

DIGITAL LOGIC

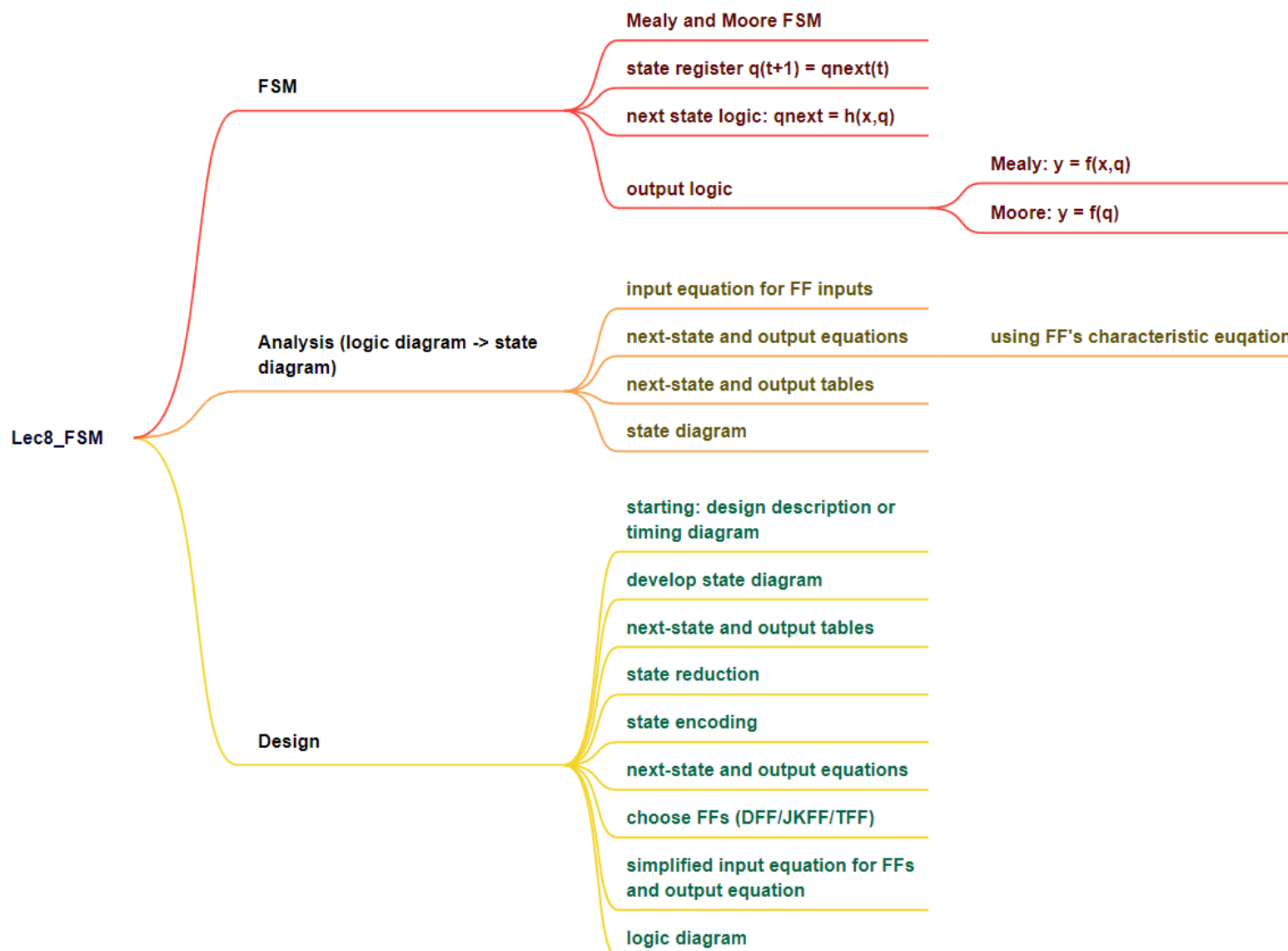
Chapter 6 : Registers & Counters

2023 Fall

Today's Agenda

- Recap
- Context
 - Registers
 - Design a sequence generator
 - Counters
 - Design a counter
- Reading: Textbook, Chapter 6.1-6.9

Recap

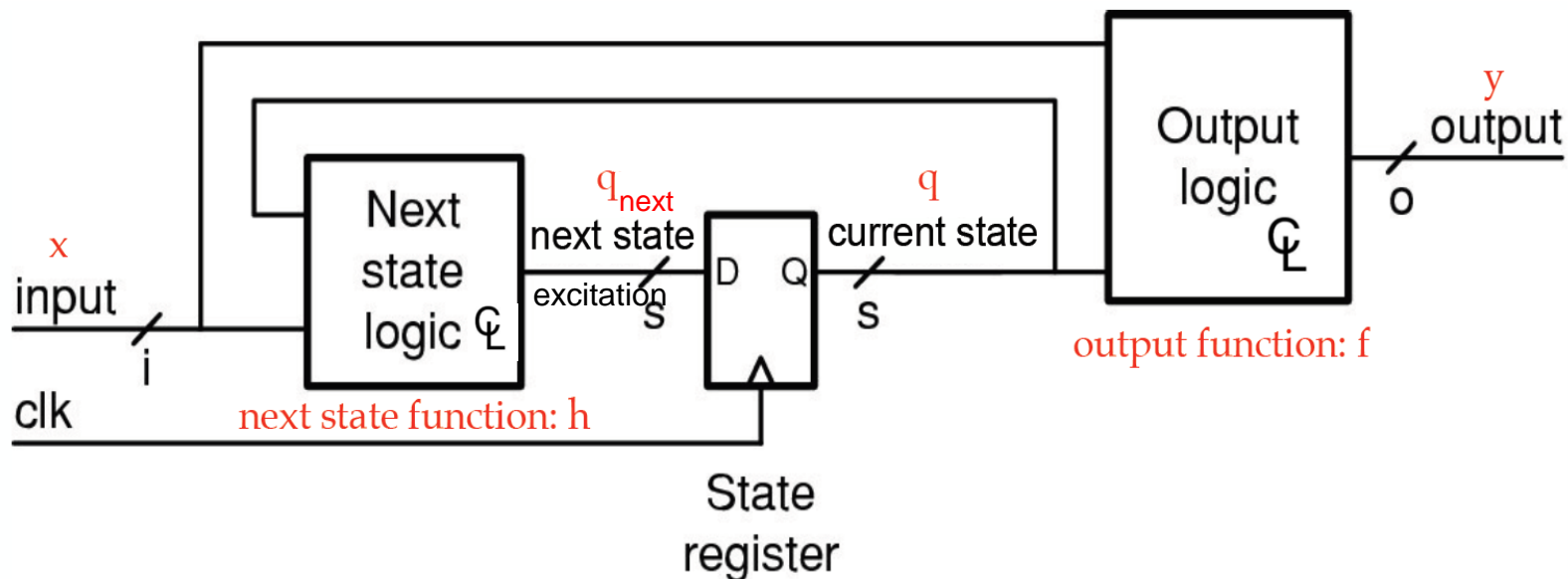


Outline

- **Various types of registers**
- Sequence generator with shift register
- Asynchronous counter
- Synchronous counter
- Design a synchronous counter

Clocked Sequential Circuits

- Clocked sequential circuits have flip-flops and combinational gates.
- FFs are essential, otherwise reduce to combinational.
- Circuits that include flip-flops are usually classified by the function they perform rather than by the name of the sequential circuit.
- Two such circuits are **registers** and **counters**.

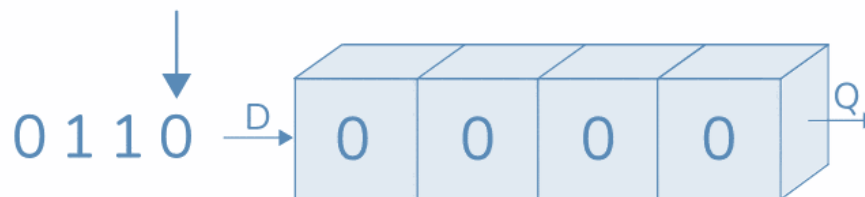


Registers and Counters

- Register
 - A group of binary cells (FFs) suitable for holding binary data information
 - In addition to the FFs, a register may have combinational gates to control when and how the new information is transferred into the register (MUXes, ...)
- Counter
 - A register that goes through a predetermined sequence of states upon the application of input pulses
 - The gates in a counter are connected in such a way as to produce a pre-described sequence of binary states in the register (arithmetic circuits)

Shift Register

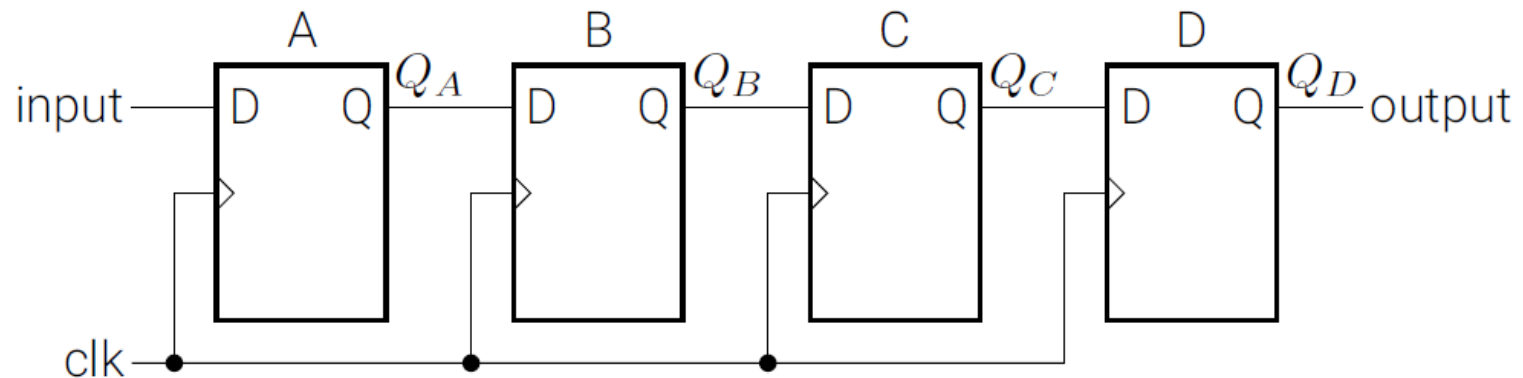
- Register: bitwise extension of a FF.
 - The shift register permits the stored data to move from a particular location to some other location within the register.
 - All the n FFs are driven by the common clock signal
 - sometimes with load control
- Type: Based on input & output
 - Serial-in to Serial-out (SISO)
 - Serial-in to Parallel-out (SIPO)
 - Parallel-in to Serial-out (PISO)
 - Parallel-in to Parallel-out (PIPO)
- Direction
 - Left shift
 - Right shift
 - Rotate (right or left)
 - Bidirectional
- Universal shift register



Bits moving through a SISO shift register

4-bit Serial-in to Serial-out

- SISO: the data is shifted serially “IN” and “OUT” of the register, one bit at a time in either a left or right direction under clock control.

