
# DIGITAL LOGIC

# 数字逻辑

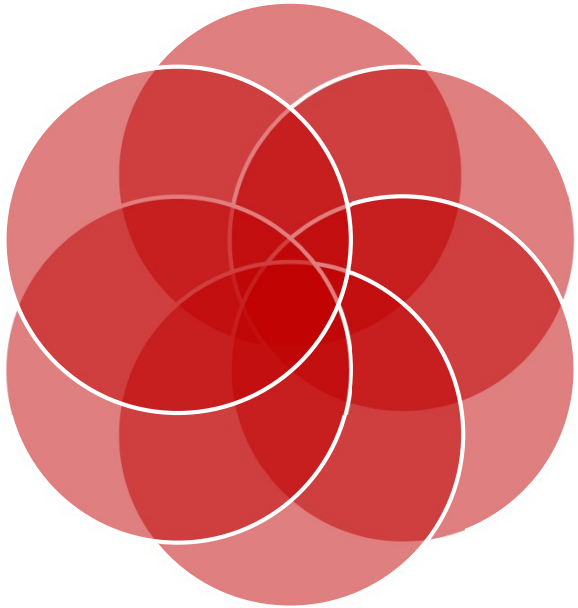
01

# Course Content

# Course Content

Ripple Counter

BCD Counter, Arbitrary sequence Counters



Synchronous Binary Counters

Binary Counter with Parallel Load

Serial Vs. Parallel Counters

Up-down Binary Counter

# Counters



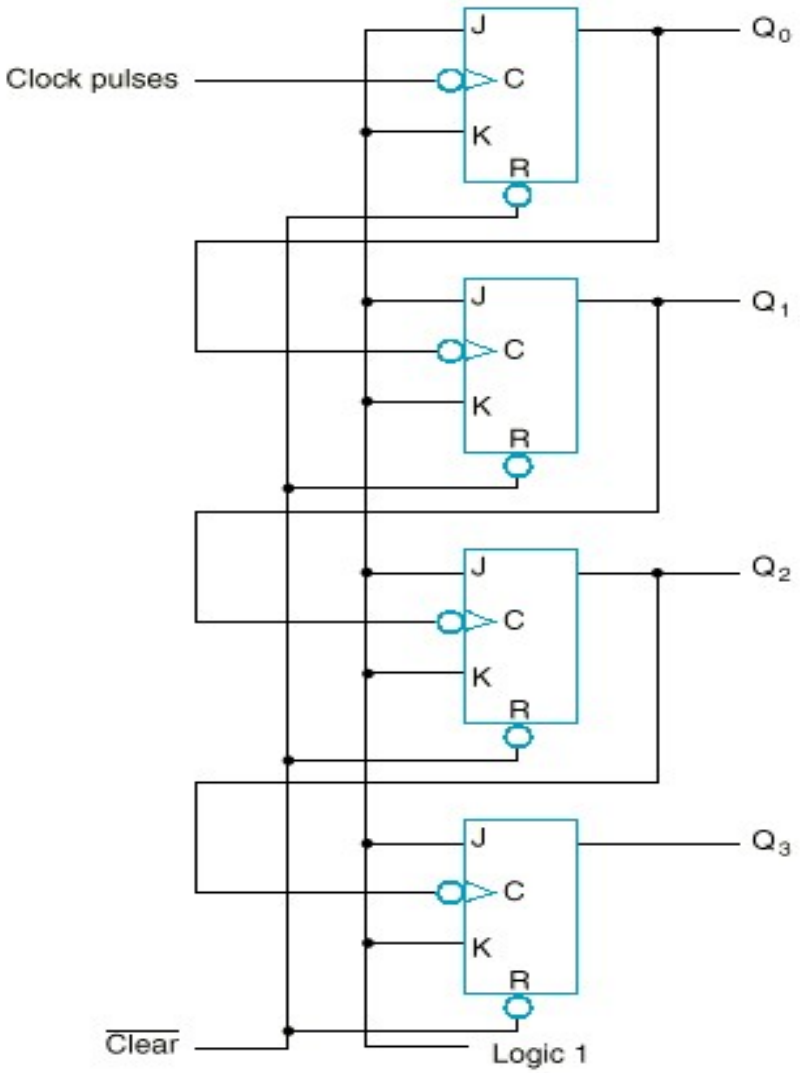
- A *counter* is a register that goes through a predetermined sequence of states upon the application of clock pulses.
- Counters are categorized as:
  - Ripple Counters:
    The FF output transition serves as a source for triggering other FFs. No common clock.
  - Synchronous Counter:
    All FFs receive the common clock pulse, and the change of state is determined from the present state.

