Digital Logic Design

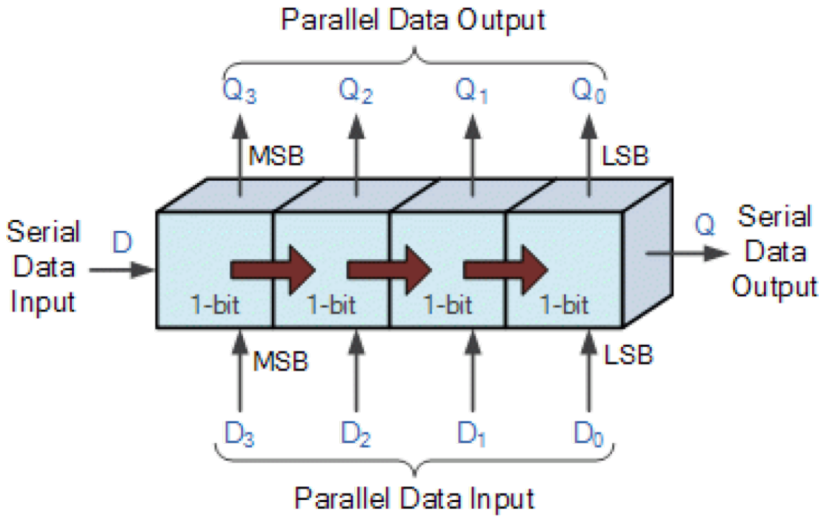
Hajar Falahati

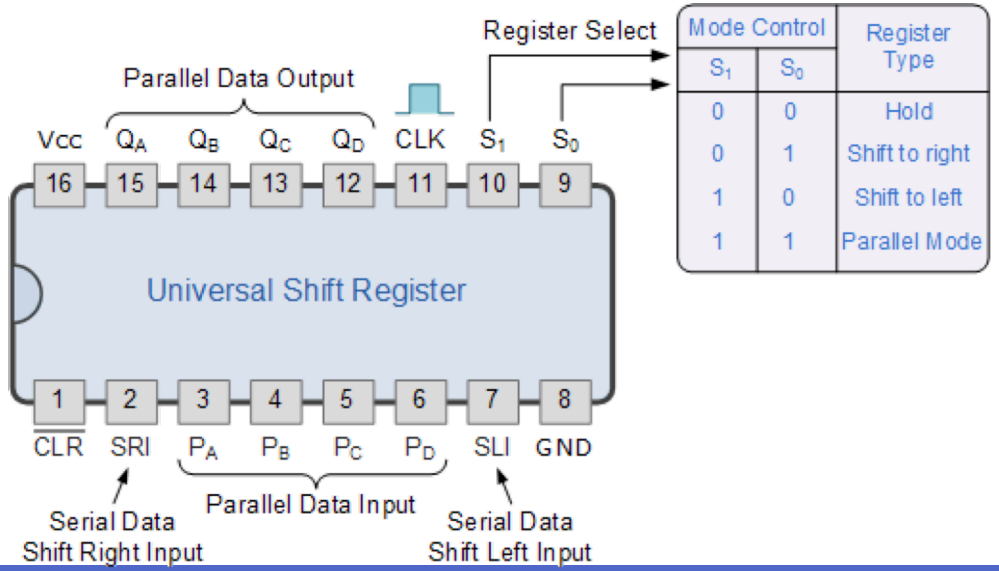
Department of Computer Engineering
IRAN University of Science and Technology

hfalahati@iust.ac.ir

Shift Register: Types

- Shift register
  - Parallel-in Parallel-out
  - Serial-in Serial-out
  - Serial-in Parallel-out
  - Parallel-in Serial-out





Outline

- Counters

Counters

Counter

- A circuit that **cycles** through a **specified** number of **states**
- Applications
  - Counting!
  - Chronometer
  - Traffic light

Counter: Type

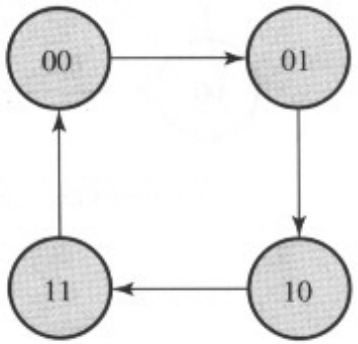
- Synchronous
  - Applies the same clock to all flip flops
  - Parallel counter

- Asynchronous
  - Do not have a common clock pulse
  - A.k.a., ripple counter
    - Ripple clock pulse
    - Flip-flop outputs can be used as a source of clock for other flip-flops

Asynchronous Counters

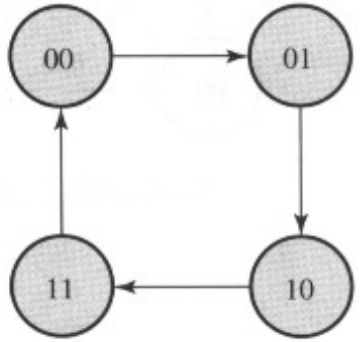
2-bit Ripple Counter

- Design a 2-bit ripple binary counter



2-bit Ripple Counter: Truth Table

- We need two output
  - Q_0 toggles every cycle
  - Q_1 toggles every two cycle

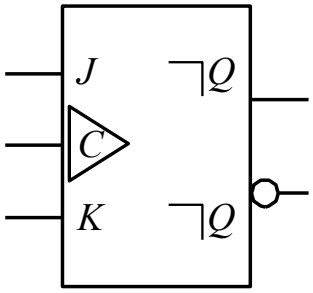


Q_1	Q_0
0	0
0	1
1	0
1	1

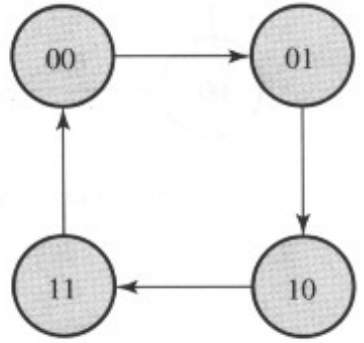
Recycle

2-bit Ripple Counter: Flip Flops

- Flip flops candidates
  - JK



J	K	Q	C	Q^*	
0	0	0	⎍	0	Hold
0	0	1	⎍	1	
0	1	0	⎍	0	Reset
0	1	1	⎍	0	
1	0	0	⎍	1	Set
1	0	1	⎍	1	
1	1	0	⎍	1	Toggle
1	1	1	⎍	0	

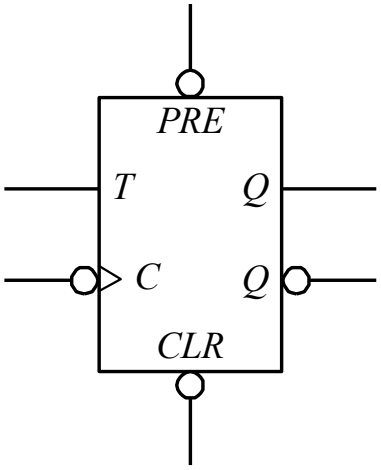


Q_1	Q_0
0	0
0	1
1	0
1	1

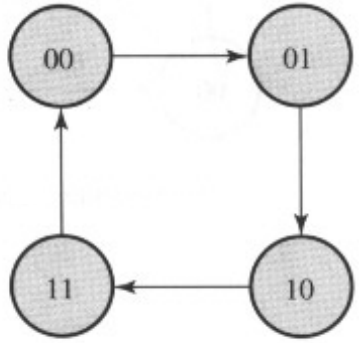
Recycle

2-bit Ripple Counter: Flip Flops

- Flip flops candidates
  - JK
  - T



T	Q	C	Q^*	
0	0	↓	0	Hold
0	1	↓	1	
1	0	↓	1	Toggle
1	1	↓	0	

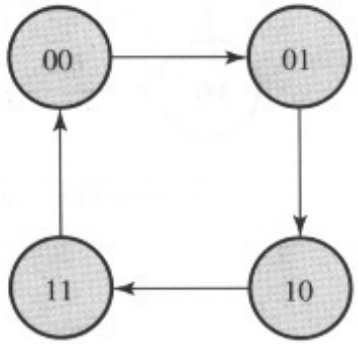


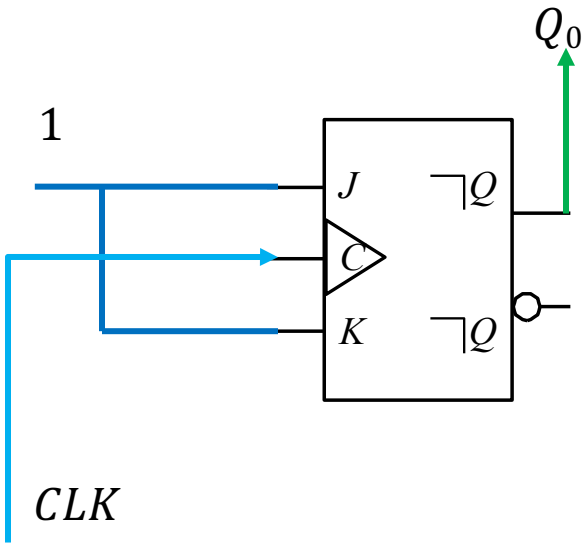
Q_1	Q_0
0	0
0	1
1	0
1	1

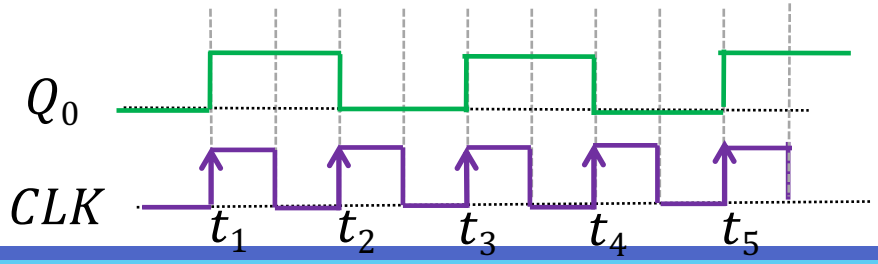
Recycle

2-bit Ripple Counter: JK FF

- Outputs
  - Q_0 toggles every cycle
  - Q_1 toggles every two cycle



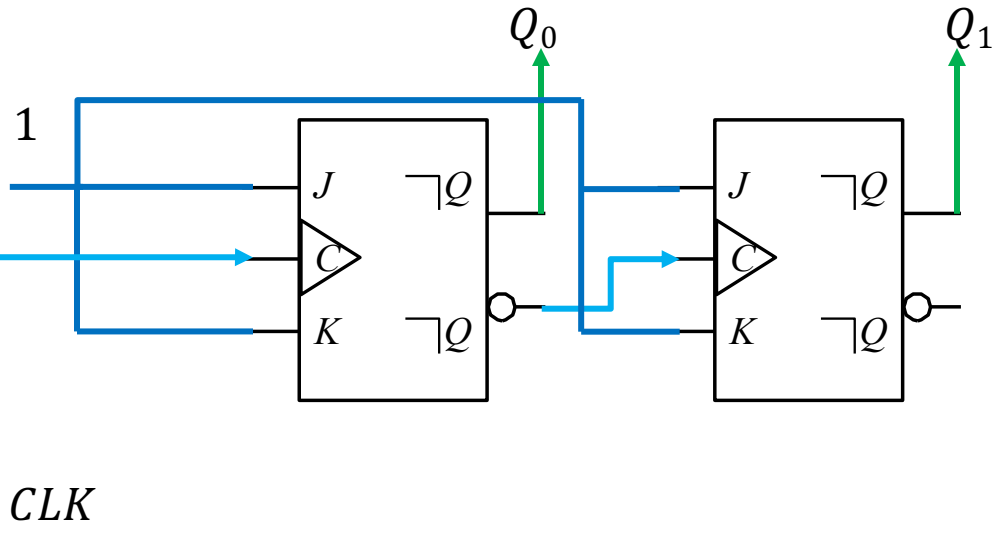


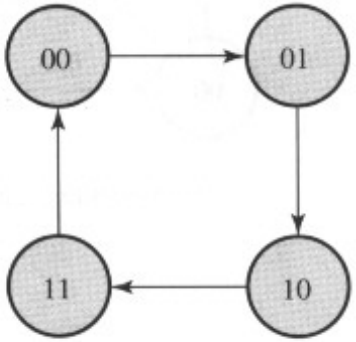


CLK	Q_0
↑	0
↑	1
↑	0
↑	1

2-bit Ripple Counter: JK FF (cont'd)