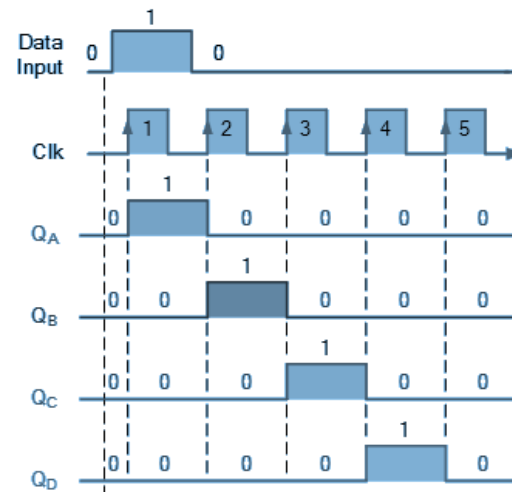
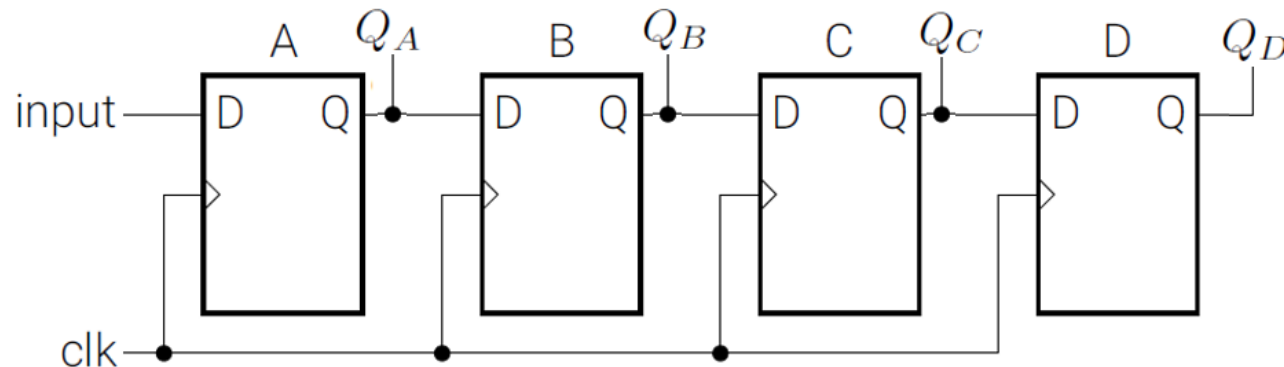


- An example of **1011** into the register. (Input from LSB to MSB, Right shift)

CLK	In	Q_A	Q_B	Q_C	Q_D	Out
initial		0	0	0	0	0
↑	1	1	0	0	0	0
↑	1	1	1	0	0	0
↑	0	0	1	1	0	0
↑	1	1	0	1	1	1

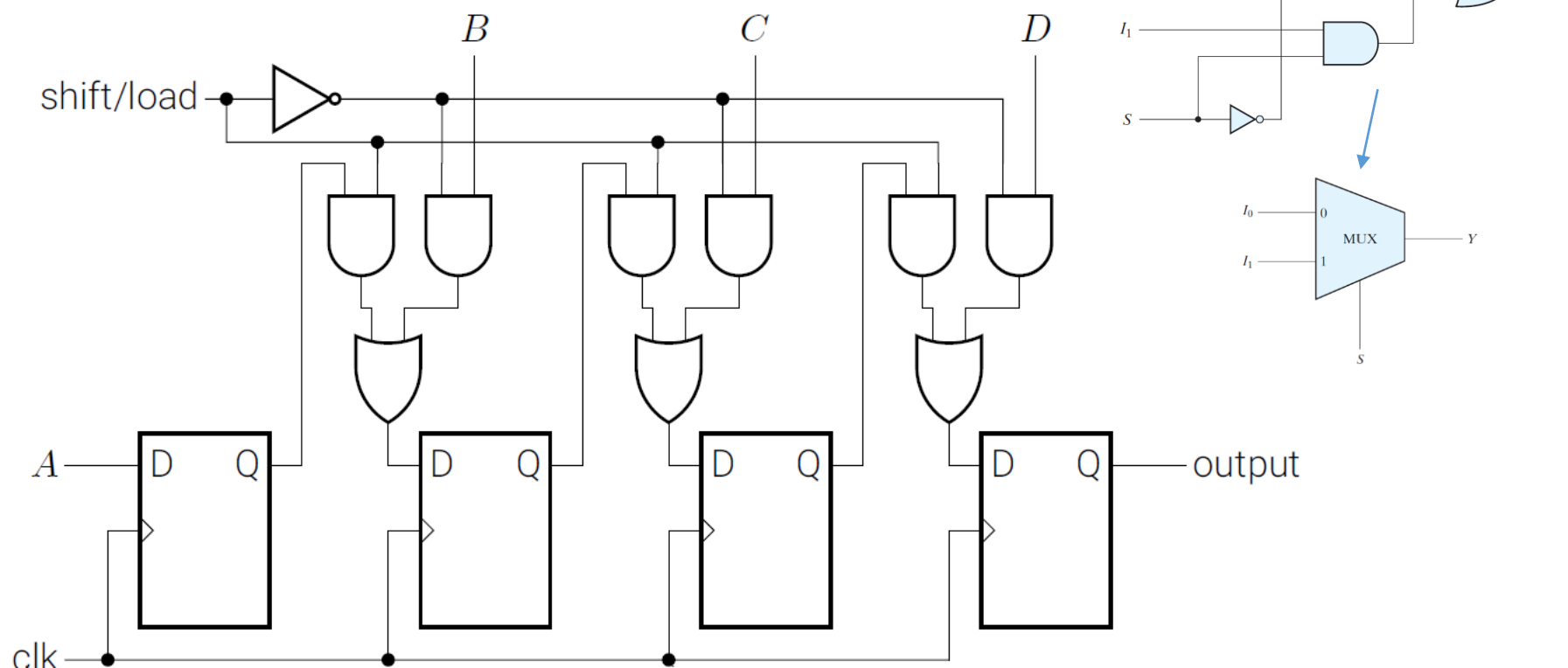
4-bit Serial-in to Parallel-out

- SIPO: the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.



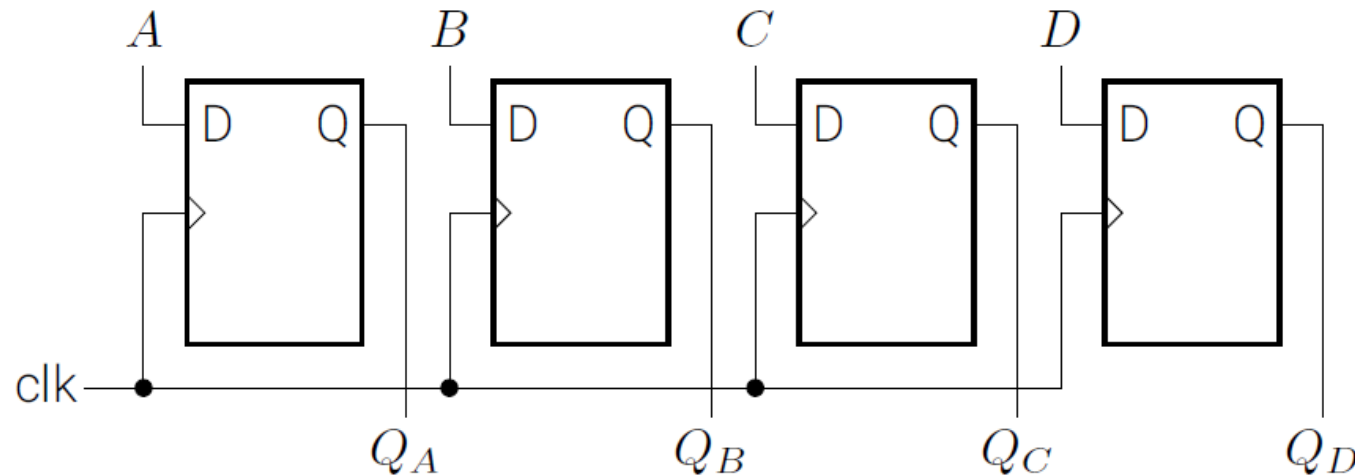
4-bit Parallel-in to Serial-out

- PISO: the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.



4-bit Parallel-in to Parallel-out

- PIPO: the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

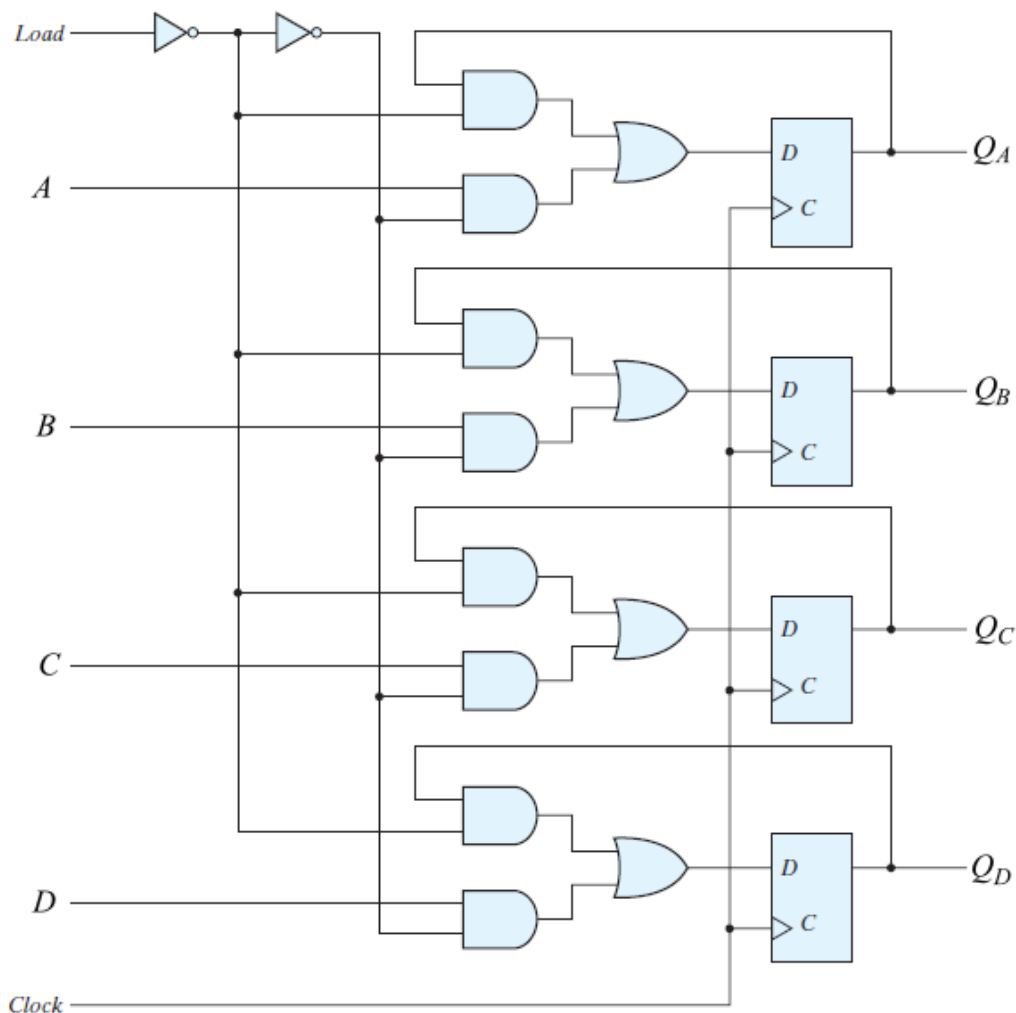


PIPO with Parallel Load

- 4-bit Parallel-in to Parallel-out Register with Parallel Load Logic diagram

function table

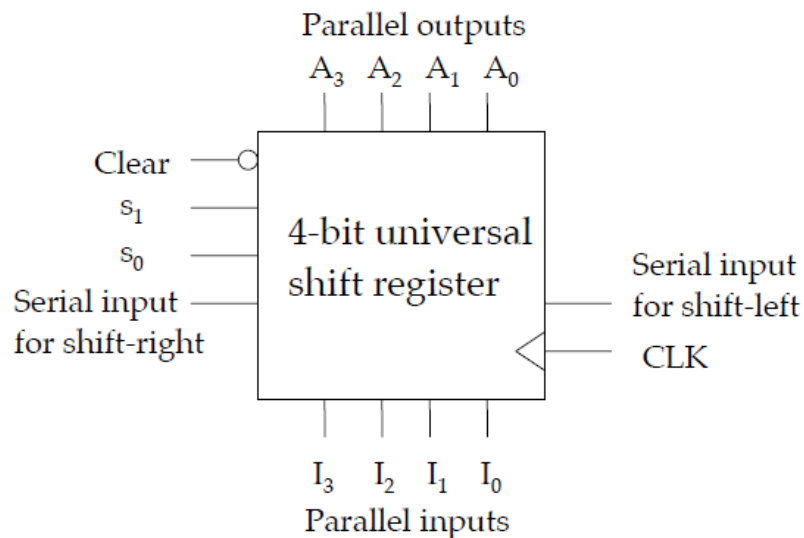
Present state Load	Next state			
	Q_A	Q_B	Q_C	Q_D
0	No change			
1	A	B	C	D



Universal Shift Register

- Universal shift register
 - Capable of both-direction shifting and parallel load/out