# Example: A 4-bit Upward Counting Ripple Counter



**Less Significant Bit output is Clock for Next Significant Bit!**
**(Clock is active low)**

| (a) *JK* Flip-Flop | | | |
|---|---|---|---|
| **J** | **K** | ***Q*** **(*t* 1 1)** | **Operation** |
| 0 | 0 | $Q(t)$ | No change |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | $\overline{Q}(t)$ | Complement |



# Example (cont.)



- The output of each FF is connected to the C input of the next FF in sequence.
- The FF holding the least significant bit receives the incoming clock pulses.
- The J and K inputs of all FFs are connected to a permanent logic 1.
- The bubble next to the C label indicates that the FFs respond to the negative-going transition of the input.

# Example (cont.)

Operation:

- The least significant bit ($Q_0$) is complemented with each negative-edge clock pulse input.
- Every time that $Q_0$ goes from 1 to 0, $Q_1$ is complemented.
- Every time that $Q_1$ goes from 1 to 0, $Q_2$ is complemented.
- Every time that $Q_2$ goes from 1 to 0, $Q_3$ is complemented, and so on.

**Upward Counting Sequence**

| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 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 |

# Synchronous Binary Counters



- The design procedure for a binary counter is the same as any other synchronous sequential circuit.
- The primary inputs of the circuit are the CLK and any control signals (EN, Load, etc).
- The primary outputs are the FF outputs (present state).
- Most efficient implementations usually use T-FFs or JK-FFs. We will examine JK and D flip-flop designs.

# Synchronous Binary Counters

SICHUAN UNIVERSITY

## J-K Flip Flop Design of a 4-bit Binary Up Counter

| Present state | | | | Next state | | | |
|---|---|---|---|---|---|---|---|
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 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 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

**J–K flip-flop characteristic table and equation**

| Present J | Present K | Present $Q_t$ | Next $Q_{t+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$Q_{t+1} = JQ_t' + K'Q_t$

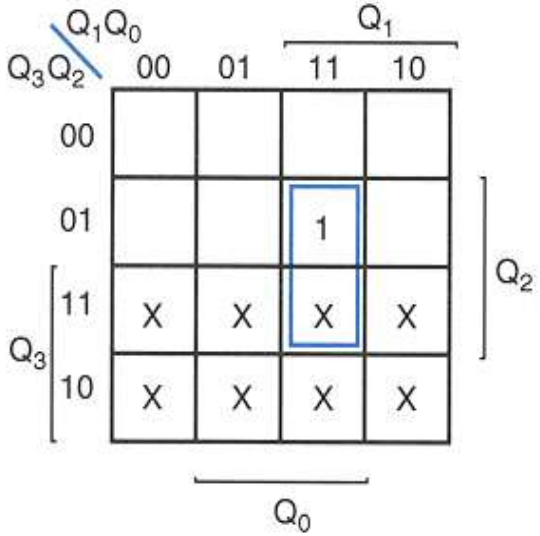
**J-K flip-flop excitation table**

| $Q_t$ | $Q_{t+1}$ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | d |
| 0 | 1 | 1 | d |
| 1 | 0 | d | 1 |
| 1 | 1 | d | 0 |

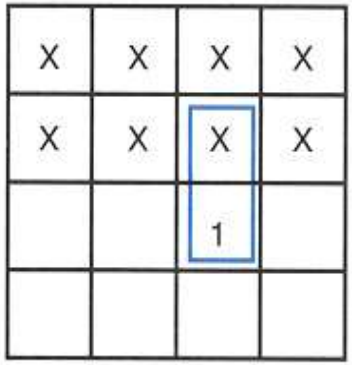
# Synchronous Binary Counters

## J-K Flip Flop Design of a Binary Up Counter (cont.)

| Present state | | | | Next state | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_{Q3}$ | $K_{Q3}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | X |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | X |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | X |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | X |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | X |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | X |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | X | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | X | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | X | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | X | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | X | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | X | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | X | 1 |




