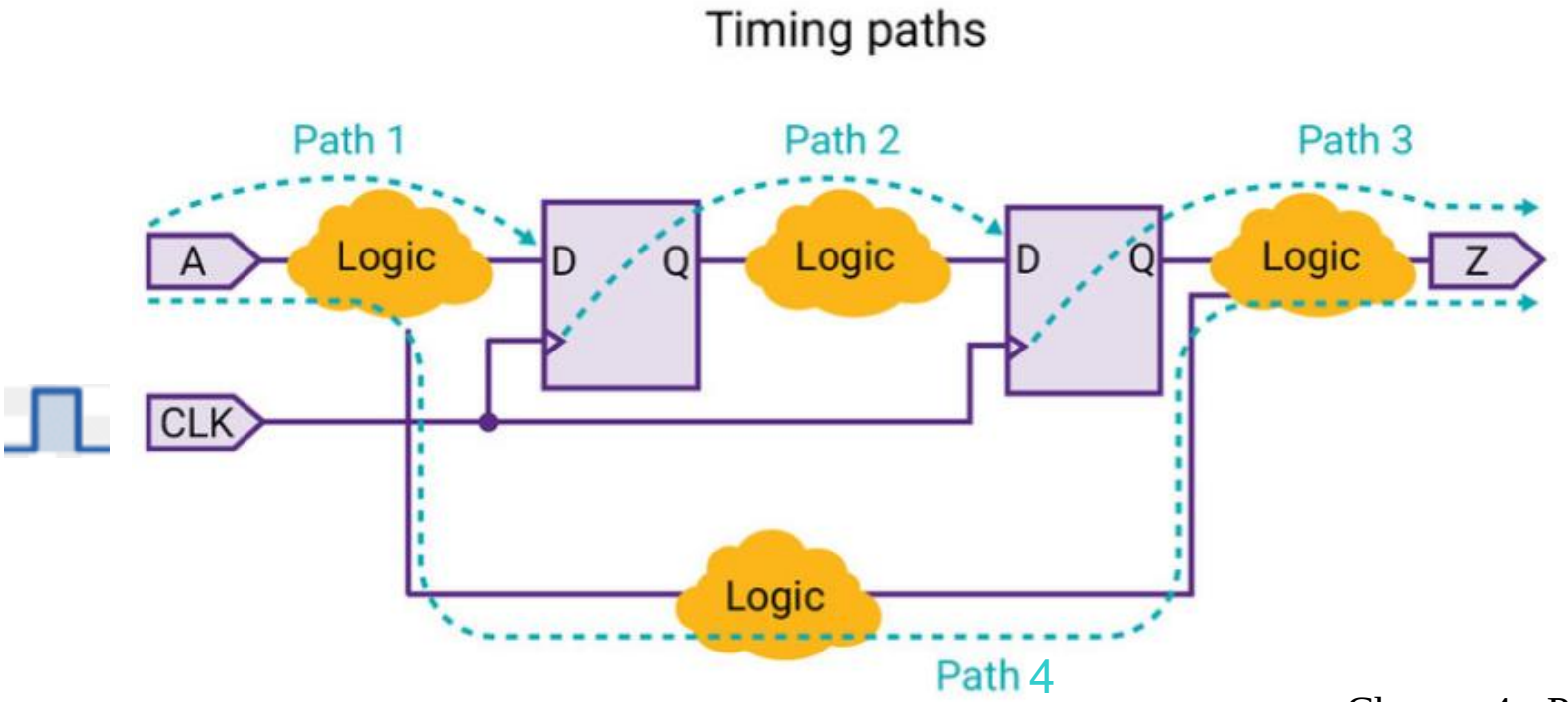
Example 2: State Diagram



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

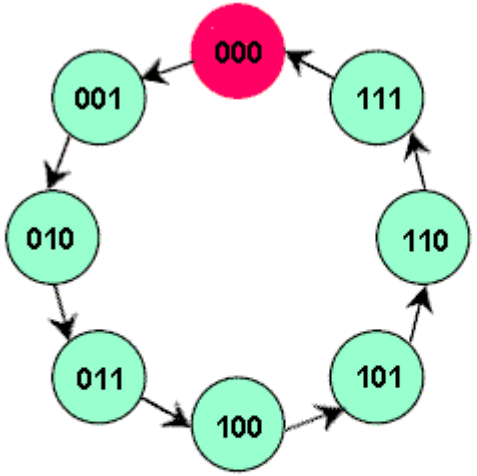
Timing Analysis of Sequential Circuits

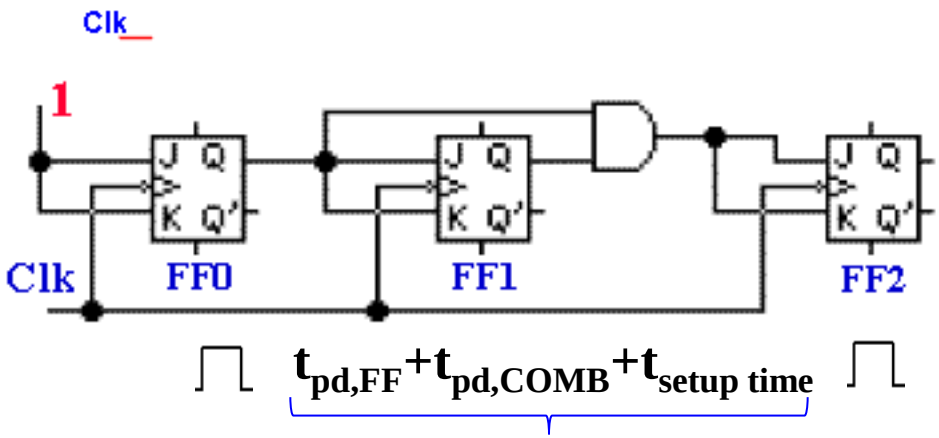
- It is important to analyze the timing behavior of a sequential circuit.
- The ultimate goal of timing analysis is to determine the maximum clock frequency of the circuit.



Timing Analysis of Sequential Circuits (continued)

- **Consider a 3-bit binary counter.**
- **If the clock period is too short, data changes may not be able to propagate through the circuit to flip-flop inputs before the setup time interval begins.**





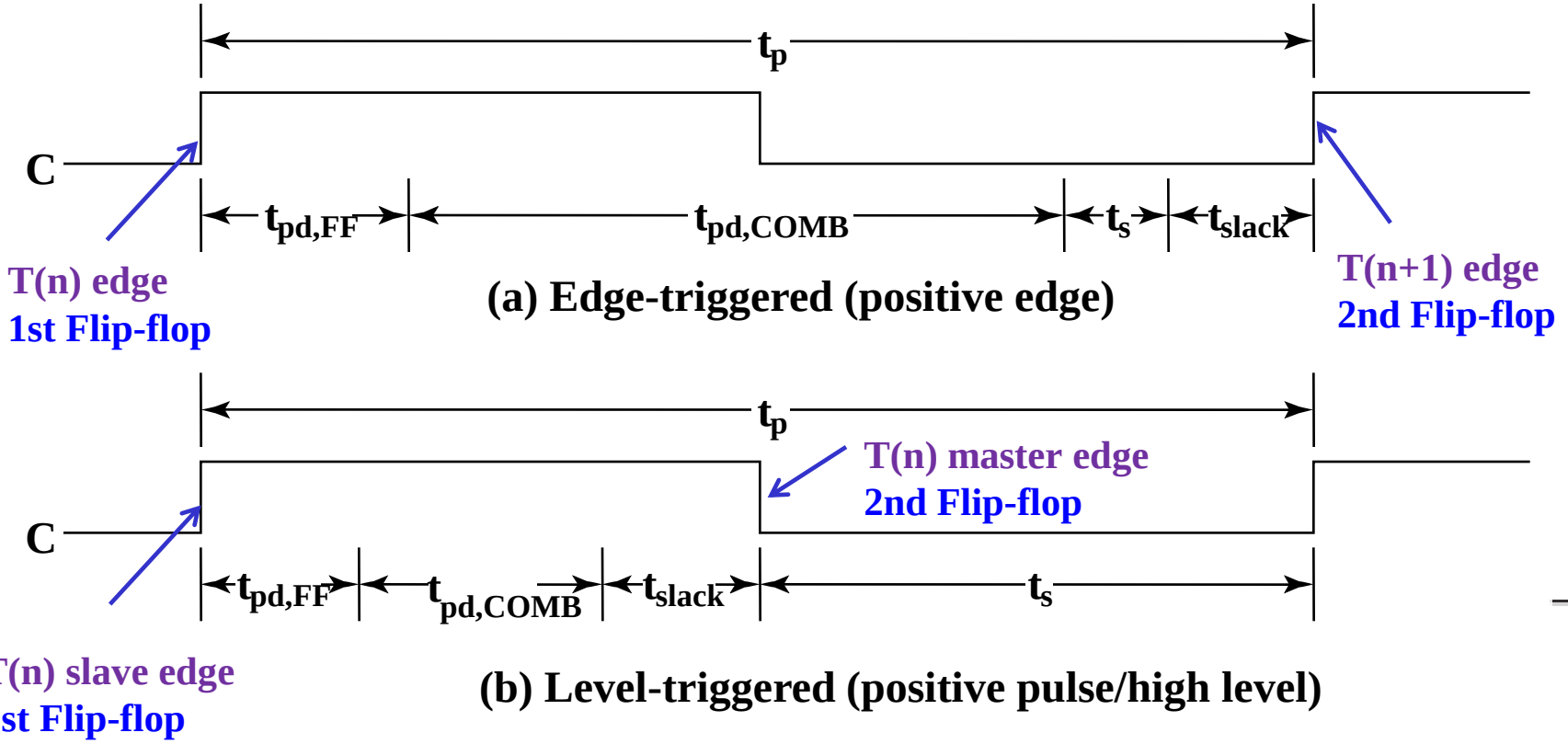
Timing Analysis of Sequential Circuits (continued)

- **Timing Constraints Components**
  - t_p - **clock period** - **The interval between occurrences of a specific clock edge in a periodic clock**
  - $t_{pd,FF}$ - **flip-flop propagation delay** - **The amount of time from clock edge to when the flip-flop output becomes stable**
  - $t_{pd,COMB}$ - **combinational logic delay** – **The total delay of combinational logic along the path from flip-flop output to flip-flop input**
  - t_s – **flip-flop setup time** - **The amount of time data input should be held steady before the clock event.**
  - t_{slack} - **extra time in the clock period**

time delays along the path

Timing Analysis of Sequential Circuits (continued)

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



Timing Analysis of Sequential Circuits (continued)

- **Timing Equations**

  $t_{pi} \geq t_{slack} + (t_{pd,FF} + t_{pd,COMB} + t_s)$ ⟵ for every Flip-flop

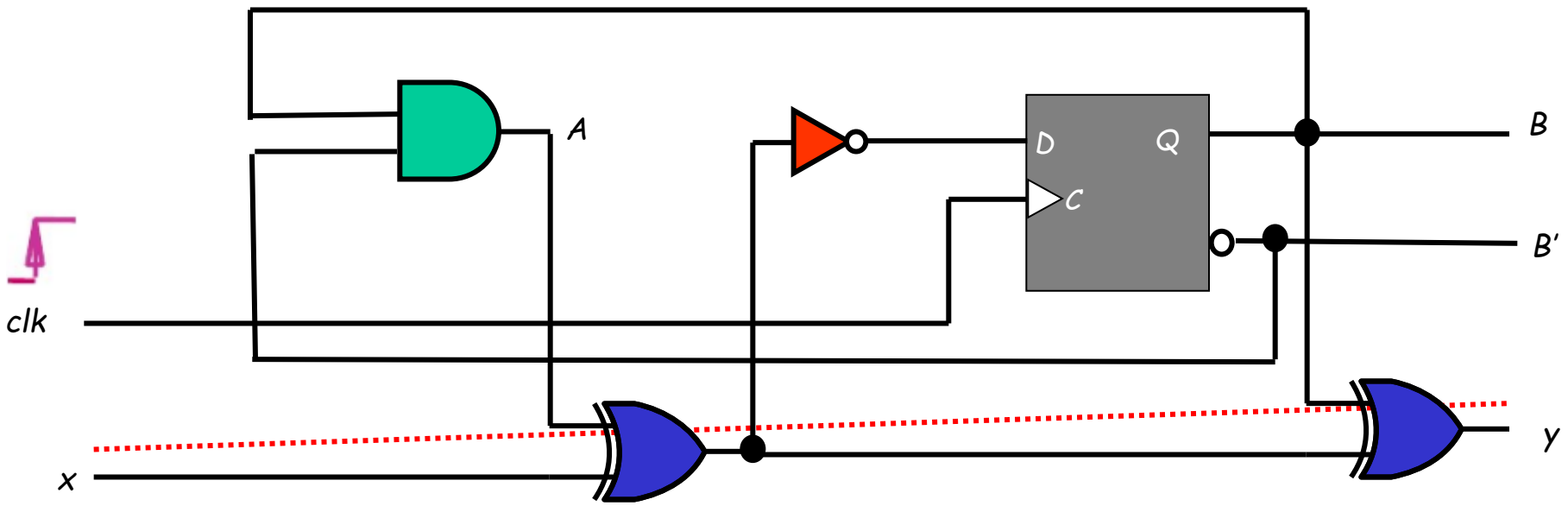
  - For t_{slack} greater than or equal to zero,

