

# **ECE 341**

## **Lecture # 3**

**Instructor: Zeshan Chishti**  
**zeshan@ece.pdx.edu**

**October 6, 2014**

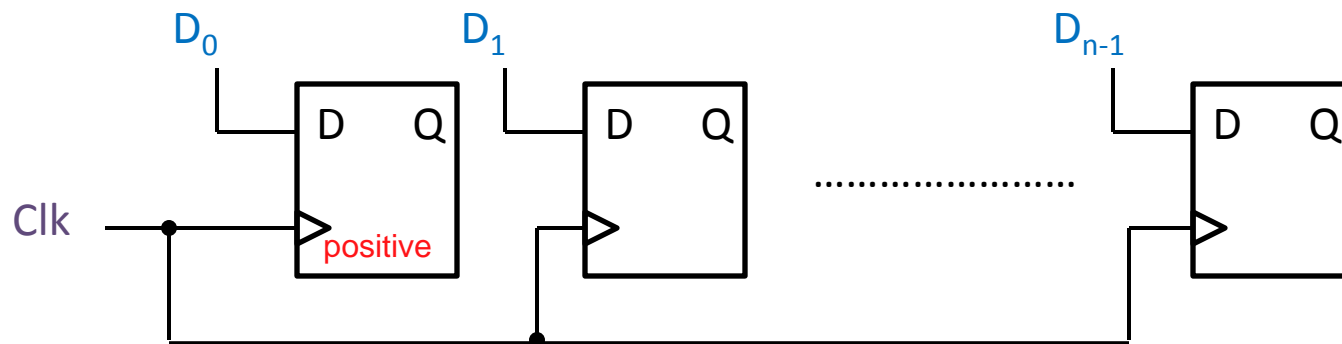
**Portland State University**

# Lecture Topics

- Registers
- Counters
- Finite State Machines
- Reference: Appendix A of the textbook, sections A.7, A.8, A.13.

# Registers

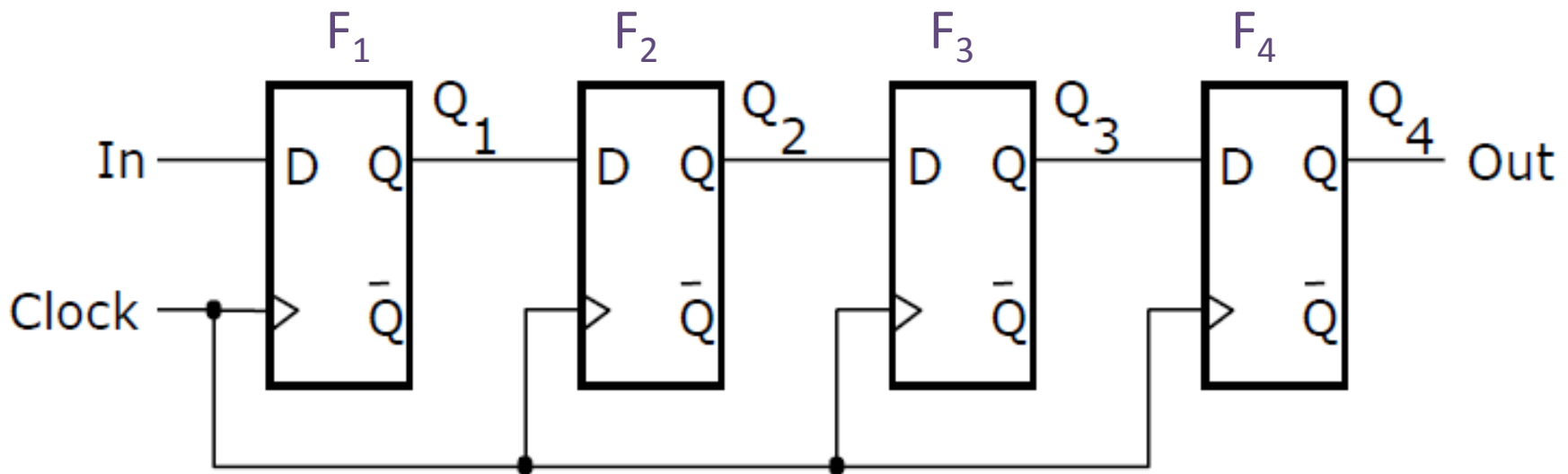
- A flip-flop stores one bit of information
- When a set of  $n$  flip-flops is used together to store  $n$  bits of data, it is referred to as a ***n-bit register***
- All flip-flops in a register are **synchronized** by a common clock
  - Data loaded and stored into all flip-flops at the same time
- Common register usage:
  - Temporary storage of data output from an arithmetic circuit



**Write operation in a  $n$ -bit register**

# Shift Register

- A register whose contents may be shifted one bit position at a time
  - To the right or left or possibly both
- Example: Shift Right Register
  - At each positive clock edge, contents of  $F_i$  are shifted to  $F_{i+1}$  (right shift)
  - Gated latch unsuitable for a shift register, no control over # bit shifts per clock