$J_{Q3} = Q_0Q_1Q_2$

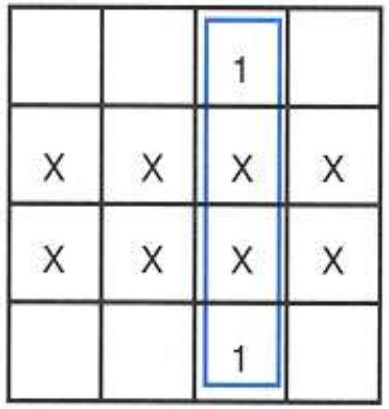



$K_{Q3} = Q_0Q_1Q_2$

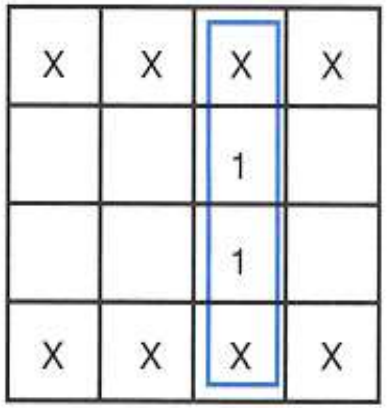
# Synchronous Binary Counters

## J-K Flip Flop Design of a Binary Up Counter (cont.)

| Present state | | | | Next state | | | | Flip-flo | |
|---|---|---|---|---|---|---|---|---|---|
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_{Q2}$ | $K_{Q2}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | × |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | × |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | × |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | × |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | × | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | × | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | × | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | × | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | × |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | × |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | × |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | × |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | × | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | × | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | × | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | × | 1 |




$J_{Q2} = Q_0Q_1$

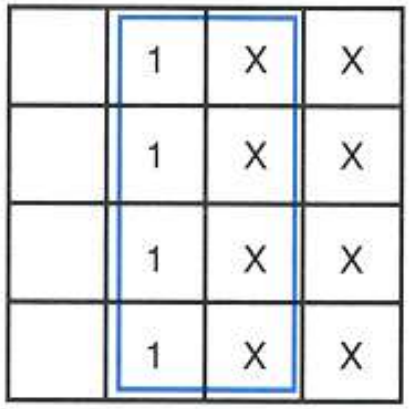



$K_{Q2} = Q_0Q_1$

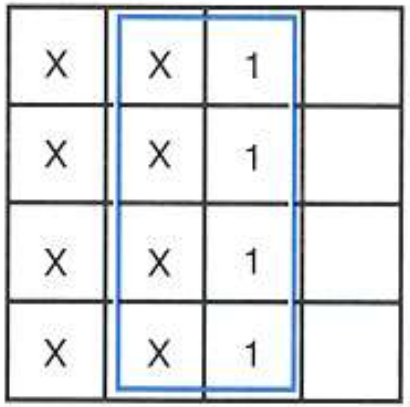
# Synchronous Binary Counters

## J-K Flip Flop Design of a Binary Up Counter (cont.)

| Present state | | | | Next state | | | | p inputs | |
|---|---|---|---|---|---|---|---|---|---|
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_{Q1}$ | $K_{Q1}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | X |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | X | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | X | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | X |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | X | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | X | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | X |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | X |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | X | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | X | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | X |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | X |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | X | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | X | 1 |