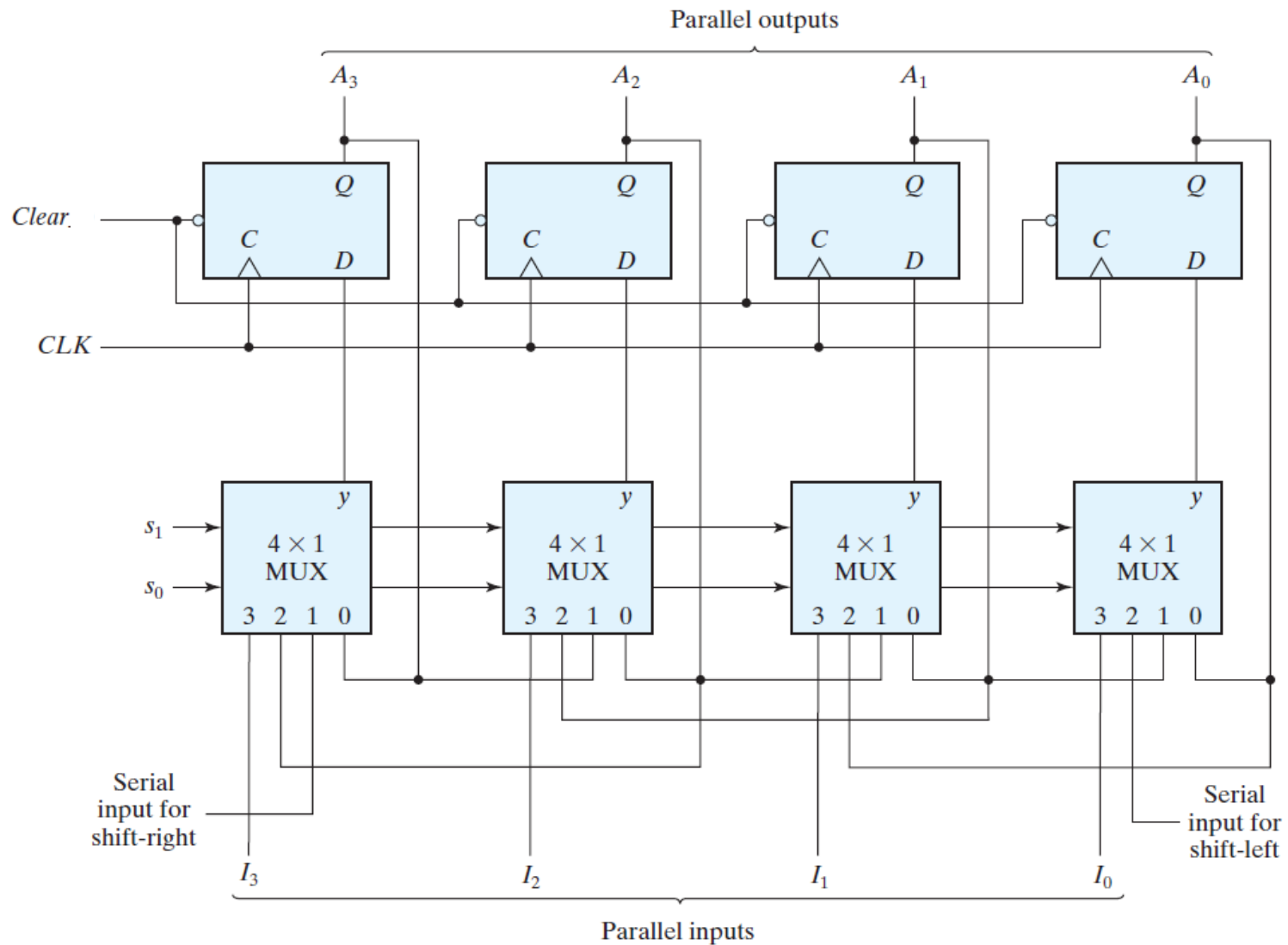
functional table



Clear	S_1	S_0	A_3^+	A_2^+	A_1^+	A_0^+	Operation
0	x	x	0	0	0	0	Clear
1	0	0	A_3	A_2	A_1	A_0	No change
1	0	1	sri	A_3	A_2	A_1	Shift right
1	1	0	A_2	A_1	A_0	sli	Shift left
1	1	1	I_3	I_2	I_1	I_0	Parallel load

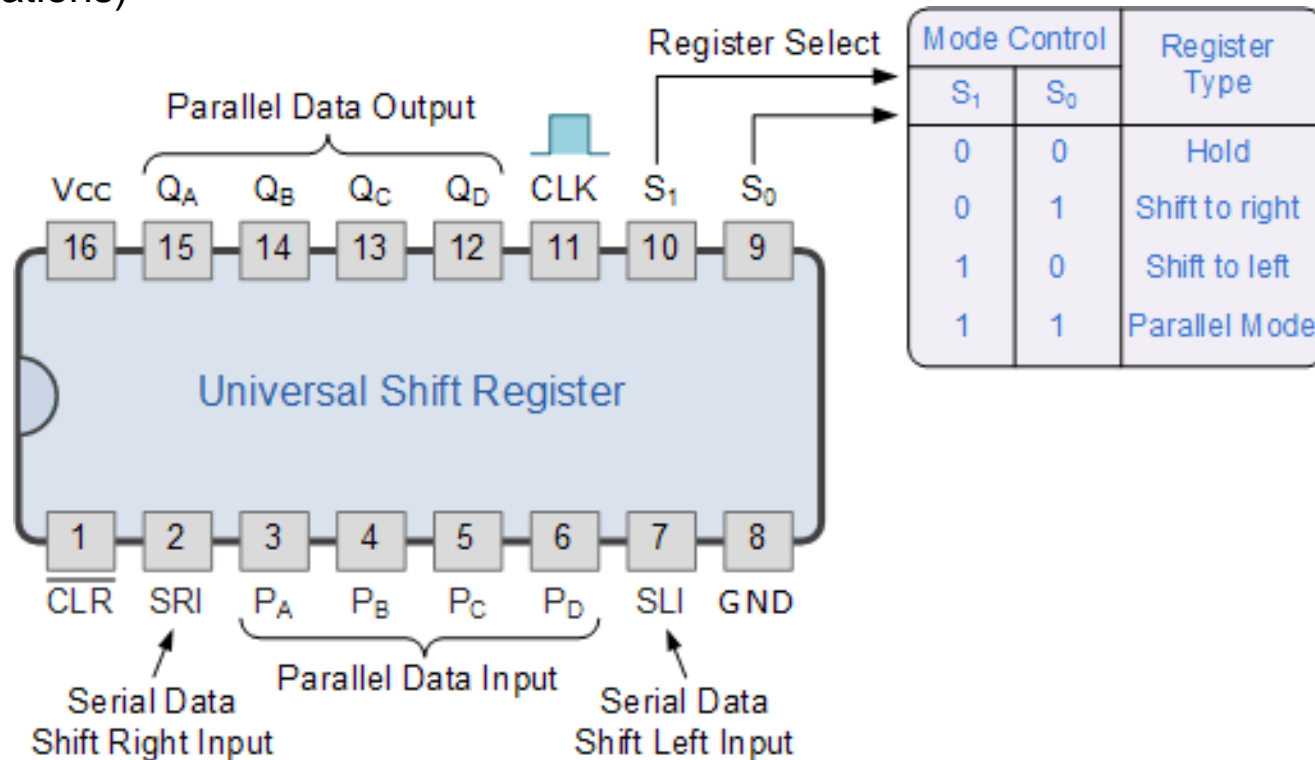
graphic symbol

Universal Shift Register



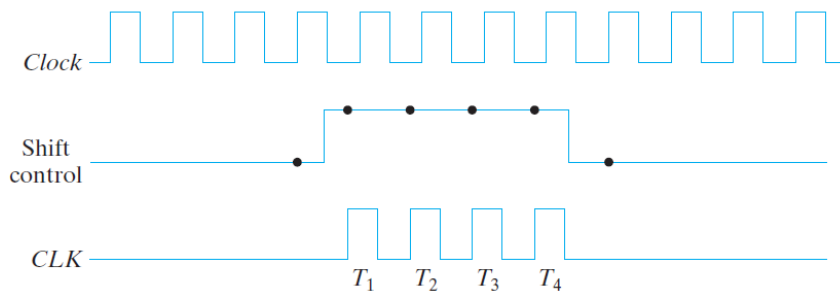
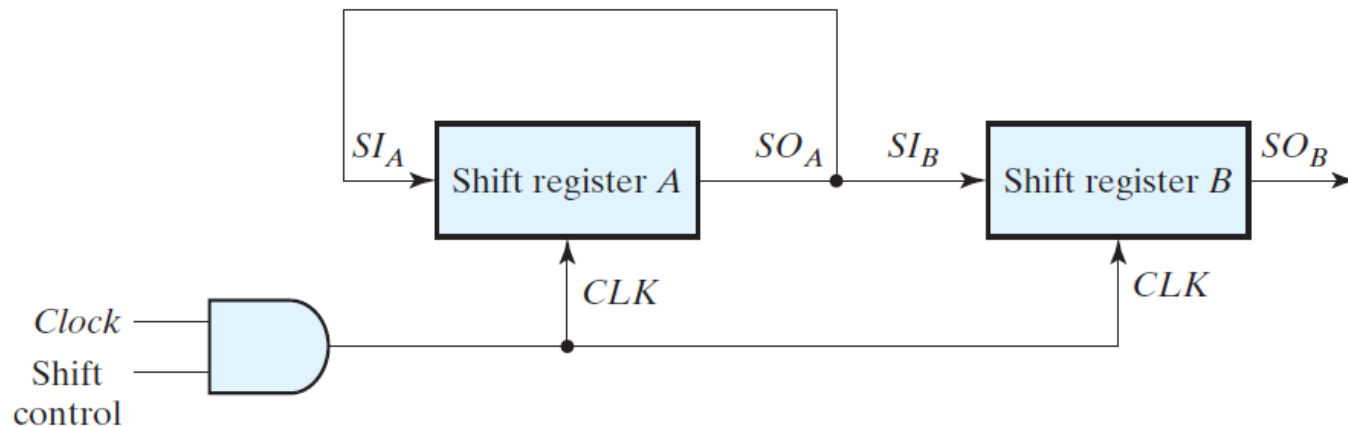
74 Series Shift Register

- Some of the IC's available for shift registers
 - 74164 – 8-bit SIPO shift register
 - 74165 – 8-bit PISO shift register
 - 74194 – 4-bit PIPO shift register
 - 74195 – 4-bit Universal shift register (can be used for SISO, SIPO, and PIPO operations)



Serial Transfer

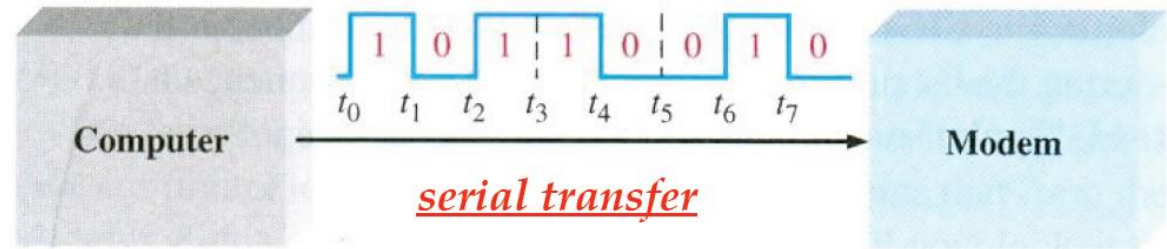
- Serial Transfer from Reg A to Reg B



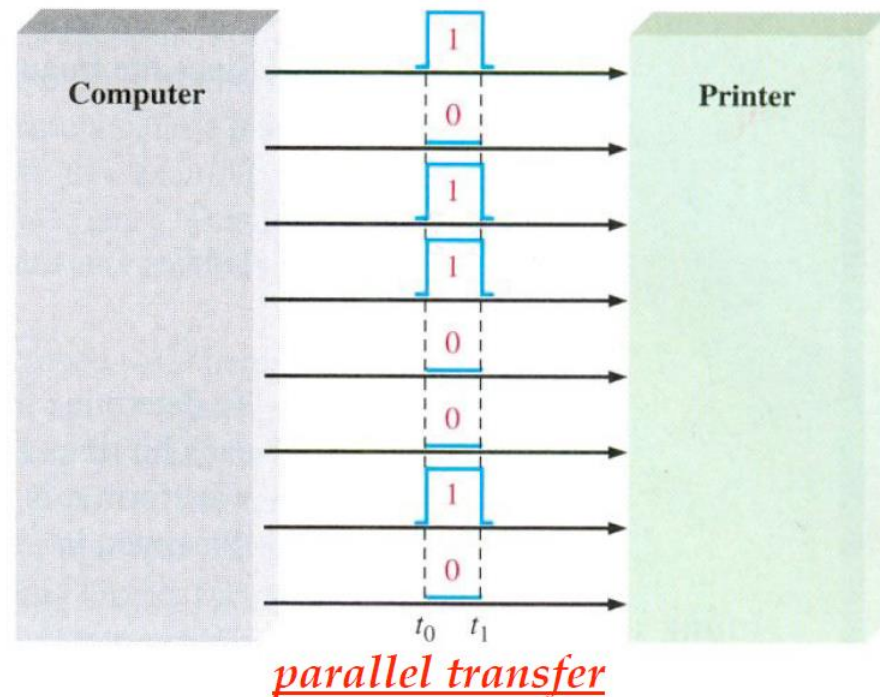
Timing Pulse	Shift Register A				Shift Register B			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

Serial/Parallel Transfer

- Shifters are useful in serializers and deserialisers that convert data from parallel to serial form and back again