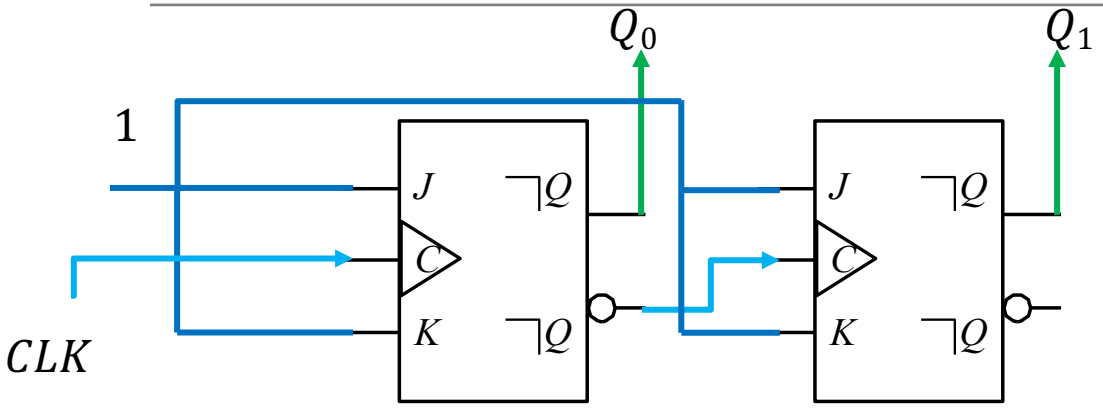
- Outputs
  - Q_0 toggles every cycle
  - Q_1 toggles every two cycle

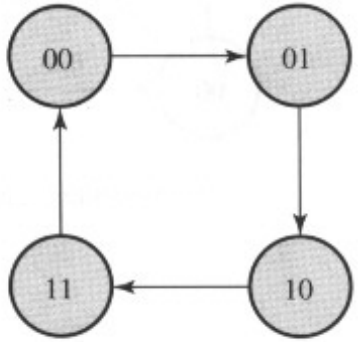




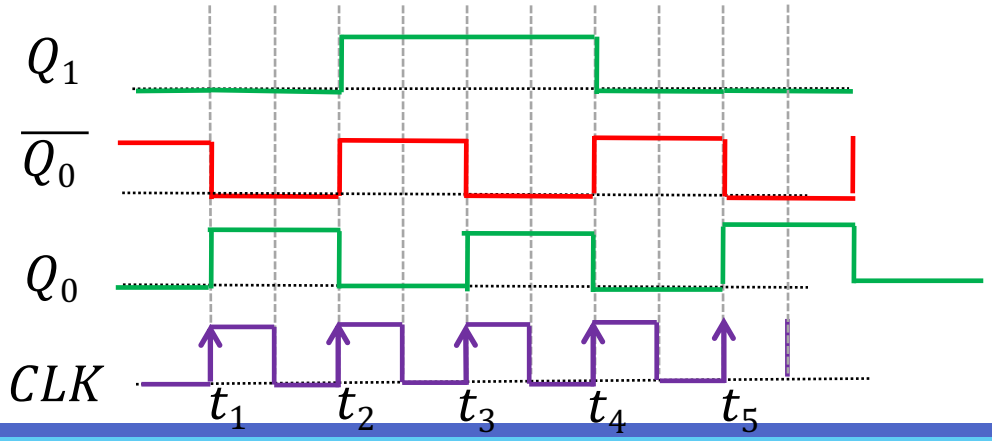
CLK	Q_1	Q_0	$\overline{Q_0}$
↑	0	0	1
↑	0	1	0
↑	1	0	1
↑	1	1	0

2-bit Ripple Counter: JK FF (cont'd)



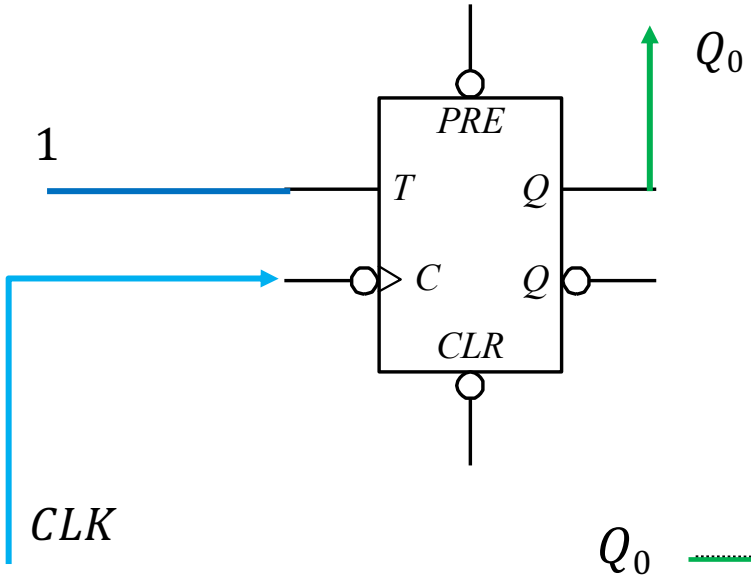


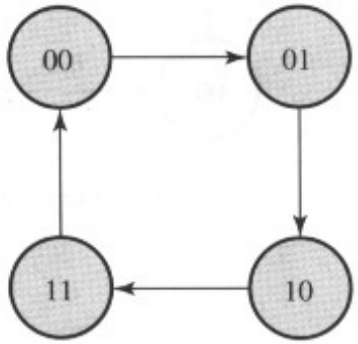
CLK	Q_1	Q_0	$\overline{Q_0}$
↑	0	0	1
↑	0	1	0
↑	1	0	1
↑	1	1	0



