Digital Circuits

ECS 371

Dr. Prapun Suksompong

prapun@siit.tu.ac.th

Lecture 18

Office Hours:
BKD 3601-7
Monday 9:00-10:30, 1:30-3:30
Tuesday 10:30-11:30

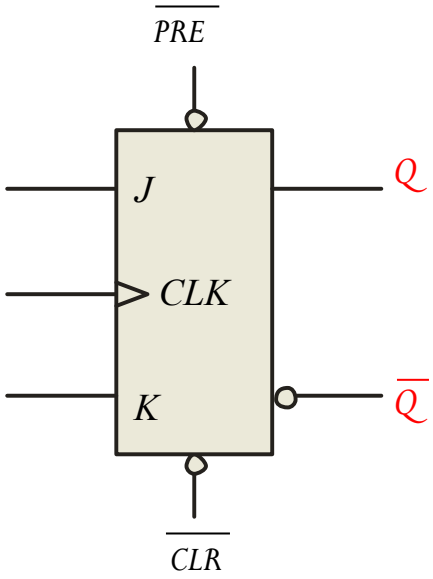
ECS371.PRAPUN.COM

Announcement

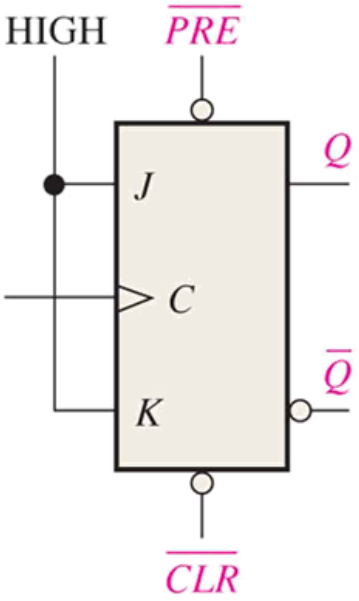
- Reading Assignment:
  - Chapter 7: 7-1, 7-2, 7-4
  - Chapter 8: 8-1, 8-2
- HW6: Chapter 7
  - Q: 2, 4, 8, 10, 12, 14, 18
  - Due: 4:30 PM, August 19 (Wednesday)

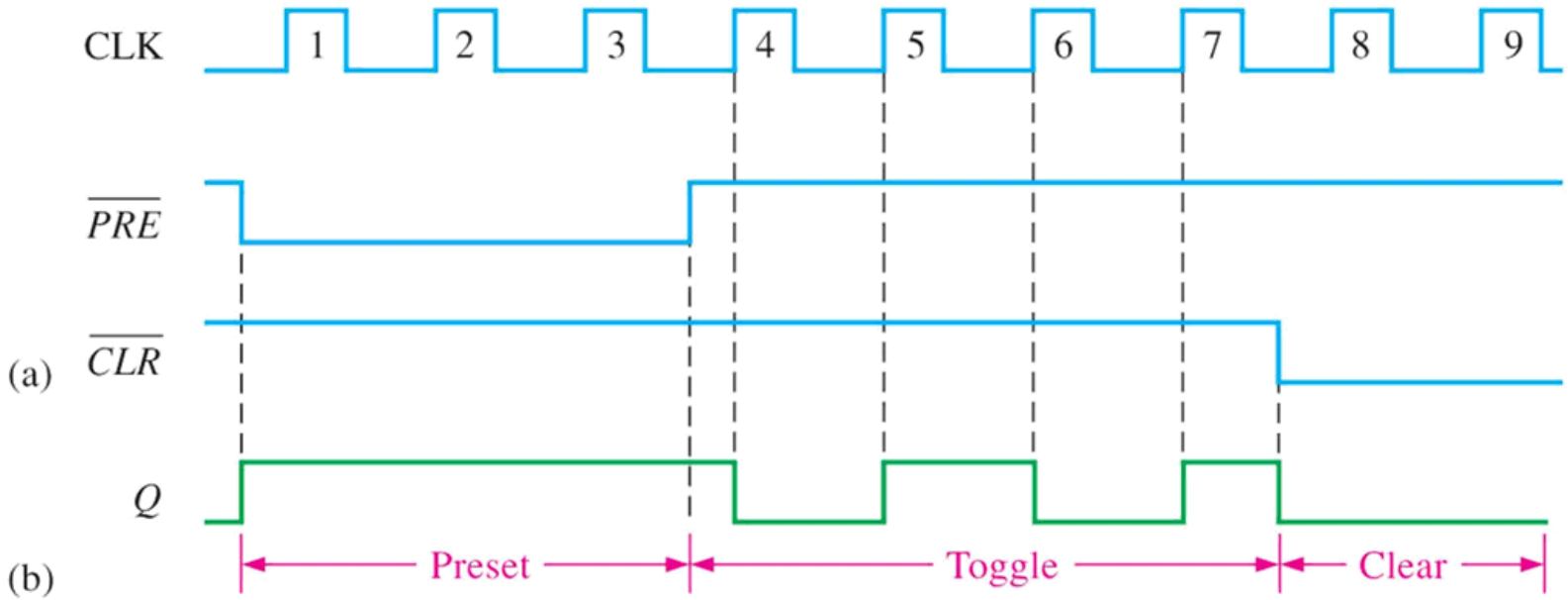
Asynchronous Inputs

- Most flip-flops have ***extra*** inputs that are *asynchronous*, meaning they affect the output independent of the clock.
- Two such inputs are normally labeled **preset (PRE)** and **clear (CLR)**.
- These inputs are usually active-LOW.
- A J-K flip flop with active-LOW preset and CLR is shown.



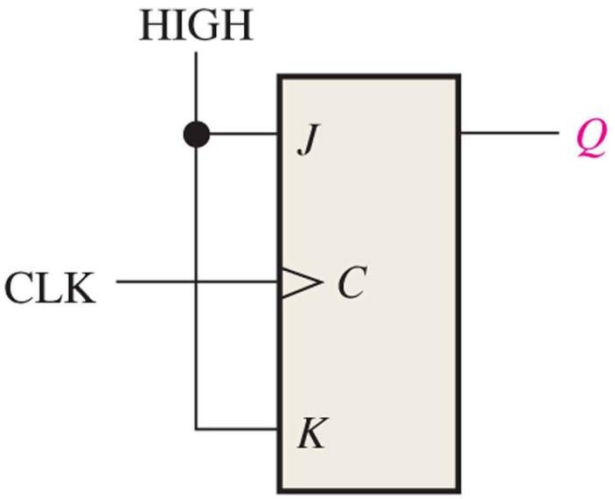
Example

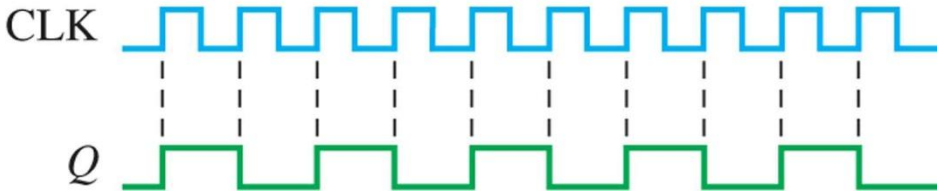

HIGH
$\overline{PRE}$
Q
J
C
$\overline{Q}$
K
$\overline{CLR}$


CLK
1
2
3
4
5
6
7
8
9
$\overline{PRE}$
$\overline{CLR}$
(a)
Q
(b)
Preset
Toggle
Clear