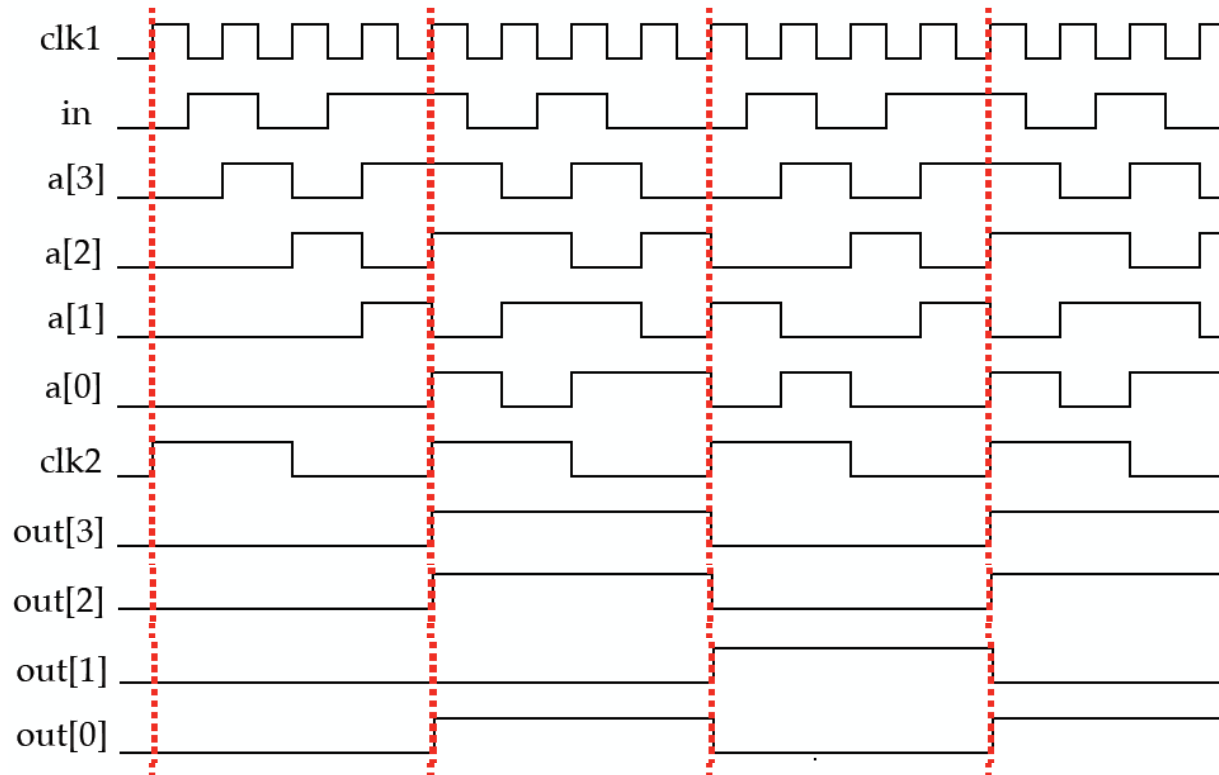
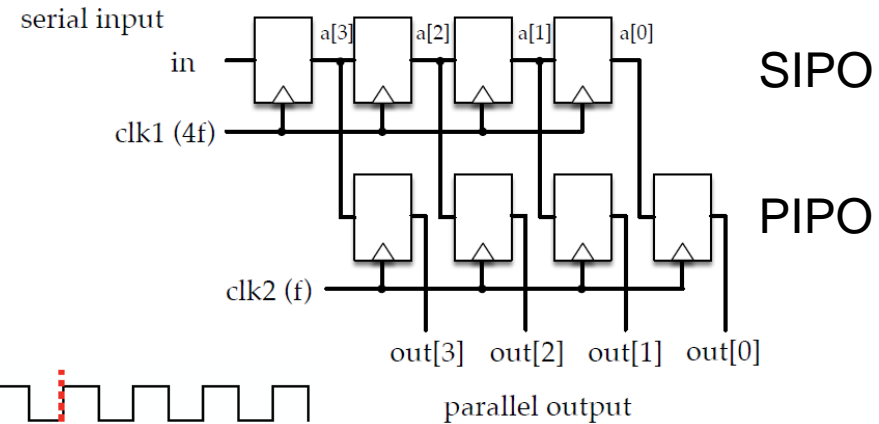


- Serial
 - One bit a time
 - Need more time, low complexity
- Parallel
 - All bits at the same time
 - Transfer faster, higher complexity



Serial to Parallel Converter

- clk for SIPO is faster



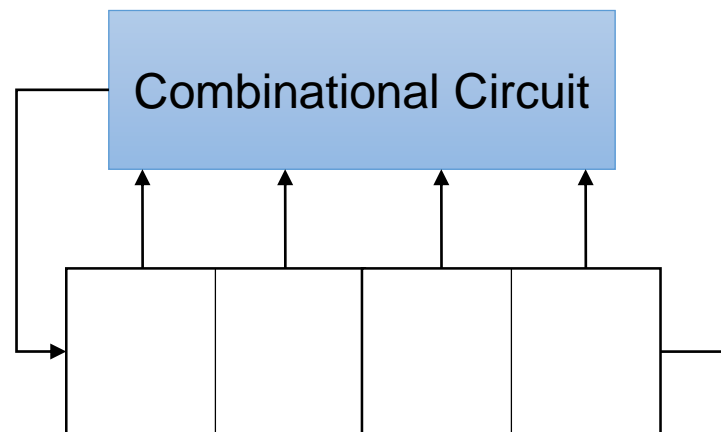


Outline

- Various types of registers
- **Sequence generator with shift register**
- Asynchronous counter
- Synchronous counter
- Design a synchronous counter

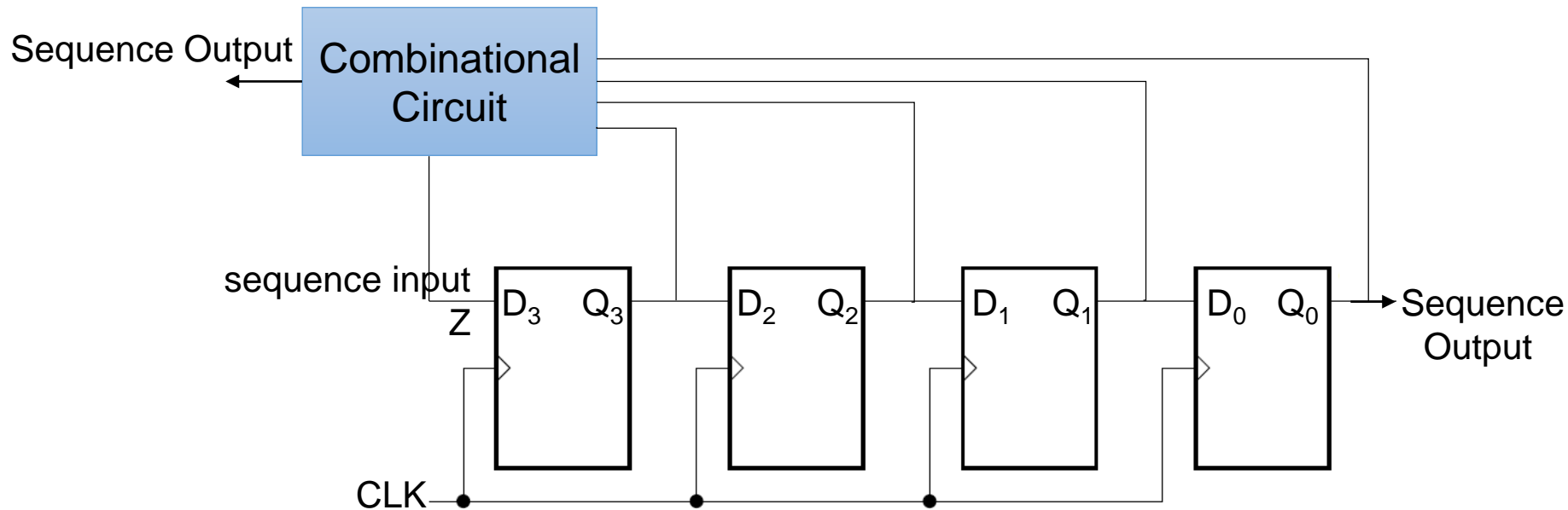
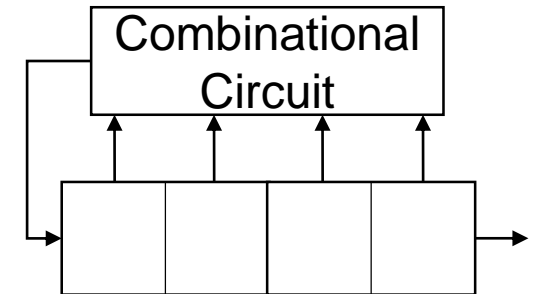
Sequence Generator

- A sequence generator is a circuit that generates a desired sequence of bits in synchronization with a clock.
- Can be used as a random bit generator, code generator, and prescribed period generator.
- The output of the combinational circuit is a function of the shift register state and is connected to the serial input of the shift register



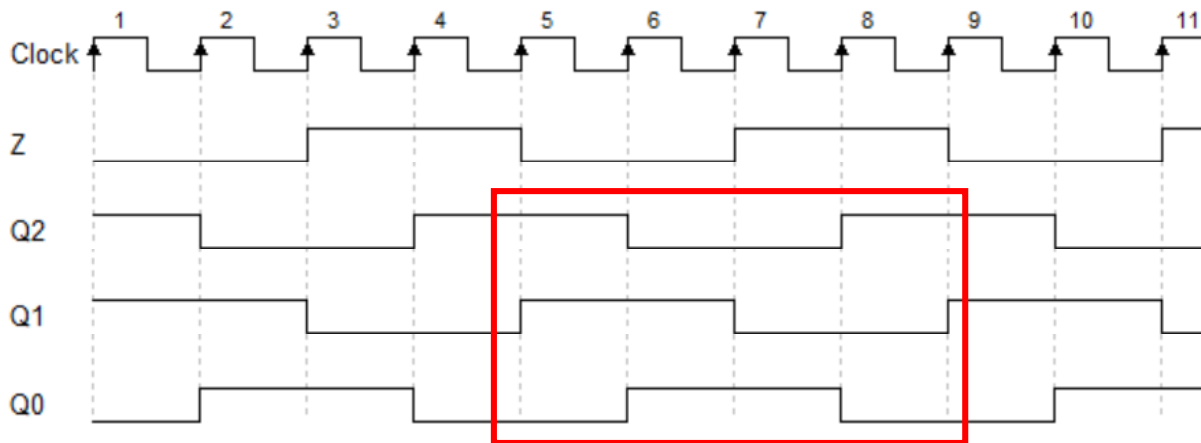
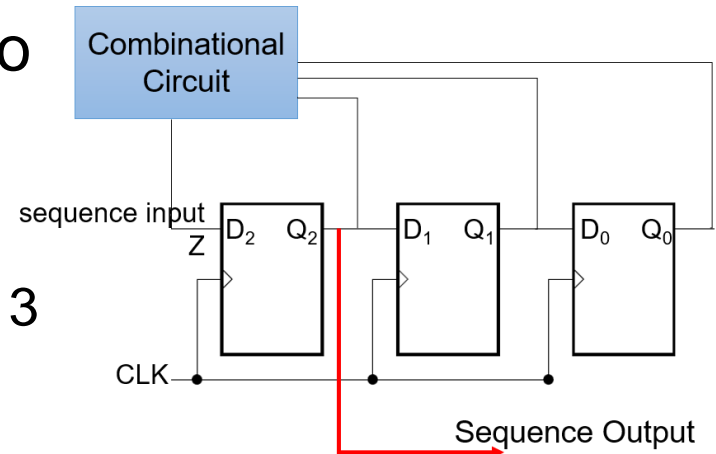
Sequence Generator

- n FFs can generate a sequence with the maximum length of $N = 2^n - 1$
- The required sequence can be obtained from the output of any FF, or the output of the combinational circuit



Example: Design a 4-bit Sequence Generator

- Design of a sequence generator to generate a sequence of **1001**.
 - The minimum number of flip-flops required to generate a sequence of length N is given by $N \leq 2^n - 1 \rightarrow n = 3$ (3 FFs)