    $t_p \geq \max (t_{pd,FF} + t_{pd,COMB} + t_s)$ ⟵ for all Flip-flops

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

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

Timing Analysis Example (1/5)



$t_{pd,NOT} = 0.5$ ns $t_{pd,FF} = 2.0$ ns

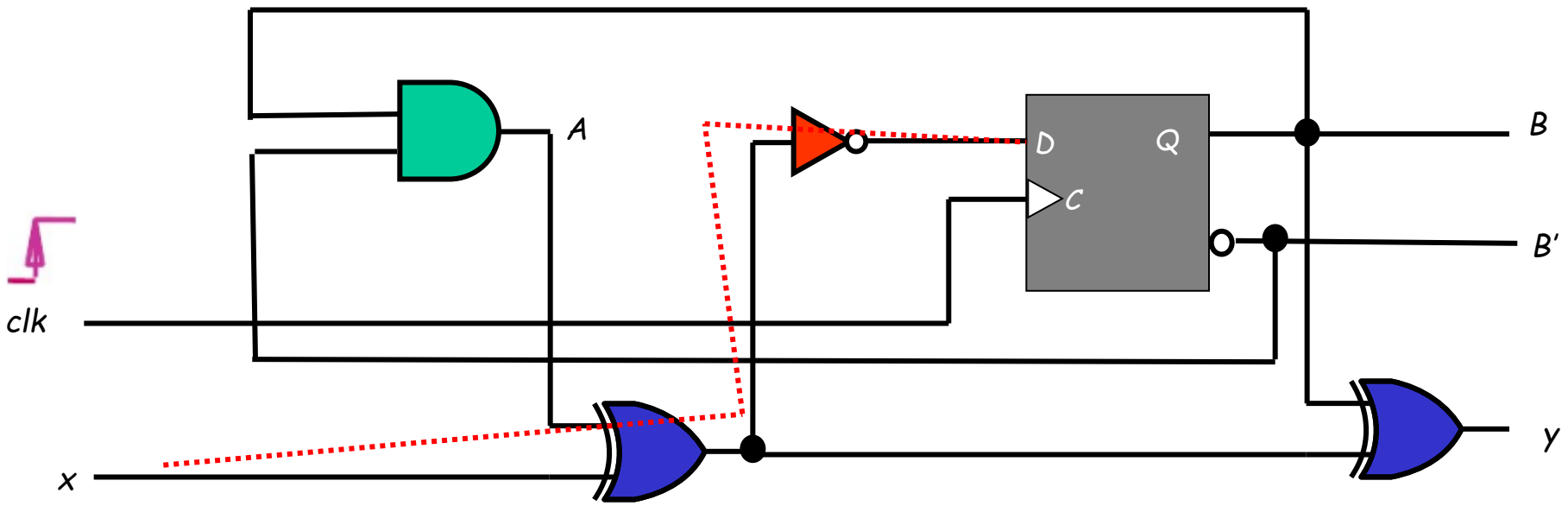
$t_{pd,XOR} = 2.0$ ns $t_s = 1.0$ ns

$t_{pd,AND} = 1.0$ ns $t_h = 0.25$ ns

- **Find the longest path delay from external input to the output.**

$$t_{pd,XOR} + t_{pd,XOR} = 2.0 + 2.0 = 4.0 \text{ ns}$$

Timing Analysis Example (2/5)



$t_{pd,NOT} = 0.5$ ns $t_{pd,FF} = 2.0$ ns

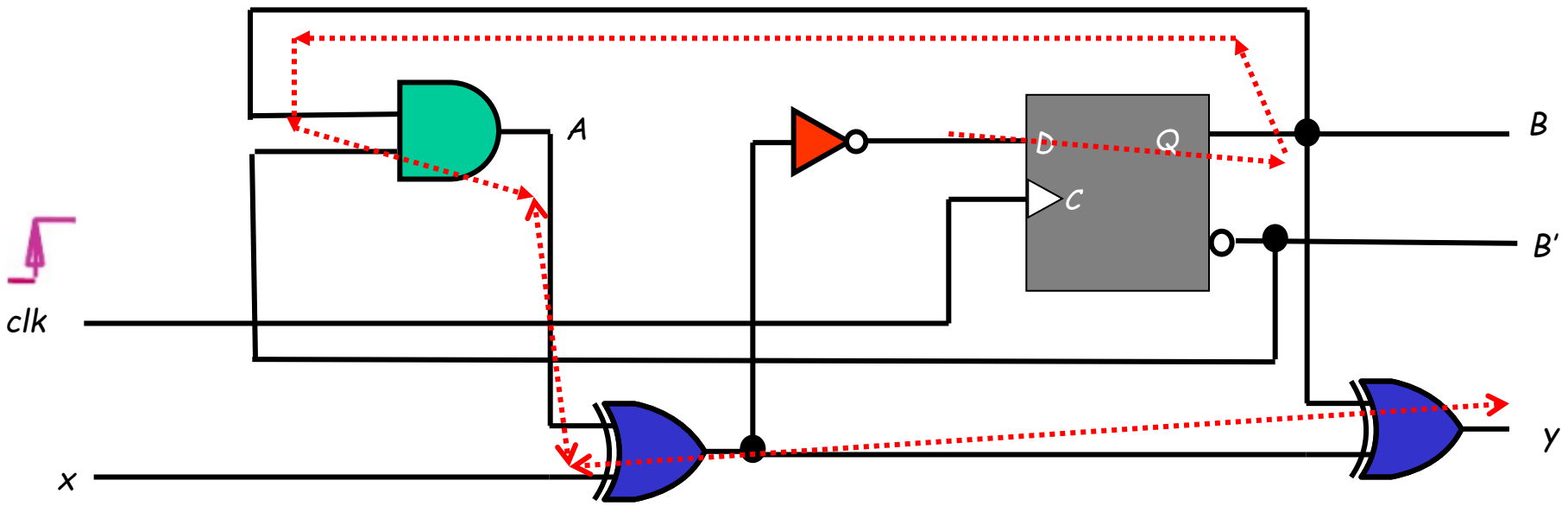
$t_{pd,XOR} = 2.0$ ns $t_s = 1.0$ ns

$t_{pd,AND} = 1.0$ ns $t_h = 0.25$ ns

- **Find the longest path delay in the circuit from external input to positive clock edge.**

$$t_{pd,XOR} + t_{pd,NOT} + t_s$$
$$= 2.0 + 0.5 + 1.0 = 3.5 \text{ ns}$$

Timing Analysis Example (3/5)



$t_{pd,NOT} = 0.5$ ns $t_{pd,FF} = 2.0$ ns

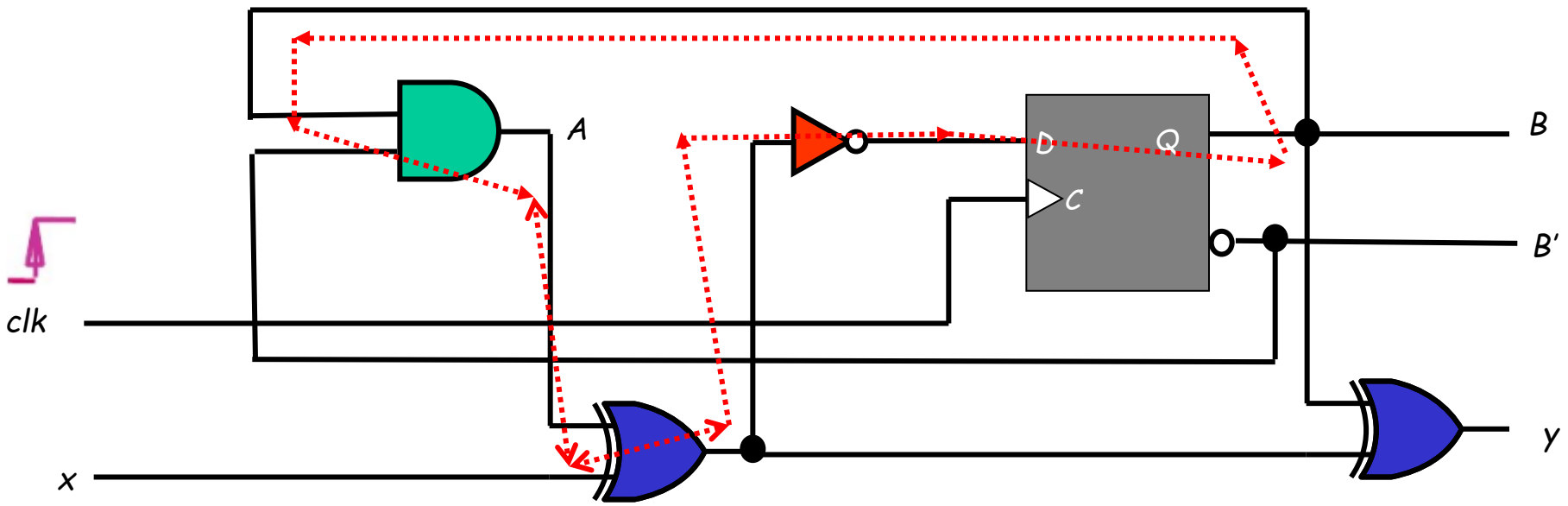
$t_{pd,XOR} = 2.0$ ns $t_s = 1.0$ ns

$t_{pd,AND} = 1.0$ ns $t_h = 0.25$ ns

- **Find the longest path delay from positive clock edge to output.**

$$t_{pd,\,FF} + t_{pd,\,AND} + t_{pd,\,XOR} + t_{pd,\,XOR}$$
$$= 2.0 + 1.0 + 2.0 + 2.0 = 7 \text{ ns}$$

Timing Analysis Example (4/5)