Some Applications

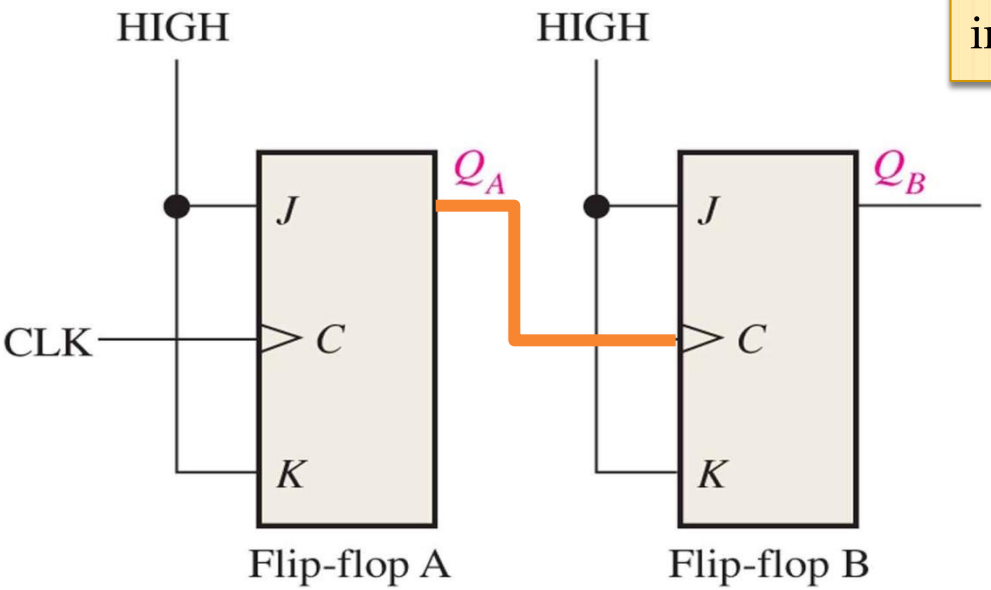
- Divide the clock frequency by 2

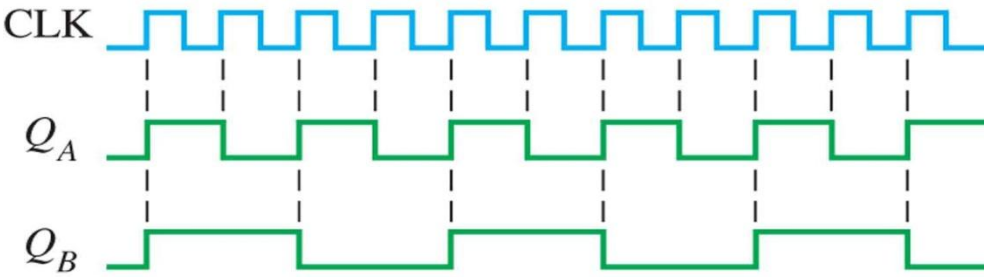




Some Applications

- Divide the clock frequency by 4

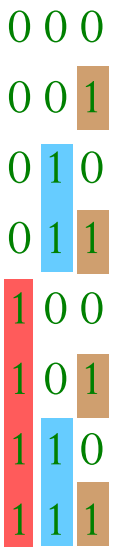




Note that FF B will be synced to the positive-edge of Q_A instead of CLK!

Counting in Binary

- We have seen that the binary count sequence follows a familiar pattern of 0's and 1's.



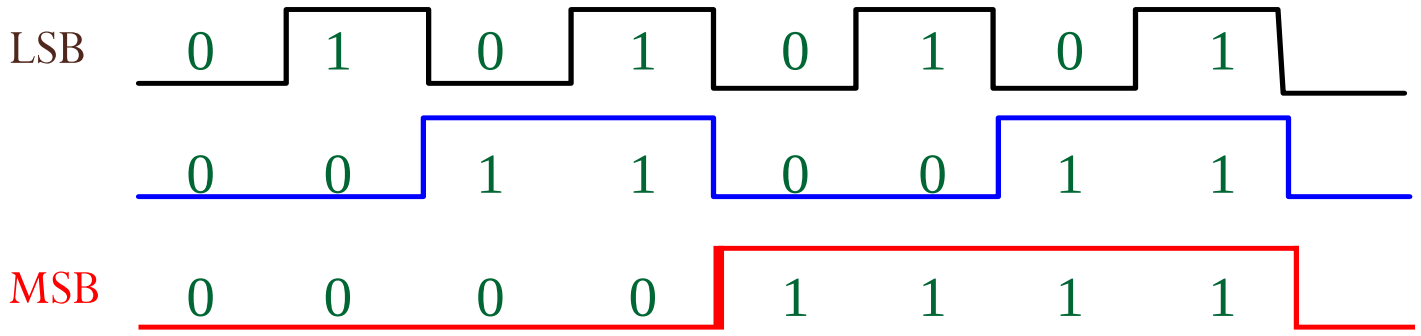
LSB changes on every number.

The next bit changes on every other number.