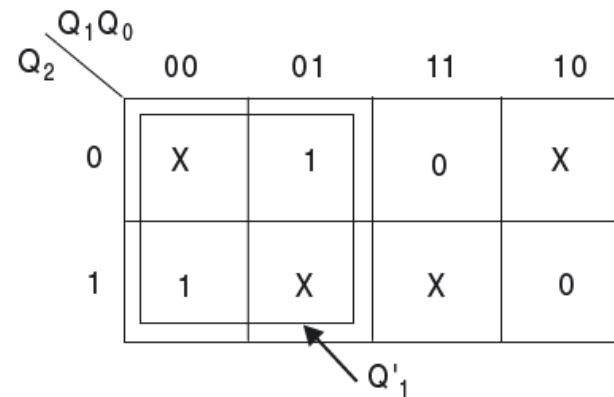


Clk	Z	Q ₂	Q ₁	Q ₀
↑		1	1	0
↑		0	1	1
↑		0	0	1
↑		1	0	0

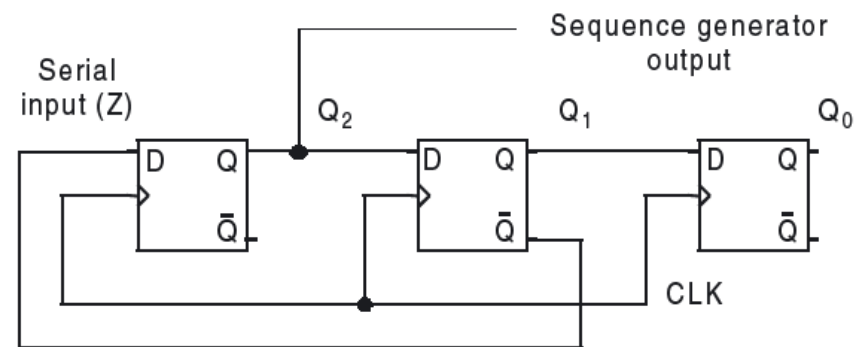
- Design of a sequence generator to generate a sequence of **1001**.
 - The minimum number of flip-flops required to generate a sequence of length N is given by $N \leq 2^n - 1 \rightarrow n = 3$ (3 FFs)

CLK	Z	Q ₂	Q ₁	Q ₀
↑		1	1	0
↑		0	1	1
↑		0	0	1
↑		1	0	0

CLK	Z	Q_2	Q_1	Q_0
↑	0	1	1	0
↑	0	0	1	1
↑	1	0	0	1
↑	1	1	0	0



$$Z = Q_1'$$



- Design of a sequence generator to generate a sequence of **10011**.
 - The minimum number of flip-flops required to generate a sequence of length N is given by $N \leq 2^n - 1 \rightarrow n = 3$ (3 FFs)

The Karnaugh map for the output Z is shown above the circuit diagram. The map has columns for Q_1Q_0 (00, 01, 11, 10) and rows for Q_2 (0, 1). The values in the cells are: (0,0)=X, (0,1)=1, (0,11)=0, (0,10)=X, (1,00)=1, (1,01)=X, (1,11)=0, (1,10)=1. The prime implicants $Q_1' + Q_0'$ are highlighted with dashed boxes, and arrows point to the corresponding terms in the Boolean expression $Z = Q_1' + Q_0'$.

The circuit diagram shows three D flip-flops labeled Q_2 , Q_1 , and Q_0 . The output of Q_0 is the "Sequence generator output". The clock (CLK) is connected to the clock input of all three flip-flops. The D input of Q_2 is the output of an AND gate with inputs Q_1 and Q_0 . The D input of Q_1 is Q_0 . The D input of Q_0 is Q_1 . The \bar{Q} outputs of each flip-flop are connected to the D input of the next flip-flop in the sequence: \bar{Q}_2 to D_{Q_1} , \bar{Q}_1 to D_{Q_0} , and \bar{Q}_0 to D_{Q_2} .

Example: Design a 6-bit Sequence Generator

- Design of a sequence generator to generate a sequence of **110101**
 - The minimum number of flip-flops required to generate a sequence of length N is given by $N \leq 2^n - 1$
 - The minimum value of n to satisfy the above condition is 3.

CLK	Z	Q_2	Q_1	Q_0
↑		1	1	0
↑		1	1	1
↑		0	1	1
↑		1	0	1
↑		0	1	0
↑		1	0	1

State 101 occurs twice!

3 flip-flops are not sufficient to generate the given sequence.

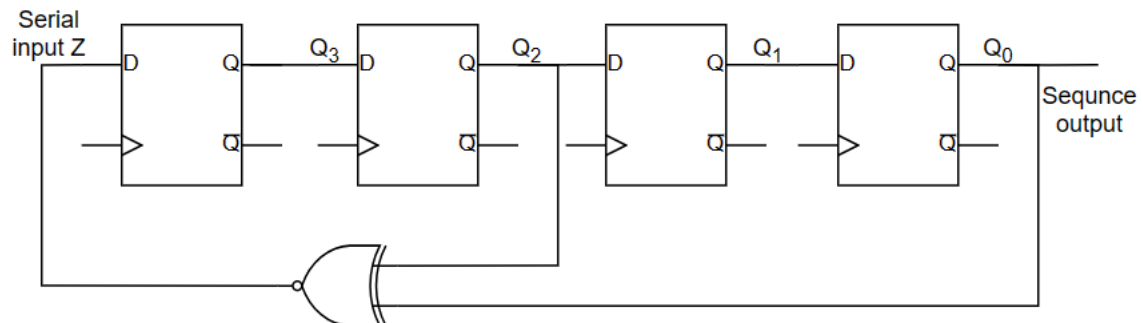
Example: Design a 6-bit Sequence Generator

- Needs 4 FFs to generate 110101

clk	Z	Q ₃	Q ₂	Q ₁	Q ₀
↑	1	1	1	0	1
↑	0	1	1	1	0
↑	1	0	1	1	1
↑	0	1	0	1	1
↑	1	0	1	0	1
↑	1	1	0	1	0

Q ₁ Q ₀		Q ₁			
		00	01	11	10
Q ₃ Q ₂	00	X	X	X	X
	01	X	1	1	X
	11	X	1	X	0
	10	X	X	0	1

$$Z = Q_2Q_0 + Q_2'Q_0' = (Q_2 \oplus Q_0)'$$





Outline

- Various types of registers
- Sequence generator with shift register
- **Asynchronous counter**
- Synchronous counter
- Design a synchronous counter



Counters

- A counter is a special type of register that counts upward, downward, or in any pre-specified sequence
 - Ripple counters (asynchronous counter)
 - The output transition of flip-flop serves as a source for triggering other flip-flops
 - Synchronous counters
 - The clock inputs of all flip-flops receive a common clock

Asynchronous Counters

- aka. Serial or ripple counters
- All the flip-flops are not driven by the same clock pulse.
 - The successive flip-flop is triggered by the output of the previous flip-flop.
 - Hence the counter has cumulative settling time, which limits its speed of operation



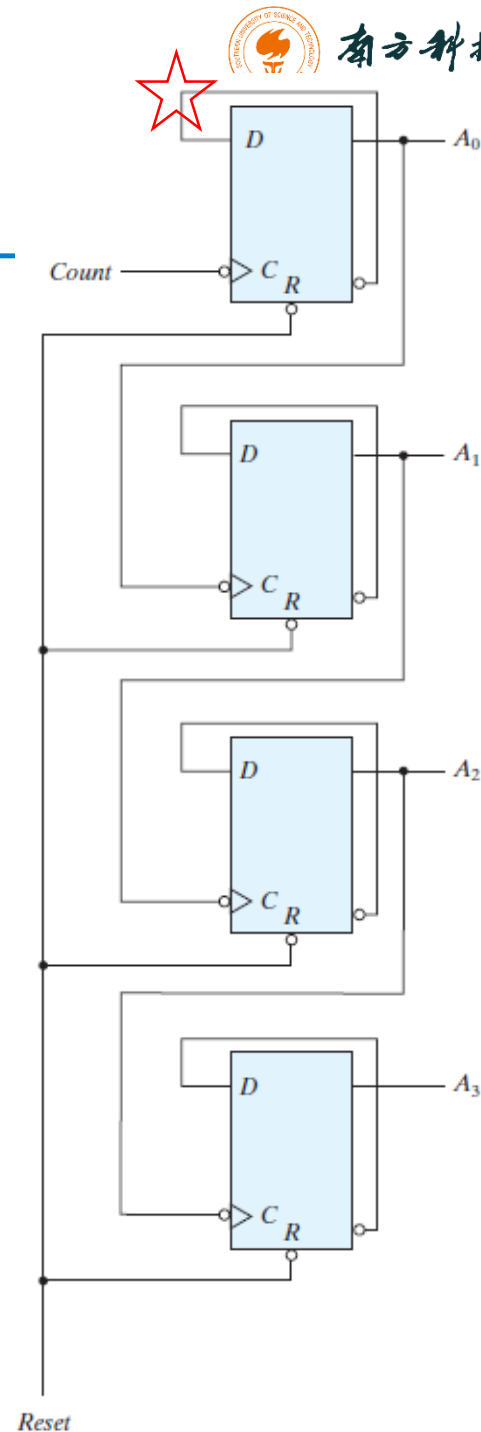
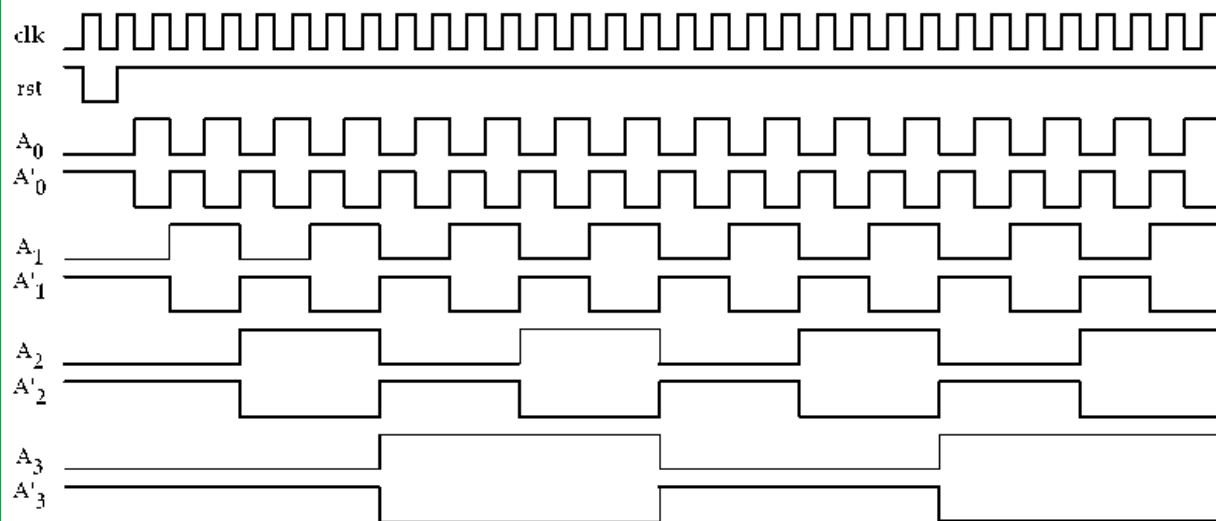
Ripple Counter Example: Slip-Flap Clock

4-bit Binary Ripple Counter

- Binary Up count sequence

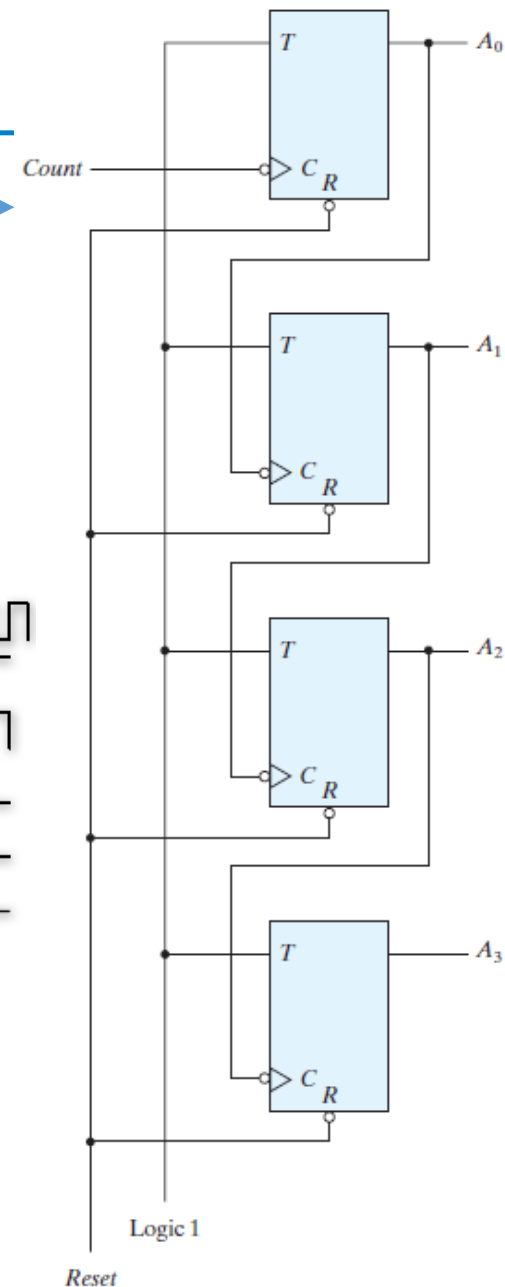
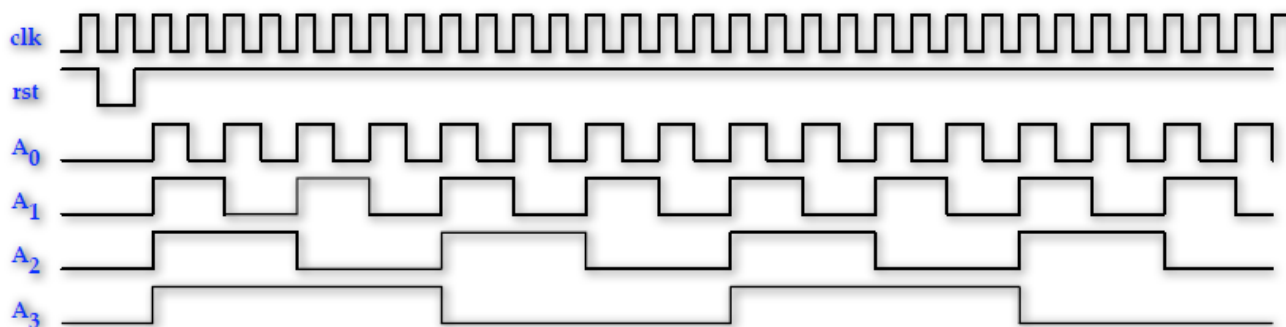
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