$t_{pd,NOT} = 0.5$ ns $t_{pd,FF} = 2.0$ ns

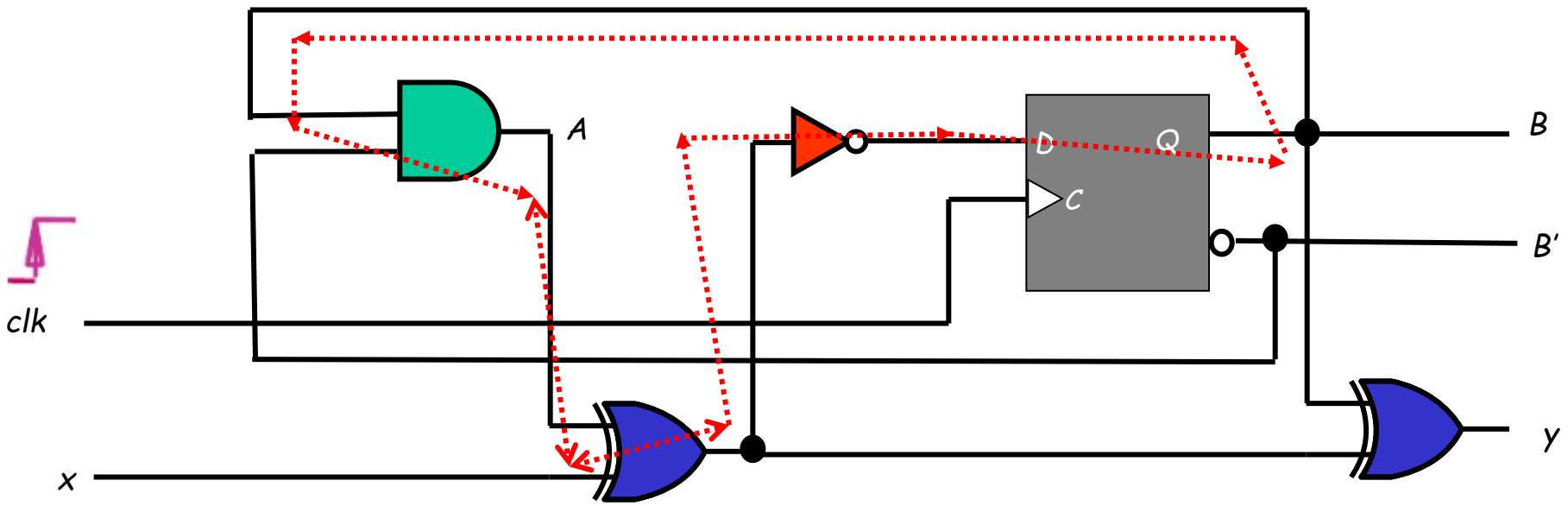
$t_{pd,XOR} = 2.0$ ns $t_s = 1.0$ ns

$t_{pd,AND} = 1.0$ ns $t_h = 0.25$ ns

- **Find the longest path delay from positive clock edge to the positive clock edge.**

$$t_{pd,FF} + t_{pd,AND} + t_{pd,XOR} + t_{pd,NOT} + t_s$$
$$= 2.0 + 1.0 + 2.0 + 0.5 + 1.0 = 6.5 \text{ ns}$$

Timing Analysis Example (5/5)



$t_{pd,NOT} = 0.5$ ns $t_{pd,FF} = 2.0$ ns

$t_{pd,XOR} = 2.0$ ns $t_s = 1.0$ ns

$t_{pd,AND} = 1.0$ ns $t_h = 0.25$ ns

- **Determine the maximum frequency of the circuit in megahertz (MHz).**

$$t_{pd,FF} + t_{pd,AND} + t_{pd,XOR} + t_{pd,NOT} + t_s$$
$$= 2.0 + 1.0 + 2.0 + 0.5 + 1.0 = 6.5 \text{ ns}$$
$$f_{max} = 1/(6.5 \times 10^{-9}) \approx 154 \text{ MHz}$$

Assignments

Reading:

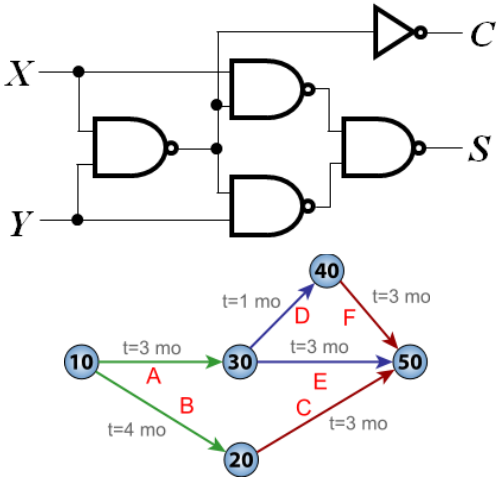
- **4.1-4.4, 4.9-4.10**

Problem assignment:

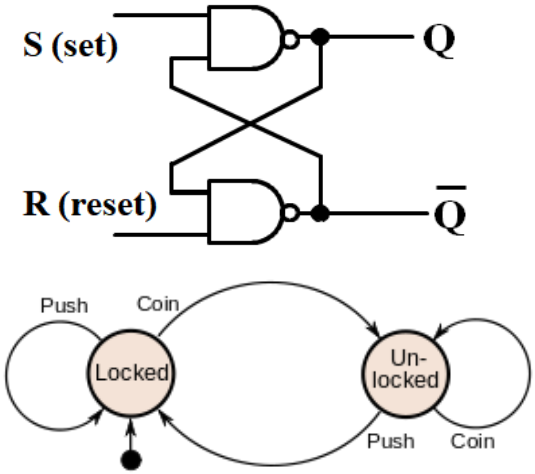
- **4-7; 4-9; 4-12(b); 4-58; 4-59**

Appendix A: Going Beyond Combinational Logic (continued)

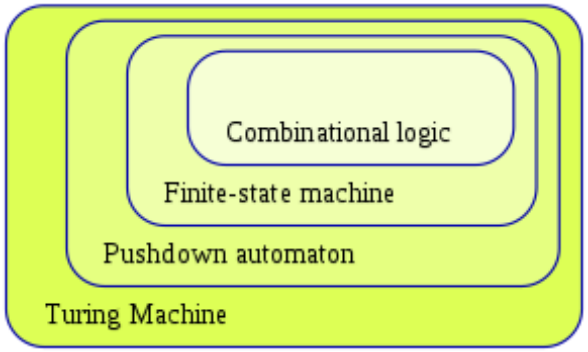
- **Automata theory is the study of abstract machines and the computational problems.**
- **Classes of automata theory:**
  - **Combinational logic (time-independent logic) is defined as a directed acyclic graph**
  - **Finite state machine (directed graph) is introduced for modelling time-dependent logic.**



Combinational circuit

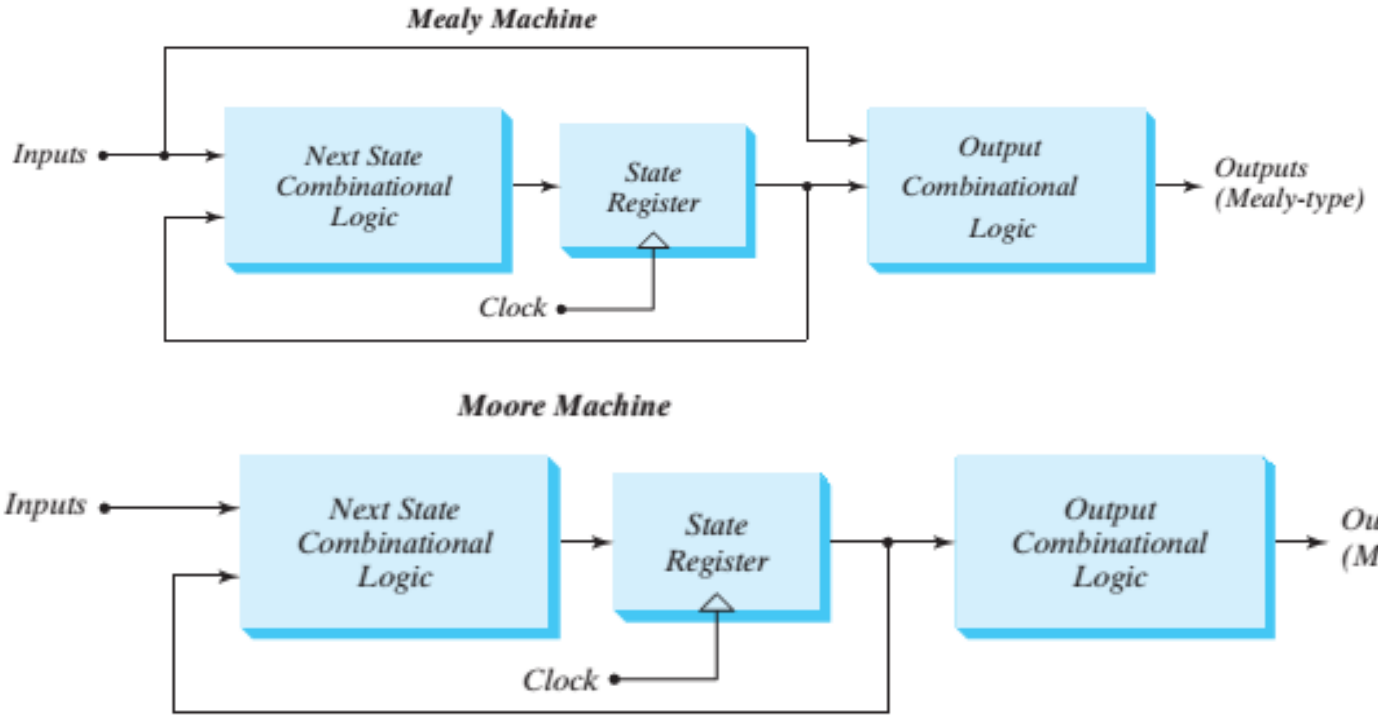


Sequential circuit

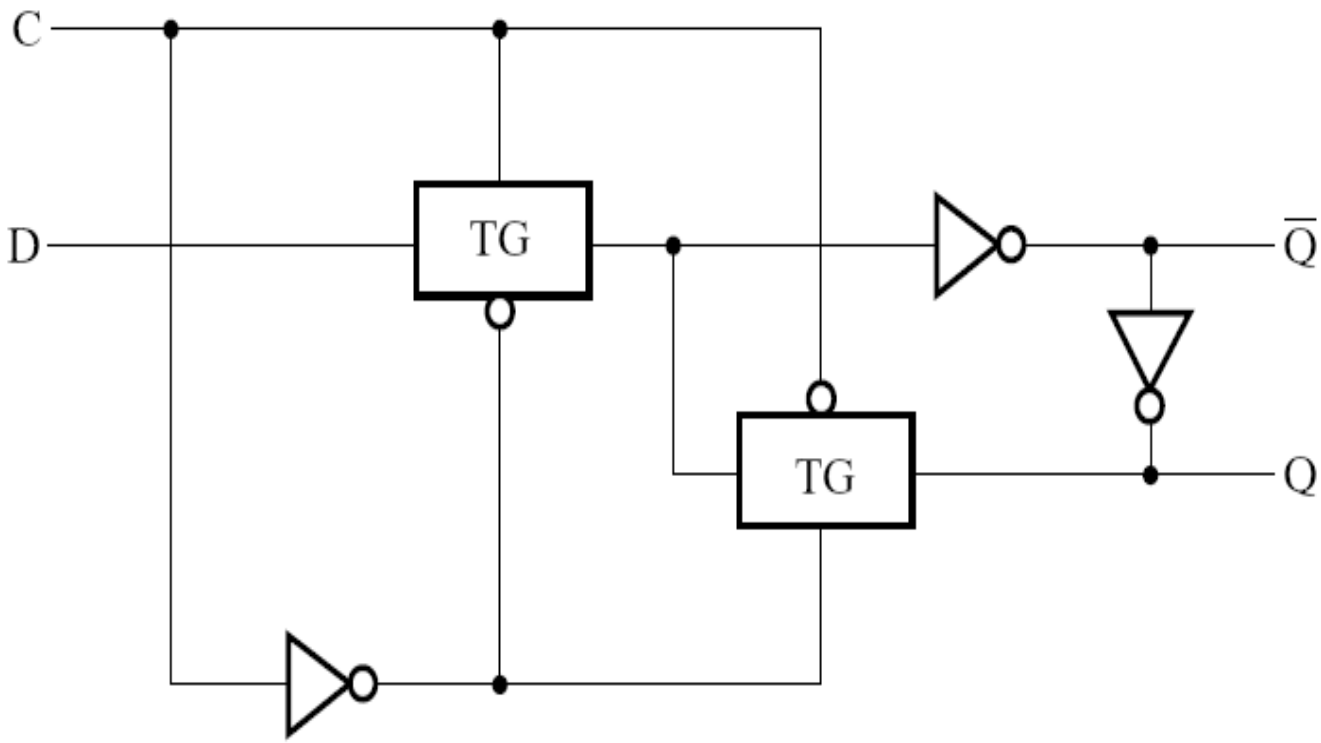


Introduction to Sequential Circuits

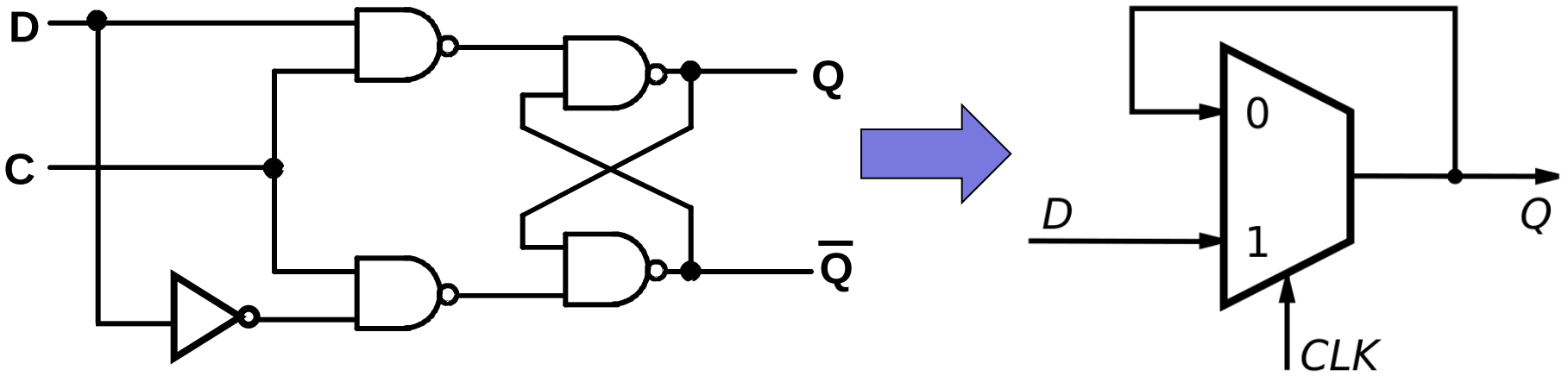
- **Combinatorial Logic**
  - ***Next state function*: Next State = f(Inputs, State)**
  - ***Output function* (Mealy): Outputs = g(Inputs, State)**
  - ***Output function* (Moore): Outputs = h(State)**
- **Output function type depends on specification and affects the design significantly**