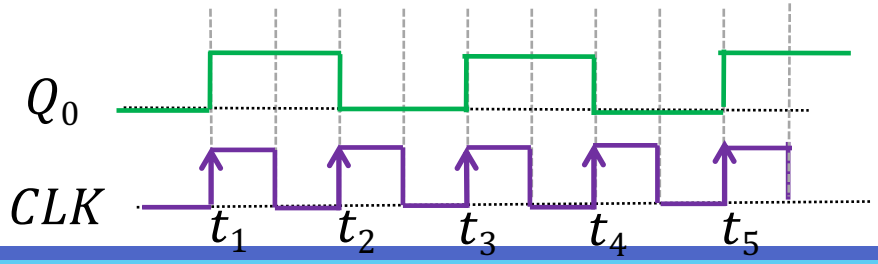
2-bit Ripple Counter: T FF

- Outputs
  - Q_0 toggles every cycle
  - Q_1 toggles every two cycle



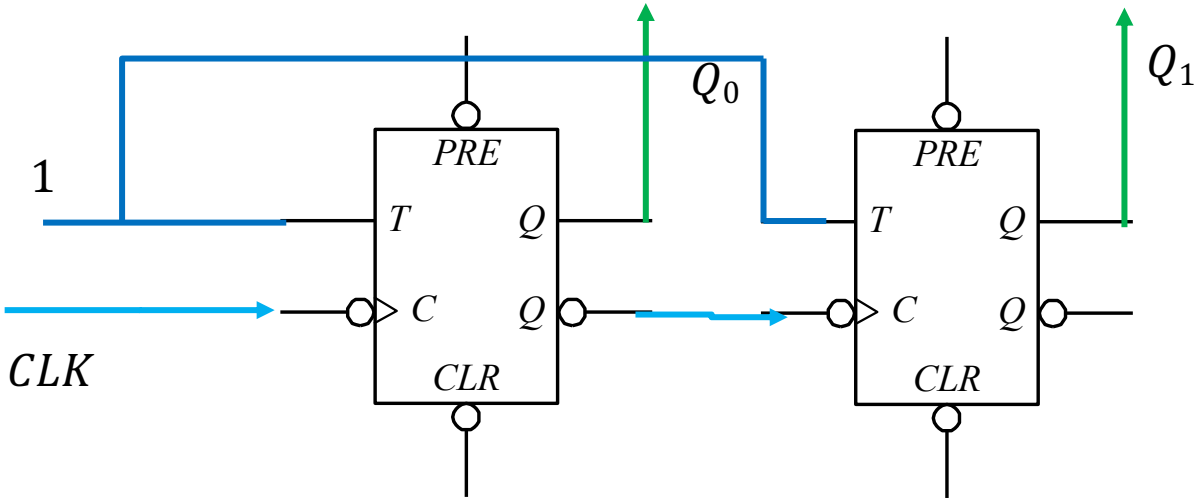


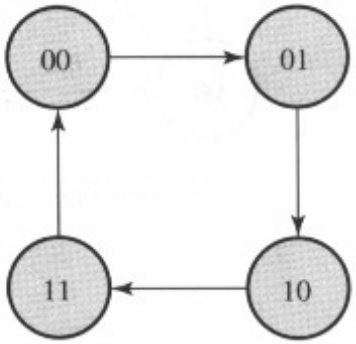
CLK	Q_0
↑	0
↑	1
↑	0
↑	1



2-bit Ripple Counter: T FF

- Outputs
  - Q_0 toggles every cycle
  - Q_1 toggles every two cycle

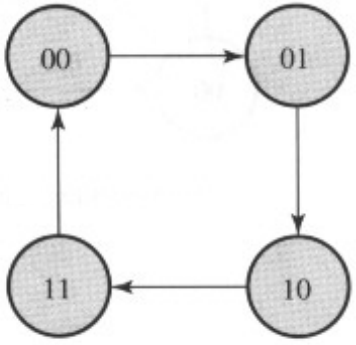




CLK	Q_0
↑	0
↑	1
↑	0
↑	1

2-bit Ripple Counter: D FF

- Flip flops candidates
  - D

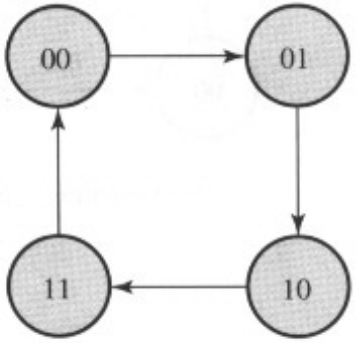


Q_1	Q_0
0	0
0	1
1	0
1	1

Recycle

2-bit Ripple Counter: D FF

- Flip flops candidates
  - D
  - Negative edge

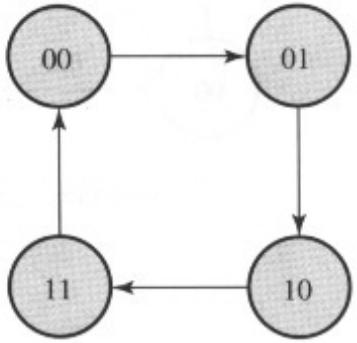


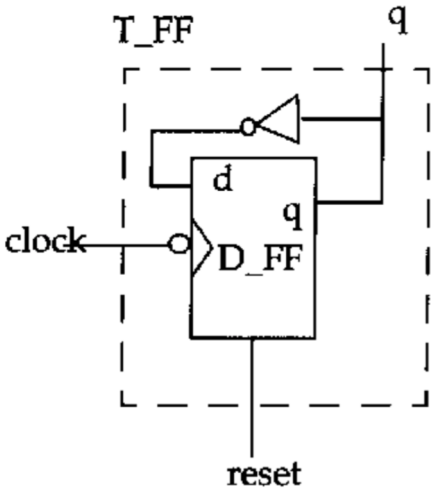
Q_1	Q_0
0	0
0	1
1	0
1	1

Recycle

2-bit Ripple Counter: D FF

- Flip flops candidates
  - D
  - Negative edge



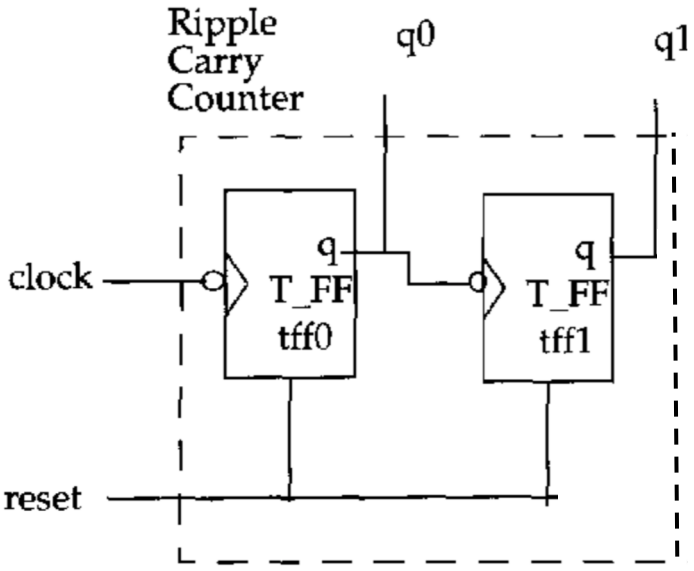


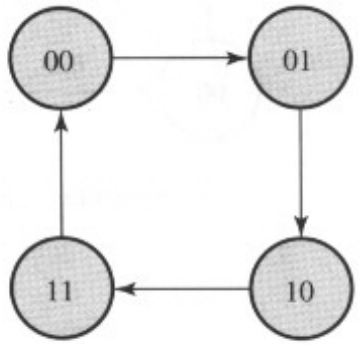
Q_1	Q_0
0	0
0	1
1	0
1	1

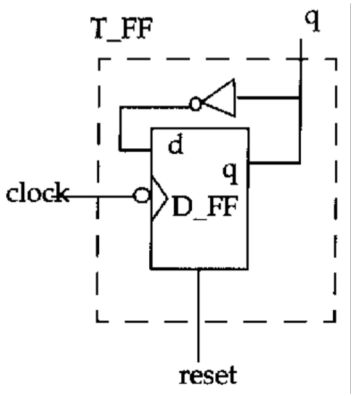
Recycle

2-bit Ripple Counter: D FF

- Flip flops candidates
  - D
  - Negative edge





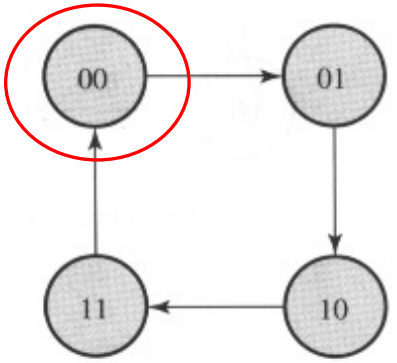


Q_1	Q_0
0	0
0	1
1	0
1	1

Recycle

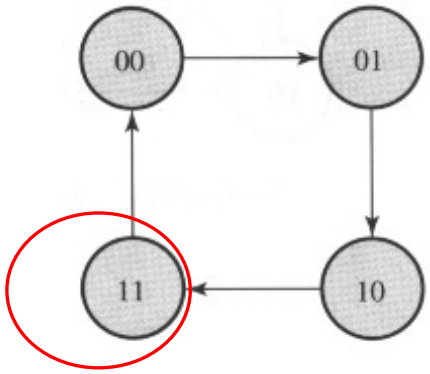
Asynchronous Up/Down Counter

- Up counters
  - Counts upward from zero to the maximum value
- Down counters
  - Counts down from the maximum to zero value