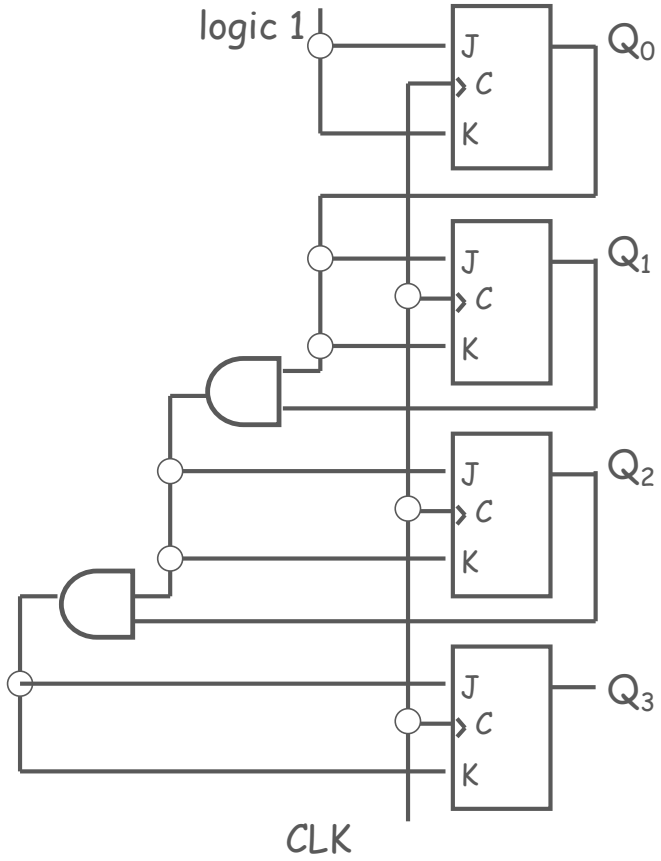

$J_{Q1} = Q_0$




$K_{Q1} = Q_0$

# Synchronous Binary Counters

## J-K Flip Flop Design of a Binary Up Counter (cont.)




$J_{Q0} = 1$
$K_{Q0} = 1$

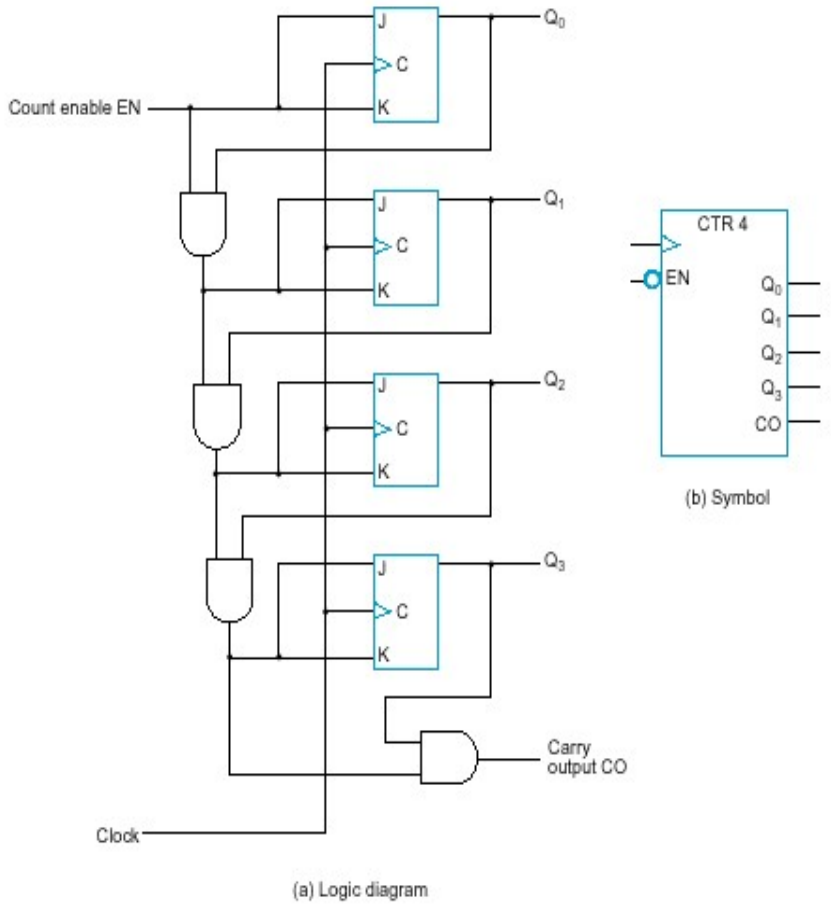
$J_{Q1} = Q_0$
$K_{Q1} = Q_0$

$J_{Q2} = Q_0 Q_1$
$K_{Q2} = Q_0 Q_1$

$J_{Q3} = Q_0 Q_1 Q_2$
$K_{Q3} = Q_0 Q_1 Q_2$

# Synchronous Binary Counters

## J-K Flip Flop Design of a Binary Up Counter with EN and CO




EN = enable control signal, when 0 counter remains in the same state, when 1 it counts