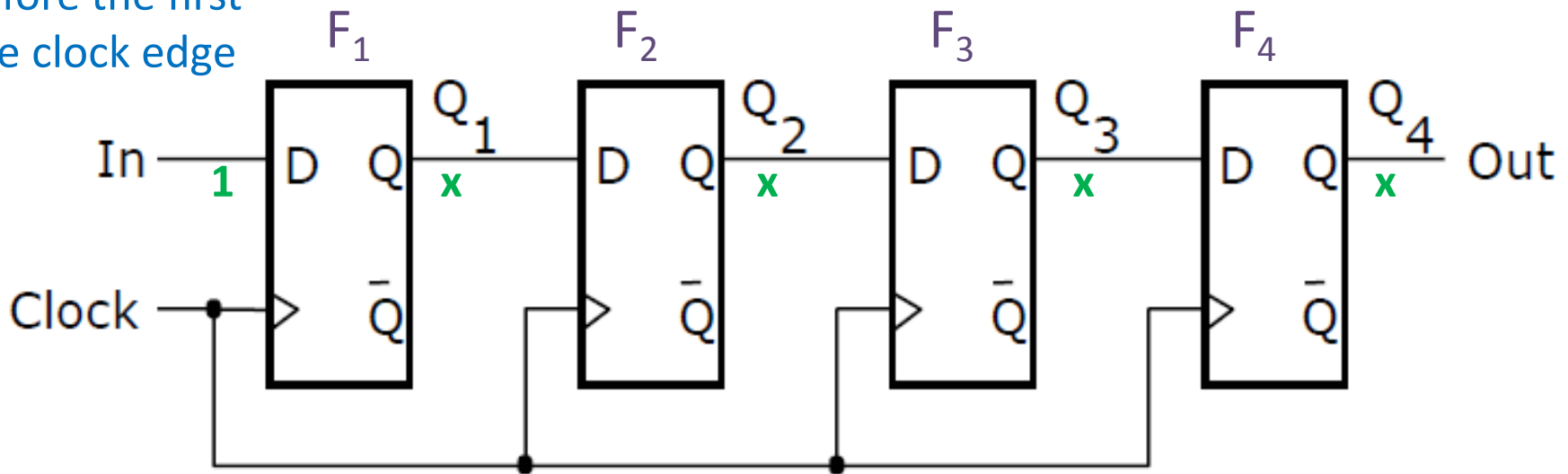


**Shift Right Register**

# Example: Shift Right Register

- Example: Data 1001 needs to be written to a 4-bit shift right register

Before the first  
+ve clock edge

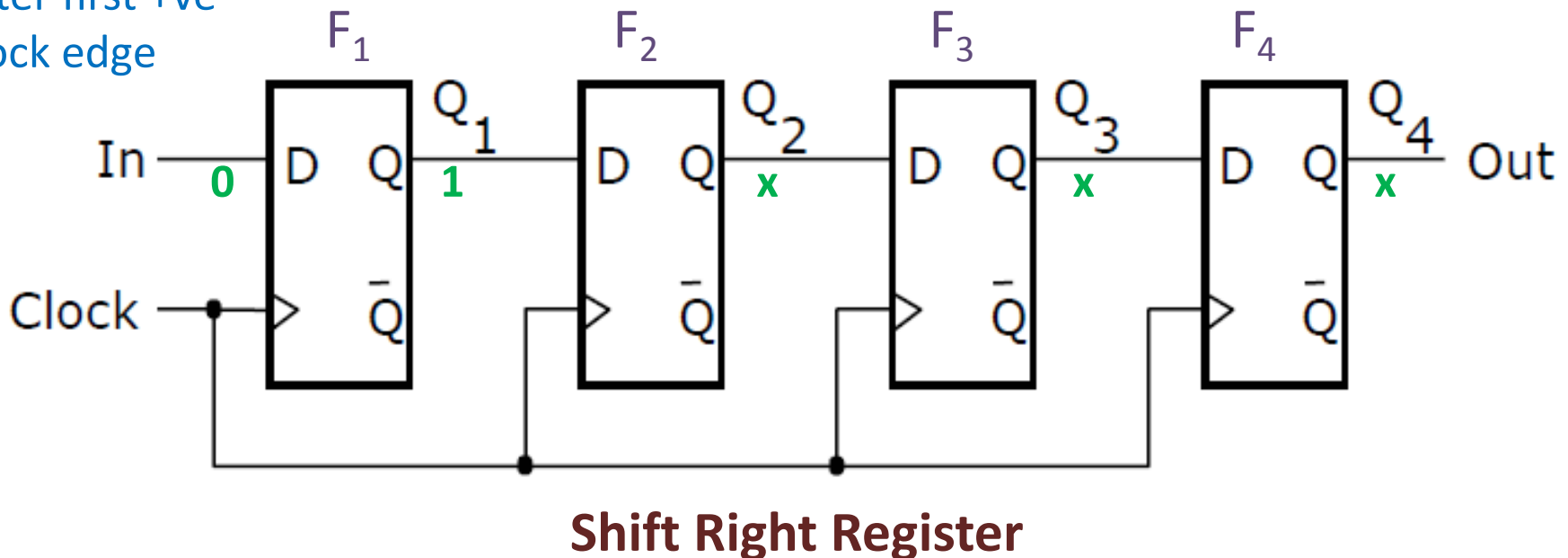


Shift Right Register

# Example: Shift Right Register

- Example: Data 1001 needs to be written to a 4-bit shift right register

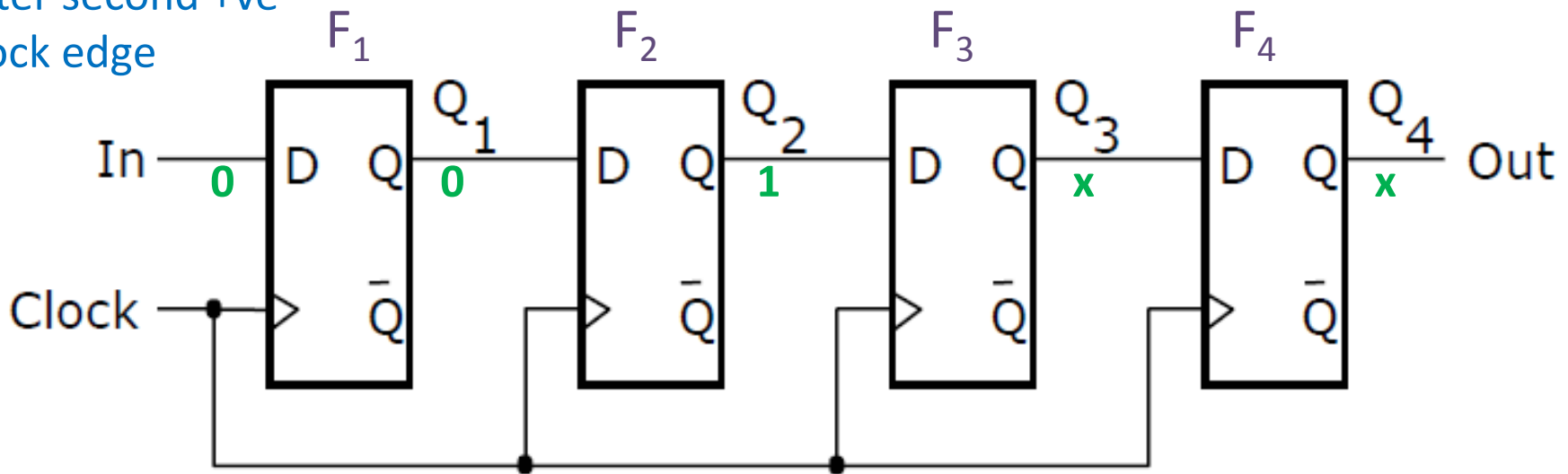
After first +ve  
clock edge