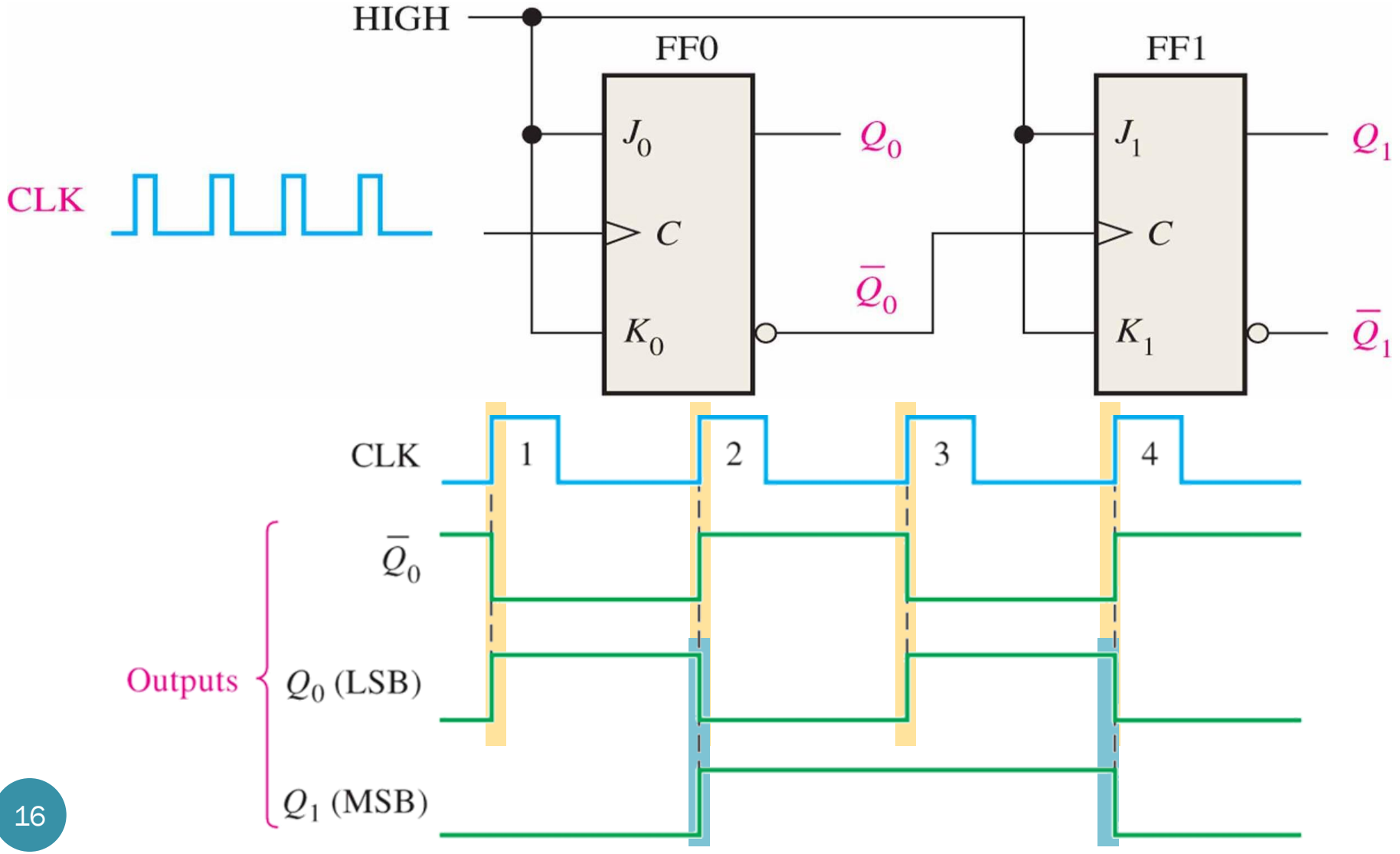
The next bit changes on every fourth number.

Counter

- A counter can form the same pattern of 0's and 1's with logic levels.
- The first stage in the counter represents the least significant bit (LSB) – notice that these waveforms follow the same pattern as counting in binary.



Ex: 2-bit Asynchronous Binary Counter

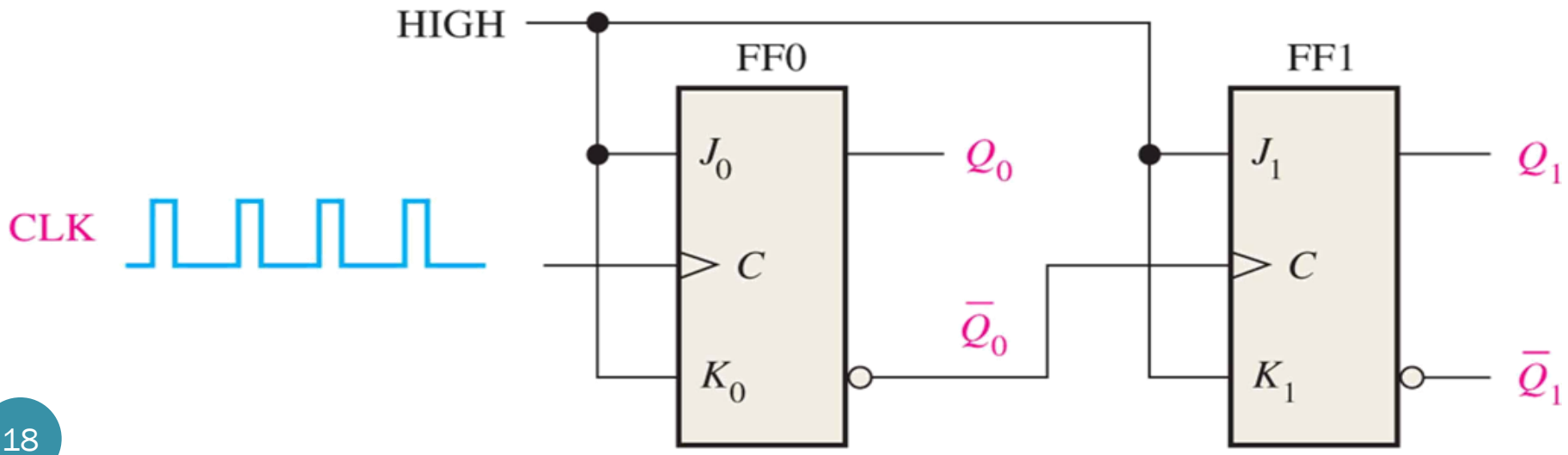


Asynchronous counter

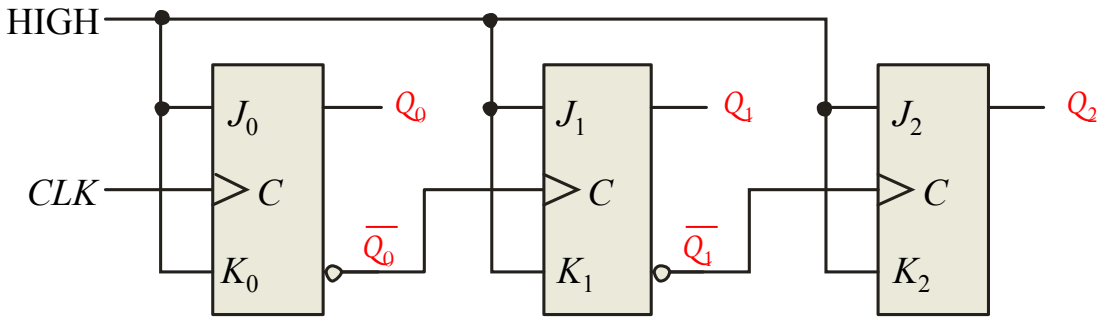
- Counters are classified into two broad categories according to the way they are clocked:
  1. asynchronous
  2. Synchronous
- The term **asynchronous** refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time.
- **Asynchronous counters**
  - The FFs within the counter do not change states at *exactly* the same time because they do not have a common clock pulse.

Asynchronous counter (con't)

- Commonly called **ripple counters**
- The FFs are connected for toggle operation (J = 1, K = 1)
- The first FF is clocked by the external clock pulse (CLK).
- Each successive FF is clocked by the output of the preceding FF.