*CO* = carry output signal, used to extend the counter to more stages

$$J_{Q0} = 1 \cdot EN$$
$$K_{Q0} = 1 \cdot EN$$
$$J_{Q1} = Q_0 \cdot EN$$
$$K_{Q1} = Q_0 \cdot EN$$
$$J_{Q2} = Q_0 Q_1 \cdot EN$$
$$K_{Q2} = Q_0 Q_1 \cdot EN$$
$$J_{Q3} = Q_0 Q_1 Q_2 \cdot EN$$
$$K_{Q3} = Q_0 Q_1 Q_2 \cdot EN$$
$$CO = Q_0 Q_1 Q_2 Q_3 \cdot EN$$

# Synchronous Binary Counters using D flip-flops

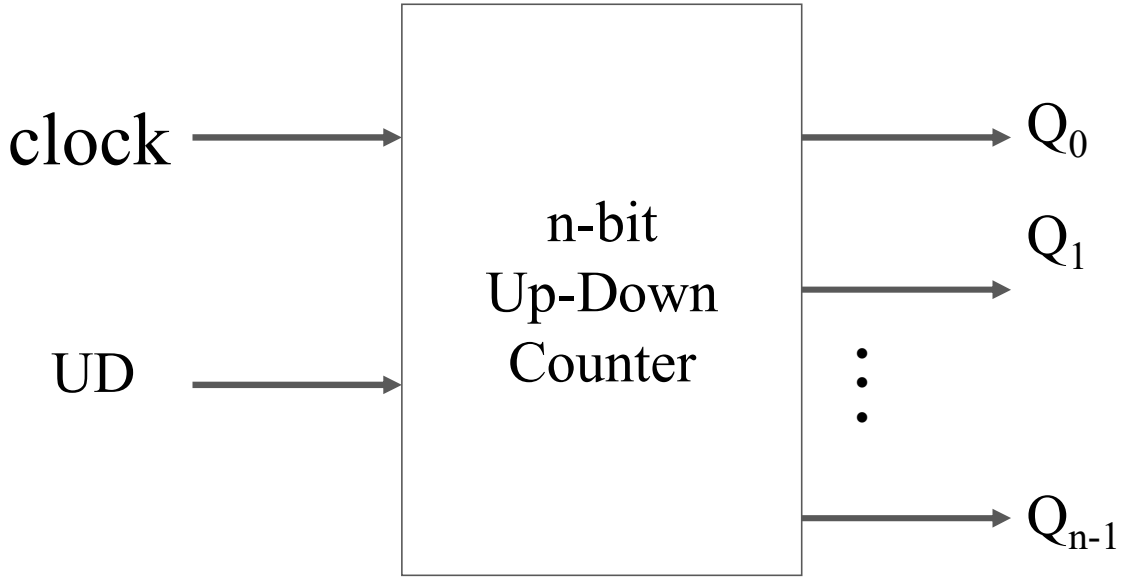
- $D_{Q0} = Q_0 \otimes EN$
- $D_{Q1} = Q_1 \otimes ( Q_0 \cdot EN)$
- $D_{Q2} = Q_2 \otimes ( Q_0 Q_1 \cdot EN )$
- $D_{Q3} = Q_3 \otimes ( Q_0 Q_1 Q_2 \cdot EN )$
- $C0 = Q_0 Q_1 Q_2 Q_3 \cdot EN$