# Example: Shift Right Register

- Example: Data 1001 needs to be written to a 4-bit shift right register

After second +ve  
clock edge

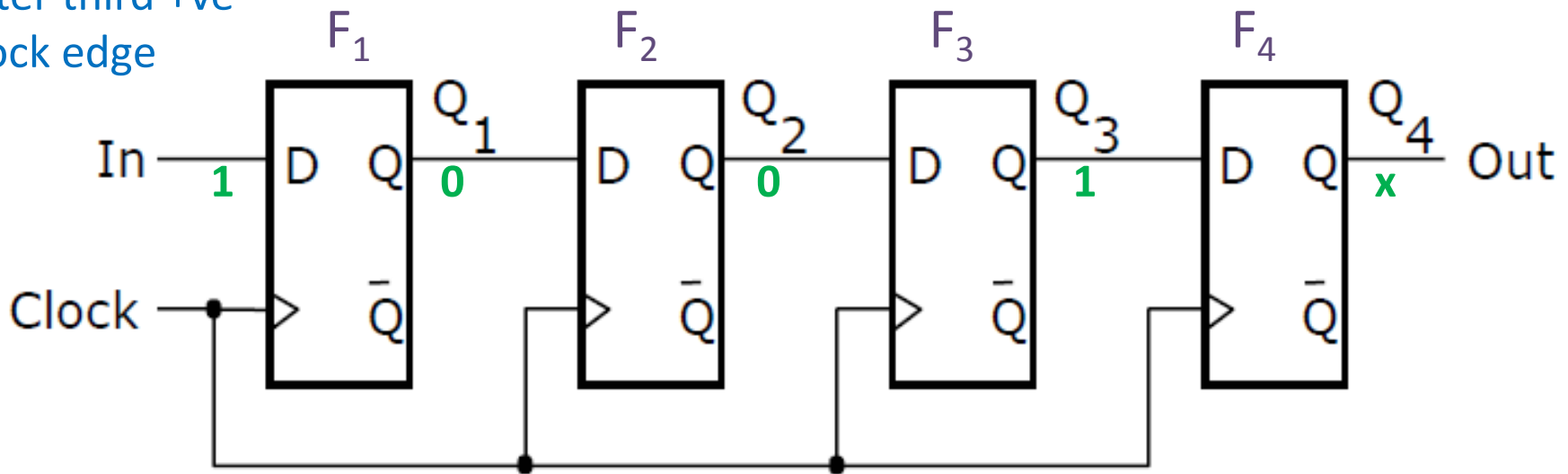


Shift Right Register

# Example: Shift Right Register

- Example: Data 1001 needs to be written to a 4-bit shift right register

After third +ve  
clock edge



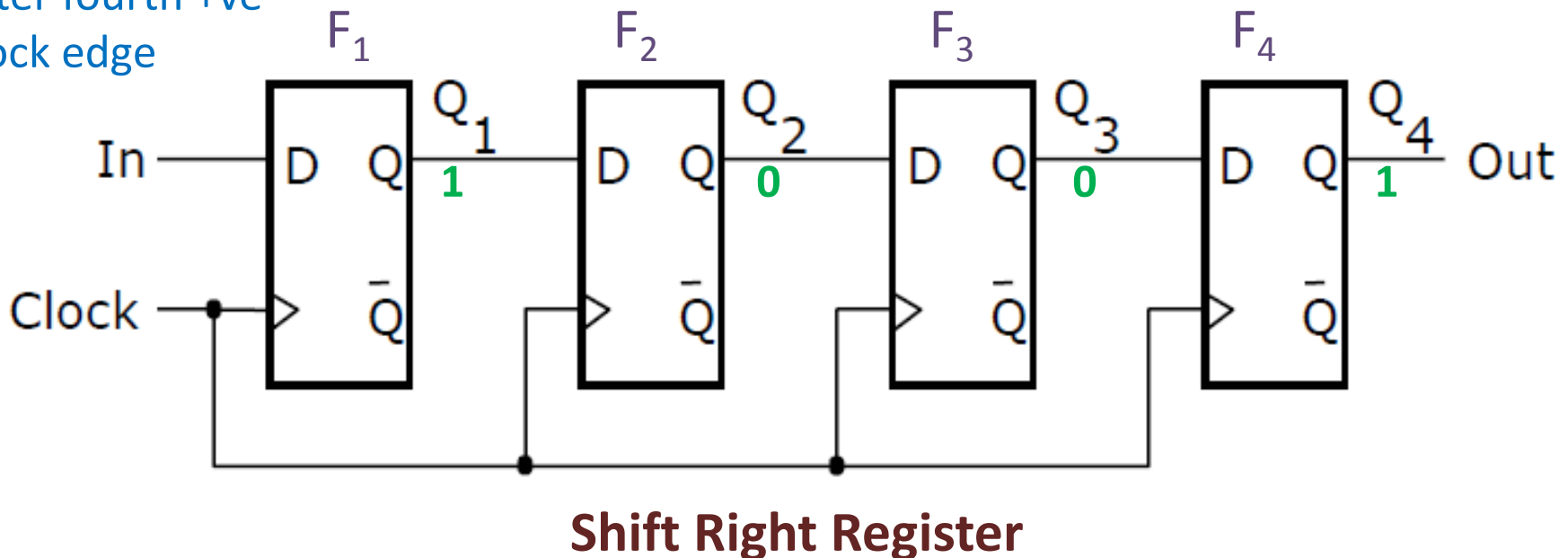
Shift Right Register



# Example: Shift Right Register

- Example: Data 1001 needs to be written to a 4-bit shift right register

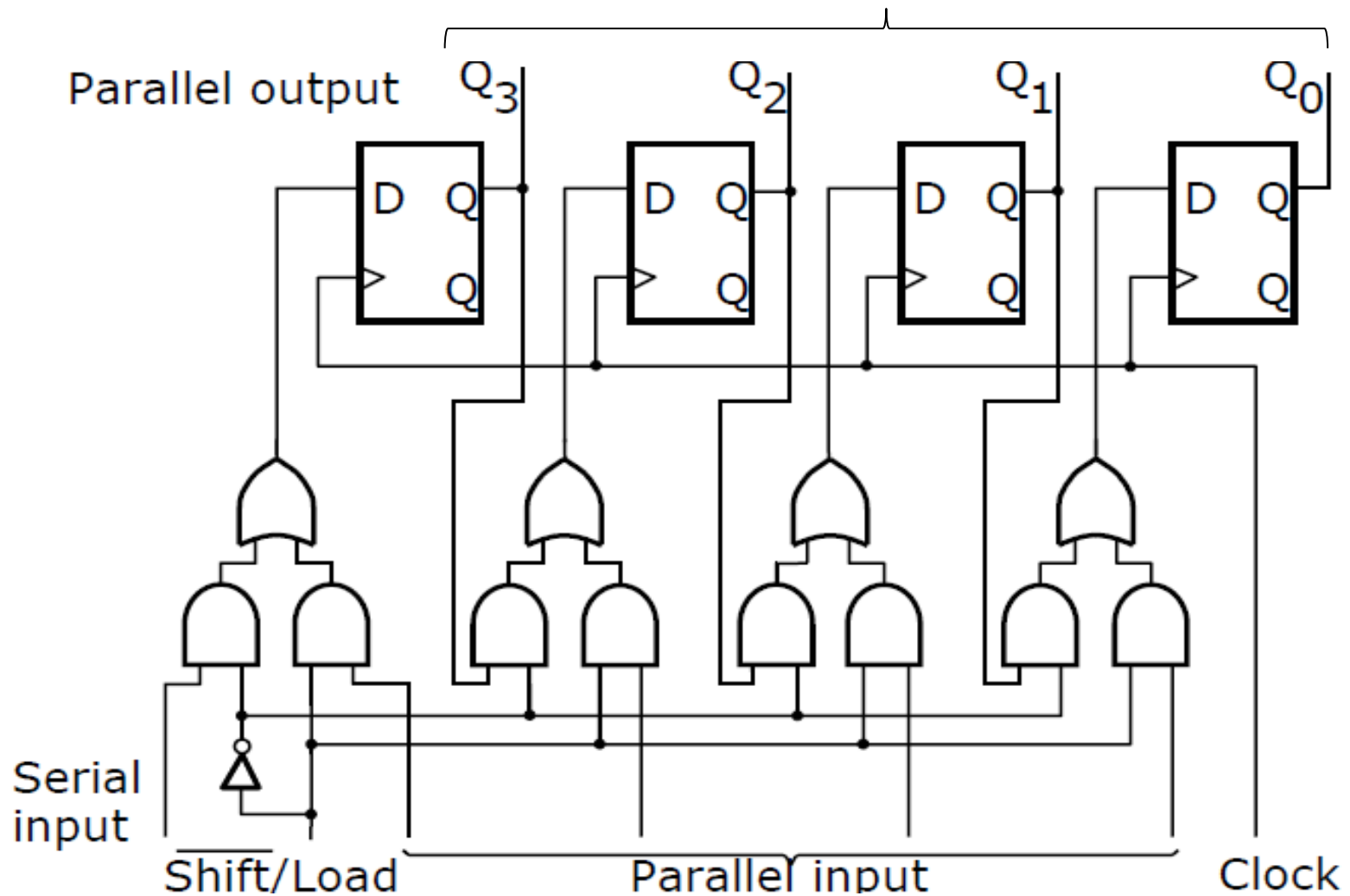
After fourth +ve  
clock edge



# Parallel Access Shift Register

- Data transfer in computer systems is of two types:
  - If the transfer is **1-bit at a time**, it is said to be **serial**
  - If the transfer is **n-bits ( $n > 1$ ) at a time**, it is said to be **parallel**
- To read/write a register in *serial* fashion, data is read/written bit-by-bit and is shifted by one bit position each cycle
- To read/write a register in *parallel* fashion, all the n-bits are read/written during the same clock cycle