- Timing diagram

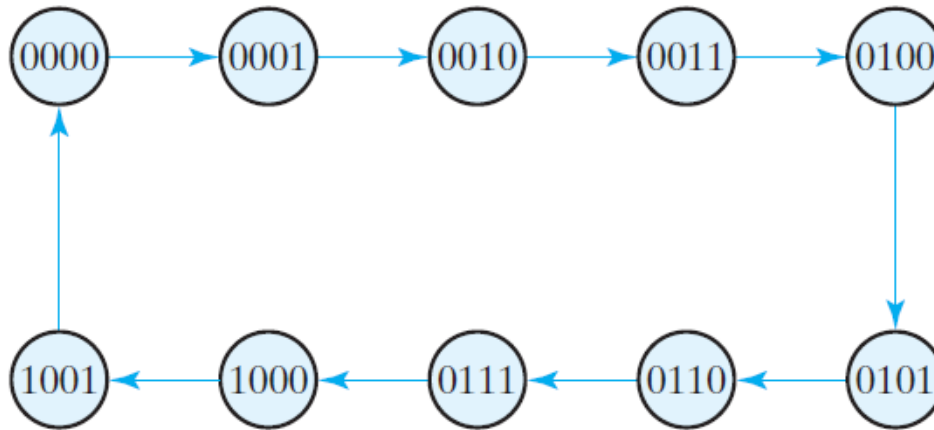


4-bit Binary Ripple Counter

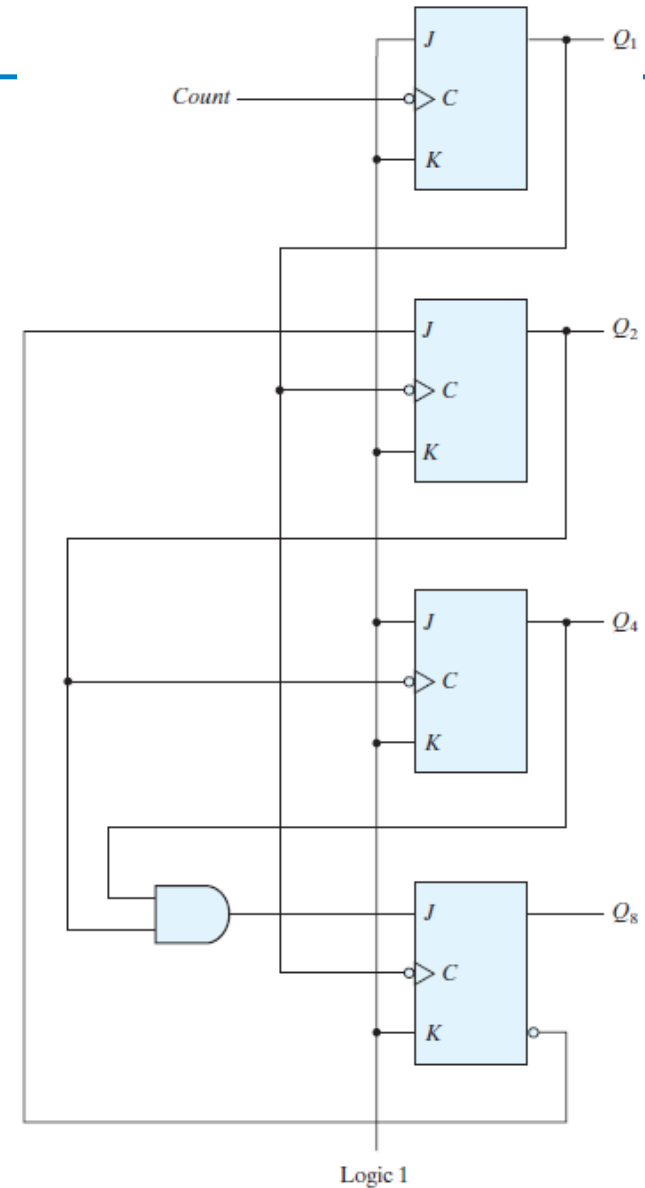
- Binary Up counter using TFF
- What if design a down counter?
 - just replace negative-edge triggered clock with positive-edge triggered clock



BCD Ripple Counter



Present State				Next State			
Q8	Q4	Q2	Q1	Q8	Q4	Q2	Q1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0





Outline

- Various types of registers
- Sequence generator with shift register
- Asynchronous counter
- **Synchronous counter**
- Design a synchronous counter

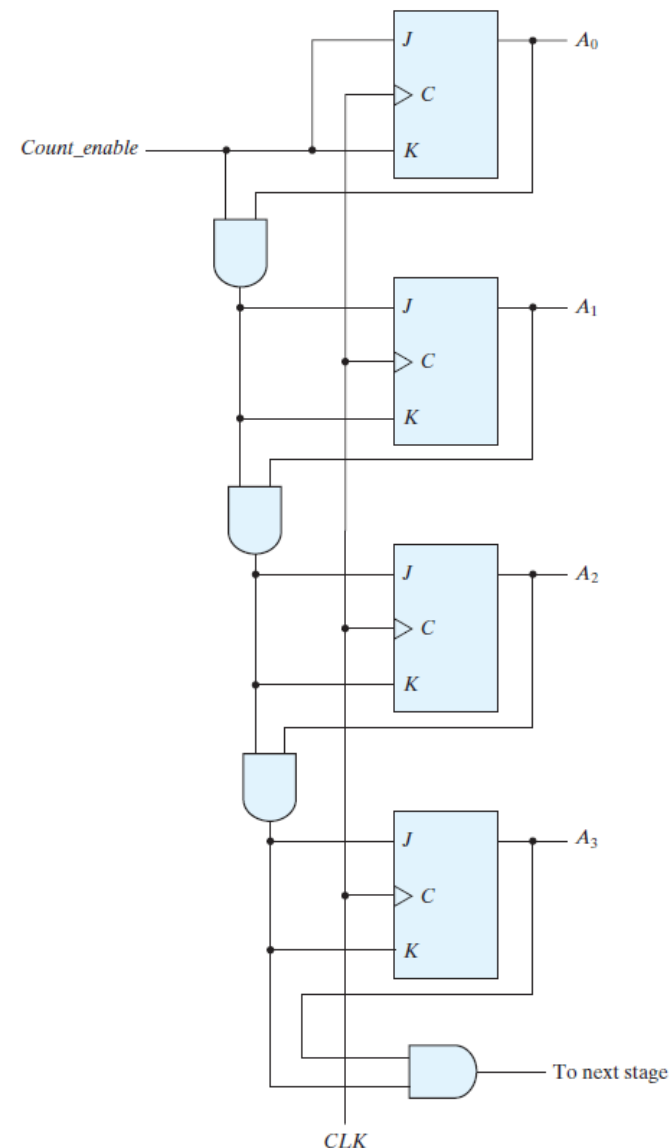
Synchronous counters

- The ripple or asynchronous counter is the simplest to build, but its highest operating frequency is limited because of ripple action.
 - delay time
 - glitches
- Both of these problems can be overcome, if all the flip-flops are clocked synchronously.
- The resulting circuit is known as a synchronous counter.

4-bit Synchronous Binary Counters

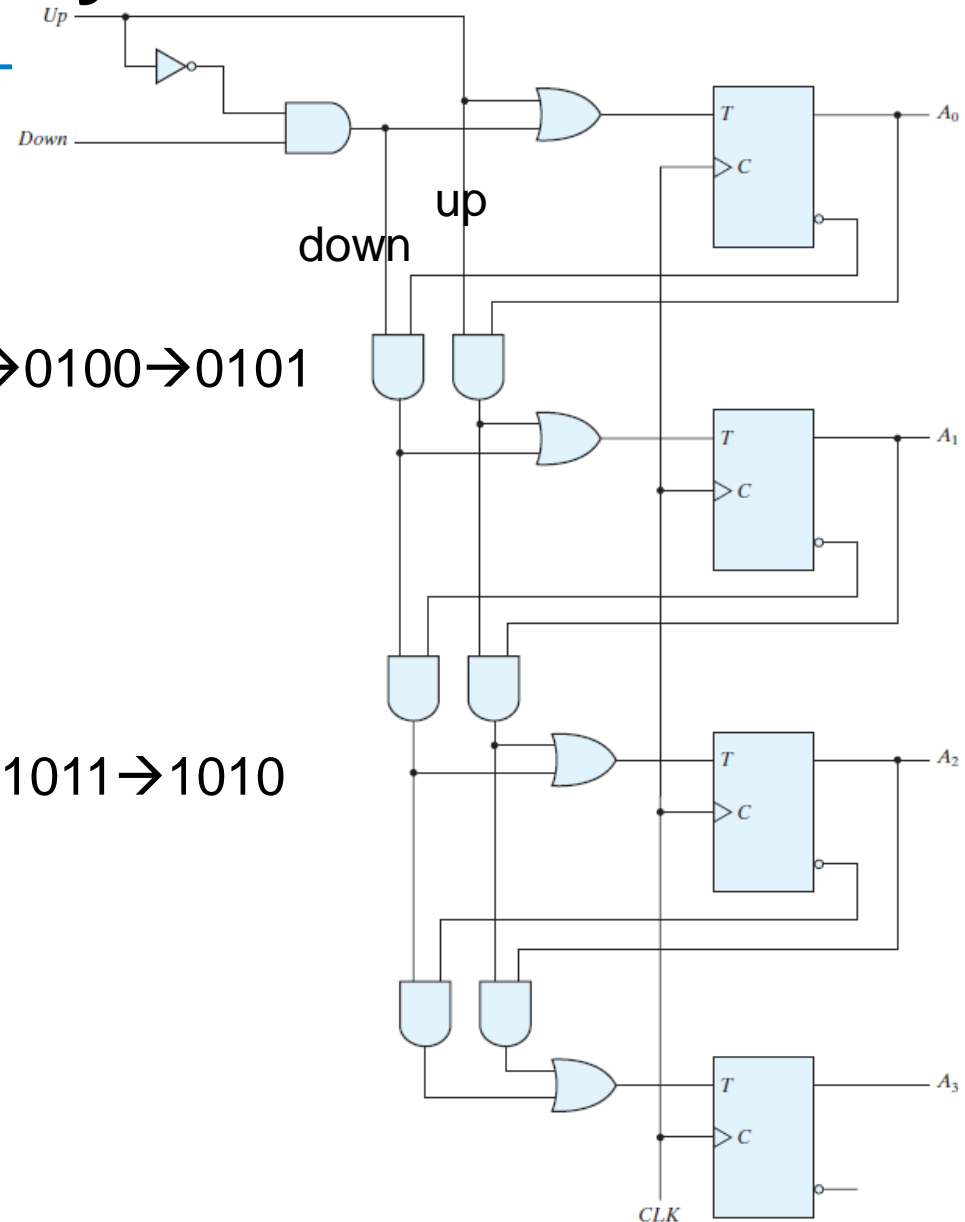
- Count up
 - As the number of stages increases, the number of AND gates also increases, along with the number of inputs for each of those AND gates.