JK-FF equations

JK-based design calls for 4 AND gates

D-based design calls for 4 AND and 4 XOR gates

# Up-Down Binary Counter




UD = 0: count up
UD = 1: count down

# Up-Down Binary Counter (cont.)

| UD | Q2 | Q1 | Q0 | Q2.D | Q1.D | Q0.D | UD | Q2 | Q1 | Q0 | Q2.D | Q1.D | Q0.D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Up-Counter

Down-Counter

Finish the design in class

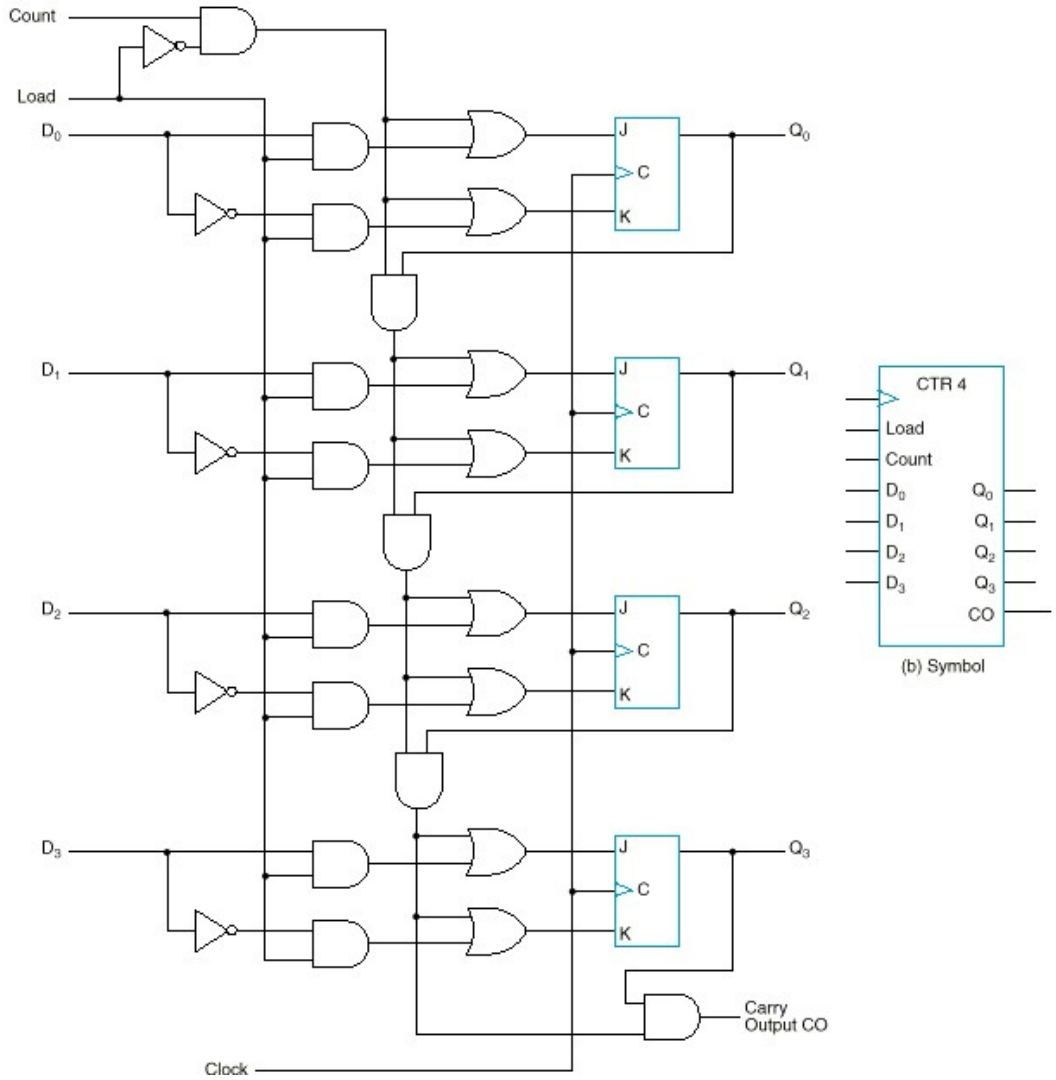
# Binary Counter with Parallel Load



- (Next slide) gives the logic diagram and symbol of a 4-bit synchronous binary counter with parallel load capability. The function table for this binary counter is

| Load | Count | Operation |
|---|---|---|
| 0 | 0 | Nothing |
| 0 | 1 | Count |
| 1 | x | Load |