# Parallel Access Shift Register



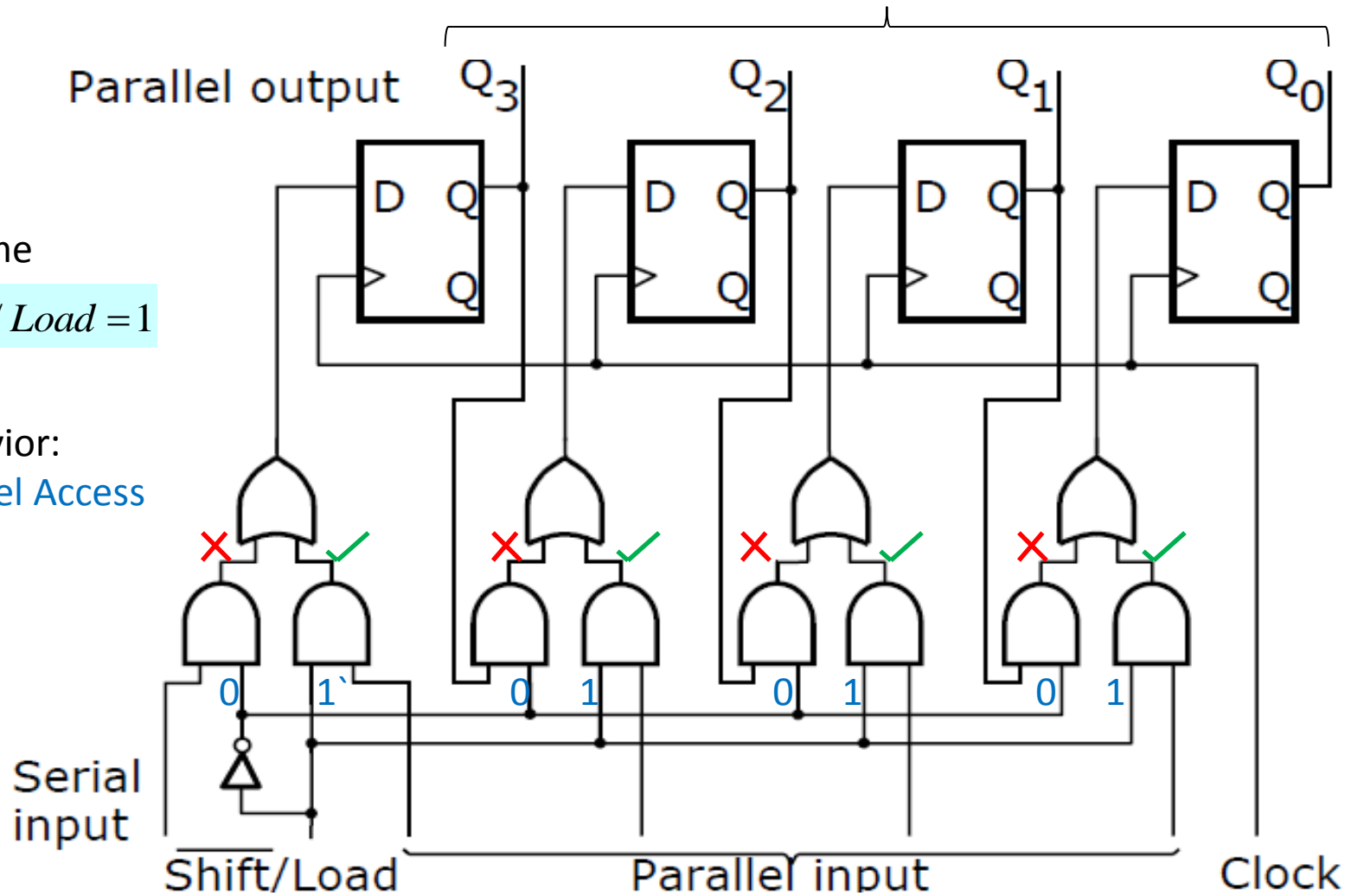
# Parallel Access Shift Register

Assume

$\overline{\text{Shift}} / \text{Load} = 1$

Behavior:

Parallel Access



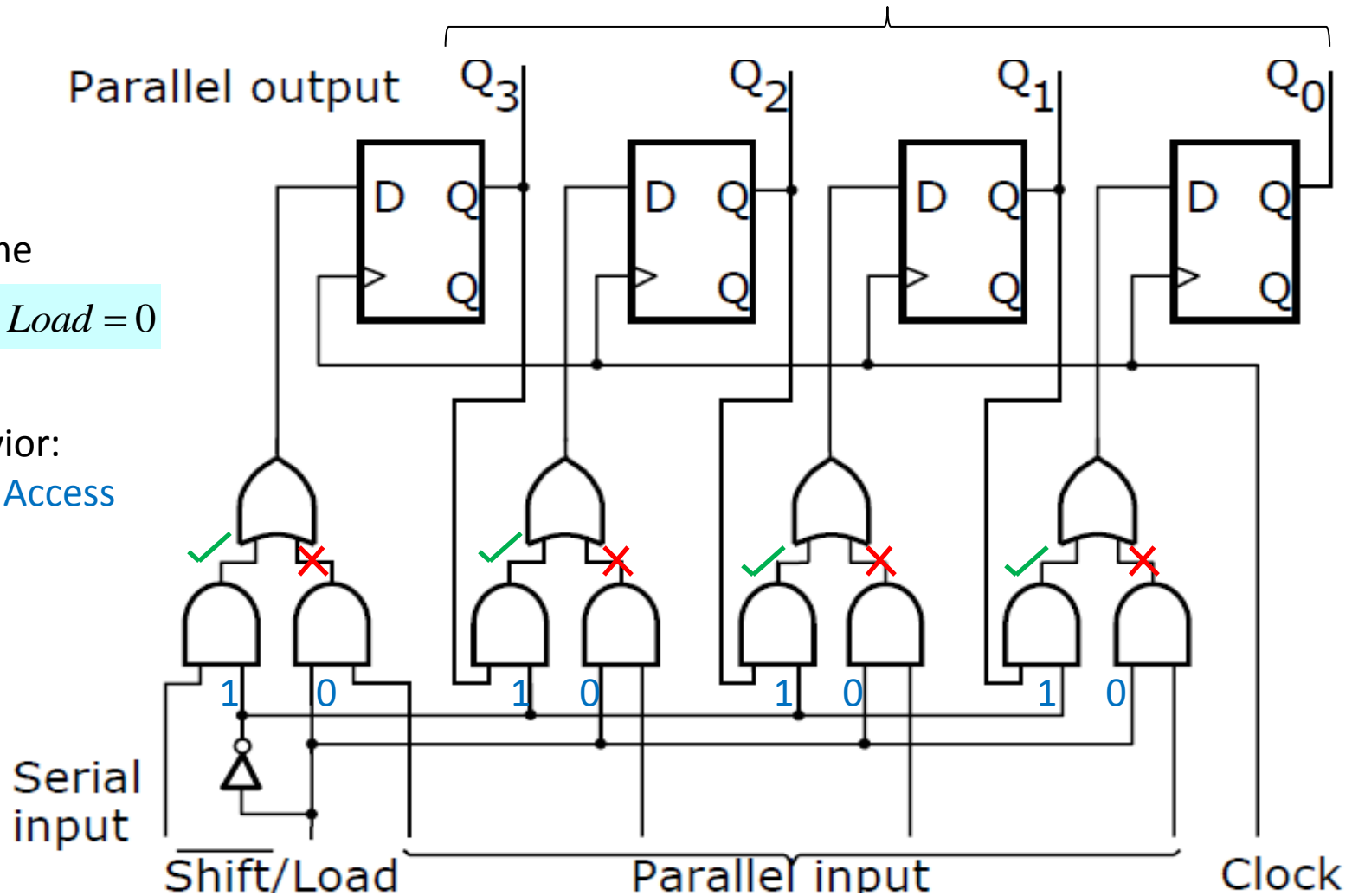
# Parallel Access Shift Register

Assume


$\overline{\text{Shift}} / \text{Load} = 0$

Behavior:

Serial Access



# Counters

- Counters are arithmetic circuits used for the purpose of *counting*
  - Can increment or decrement by 1 each cycle
- Counters often implemented with **T flip-flops**
  - Toggle feature naturally suited for counting operation
- Applications of counters
  - Count occurrences of certain events, for example, no. of add instructions
  - Track elapsed time between events
  - Generate control and timing signals, for example, to produce signals whose frequencies are **multiples of original clock frequency** 

# A 3-bit Up-counter

LSB

Consider a 3-bit counter

$x_2x_1x_0$  shown in table.

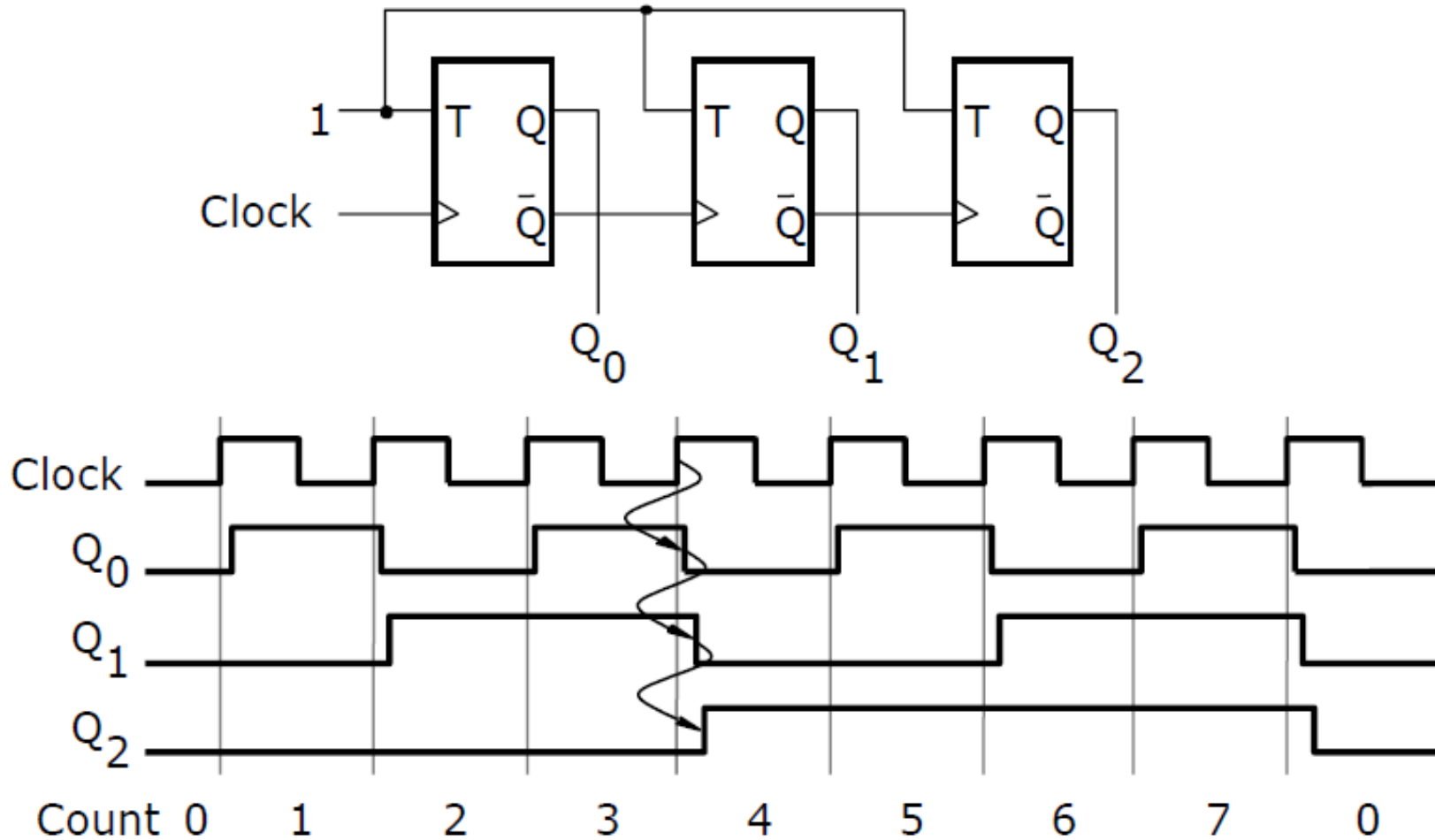
- The least significant bit  $x_0$  toggles at every increment of counter

- $x_1$  toggles on 1- $\rightarrow$ 0 transitions of  $x_0$  (half the rate of toggling of  $x_0$ )

- $x_2$  toggles on 1- $\rightarrow$ 0 transitions of  $x_1$  (half the rate of toggling of  $x_1$ )

Counter Value	$x_2$	$x_1$	$x_0$
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

# A 3-bit Up-counter





# Practice Exercise

- Design a 3-bit down counter with T flip-flops. How does the down counter circuit differ from the up-counter circuit?