Q_1	Q_0
1	1
1	0
0	1
0	0

Recycle



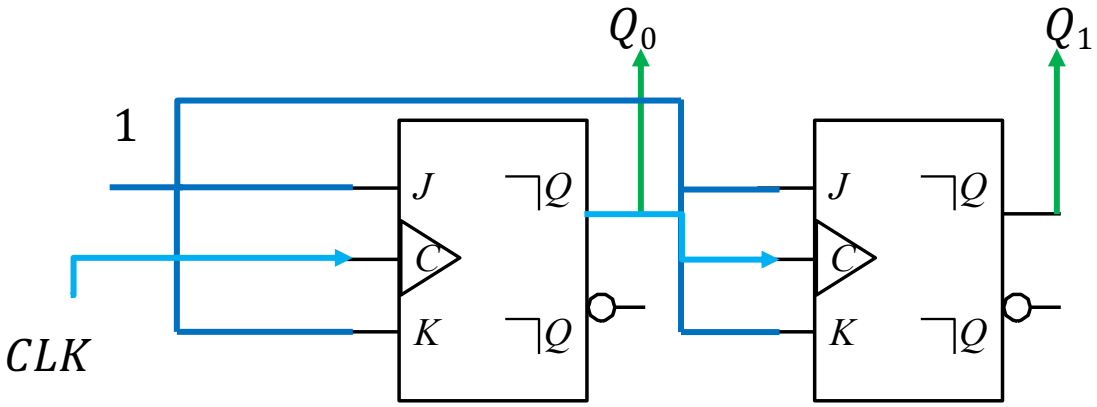
Q_1	Q_0
0	0
0	1
1	0
1	1

Recycle

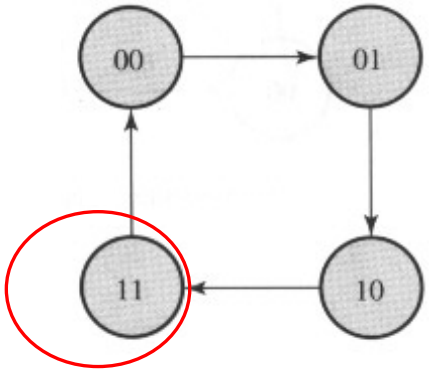
Asynchronous Up/Down Counter

- 2-bit down counter

Q_1	Q_0
1	1
1	0
0	1
0	0

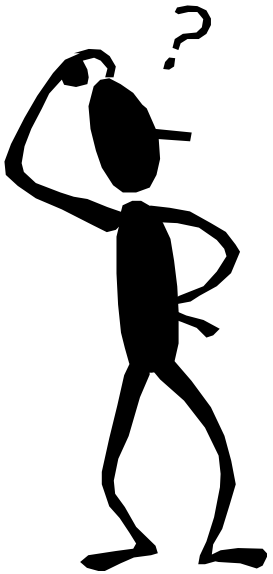


Recycle



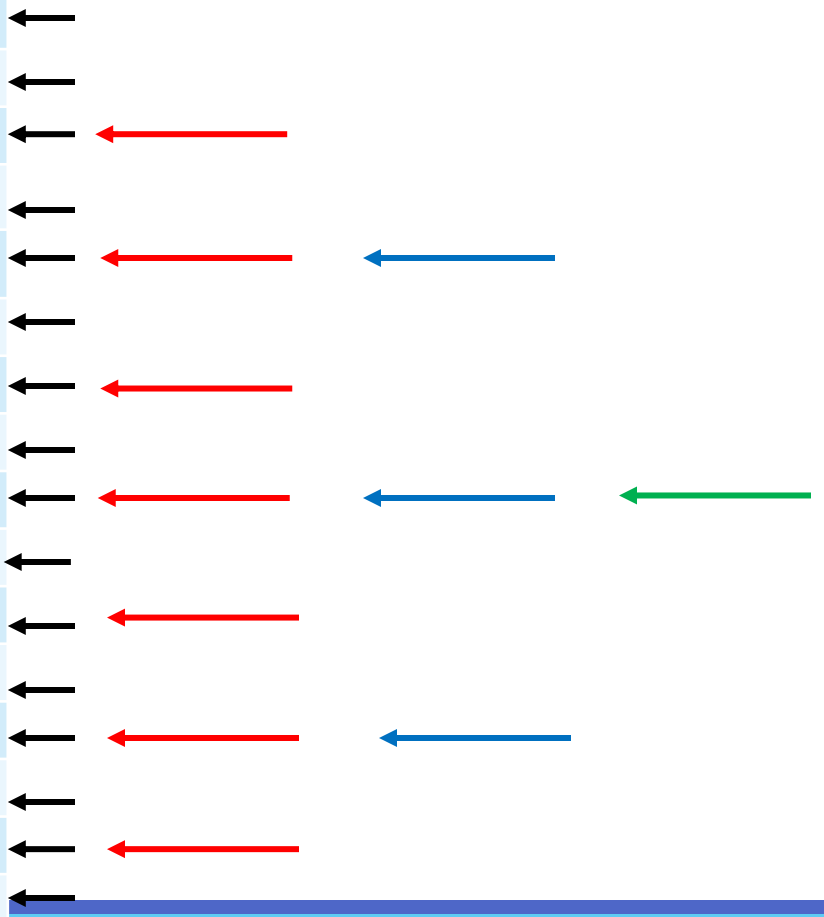
Design an Asynchronous Counter

- Design an asynchronous 4-bit counter
  - Counting upward

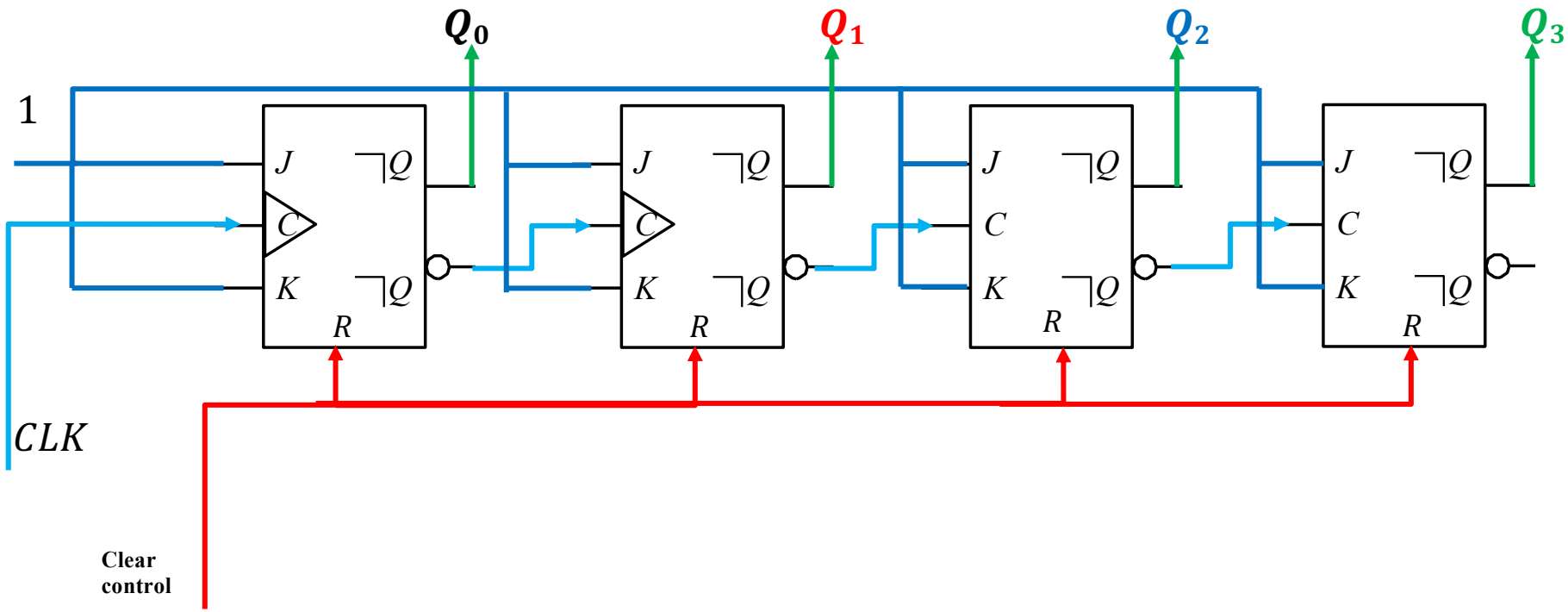


Asynchronous 4-bit Counter

Q_3	Q_2	Q_1	Q_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
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

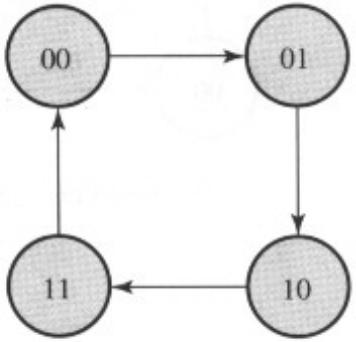


Asynchronous 4-bit Counter

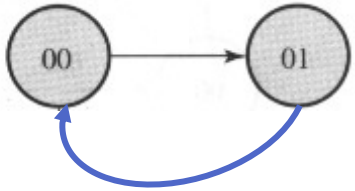


Asynchronous Counter with Mod K

- Mod-2^n counters
  - Counts through 2^n states
  - n flip flops

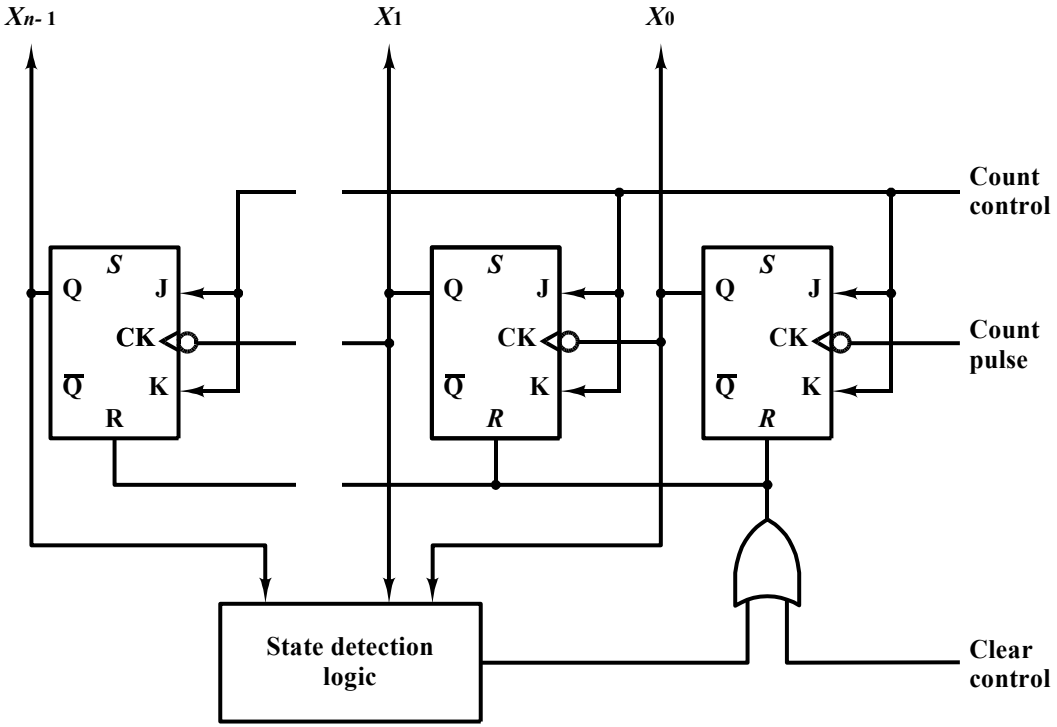


- Mod-x counters
  - $x<2^n$
  - Counts through x states
  - Truncated sequence



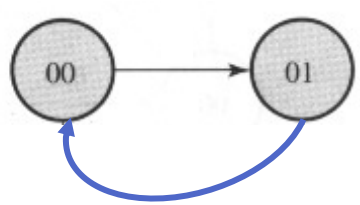
Asynchronous Counter with Mod K (cont'd)

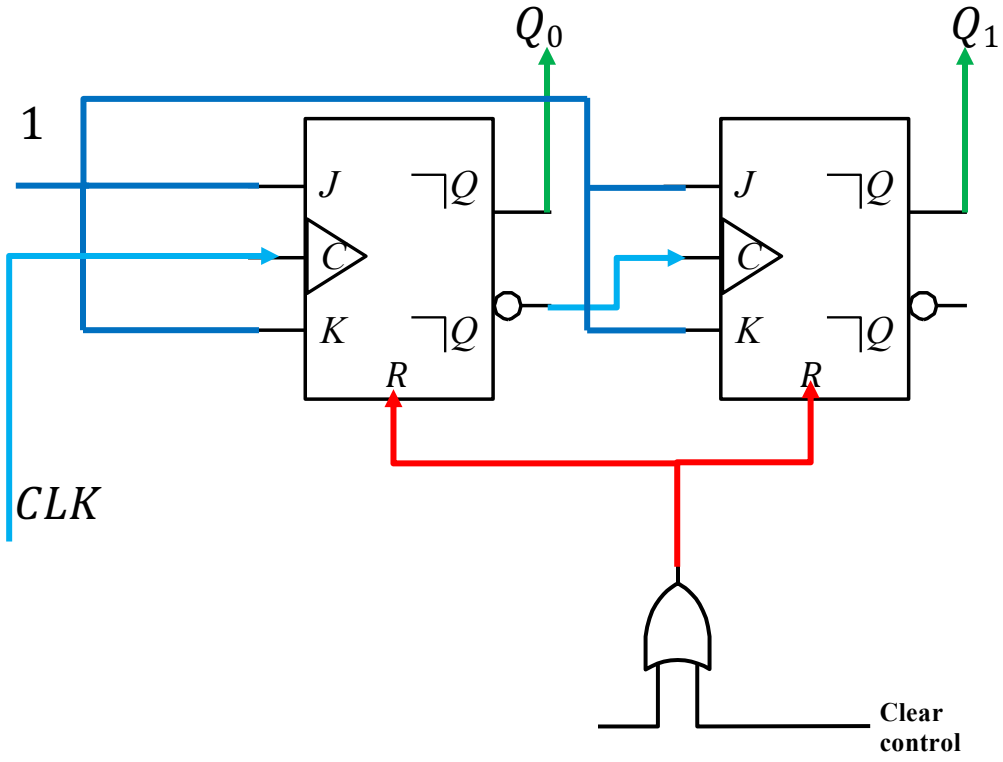
- How to design a Mod-x counters
  - Recycle the counter *at a special state*
  - *Not traverse through all of the states* in the binary sequence