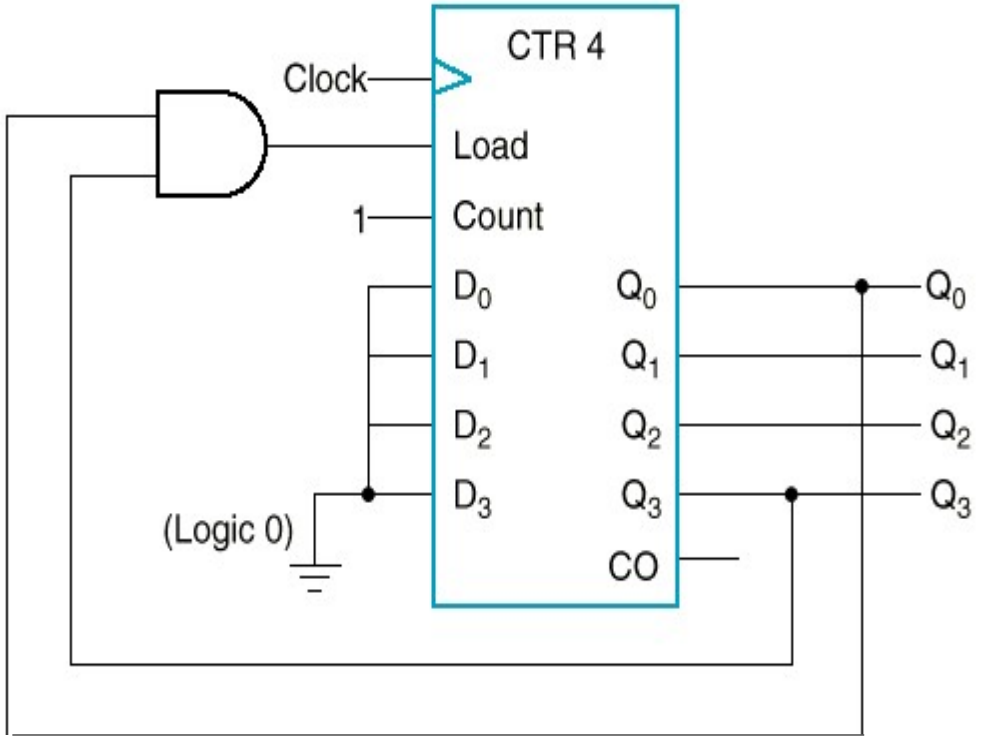
# Binary Counter with Parallel Load



# BCD counter

- The binary counter with parallel load can be converted into a synchronous BCD counter by connecting an external AND gate to it.

| Load | Count | Operation |
|---|---|---|
| 0 | 0 | Nothing |
| 0 | 1 | Count |
| 1 | x | Load |



# BCD counter（cont.）



- The counter starts with an all-zero output.
- As long as the output of the AND gate is 0, each positive clock pulse transition increments the counter by one.
- When the output reaches the count of 1001, both $Q_0$ and $Q_3$ become 1, making the output of the AND gate equal to 1. This condition makes Load active, so on the next clock transition, the counter does not count, but is loaded from its four inputs.
- The value loaded then is 0000.

# Arbitrary Sequence Counter



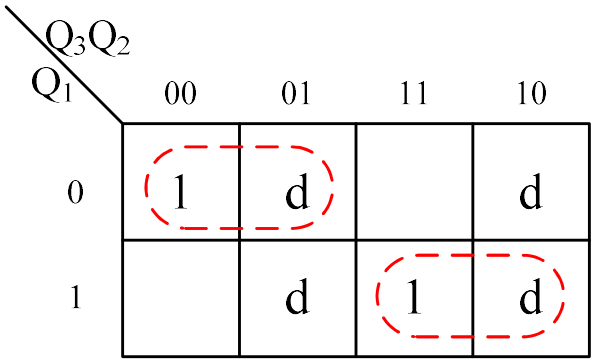
- Given an arbitrary sequence, design a counter that will generate this sequence.
- Procedure:
  - Derive state table/diagram based on give sequence
  - Simplify (using K-maps, etc)
  - Draw logic diagram
- Example: Use D-FFs to draw the logic diagram for sequence generator (counter) for: 0 → 7 → 6 → 1 → 0 (000 → 111 → 110 → 001 → 000)