Ex: 3-bit Asynchronous Counter



CLK

(LSB) Q_0

Q_1

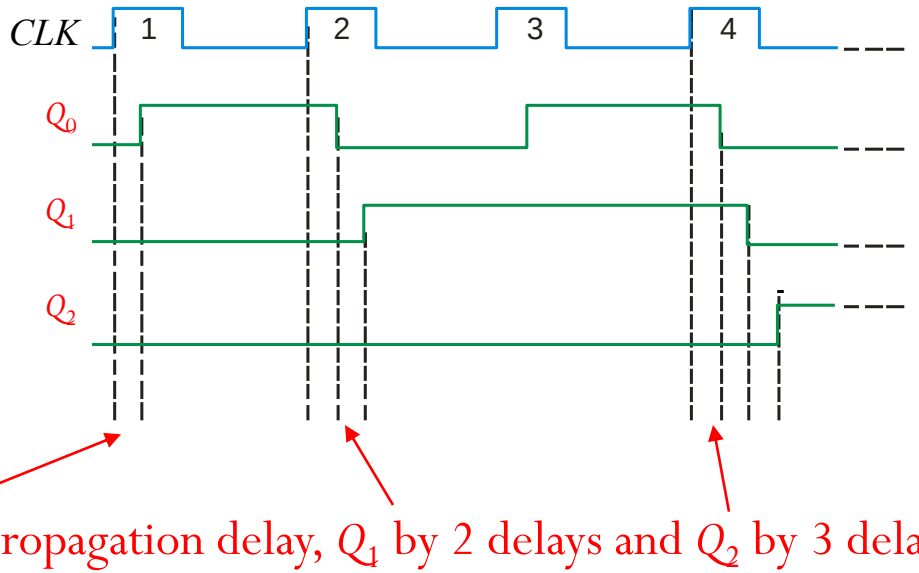
(MSB) Q_2

Recycle back to 0

Propagation Delay

- Asynchronous counters are sometimes called **ripple** counters, because the stages do not all change together.
- For certain applications requiring high clock rates, this is a major disadvantage.
- The maximum cumulative delay must be less than the period of the clock waveform.

Notice how delays are cumulative as each stage in a counter is clocked later than the previous stage.



Q_0 is delayed by 1 propagation delay, Q_1 by 2 delays and Q_2 by 3 delays.

It takes three propagation delay times for the effect of the clock pulse, CLK4, to ripple through the counter and change Q_2 from LOW to HIGH.

Modulus

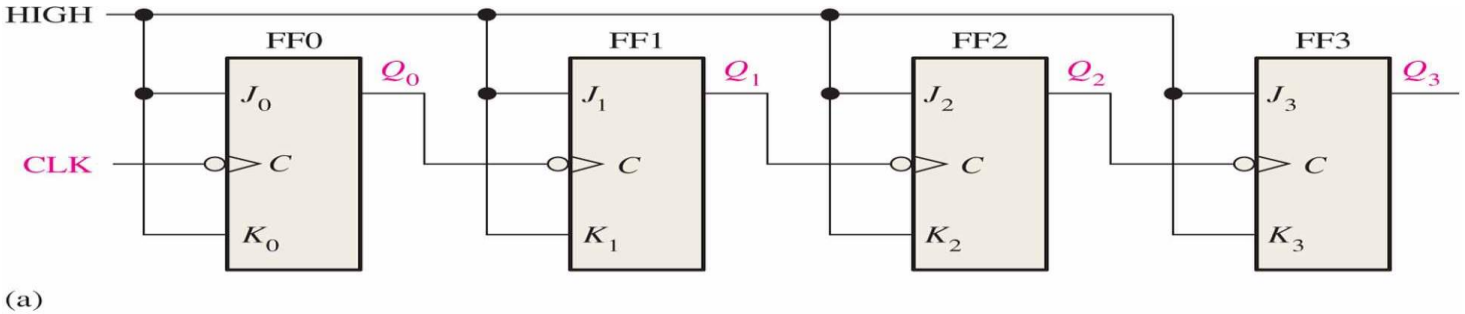
- The **modulus** of a counter is the number of unique states through which the counter will sequence.
- The maximum possible number of states (maximum modulus) of a counter is 2^n,
  - where n is the number of FFs in the counter.
- Counters can be designed to have a number of states in their sequence that is ***less*** than the maximum of 2^n.
  - This type of sequence is called a **truncated sequence**.
- To obtain a truncated sequence, it is necessary to force the counter to recycle before going through all of its possible states.

Counters with Truncated Sequences

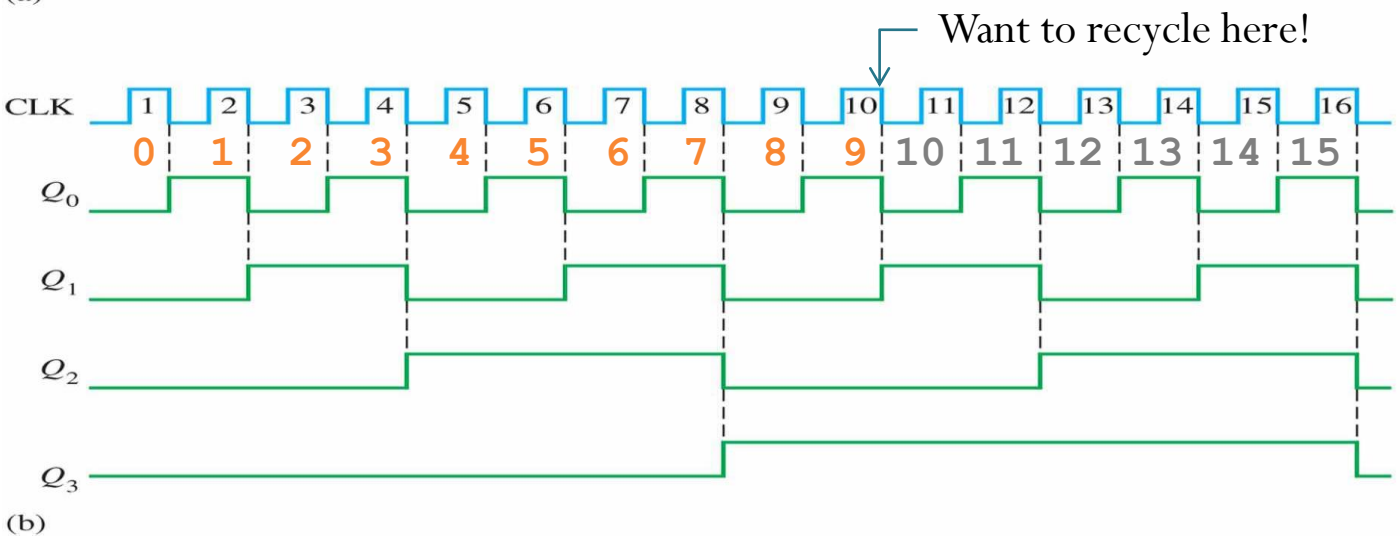
- One common modulus for counters with truncated sequences is ten (called **MOD10**).
- Counters with ten states in their sequence are called **decade counters**.
- A decade counter with a count sequence of **zero (0000)** through **nine (1001)** is a **BCD decade counter** because its ten-state sequence produces the BCD code.
- The BCD decade counter must recycle back to the 0000 state after the 1001 state.
- We will use a technique called **partial decoding**.