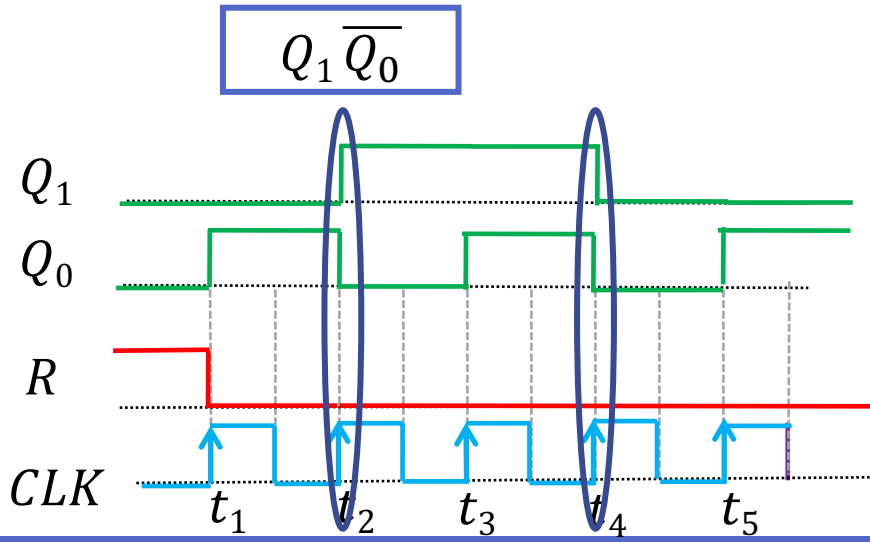


Asynchronous Counter with Mod-2

- Example
  - Design a Mode-2 2-bit counter

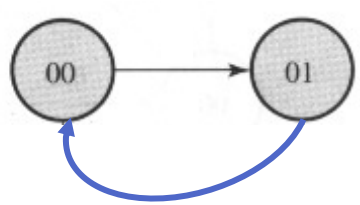


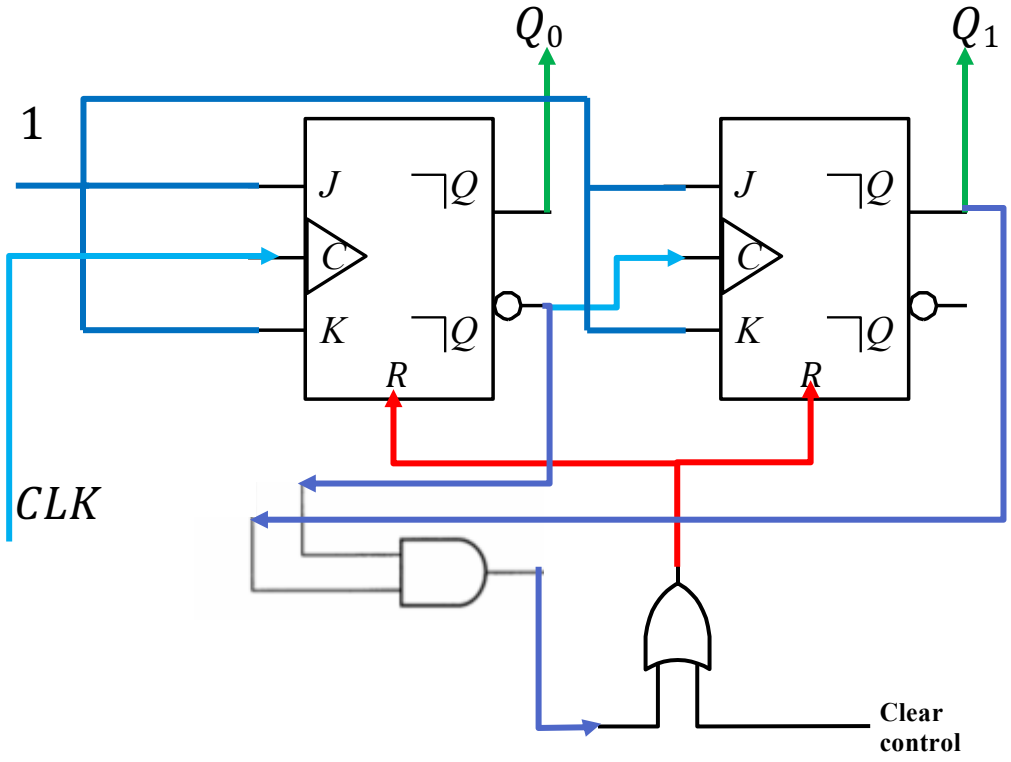


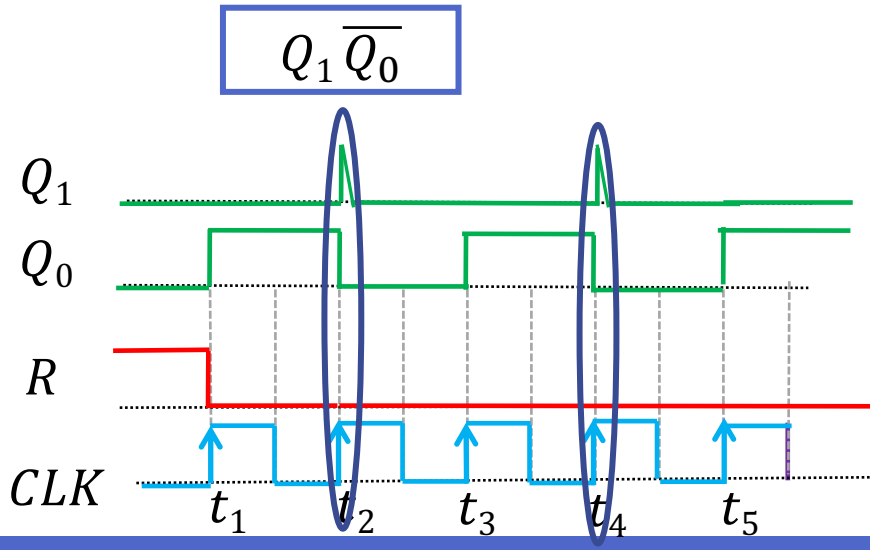


Asynchronous Counter with Mod-2

- Example
  - Design a Mode-2 2-bit counter

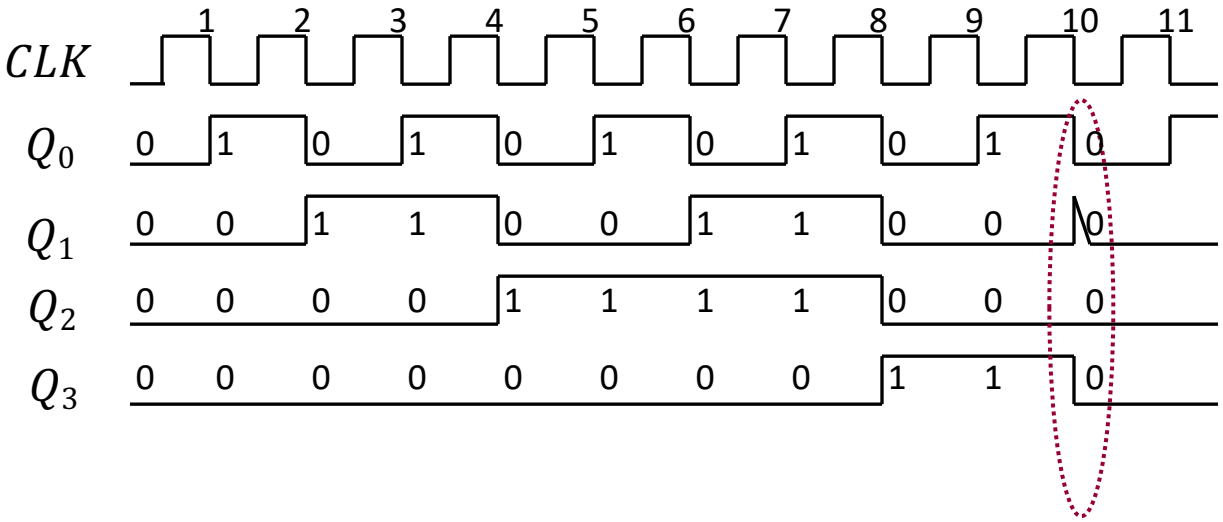






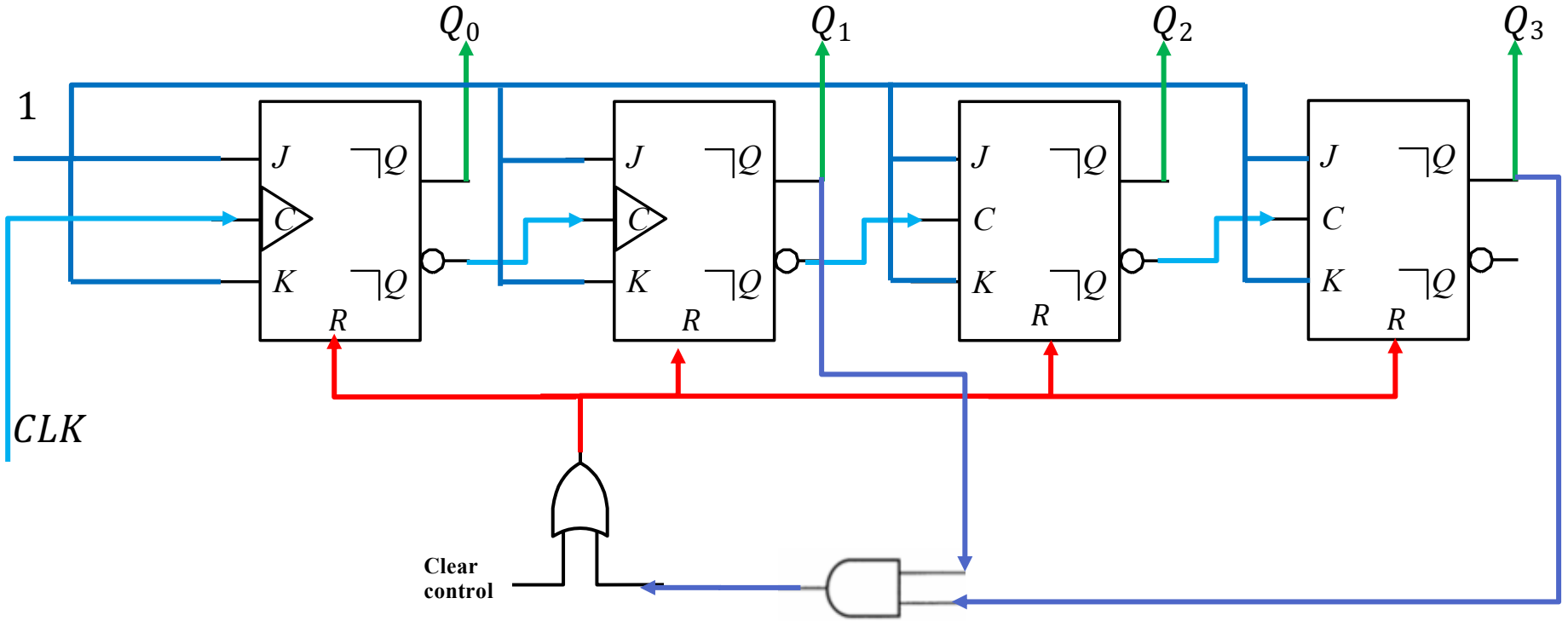
Asynchronous Counter with Mod-10

- Decade counter
- BCD counter
  - Mode-10 4-bit counter



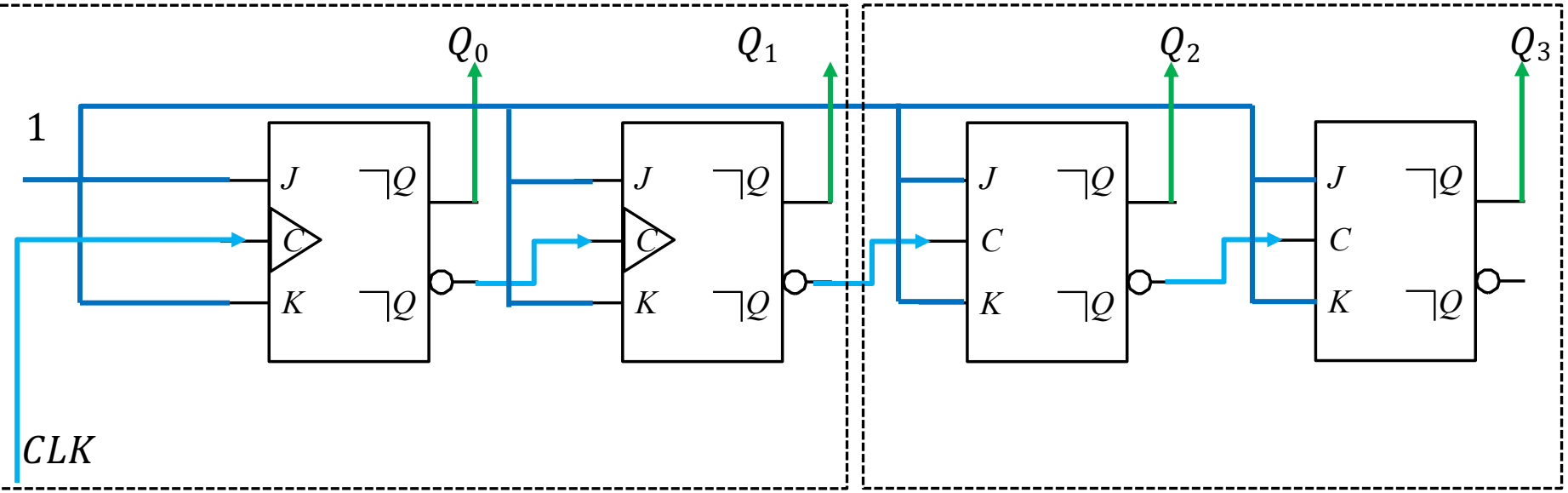
Asynchronous Counter with Mod-10 (cont'd)