Appendix B: D Latch with Transmission Gates

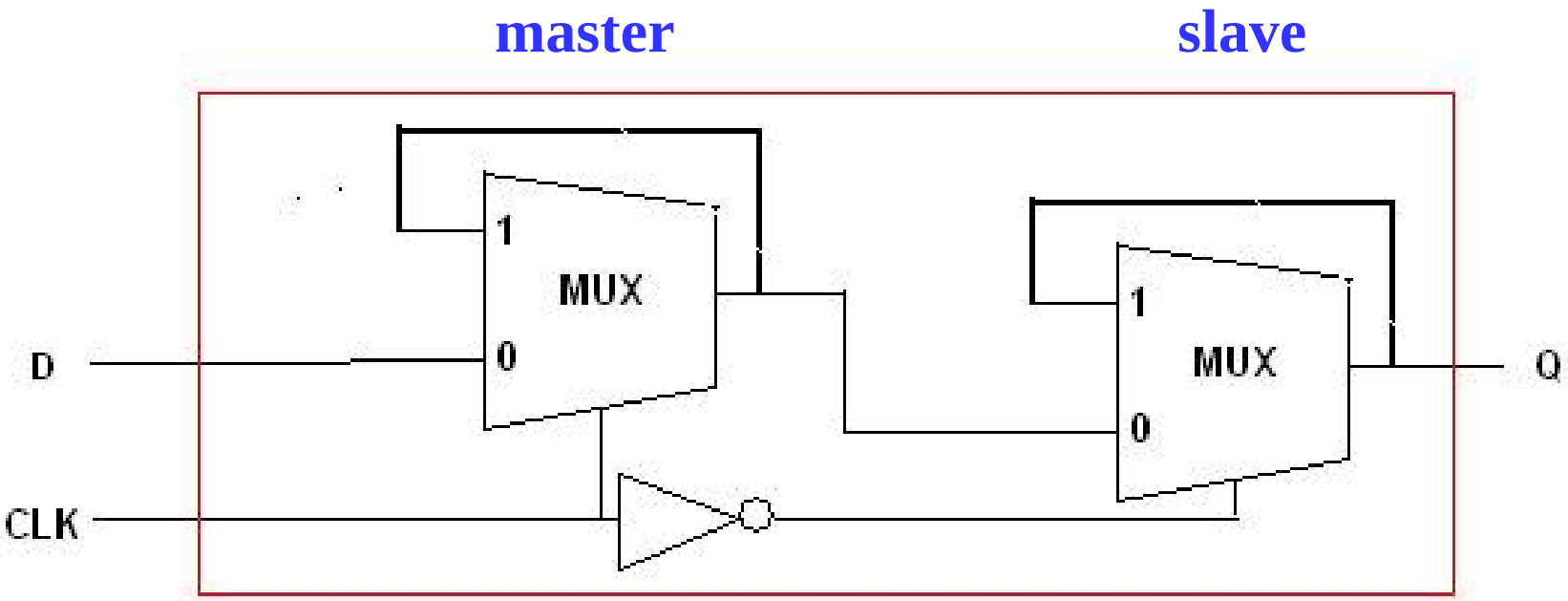


D Latch with MUX

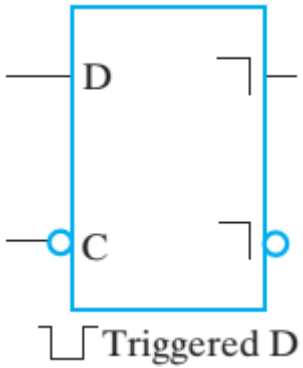


Positive level triggered D Latch

D Flip-Flop with MUXs



C	D	Q(t + 1)
0	X	No change
1	0	0: Clear Q
1	1	1: Set Q

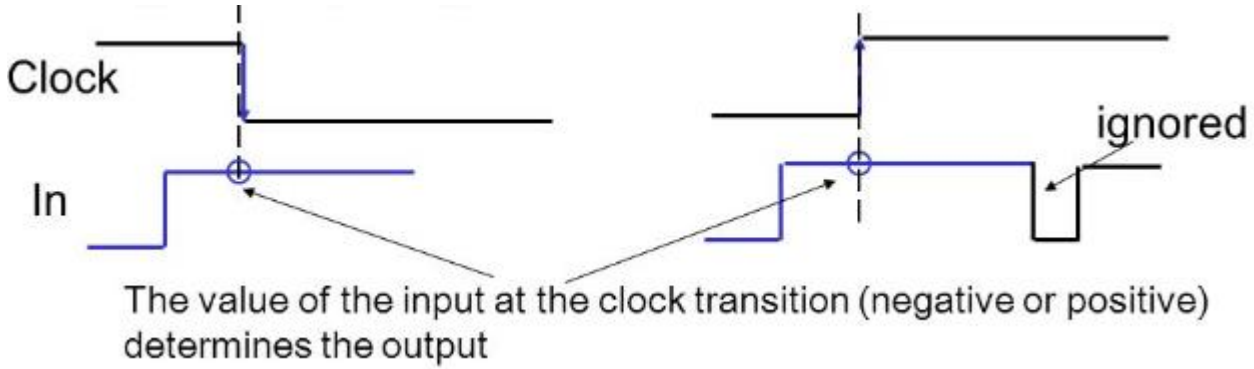


Appendix C: Summary

- **Difference between a Latch and a Flip-Flop**
  - **A latch is transparent. Its output can change as soon as the inputs do. In a flip-flop, the path from its inputs to its outputs is broken.**
  - **A latch is asynchronous, whereas flip-flop is a combination of a clock and a latch, and its output is changed according to the clock.**
  - **Latch is a level sensitive device while flip-flop is an edge sensitive device.**
  - **Latch is sensitive to glitches on enable pin, whereas flip-flop is immune to glitches.**
  - **Latches take less gates (also less power) to implement than flip-flops.**
  - **Latches are faster than flip-flops.**

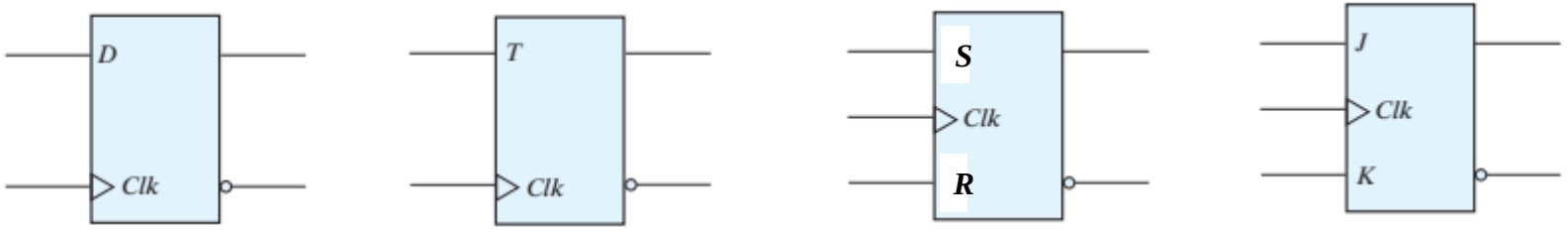
Summary (continued)

- Master-slave FFs use alternating clocks to break the path from input to output.
- The behavior of "catching" glitches (0-1-0/1-0-1) by master stage is called 1s catching.
- Solutions for solving 1s catching problem:
  - D master-slave FFs and edge-triggered FFs
- An edge-triggered flip-flop responds to its input at a well-defined moment (at the clock-transition).

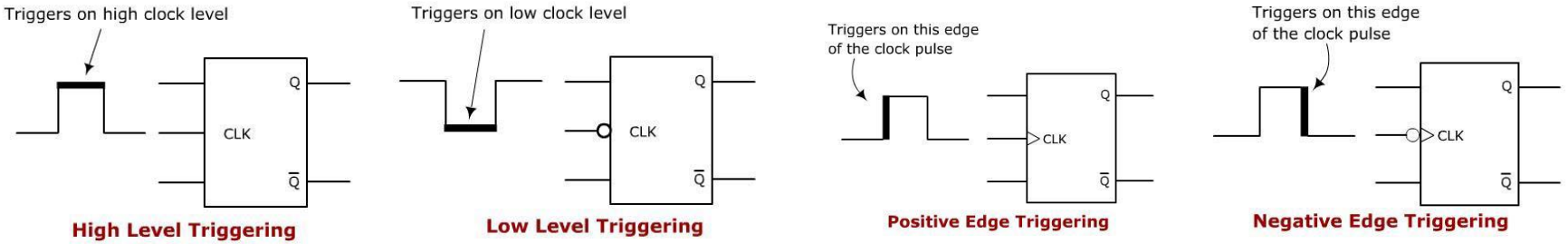


Summary (continued)

- **Four types of latches and flip-flops:**
  - **D-type (Data or Delay)**
  - **T-type (Toggle)**
  - **SR-type (S-Set, R-Reset)**
  - **JK-type (J-Set, K-Reset)**

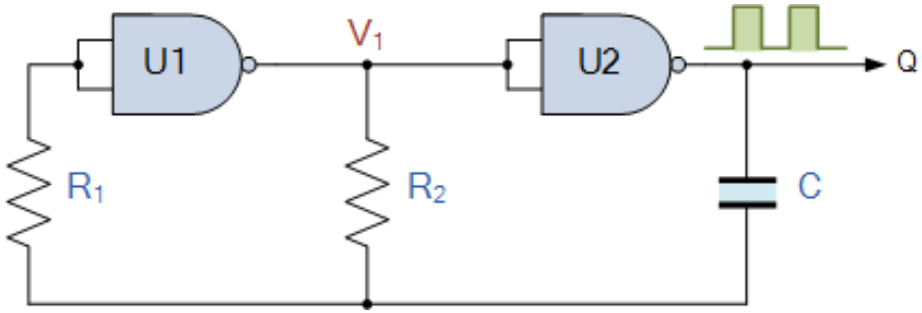


- **Four types of pulse-triggering methods:**
  - **High Level Triggering**
  - **Low Level Triggering**
  - **Positive Edge Triggering**
  - **Negative Edge Triggering**

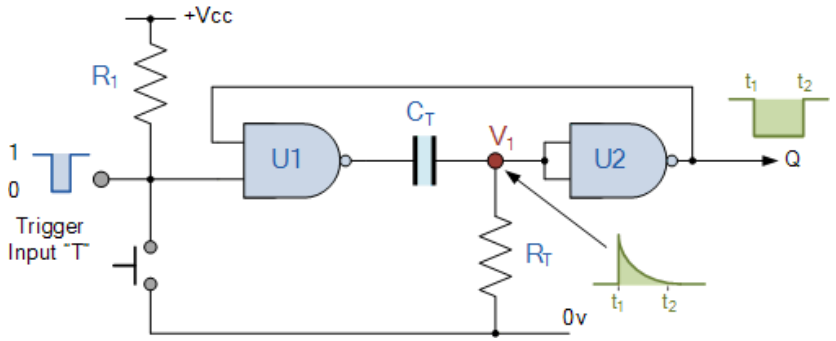


Summary (continued)