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0



4-bit Up/Down Binary Counter

- Up=1, Down=0 =>
- counting up
 - $A_3A_2A_1A_0$
 - $0000 \rightarrow 0001 \rightarrow 0010 \rightarrow 0011 \rightarrow 0100 \rightarrow 0101$
- Up=0, Down=1 =>
- counting down
 - $A_3A_2A_1A_0$
 - $1111 \rightarrow 1110 \rightarrow 1101 \rightarrow 1100 \rightarrow 1011 \rightarrow 1010$

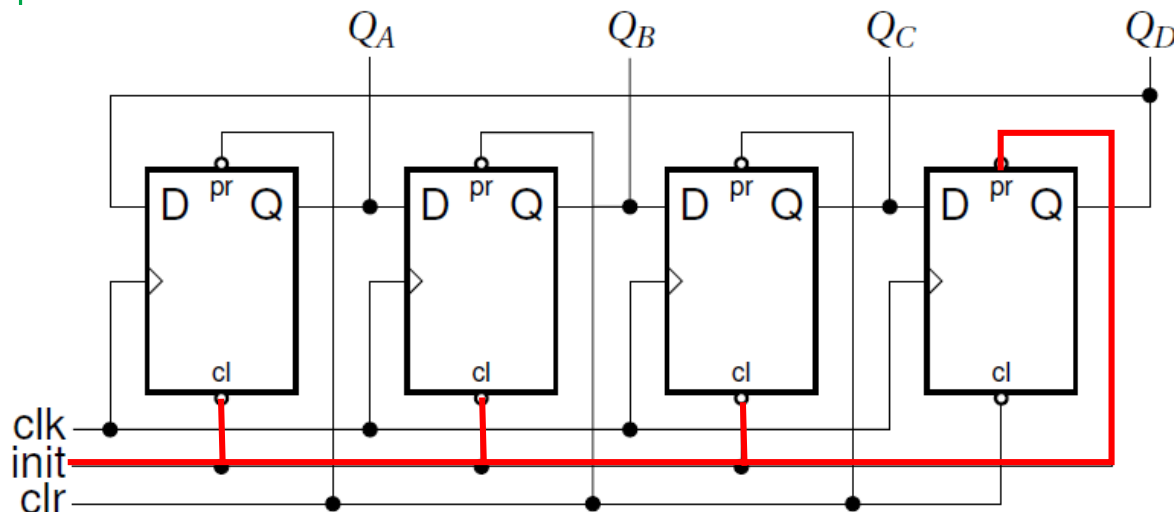


Shift Register Counters

- Shift registers may be arranged to form different types of counters.
- These shift registers use feedback, where the output of the last flip-flop in the shift register is fed back to the first flip-flop.
- Based on the type of this feedback connection, the shift register counters are classified as
 - ring counter
 - switch-tail ring counter or Johnson counter

Ring Counter

- A circular shift register with only one flip-flop being set at any particular time, all others are cleared. (initial value 0001 as in example)
- The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals

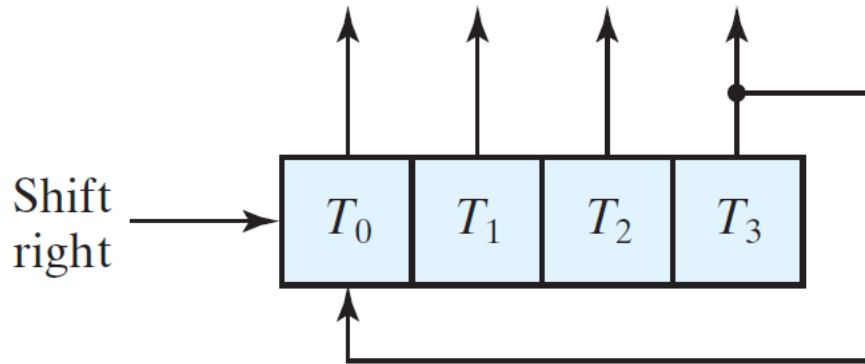


1000 → 0100 → 0010 → 0001 →

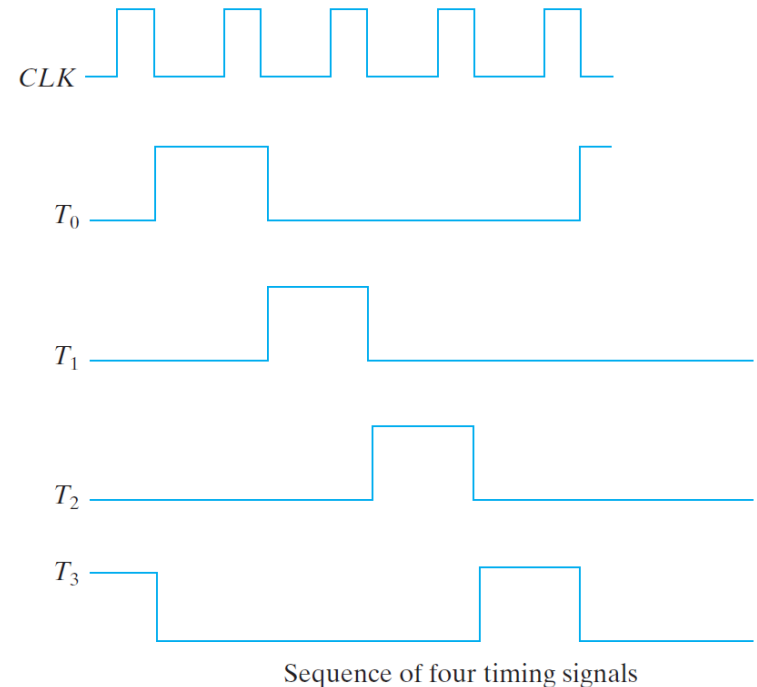
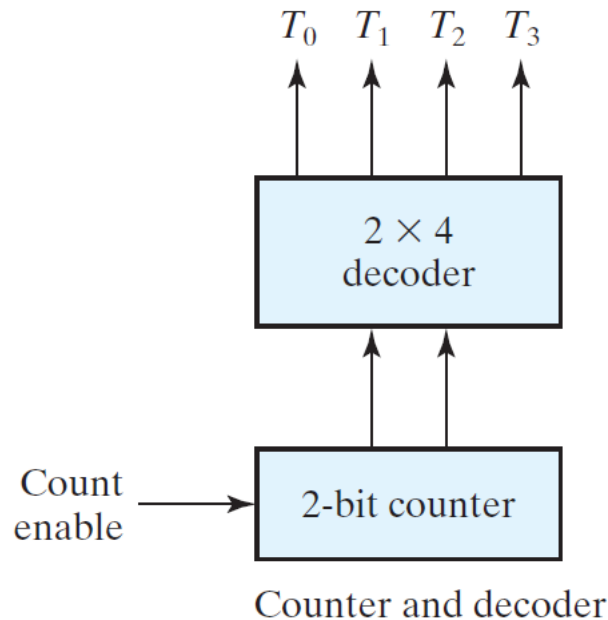
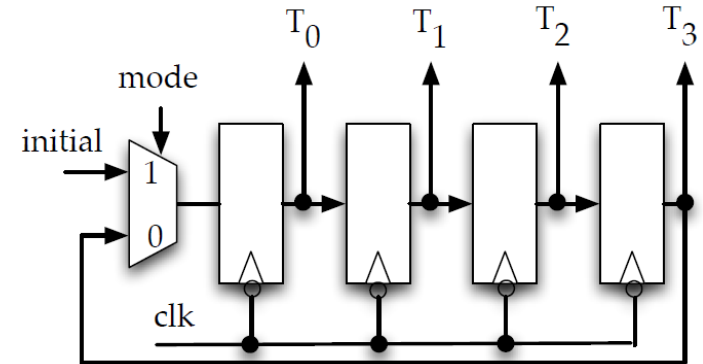
init	clk	Q_A	Q_B	Q_C	Q_D
L	X	0	0	0	1
H	↑	1	0	0	0
H	↑	0	1	0	0
H	↑	0	0	1	0
H	↑	0	0	0	1

init	clk	Q_A	Q_B	Q_C	Q_D
L	X	0	0	0	1
H	↑	1	0	0	0
H	↑	0	1	0	0
H	↑	0	0	1	0
H	↑	0	0	0	1

Ring Counter for Decoder

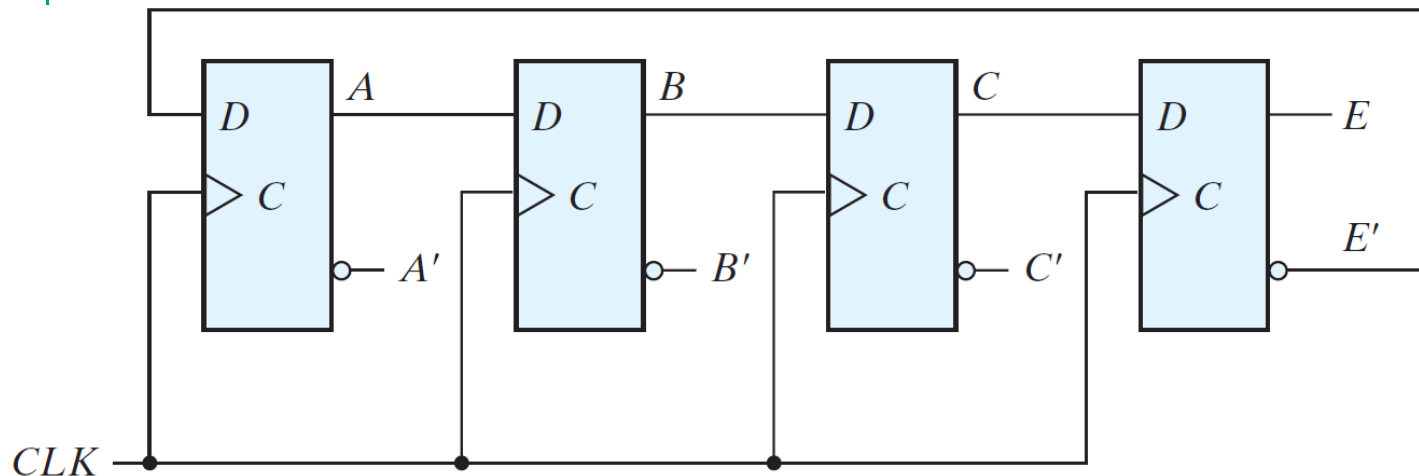
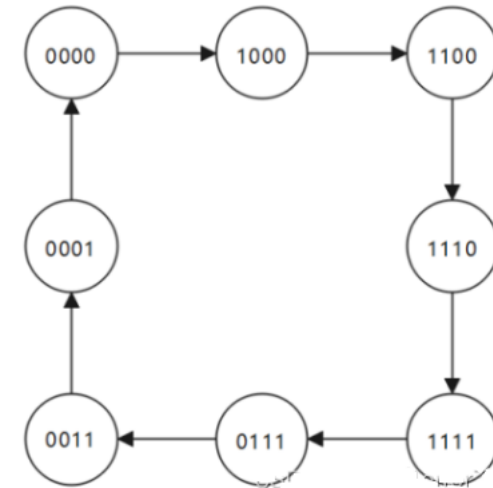


Ring-counter (initial value = 1000)



Johnson counter

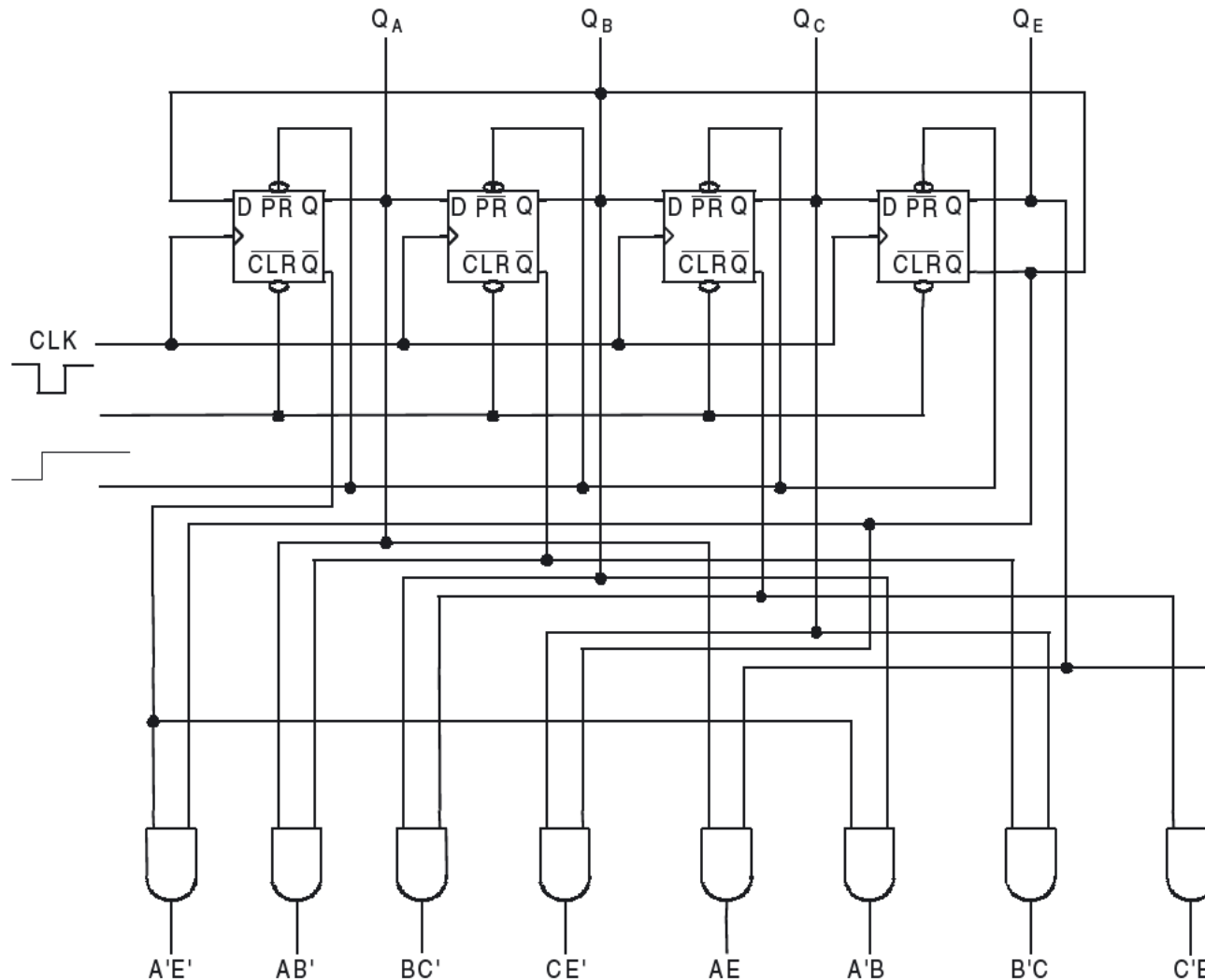
- Switch-tail ring counter: a circular shift register with its complement output of the last flip-flop connected to the input of the first flip-flop
- Johnson counter is a k-bit switch-tail ring counter will go through a sequence of $2k$ distinguishable states (initial value 0000 as in example)