- Decade counter
- BCD counter
  - Mode-10 4-bit counter



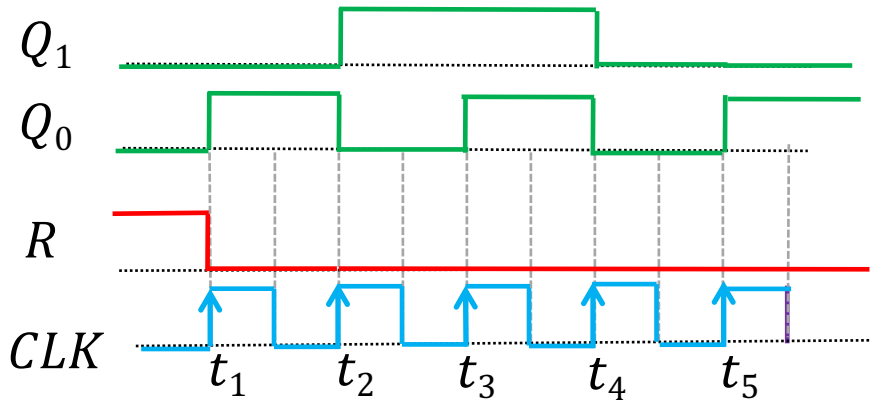
Asynchronous Counter: Cascading

- Cascading smaller ripple counters -> Larger ripple counters
  - 2 2-bit counters → 4-bit counter



Asynchronous Counter: Frequency Divider

- Frequency divider
  - Output of the last flip-flop (MSB) divides the input clock frequency by the MOD number of the counter

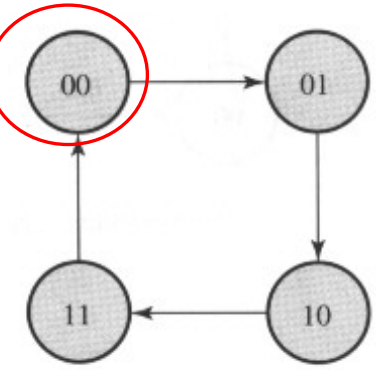


Synchronous Counters

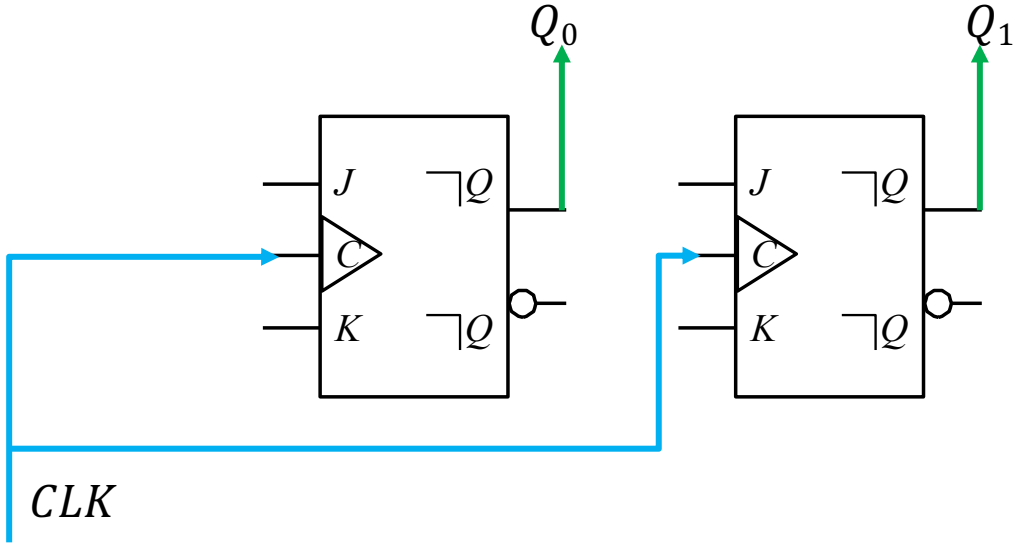
Synchronous Counter

- Parallel counter
  - A common clock pulse
  - All flip flops are clocked at the same time

- Example: 2-bit synchronous binary counter
  - *J-K FFs*

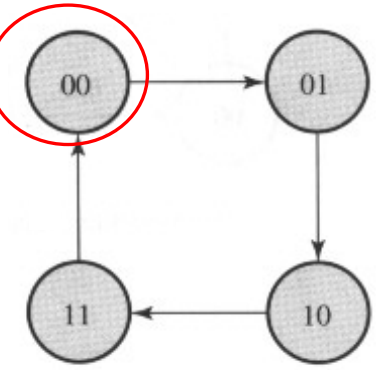


Q_1	Q_0
0	0
0	1
1	0
1	1

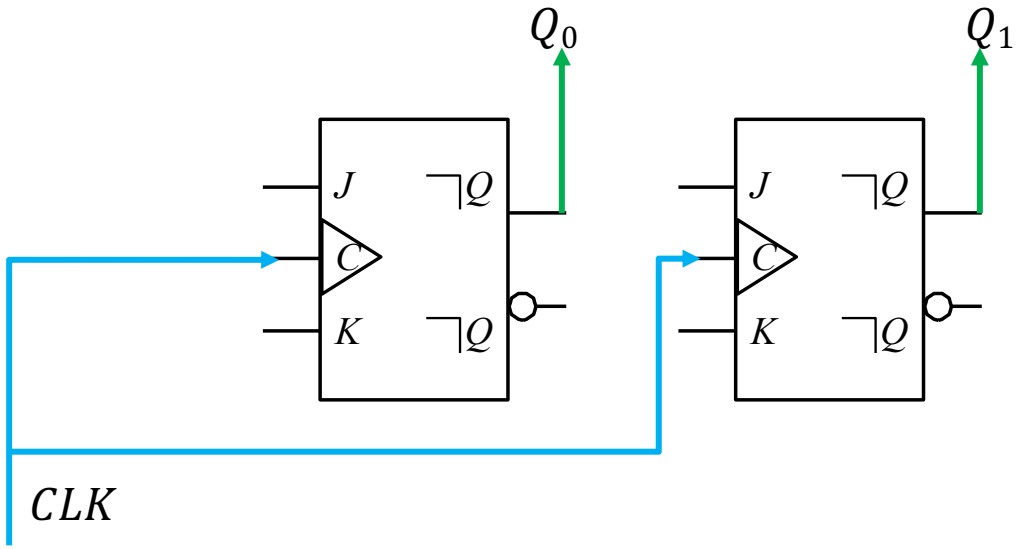


2-bit Synchronous Counter: JK-FF

- Example: 2-bit synchronous binary counter
  - J-K FFs
  - Determine flip flop inputs based on inputs and current states
  - Move from current state to the next one



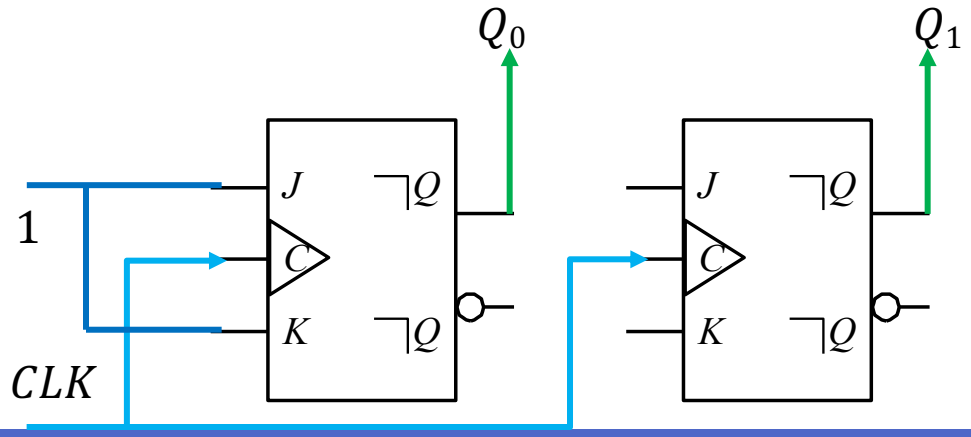
Q_1	Q_0
0	0
0	1
1	0
1	1



2-bit Synchronous Counter: JK-FF (cont'd)

Present State		Next State		FF0		FF1	
Q_1	Q_0	Q_1	Q_0	J	K	J	K
0	0	0	1	1	1		
0	1	1	0	1	1		
1	0	1	1	1	1		
1	1	0	0	1	1		

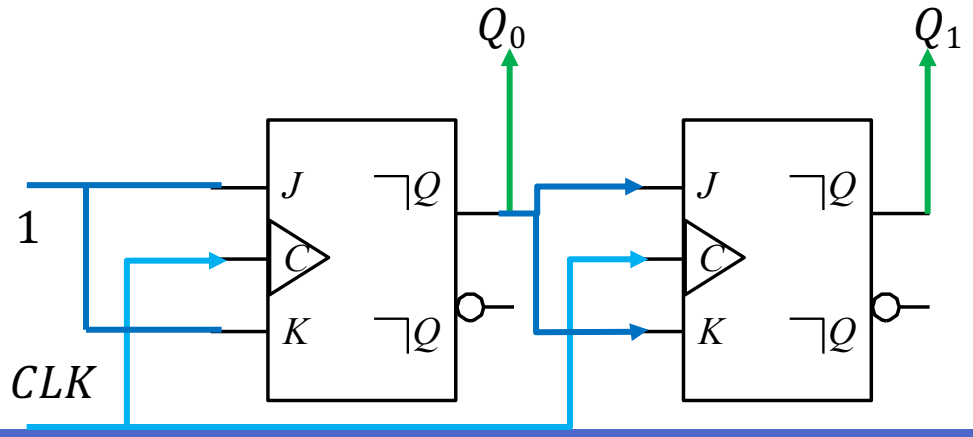
- FF0
  - Q_0 is toggling
  - FF should be in toggling state
  - => J=K =1