- **Both latch and flip-flop are a kind of multivibrators that operate between two states (0, 1).**
- **Three multivibrator types**
  - **Astable multivibrator has NO stable states.**
  - **Monostable multivibrator has only ONE stable state. By default it will stay in the stable state, but when triggered it will switch to unstable state (quasi-stable state)**
  - **Bistable multivibrator has TWO stable states.**



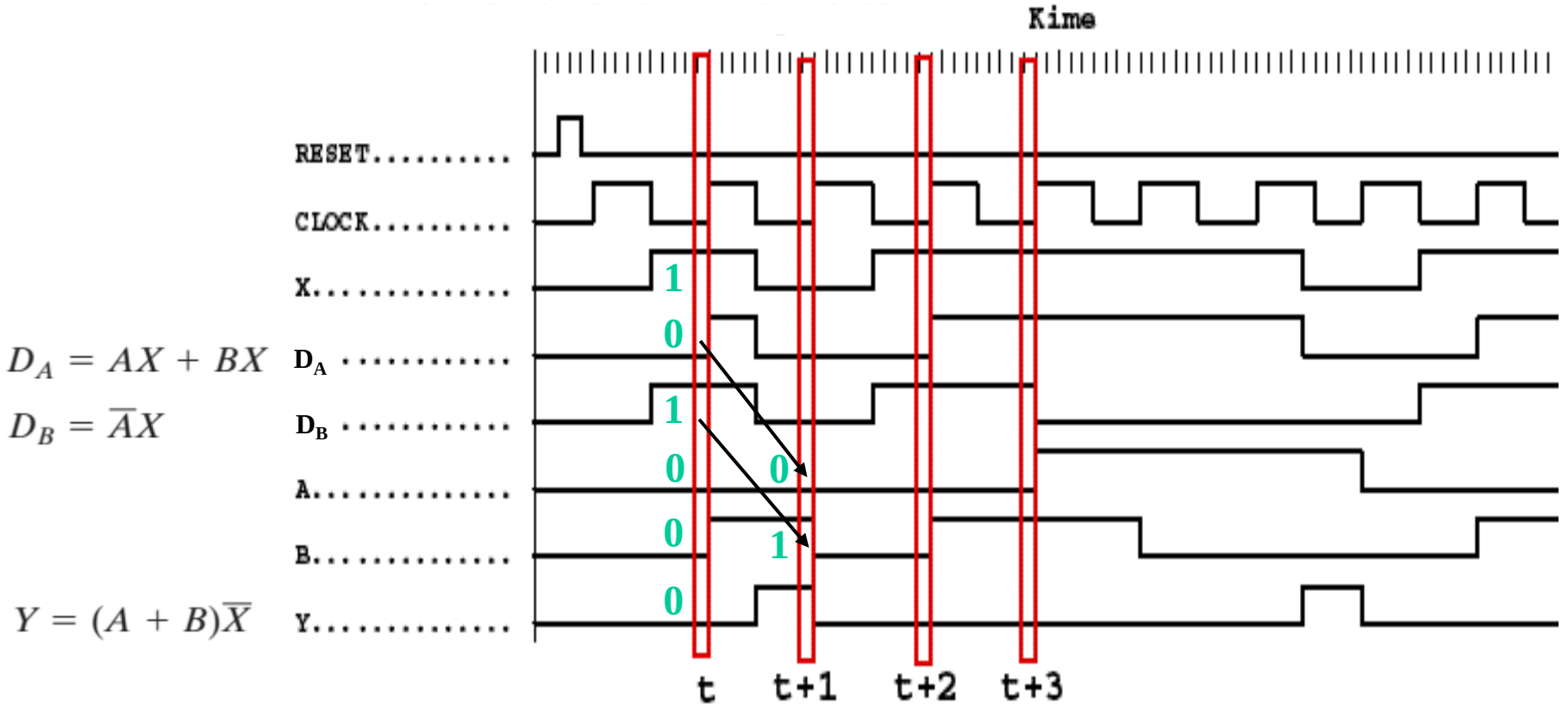
Astable multivibrator



Monostable multivibrator

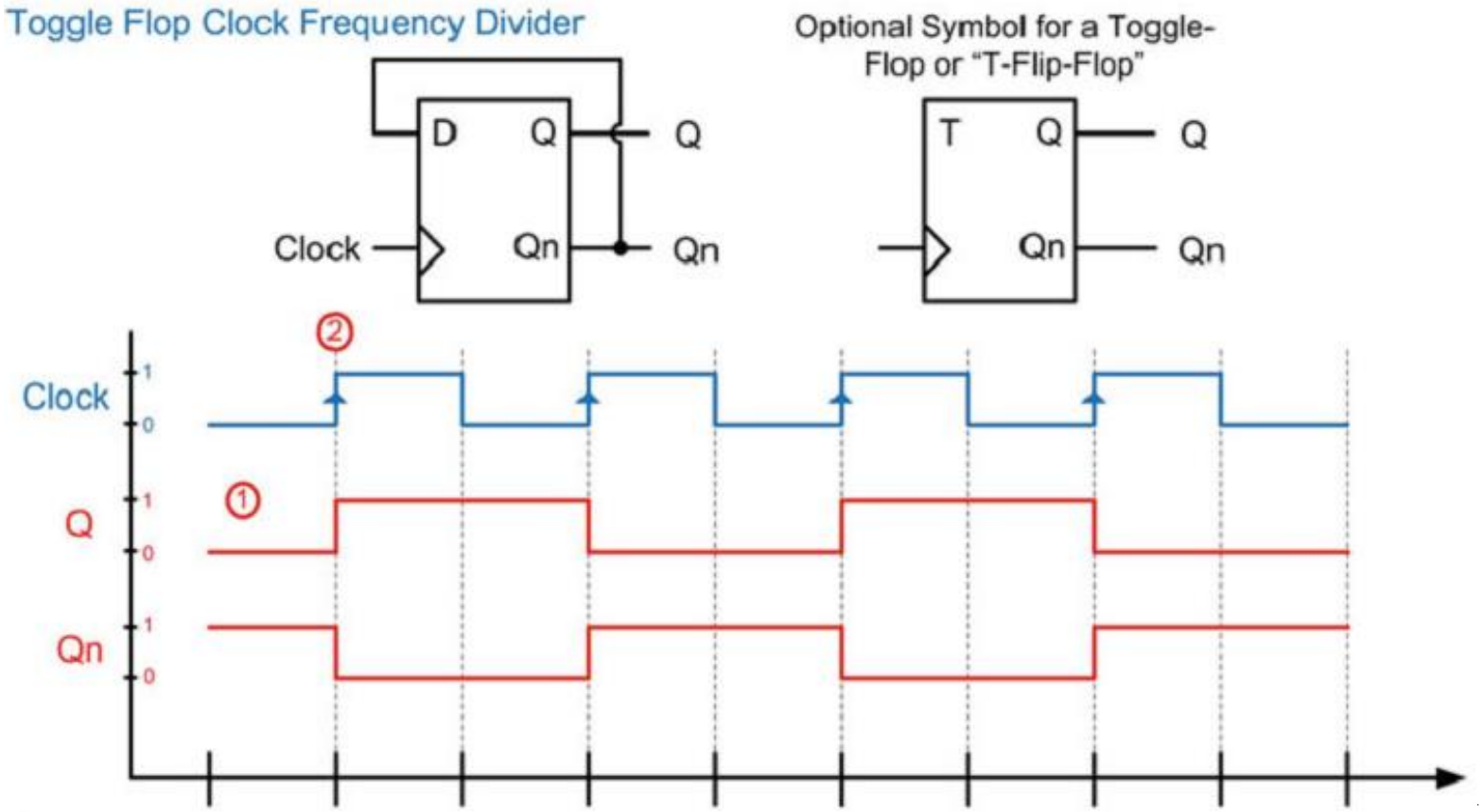
Example 1 (from Fig. 4-13) (continued)

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



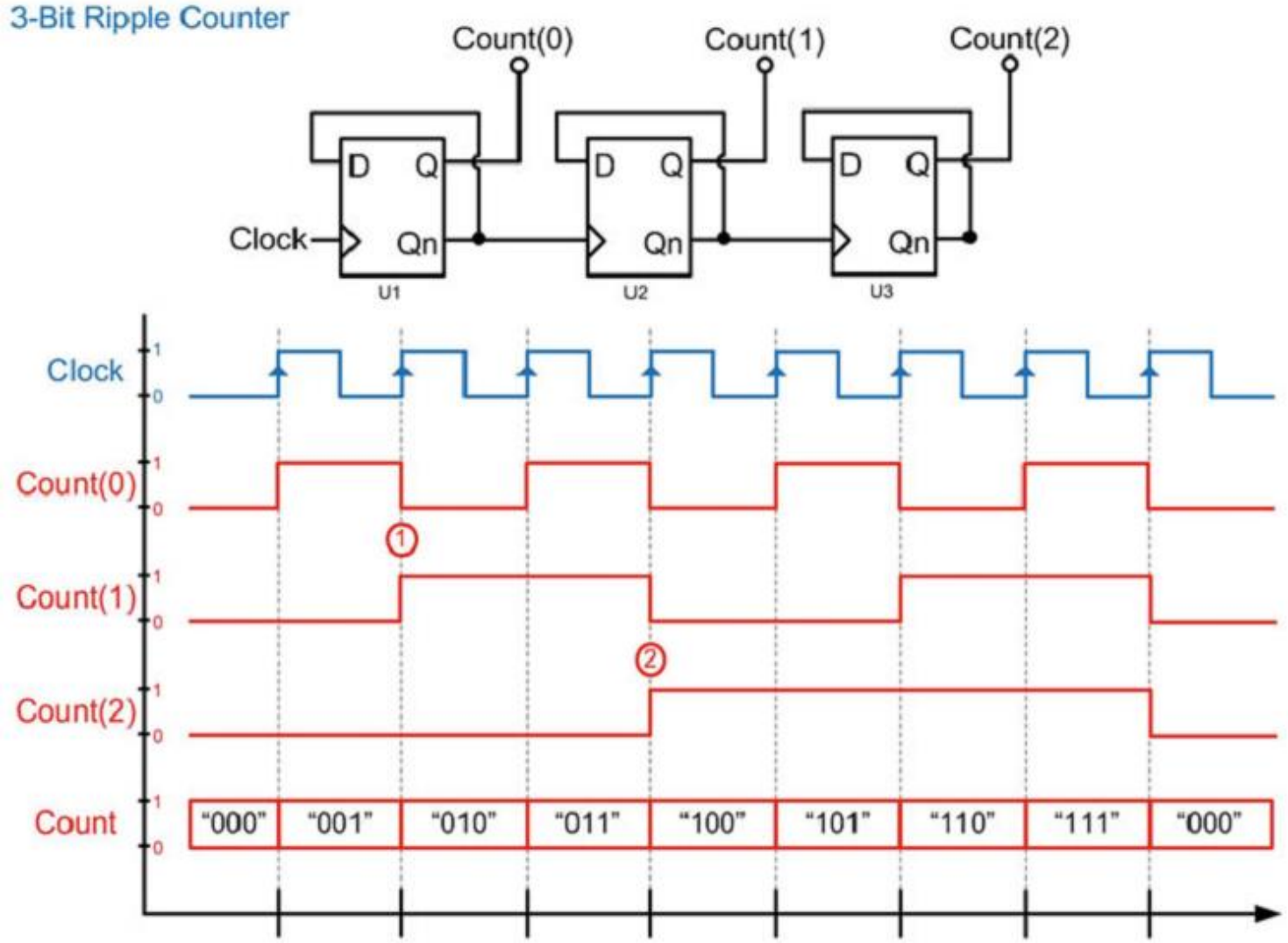
Appendix D: Circuits Based on Sequential Circuit

- **Clock Divider**
  - **A D Flip-Flop is configured with its Qn output wired back to its D, which produces outputs with half the frequency of the incoming clock.**