# A 3-bit Down-counter

- The least significant bit  $x_0$  toggles at every decrement of counter
- $x_1$  toggles on 0- $\rightarrow$ 1 transitions of  $x_0$
- $x_2$  toggles on 0- $\rightarrow$ 1 transitions of  $x_1$

Counter Value	$x_2$	$x_1$	$x_0$
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

# Practice Exercise Solution

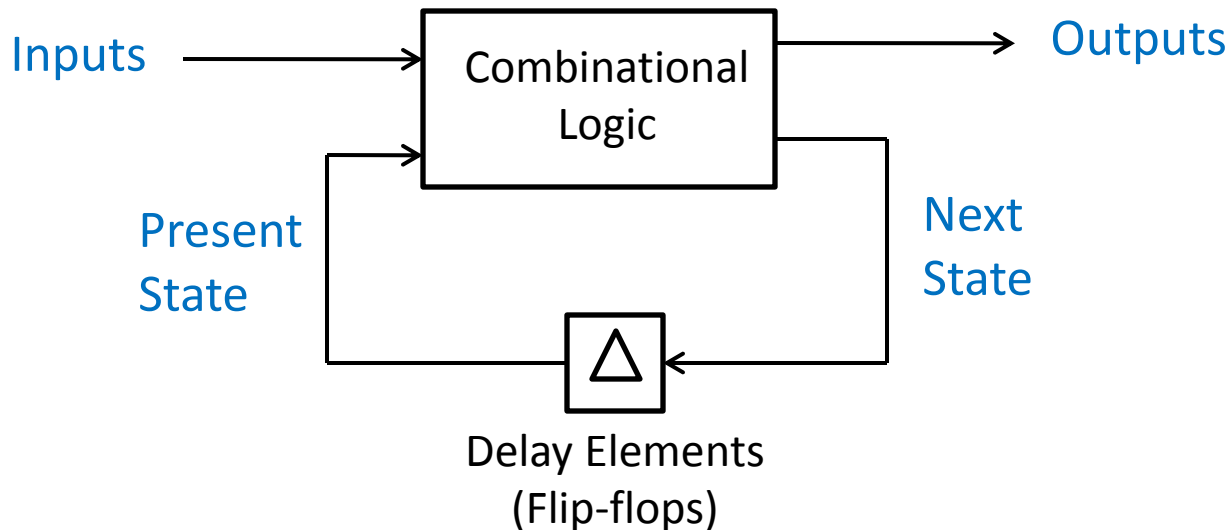
- Design a 3-bit down counter with T flip-flops. How does the down counter circuit differ from the up-counter circuit?
- Solution:
  - Two ways to convert the up-counter to a down-counter:
    1. Either, replace the +ve edge-triggered T flip-flops with –ve edge-triggered T flip-flops
    2. Or, connect the Q outputs (instead of NOT(Q) outputs) from previous flip-flops to clock inputs of next flip-flops

# Asynchronous Counters

- The previous counter is an example of ***asynchronous*** counters. Also called ***ripple*** counters
  - Input clock only connected to one flip flop
  - Clocks for other flip-flops are derived from outputs of previous flip-flops
- Asynchronous counters are slow because of cascaded clocking
  - The input clock pulse ripples from stage to stage
  - **Propagation** delay of individual flip-flops **limit speed** of operation
- Solution: Synchronous sequential circuits (finite state machines)

# Finite State Machines

- Recall that in a sequential circuit:
  - Outputs depend both on present inputs and the sequence of previous inputs
- The **state** of a sequential circuit determines its behavior when various input patterns are applied
- A **finite state machine** model formally describes a sequential circuit



# Synthesis of Finite State Machines

Synthesis of FSM involves the following steps:

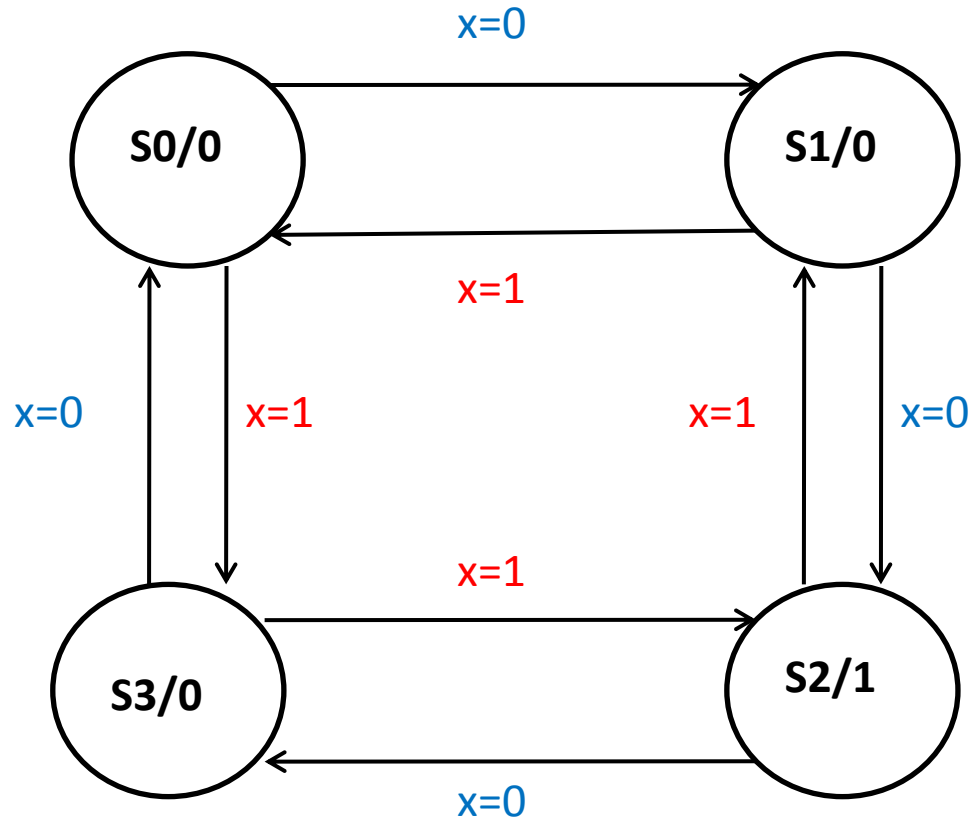
- [Step 1:](#) Develop a *state diagram* or *state table*
  - Depict how state transitions occur in response to input patterns
- [Step 2:](#) Determine # and types of needed flip flops
- [Step 3:](#) Determine *state assignment* (flip-flop values for each state)
- [Step 4:](#) Determine the state-assigned state table
- [Step 5:](#) Derive the logic expressions for next-state logic and outputs
- [Step 6:](#) Use the derived expressions to implement the circuit