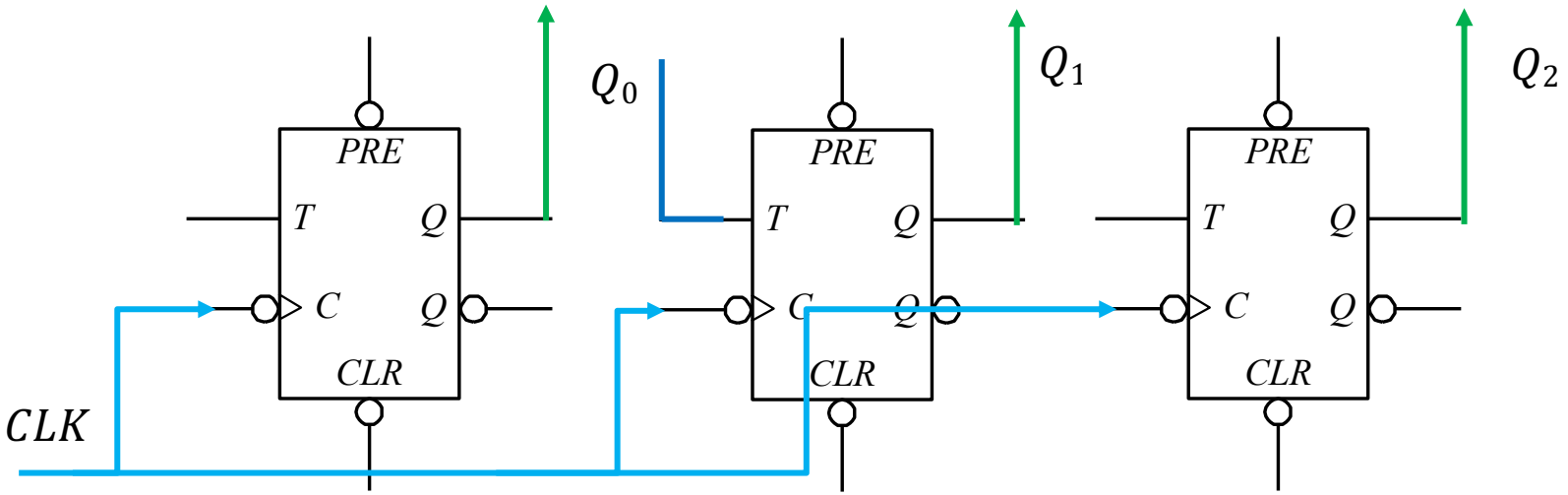
2-bit Synchronous Counter: JK-FF (cont'd)

Present State		Next State		FF0		FF1	
Q_1	Q_0	Q_1	Q_0	J	K	J	K
0	0	0	1	1	1	0	0
0	1	1	0	1	1	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	1	1

- FF1
  - J = K = Q_0



3-bit Synchronous Counter: T-FF



3-bit Synchronous Counter: T-FF (cont'd)

Present State			Next State			Flip Flops		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1			1
0	0	1	0	1	0			1
0	1	0	0	1	1			1
0	1	1	1	0	0			1
1	0	0	1	0	1			1
1	0	1	1	1	0			1
1	1	0	1	1	1			1
1	1	1	0	0	0			1

3-bit Synchronous Counter: T-FF (cont'd)

Present State			Next State			Flip Flops		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1		0	1
0	0	1	0	1	0		1	1
0	1	0	0	1	1		0	1
0	1	1	1	0	0		1	1
1	0	0	1	0	1		0	1
1	0	1	1	1	0		1	1
1	1	0	1	1	1		0	1
1	1	1	0	0	0		1	1

3-bit Synchronous Counter: T-FF (cont'd)

Present State			Next State			Flip Flops		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

$$T_0 = 1$$

$$T_1 = Q_0$$

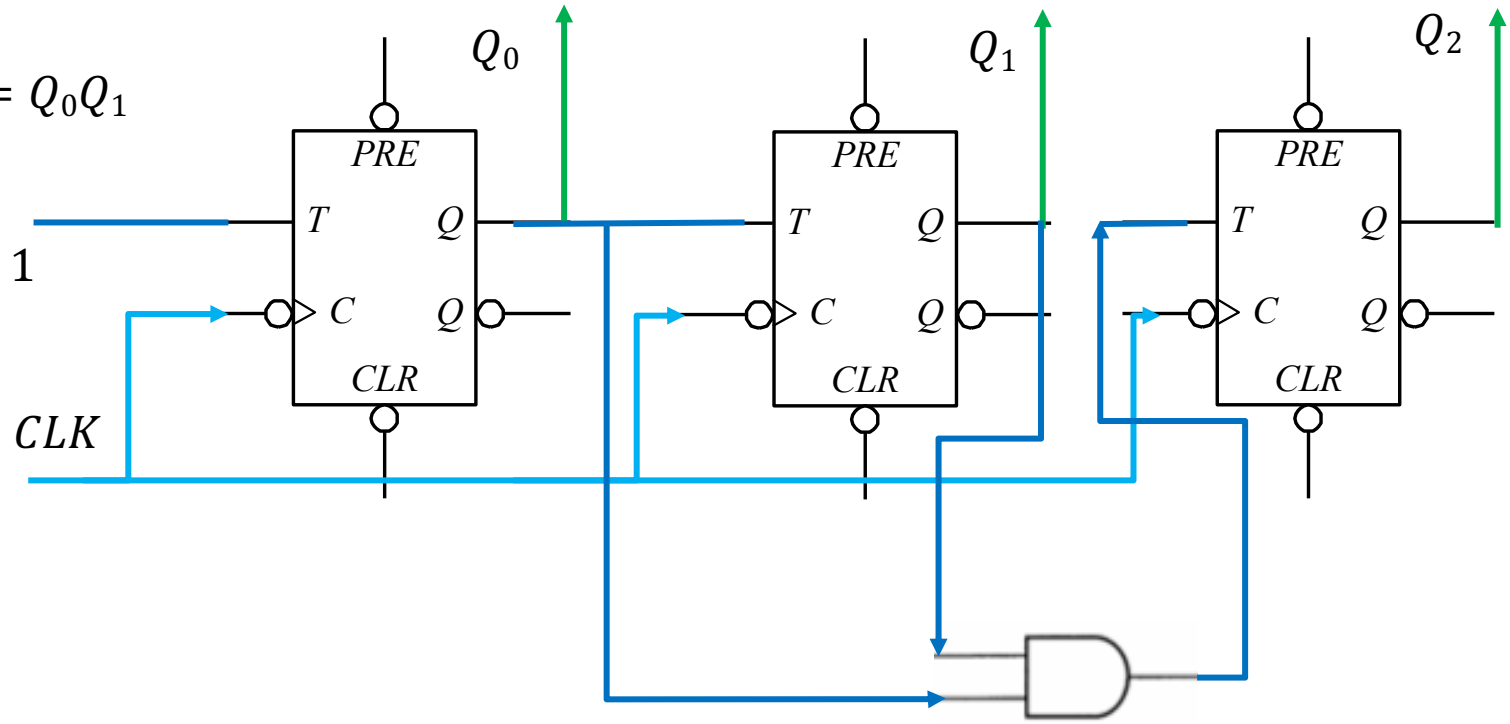
$$T_2 = Q_0 Q_1$$

3-bit Synchronous Counter: T-FF (cont'd)

$T_0 = 1$ $\quad T_n = Q_0 Q_1 \ldots Q_{n-1}$

$T_1 = Q_0$

$T_2 = Q_0 Q_1$



Synchronous BCD Counter: T-FF

Present State				Next State				Flip Flops			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

$$T_0 = 1$$

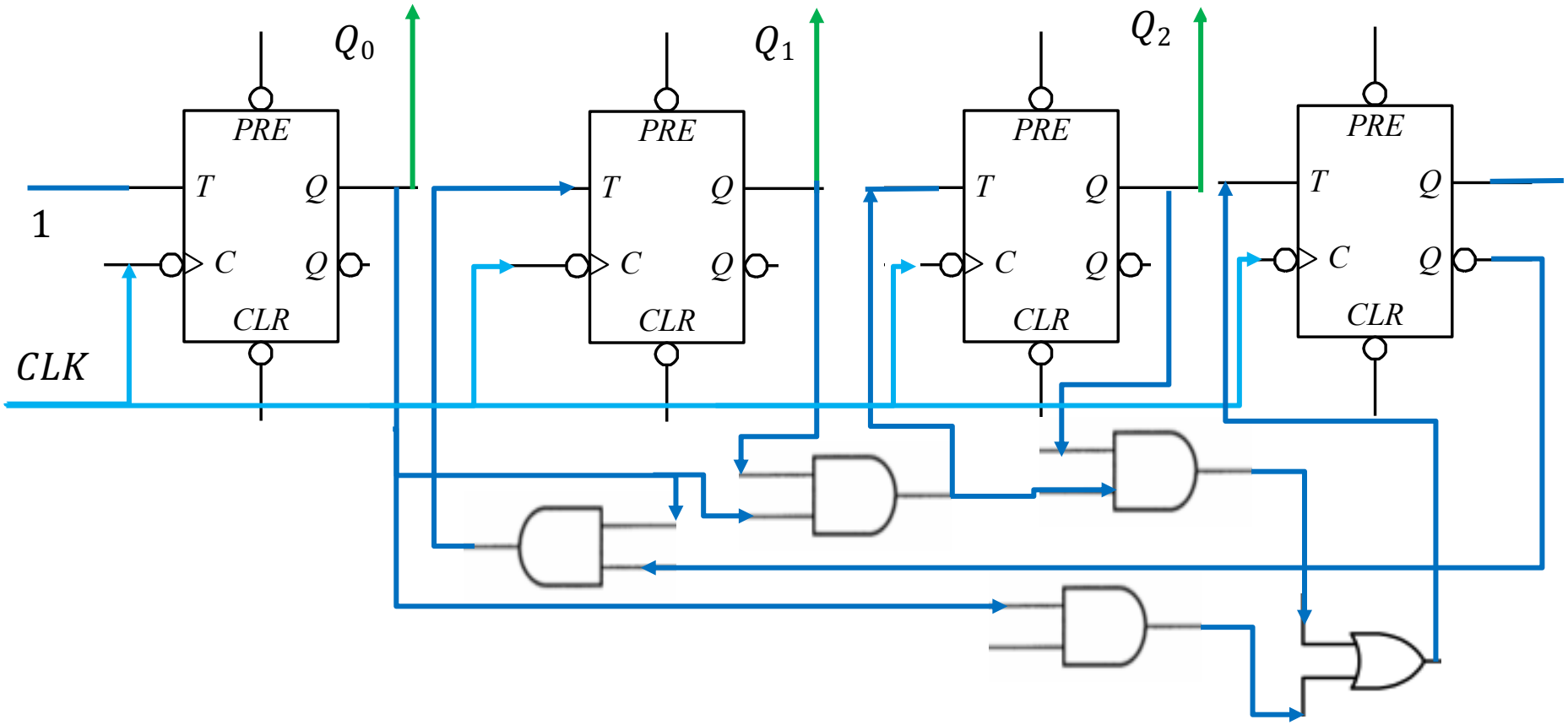
$$T_1 = \overline{Q_3} Q_0$$

$$T_2 = Q_0 Q_1$$

$$T_3 = Q_0 Q_1 Q_2 + Q_0 Q_3$$

Synchronous BCD Counter: T-FF(cont'd)