BCD Decade Counter (1)

- A decade counter requires four FFs.
  - Three FFs are insufficient because $2^3 = 8$.
- Start with 4-bit asynchronous binary counter

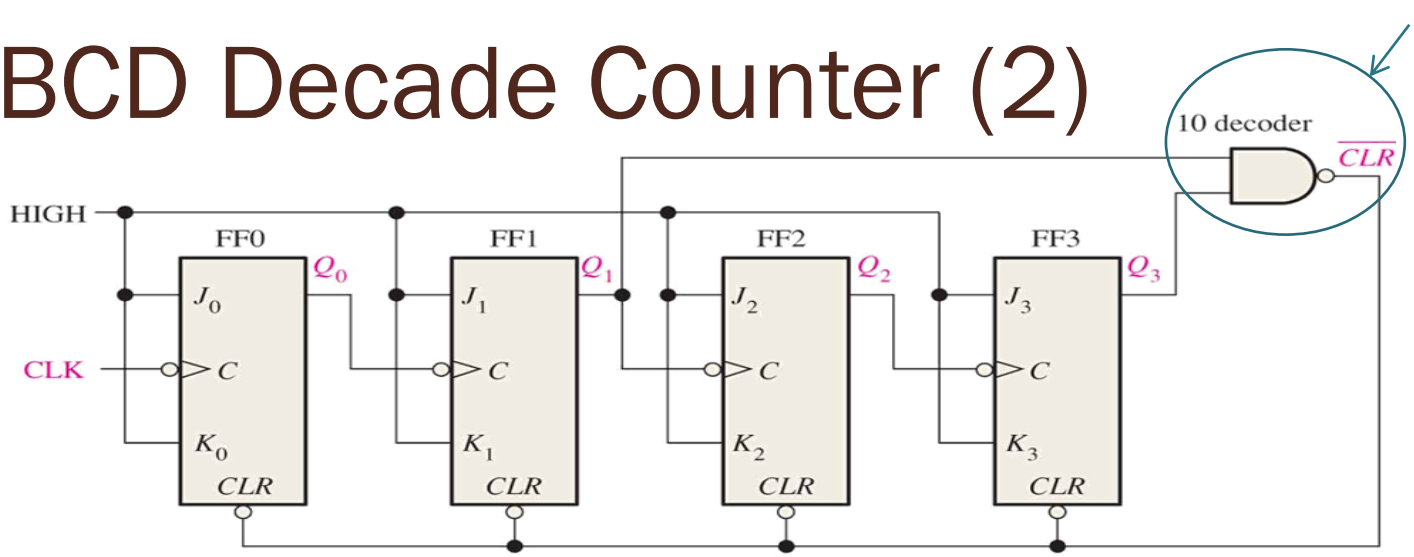


(a)

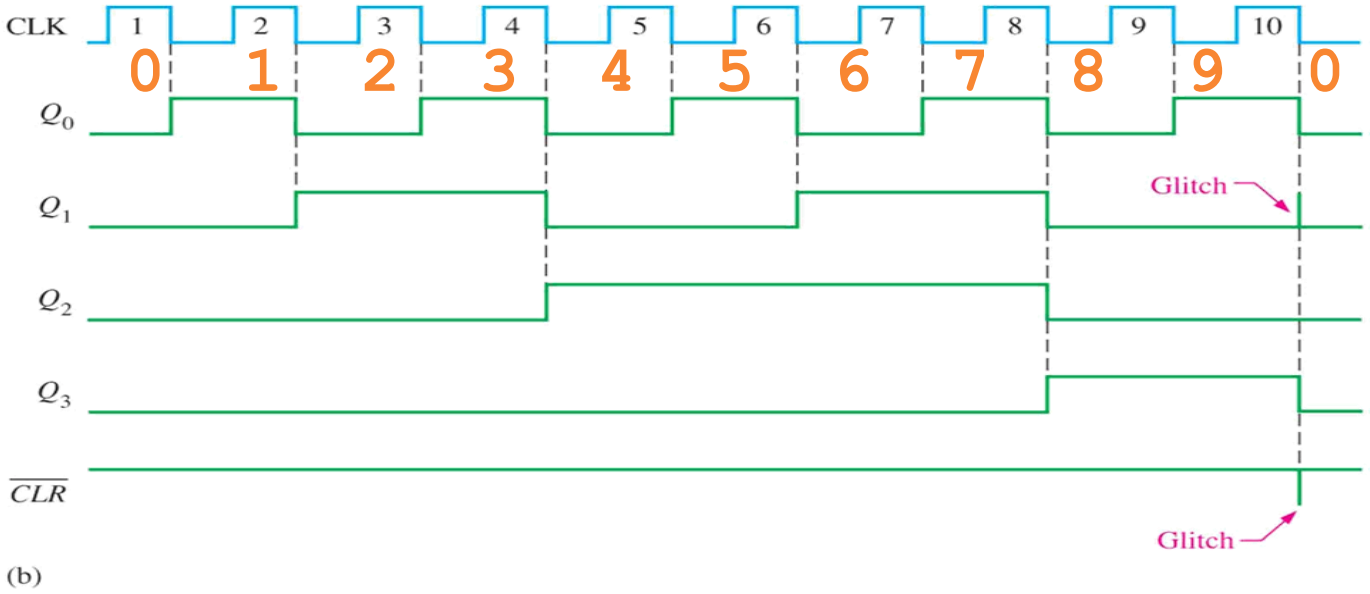


(b)

BCD Decade Counter (2)



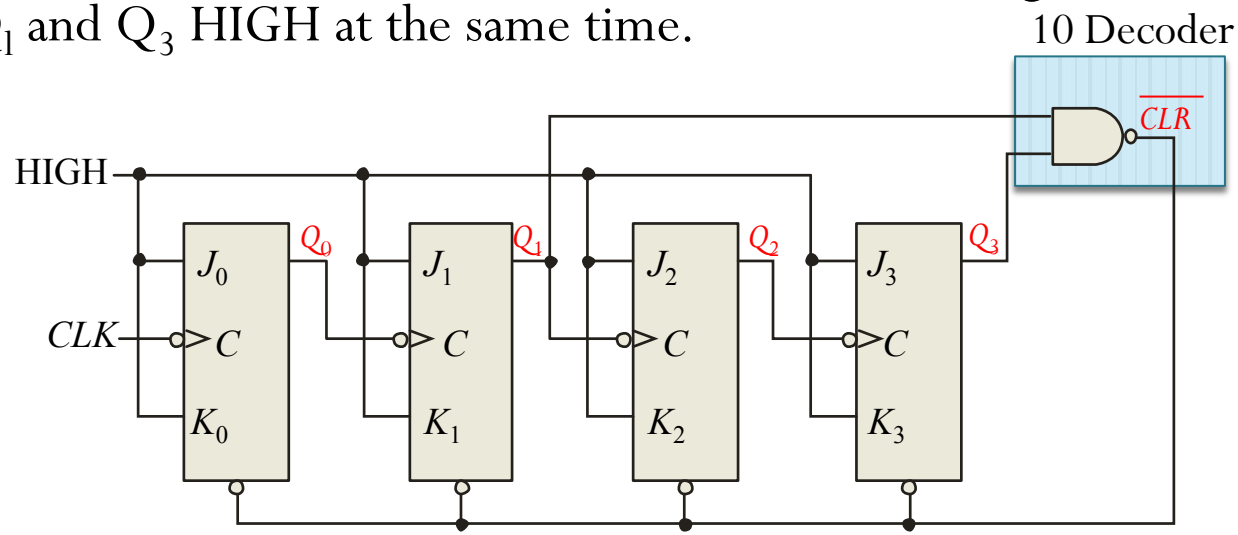
When Q_1 and Q_3 are HIGH together, the counter is cleared by a "glitch" on the *CLR* line.