# Example: Up/Down Counter with D flip-flops

Problem Statement: Design a mod-4 counter which counts up or down depending on an input and has an output of 1 if the count is equal to 2

- States: 4 states (S0, S1, S2 and S3) corresponding to 4 count values
- Input: Variable x. Count up if  $x=0$ , down if  $x=1$
- Output: Variable z. If present state is S2, then  $z=1$ , otherwise  $z=0$ ;

# State Diagram





# State Table

Present State	Next State		Output z
	x = 0	x = 1	
S0	S1	S3	0
S1	S2	S0	0
S2	S3	S1	1
S3	S0	S2	0

Need 2 state variables to represent 4 states => use 2 D flip-flops

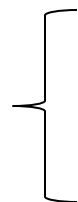
# State-Assigned State Table

- State variables  $y_1$  and  $y_2$  used to express each state as a 2-bit number  $y_2y_1$
- We choose the following state assignment  $S0=00$ ,  $S1=01$ ,  $S2=10$ ,  $S3=11$

Present State	Next State		Output $z$
	$x = 0$	$x = 1$	
$y_2y_1$	$Y_2Y_1$	$Y_2Y_1$	
00	01	11	0
01	10	00	0
10	11	01	1
11	00	10	0

Logic

Expressions



Next state

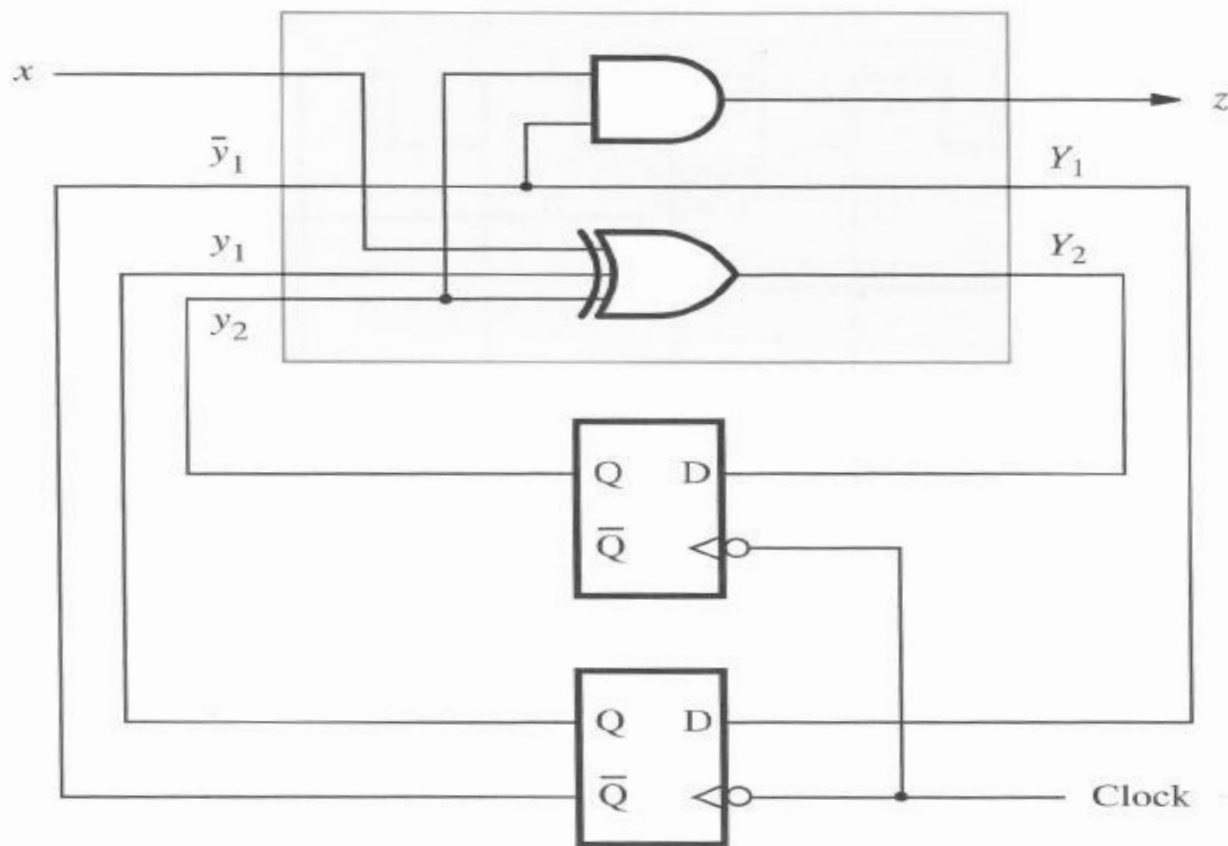
Output

$$Y_2 = y_2 \oplus y_1 \oplus x$$

$$Y_1 = \overline{y_1}$$

$$z = y_2 \overline{y_1}$$

# Logic Circuit



**Logic Expressions**

Next state

Output

$$Y_2 = y_2 \oplus y_1 \oplus x$$

$$z = y_2 \bar{y}_1$$

$$Y_1 = \bar{y}_1$$