$T_0 = 1$ $T_2 = Q_0Q_1$ $T_1 = \overline{Q_3}Q_0$ $T_3 = Q_0Q_1Q_2 + Q_0Q_3$



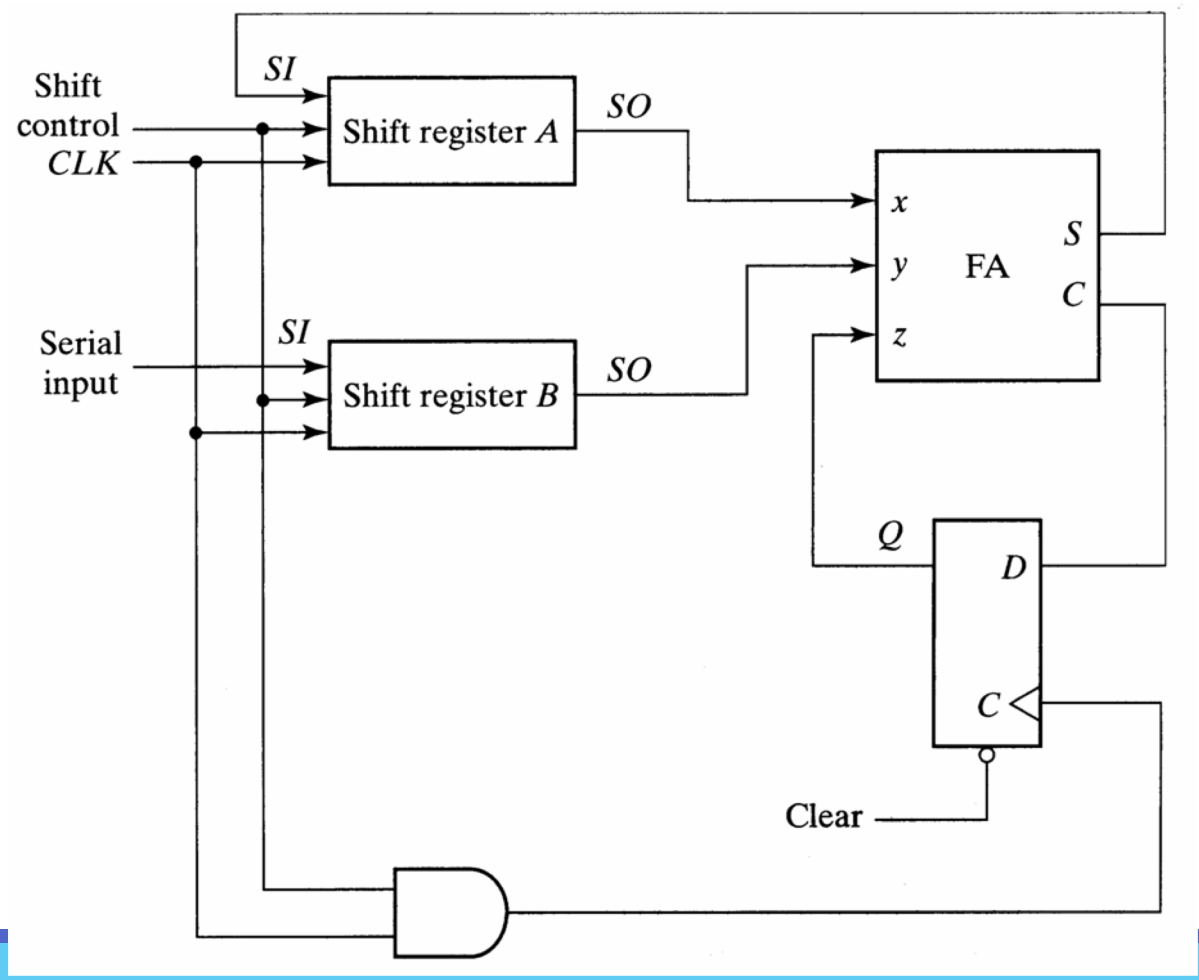
Design Sample

Sample 1

- Design an adder using flip flops.

Sample 1 (cont'd)

- Serial adder

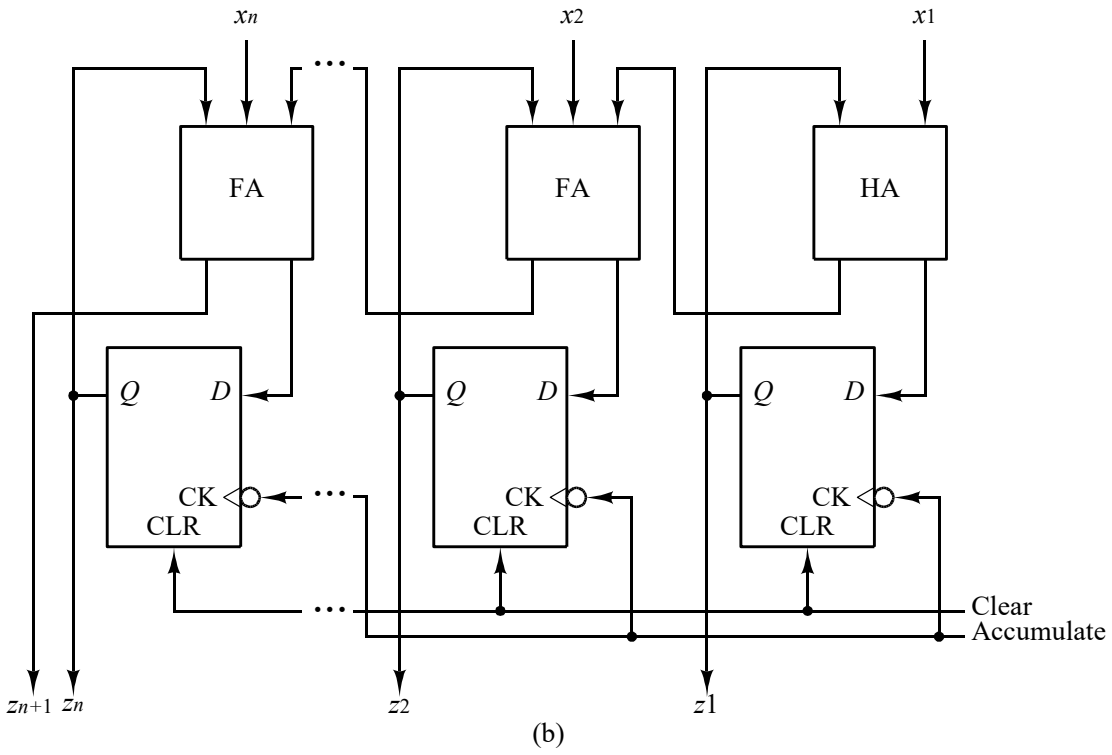


Sample 2

- Design an accumulator using flip flips.
- An adder that totals a series of binary data

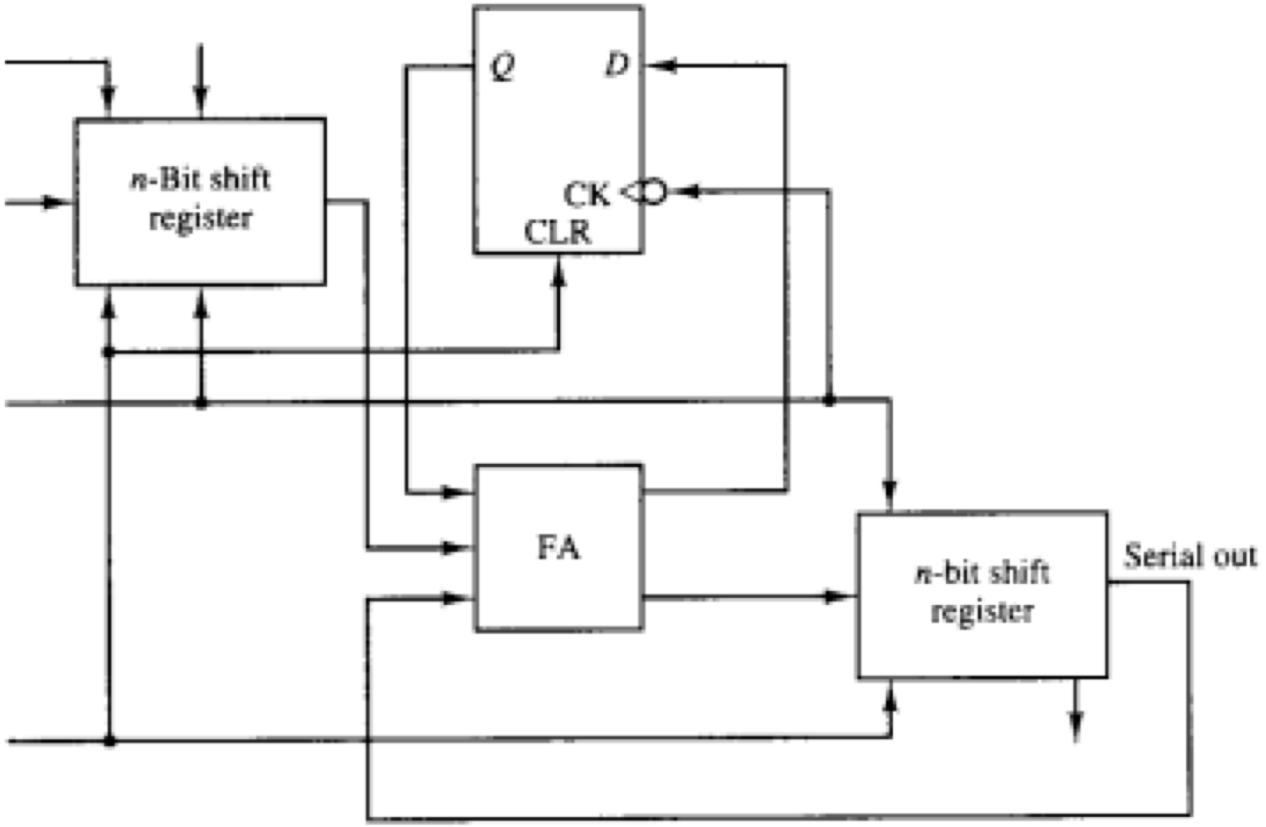
Sample 2 (cont'd)

- Parallel Accumulator using flip flips



Sample 2 (cont'd)

- Serial accumulator using flip flips



Thank You