Sequence number	Flip-flop outputs			
	A	B	C	E
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1
8	0	0	0	1

Johnson counter for decoder

- Using AND gates for one-hot outputs



AND gate required
for output

$A'E'$
 AB'
 BC'
 CE'
 AE
 $A'B$
 $B'C$
 $C'E$



Outline

- Various types of registers
- Sequence generator with shift register
- Asynchronous counter
- Synchronous counter
- **Design a synchronous counter**

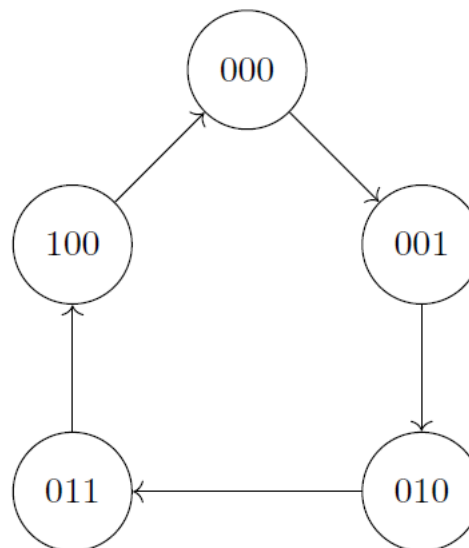
Design a Synchronous Counter

Recall: Design Procedure of Sequential Circuits

1. Specification: design description or timing diagram
2. Formulation: develop state diagram
3. Generate next-state table in form of count sequence
4. Choose type of Flip-Flop
5. Derive simplified excitation equations of FFs
6. Draw logic diagram

Modulo-N Counter

- Counters can be designed to generate any desired sequence of states. A divide-by-N counter (also known as a modulo- N counter)
- e.g. mod-5 counter
 - A counter that goes through the following binary repeated sequence: 0, 1, 2, 3, 4, 0, 1, 2, ...



Counters with Unused States

- n flops $\Rightarrow 2^n$ states
- Unused states
 - States that are not used in specifying the FSM, may be treated as don't-care conditions or may be assigned specific next states
- Self-correcting counters
 - Ensure that when a circuit enter one of its unused states, it eventually goes into one of the valid states after one or more clock pulses so that it can resume normal operation
 - Analyze the circuit to determine the next state from an unused state after it is designed

Example: Counters with Unused States

BC \ A	00	01	11	10
0	0	0	<i>x</i>	1
1	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>

J_A

BC \ A	00	01	11	10
0	0	1	<i>x</i>	<i>x</i>
1	0	1	<i>x</i>	<i>x</i>

J_B

BC \ A	00	01	11	10
0	1	<i>x</i>	<i>x</i>	0
1	1	<i>x</i>	<i>x</i>	0

J_C

BC \ A	00	01	11	10
0	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>
1	0	0	<i>x</i>	1

K_A

BC \ A	00	01	11	10
0	<i>x</i>	<i>x</i>	<i>x</i>	1
1	<i>x</i>	<i>x</i>	<i>x</i>	1

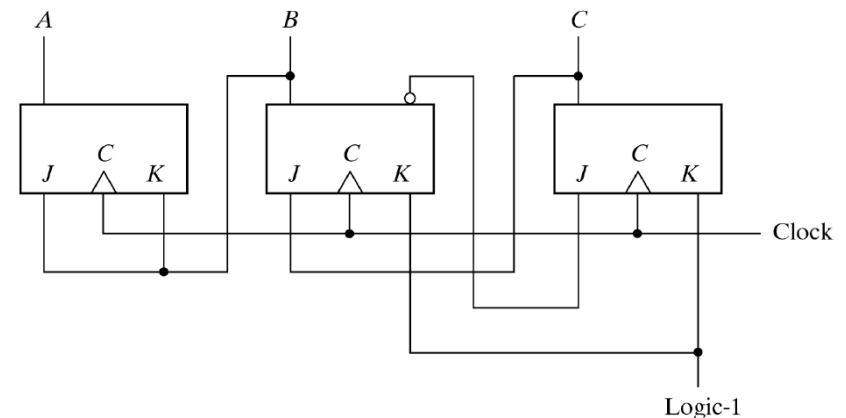
K_B

BC \ A	00	01	11	10
0	<i>x</i>	1	<i>x</i>	<i>x</i>
1	<i>x</i>	1	<i>x</i>	<i>x</i>

K_C

- $J_A = B$
- $J_B = C$
- $J_C = B'$

- $K_A = B$
- $K_B = 1$
- $K_C = 1$





Example: Counters with Unused States

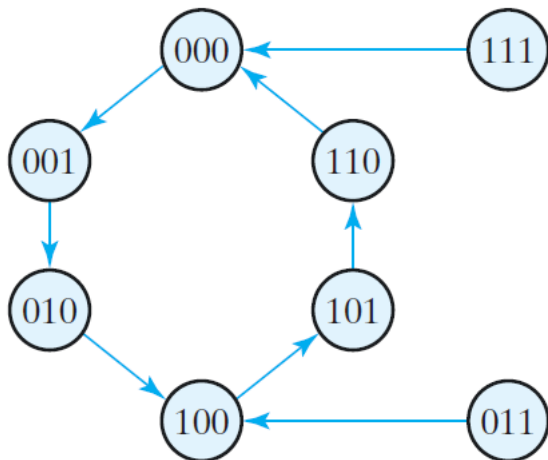
- Unused state & Self-correcting
 - What happen if the circuit gets into unused state 011 or 111 because of an error signal?
 - Let's analyze the state transition
 - $J_A = B$ $K_A = B$
 - $J_B = C$ $K_B = 1$
 - $J_C = B'$ $K_C = 1$

Present Stat			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	1	1	1	0	0	1	1	1	1	0	1
1	1	1	0	0	0	1	1	1	1	0	1

- 100 and 000 are valid state, thus this is a self-correcting counter, it eventually reaches the normal count sequence after one or more clock pulses

Example: Counters with Unused States

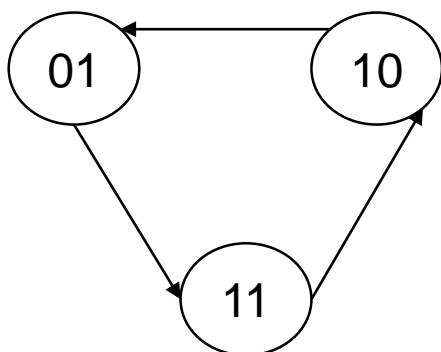
- An alternative design could use additional logic to direct every unused state to a specific next state, to make sure the counter is self-correcting
- To ensure that lock out does not occur, **assuming** $111 \rightarrow 000$, $011 \rightarrow 100$.



Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

Lock out problem

- Taking another counter example: It may be possible that the counter might go from one unused state to another and never arrive at a used state.



A(t)	B(t)	A(t+1)	B(t+1)
0	1	1	1
1	1	1	0
1	0	0	1

- $D_A = \Sigma(1,3), d = \Sigma(0)$

- $D_B = \Sigma(1,2), d = \Sigma(0)$

B	0	1
A		
0	x	1
1	0	1

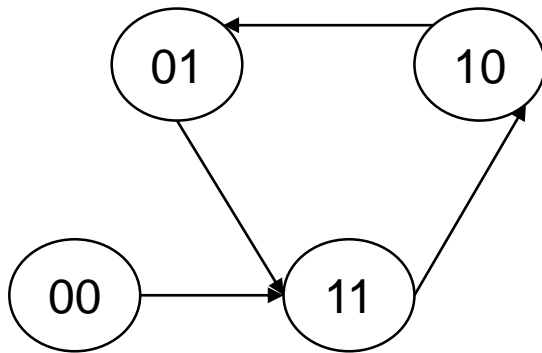
B	0	1
A		
0	x	1
1	1	0

- $D_A = B, D_B = A \oplus B$

- If state is 00, then next state is still 00 (stuck at 00)

Lock out problem

- Thus, the solution is to force state 00 transit to another valid state, for example 11

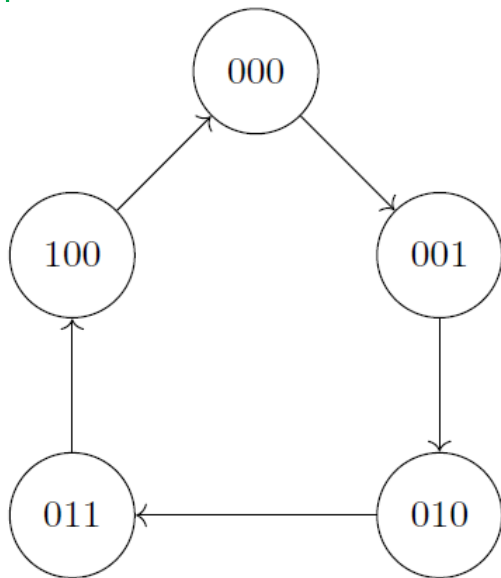


A(t)	B(t)	A(t+1)	B(t+1)
0	1	1	1
1	1	1	0
1	0	0	1
0	0	1	1

- $D_A = ?$, $D_B = ?$

Exercise: Design a mod-5 Counter

- mod-5 counter
 - A counter that goes through the following binary repeated sequence: 0, 1, 2, 3, 4, 0, 1, 2, ...



Present			Next			Flip-flop Inputs					
A_2	A_1	A_0	A_2	A_1	A_0	J_{A2}	K_{A2}	J_{A1}	K_{A1}	J_{A0}	K_{A0}
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

Exercise: Design a mod-5 Counter

A ₂ \ A ₁ A ₀				
	00	01	11	10
0			1	
1	X	X	X	X

$$J_{A2} = A_1 A_0$$

A ₂ \ A ₁ A ₀				
	00	01	11	10
0		1	X	X
1		X	X	X

$$J_{A1} = A_0$$

A ₂ \ A ₁ A ₀				
	00	01	11	10
0	1	X	X	1
1		X	X	X

$$J_{A0} = A_2'$$

A ₂ \ A ₁ A ₀				
	00	01	11	10
0	X	X	X	X
1	1	X	X	X

$$K_{A2} = 1$$

A ₂ \ A ₁ A ₀				
	00	01	11	10
0	X	X	1	
1	X	X	X	X

$$K_{A1} = A_0$$

A ₂ \ A ₁ A ₀				
	00	01	11	10
0	X	1	1	X
1	X	X	X	X

$$K_{A0} = 1$$

Exercise: Design a mod-5 Counter

- Check the unused state, we confirm that all three unused states results in a valid state after one cycle. It is by nature a self-correcting counter.

Present Stat			Next State			Flip-Flop Inputs					
A2	A1	A0	A2	A1	A0	J ₂	K ₂	J ₁	K ₁	J ₀	K ₀
1	0	1	0	1	0	0	1	1	1	0	1
1	1	0	0	1	0	0	1	0	0	0	1
1	1	1	0	0	0	1	1	1	1	0	1