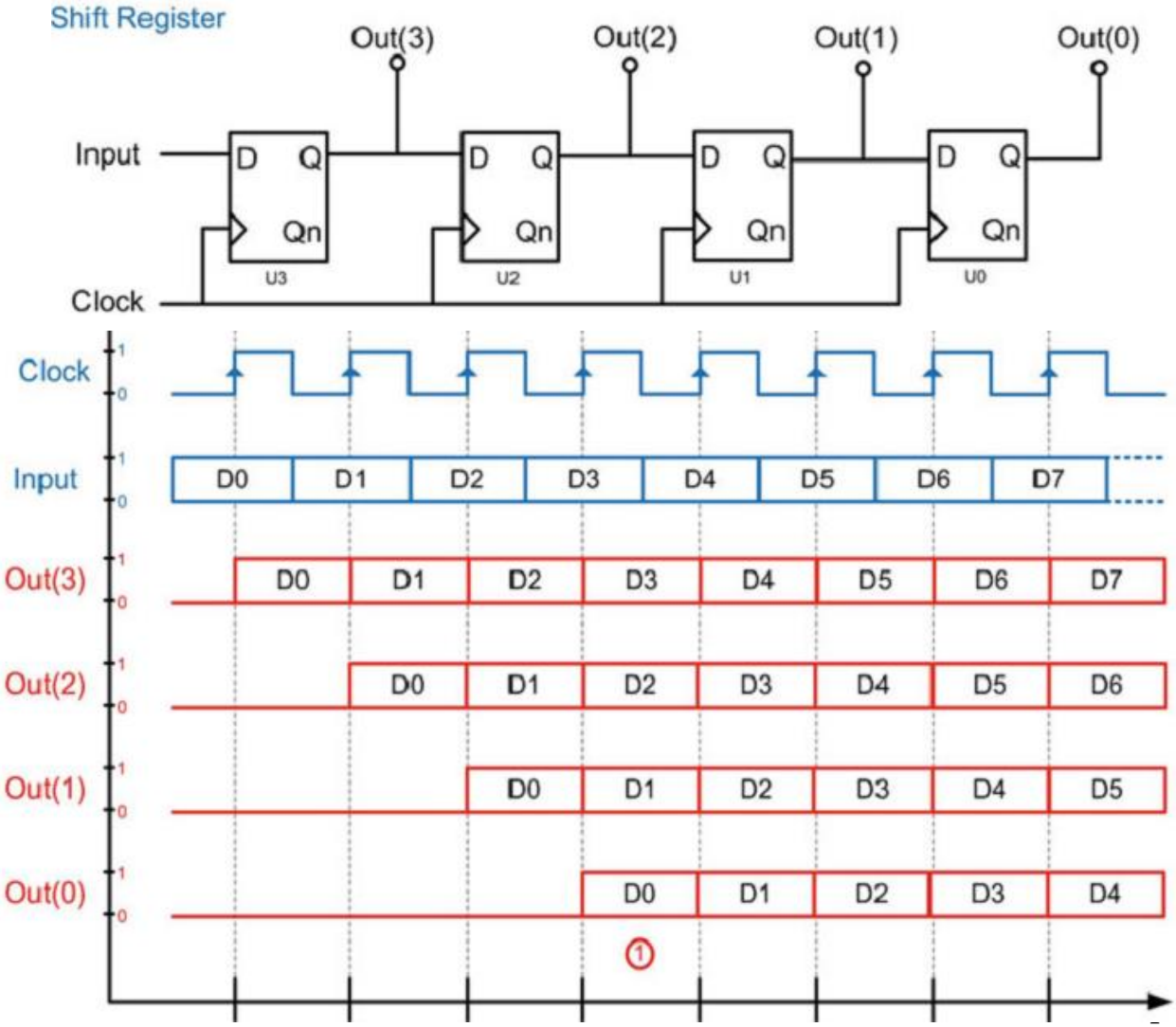
Circuits Based on Sequential Circuit (continued)

- **Ripple Counter (行波计数器)**



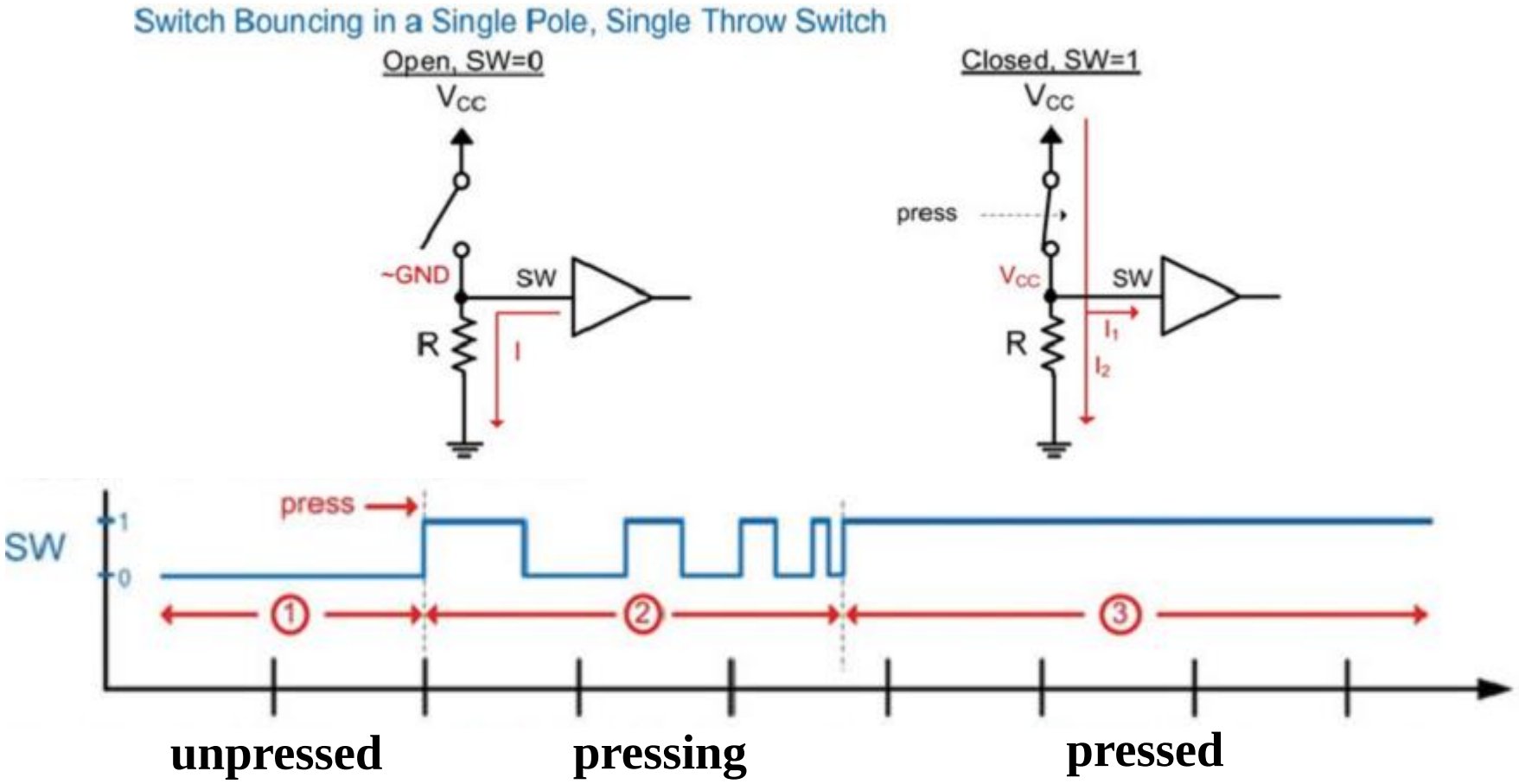
Circuits Based on Sequential Circuit (continued)

- **Shift Registers**



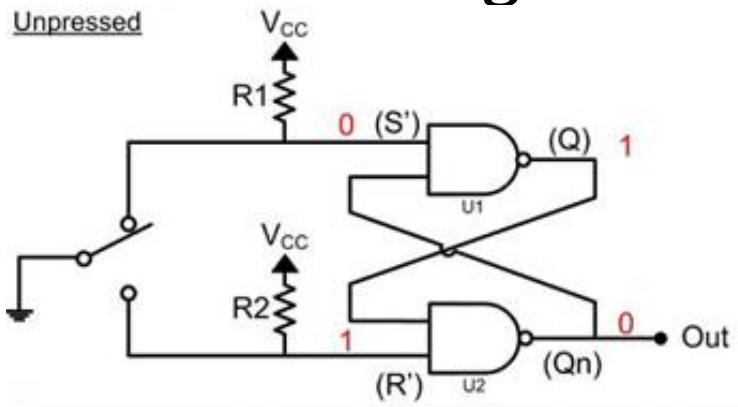
Circuits Based on Sequential Circuit (continued)

- **Switch Debouncing**



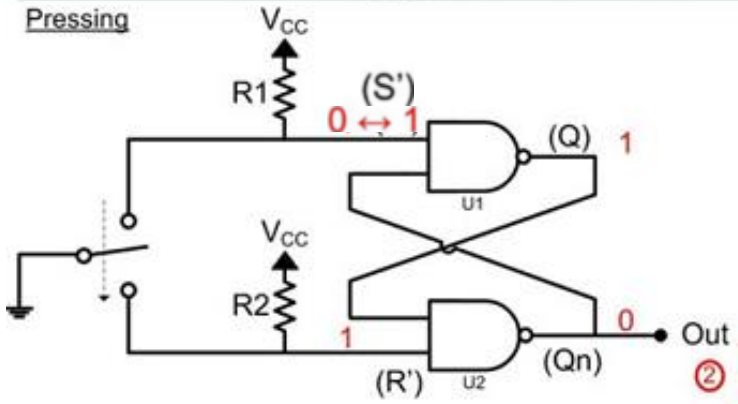
Circuits Based on Sequential Circuit (continued)

- **Switch Debouncing**



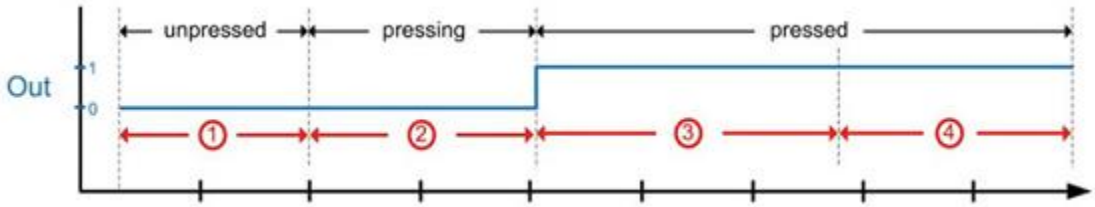
S'	R'	Q	Out Qn
0	0	1	1
0	1	1	0
1	0	0	1
1	1	Last Q	Last Qn

① The switch connects S' to GND and R2 pulls R' to V_{CC}, thus creating a solid Out=0.