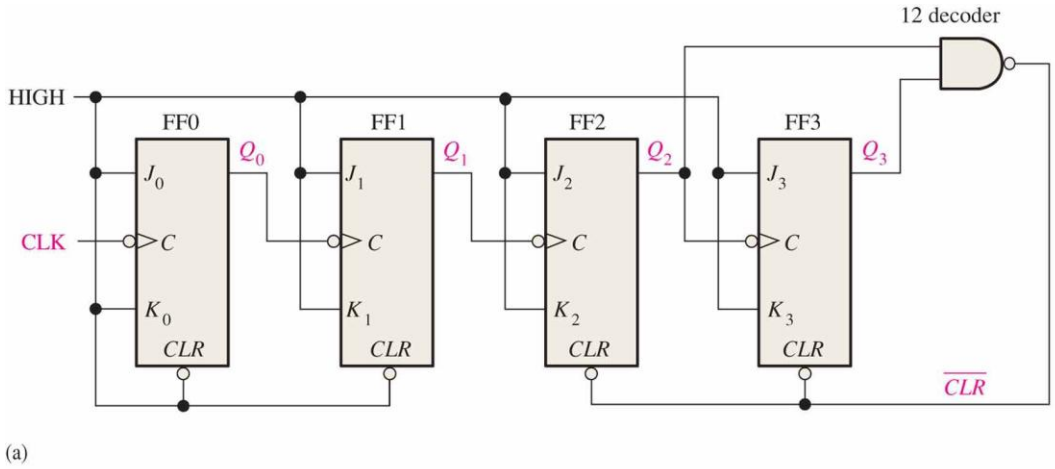
BCD Decade Counter (3)

- One way to make the counter recycle after the count of nine (1001) is to decode count ten (1010) with a NAND gate and connect the output of the NAND gate to the active-LOW CLR inputs of the FFs.
- **Partial Decoding**:
  - In our circuit, only Q_1 and Q_3 are connected to the NAND gate inputs.
  - The two unique states ($Q_1 = 1$ and $Q_3 = 1$) are sufficient to decode the count of ten because none of the other states (zero through nine) have both Q_1 and Q_3 HIGH at the same time.

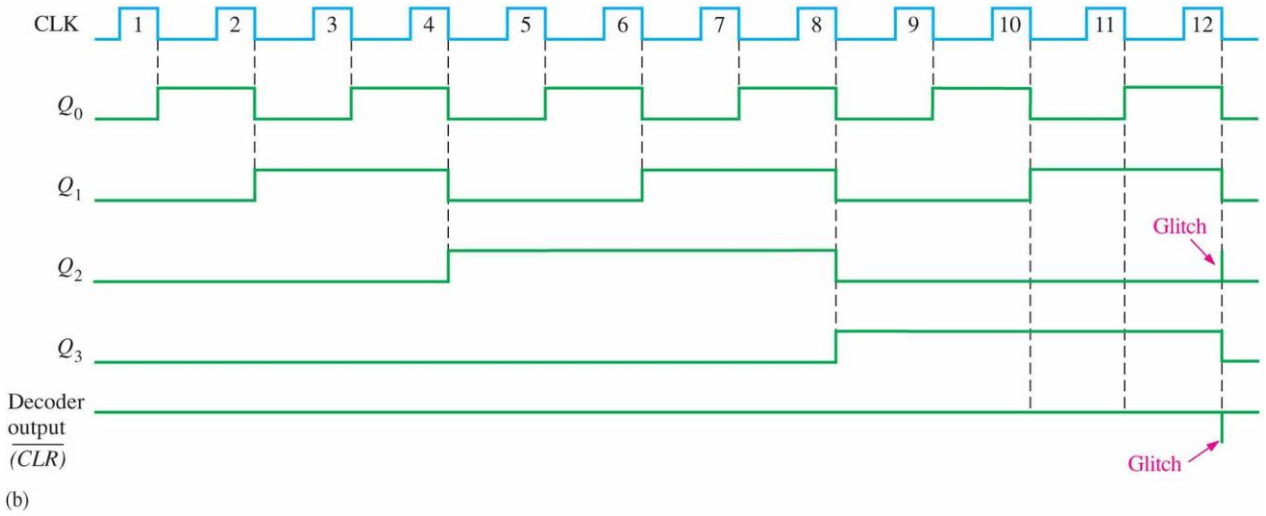
Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15



Ex: MOD12 Counter



(a)