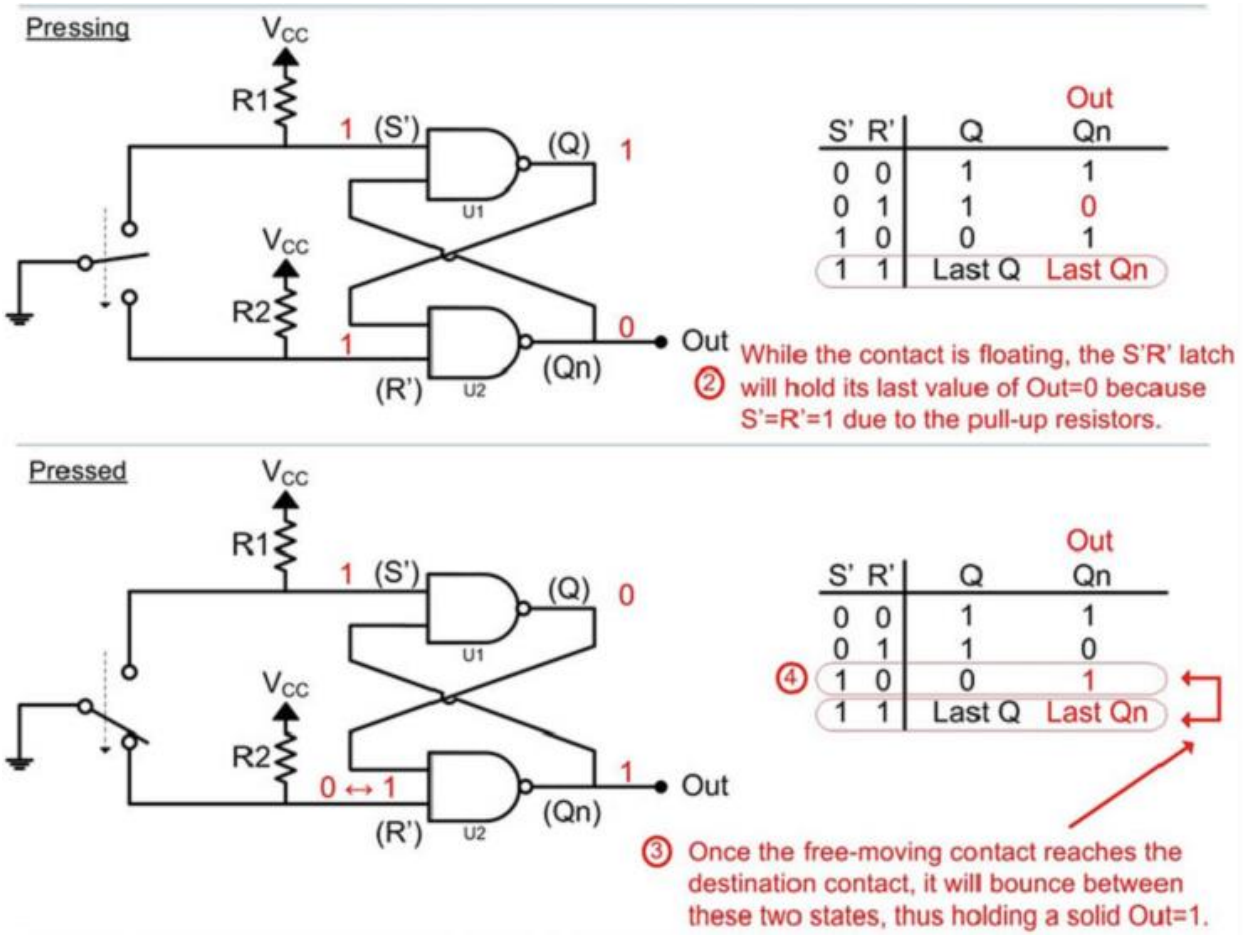


S'	R'	Q	Out Qn
0	0	1	1
0	1	1	0
1	0	0	1
1	1	Last Q	Last Qn

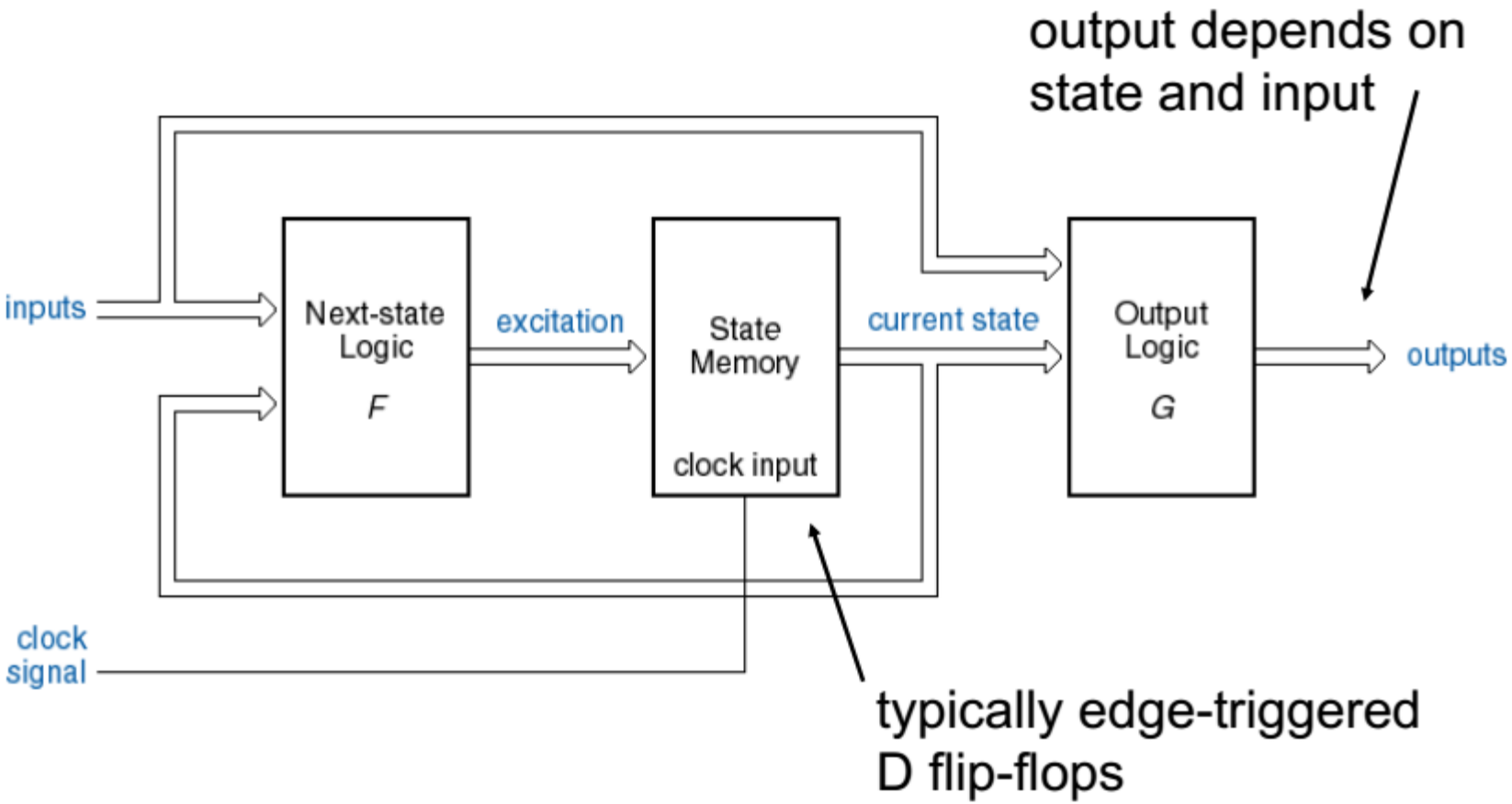
② While the contact is floating, the S'R' latch will hold its last value of Out=0 because S'=R'=1 due to the pull-up resistors.



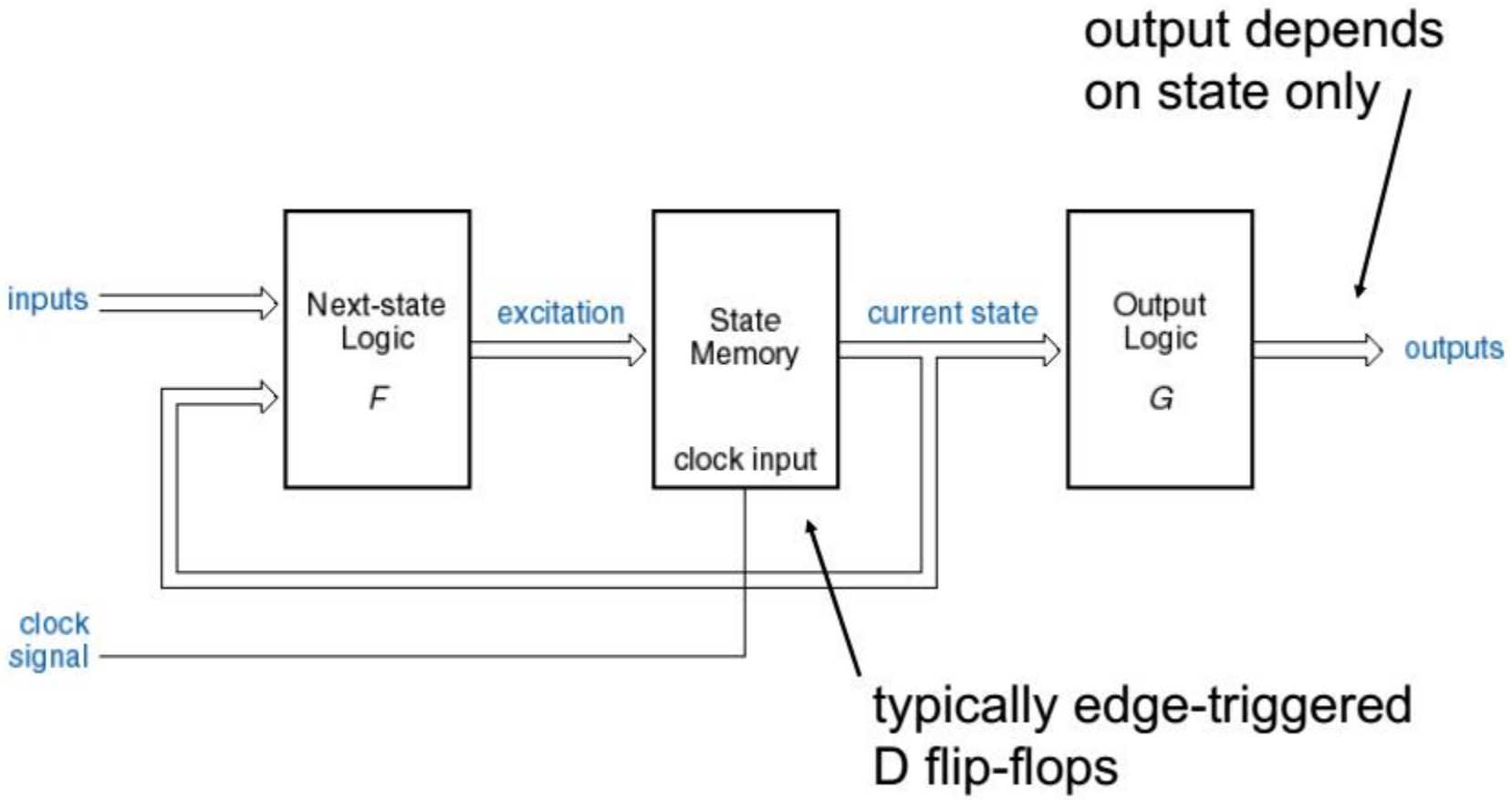
Circuits Based on Sequential Circuit (continued)



Appendix E: State Machine Structure (Mealy)



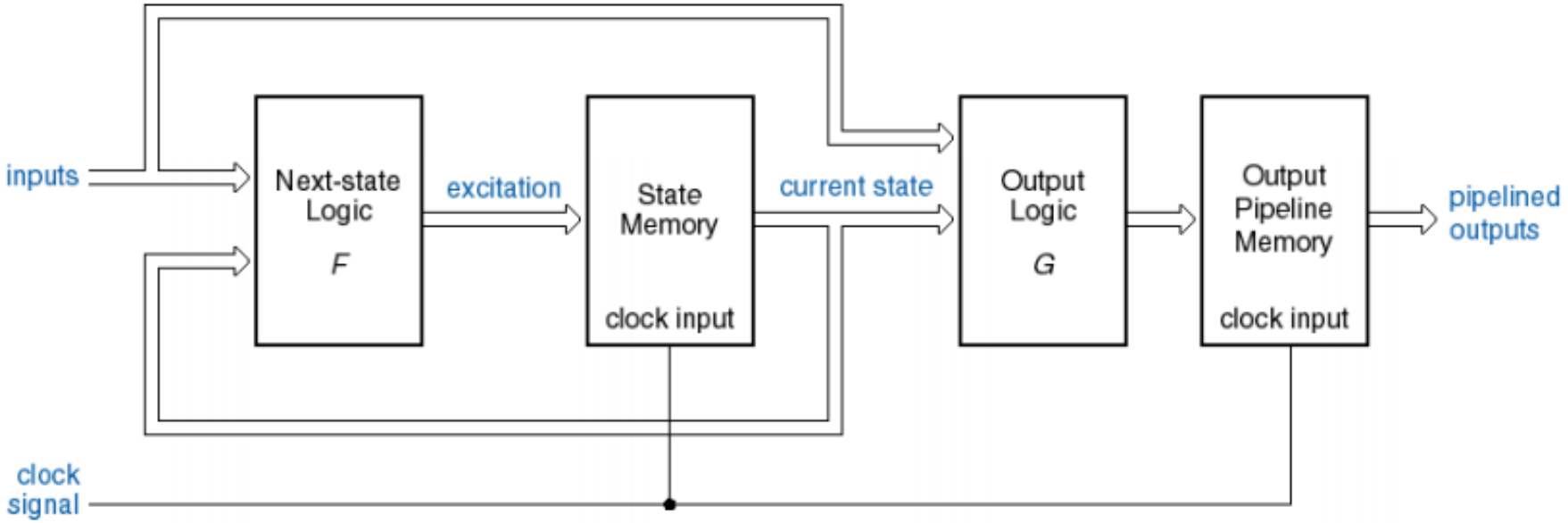
State Machine Structure (Moore)



When designing high-speed circuit, state variable can be used directly by the output pipeline, which means the output and the clock can be synchronized.