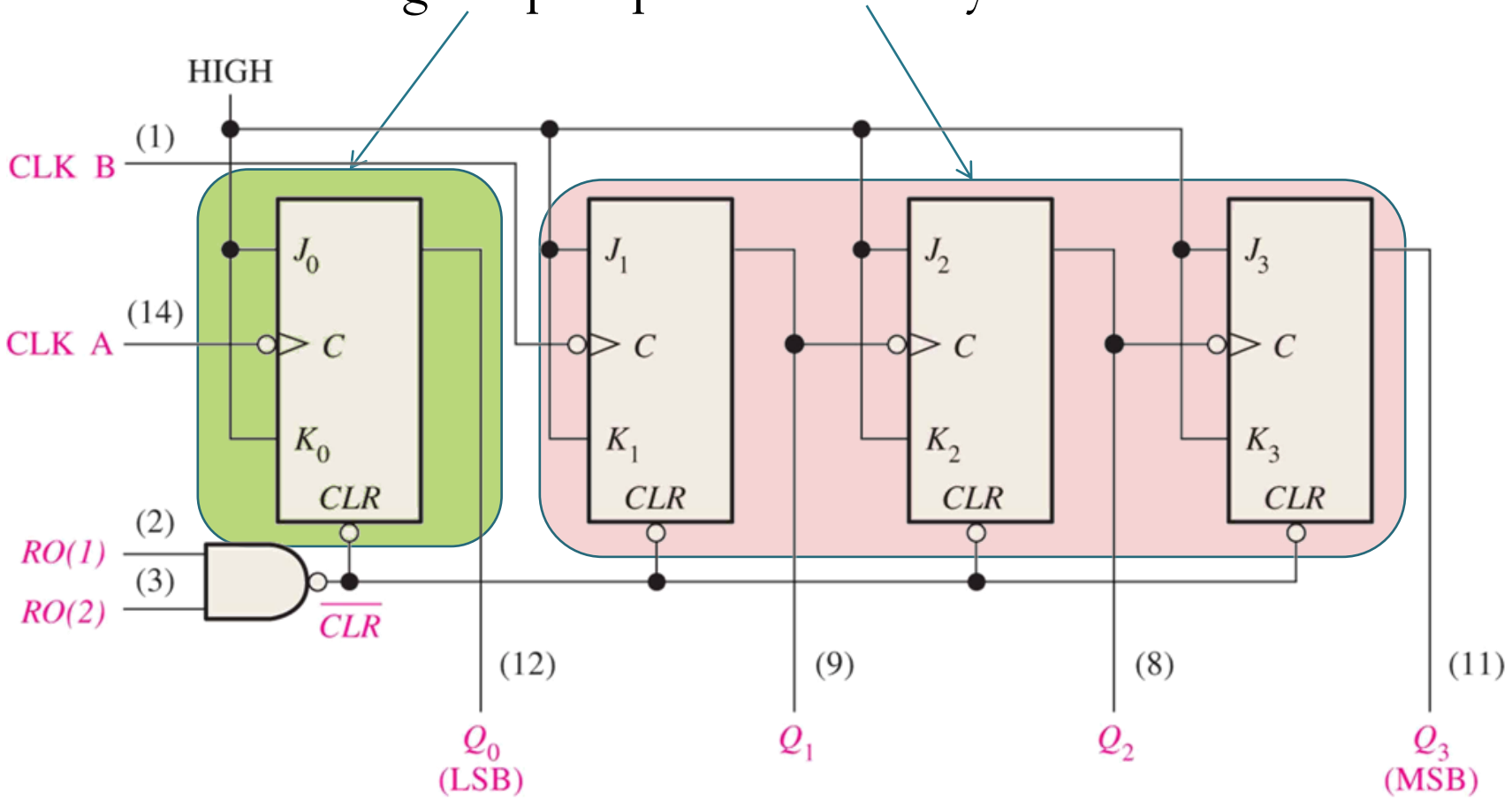


(b)

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

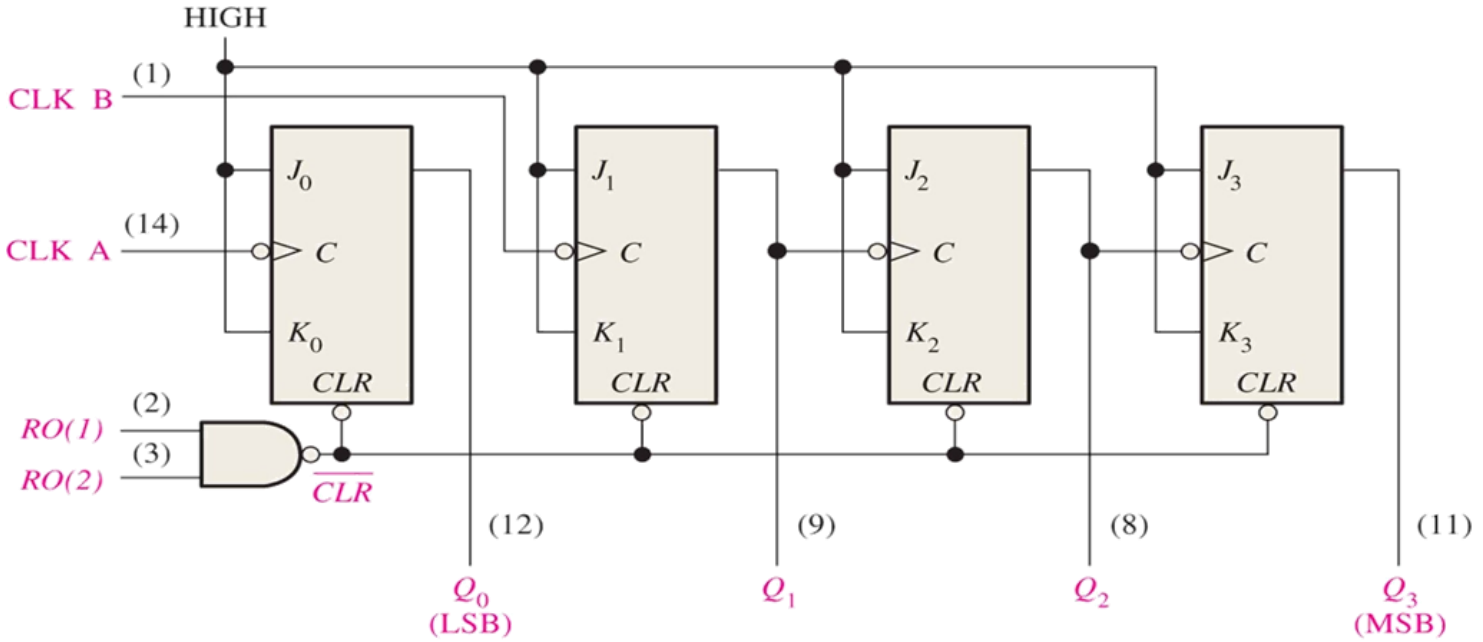
74x93

- 4-Bit Asynchronous Binary Counter
- Consist of single flip-flop and a 3-bit asynchronous counter.

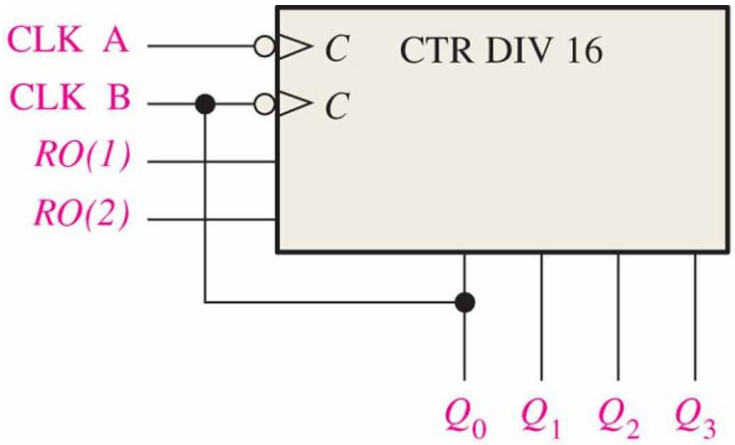


74x93

- The counter can be extended to form a 4-bit counter by connecting Q_0 to the CLK B input.
- Provide gated reset inputs, RO(1) and RO(2).
  - When both of these inputs are HIGH, the counter is reset to the 0000 state.