State Machine Structure (pipelined)

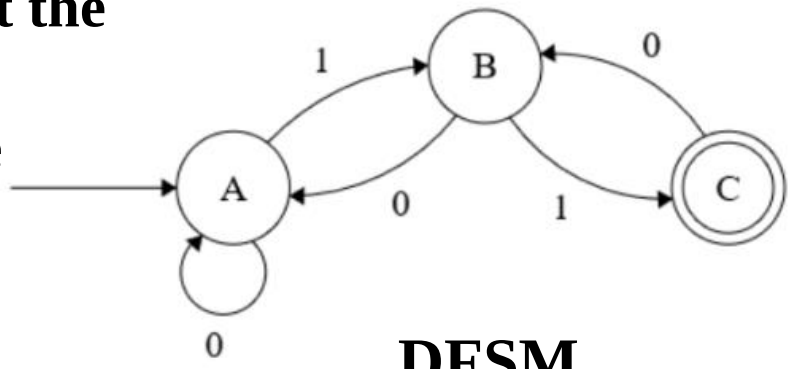
- **Structure of State Machine in pipeline--- Mealy Model**



When designing high-speed circuit, the output of state machine and the clock always need to be completely synchronized. One solution is to just use state variable as output signal.

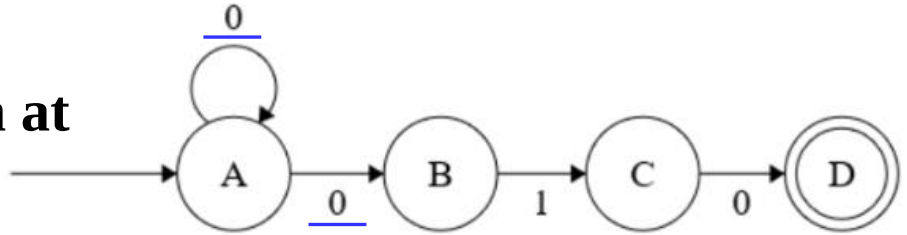
DFSM and NFSM

- ***Deterministic Finite State Machines*** **(DFSM)**
  - **Given the current we know what the next state will be**
  - **It has only one unique next state**
  - **It has no choices or randomness**



DFSM

- ***Non-deterministic Finite State Machine*** **(NFSM)**
  - **Given the current there could be multiple next states**
  - **The next state may be chosen at random**
  - **All the next states may be chosen in parallel**

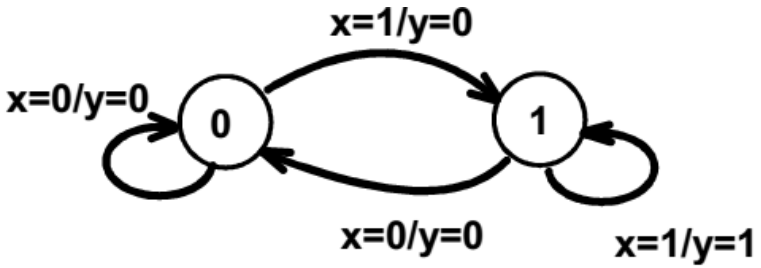


NFSM

Mealy and Moore models are two types of DFSM models.

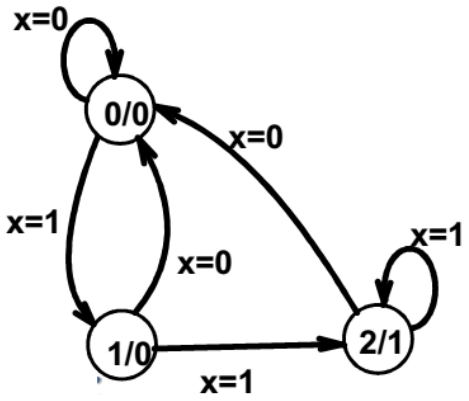
Moore and Mealy Example Tables

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



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

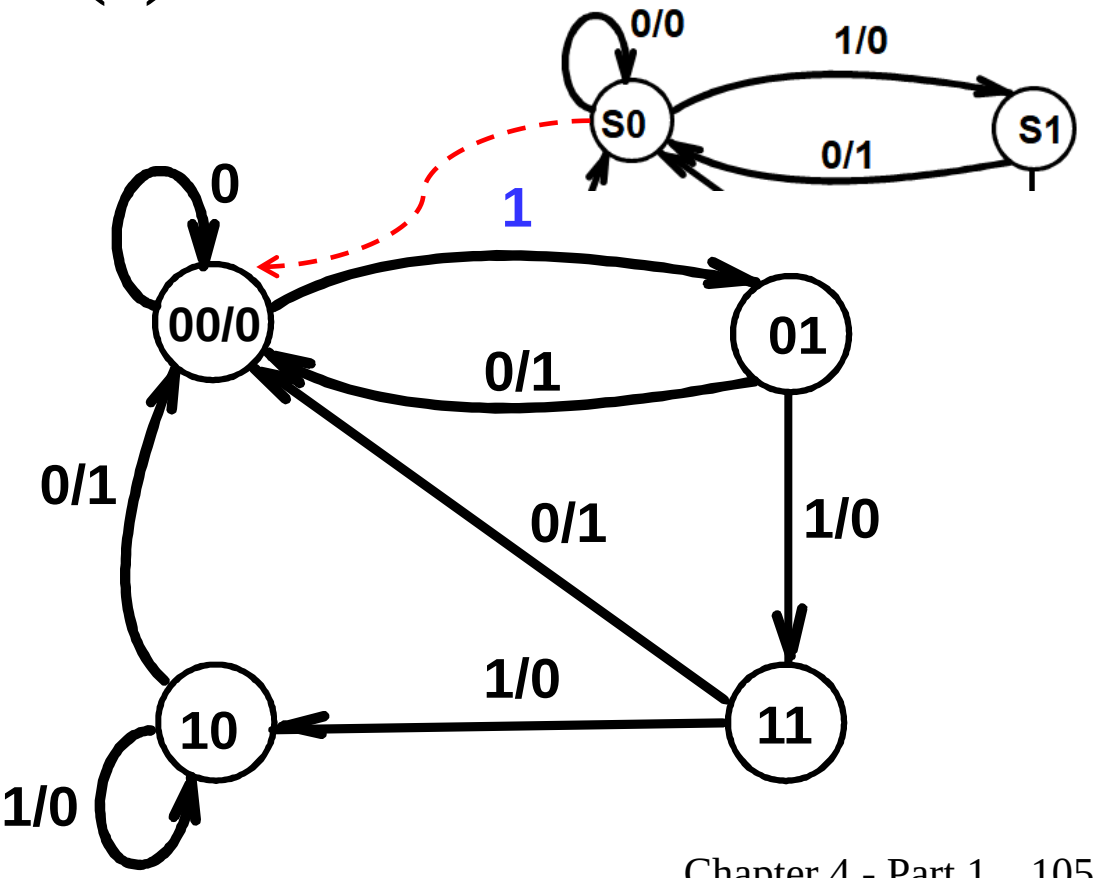
- **Moore Model state table maps states to outputs**



Present State	Next State x=0	Next State x=1	Output
0	0	1	0
1	0	2	0
2	0	2	1

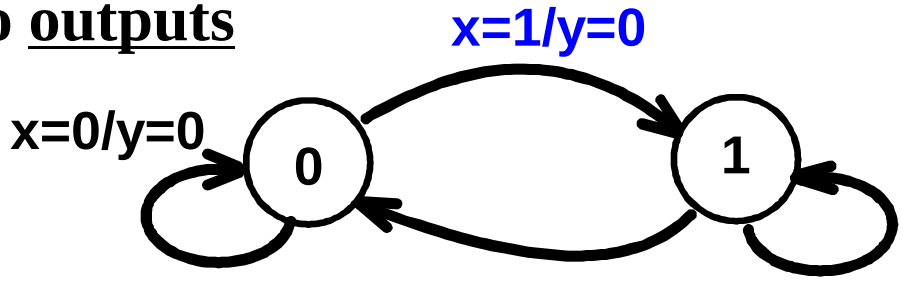
Mixed Moore and Mealy Outputs

- **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**

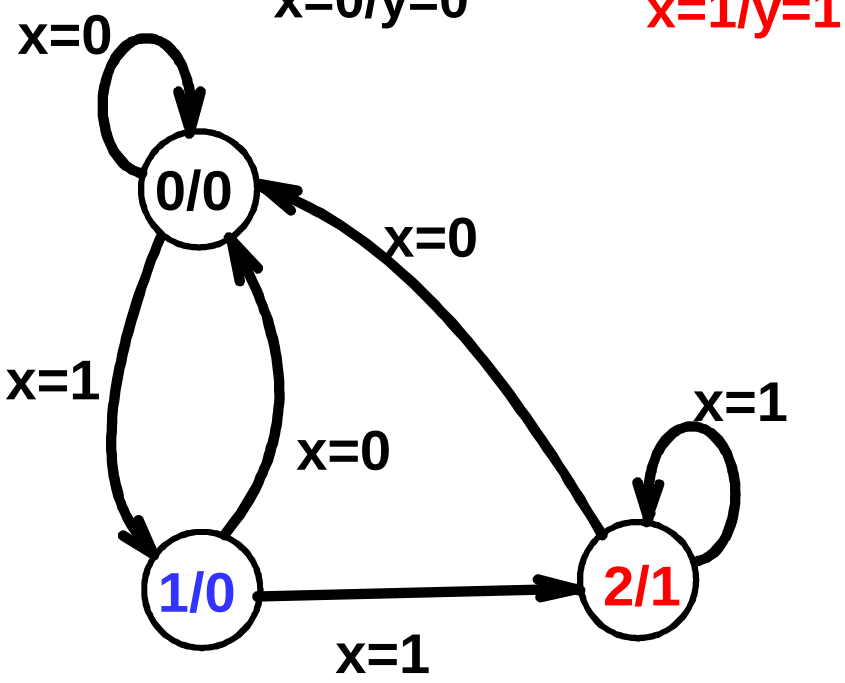


Conversion from Mealy to Moore

- **Mealy Model State Diagram maps <u>inputs and state</u> to <u>outputs</u>**



- **Moore Model State Diagram maps <u>states</u> to <u>outputs</u>**



- **Mealy models have fewer states than Moore models and react faster to input.**

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 ns**
  - t_s**(max) = 0.3 ns for edge-triggered flip-flops**
  - $t_s = t_{wH}$ **= 2.0 ns for master-slave flip-flops**
  - **Clock frequency = 250 MHz**
- **Calculations:** t_p **= 1/clock frequency = 4.0 ns**

Calculation of Allowable $t_{pd,COMB}$

- t_p = 1/clock frequency = 4.0 ns
  - Edge-triggered: $4.0 \geq 1.0 + t_{pd,COMB} + 0.3$,

    $t_{pd,COMB} \leq 2.7$ ns
  - Master-slave: $4.0 \geq 1.0 + t_{pd,COMB} + 2.0$,

    $t_{pd,COMB} \leq 1.0$ ns
- Comparison: Suppose that for a gate, t_{pd} = 0.3 ns
  - Edge-triggered: 9 gates allowed on a path
  - Master-slave: 3 gates allowed on a path