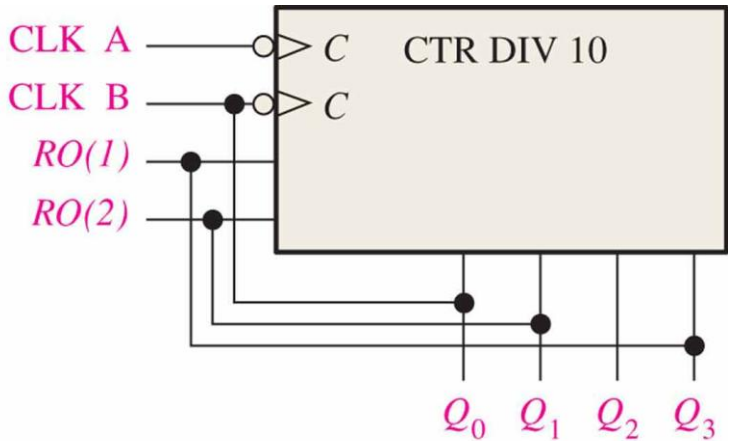


74x93 Applications



74x93 as a MOD16 counter

74x93 as a MOD10 counter

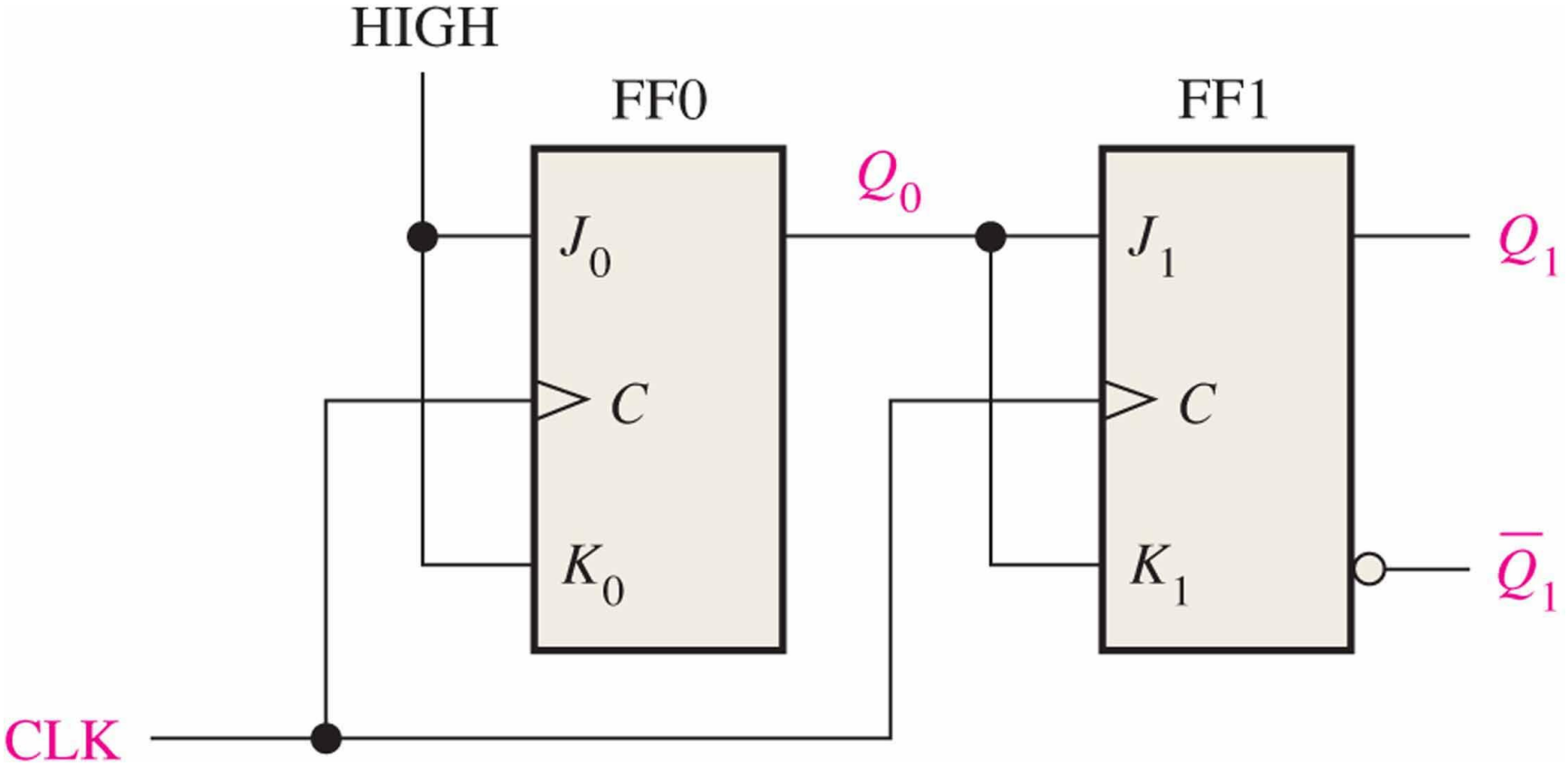


Exercise: Show how to connect a 74x93 4-bit asynchronous counter as a MOD12 counter.

Synchronous Counters

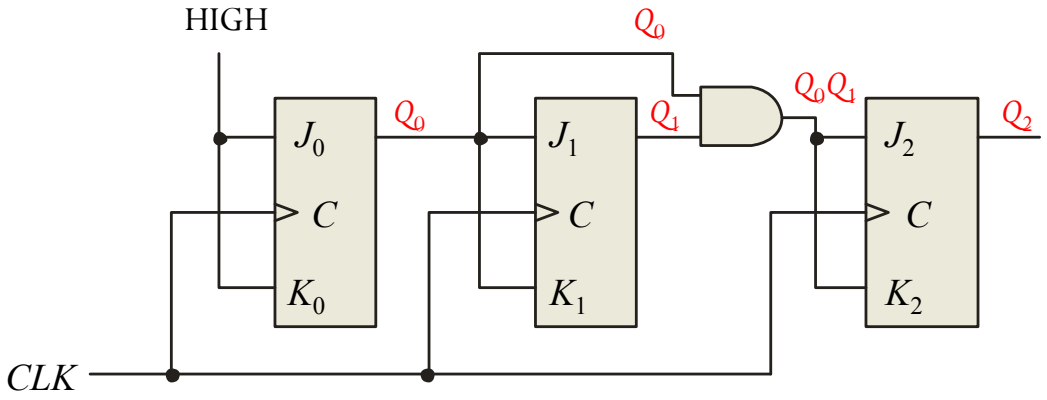
- The term **synchronous** refers to events that have a fixed time relationship with each other.
- A **synchronous counter** is one in which all the flip-flops in the counter are clocked at the same time by a **common** clock pulse.
- Synchronous counters overcome the disadvantage of accumulated propagation delays, but generally they require more circuitry to control states changes.

Ex: 2-Bit Synchronous Binary Counter



Ex: 3-bit Binary Synchronous Counter

- This 3-bit binary synchronous counter has the same count sequence as the 3-bit asynchronous counter shown previously.



Analysis of Synchronous Counters

A tabular technique for analysis is illustrated for the counter on the previous slide. Start by setting up the outputs as shown, then write the logic equation for each input. This has been done for the counter.

1. Put the counter in an arbitrary state; then determine the inputs for this state.	2. Use the new inputs to determine the next state: Q_2 and Q_1 will latch and Q_0 will toggle.	3. Set up the next group of inputs from the current output.

Outputs ← Logic for inputs →

$Q_2\,Q_1\,Q_0$	$J_2 = Q_0Q_1$	$K_2 = Q_0Q_1$	$J_1 = Q_0$	$K_1 = Q_0$	$J_0 = 1$	$K_0 = 1$
0 0 0	0	0	0	0	1	1
0 0 1	0	0	1	1	1	1
0 1 0						

4. Q_2 will latch again but both Q_1 and Q_0 will toggle.

Continue like this, to complete the table. The next slide shows the completed table…

Analysis of Synchronous Counters

Outputs ← Logic for inputs →

$Q_2\ Q_1\ Q_0$	$J_2 = Q_0Q_1$	$K_2 = Q_0Q_1$	$J_1 = Q_0$	$K_1 = Q_0$	$J_0 = 1$	$K_0 = 1$
0 0 0	0	0	0	0	1	1
0 0 1	0	0	1	1	1	1
0 1 0	0	0	0	0	1	1
0 1 1	1	1	1	1	1	1
1 0 0	0	0	0	0	1	1
1 0 1	0	0	1	1	1	1
1 1 0	0	0	0	0	1	1
1 1 1	1	1	1	1	1	1
0 0 0						

At this points all states have been accounted for and the counter is ready to recycle…