# Example

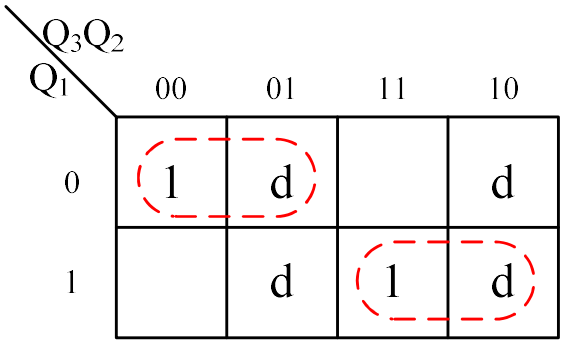
- Use D-FFs to draw the logic diagram for sequence generator (counter) for: 0 → 7 → 6 → 1 → 0 (000 → 111 → 110 → 001 → 000)

| $Q_3Q_2Q_1$ | $Q_3{+}Q_2{+}Q_1{+}$ | $D_3D_2D_1$ | Z |
|---|---|---|---|
| 000 | 111 | 111 | 0 |
| 001 | 000 | 000 | 1 |
| 010 | ddd | ddd | d |
| 011 | ddd | ddd | d |
| 100 | ddd | ddd | d |
| 101 | ddd | ddd | d |
| 110 | 001 | 001 | 0 |
| 111 | 110 | 110 | 0 |

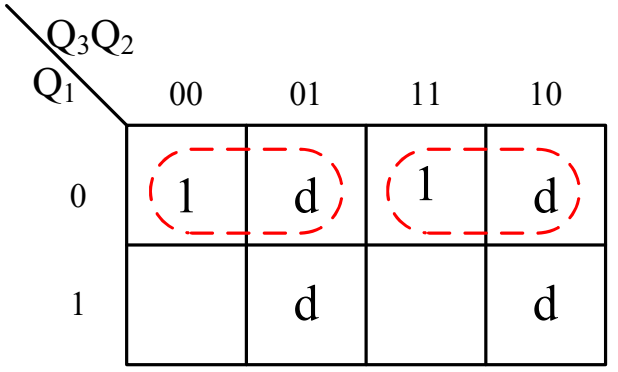
# Example



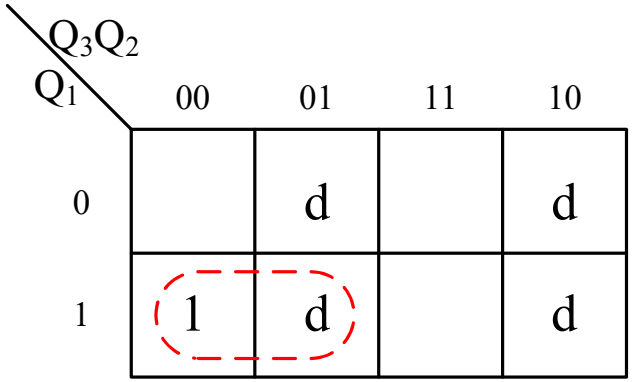

$D_3$ K-Map




$D_2$ K-Map




$D_1$ K-Map




Z K-Map

$$D_3 = Q_3'Q_1' + Q_3Q_1$$

$$D_2 = Q_3'Q_1' + Q_3Q_1$$

$$D_1 = Q_1'$$

$$Z = Q_3'Q_1$$

| $Q_3Q_2Q_1$ | $Q_3+Q_2+Q_1+$ | $D_3D_2D_1$ | Z |
|---|---|---|---|
| 000 | 111 | 111 | 0 |
| 001 | 000 | 000 | 1 |
| 010 | ddd | ddd | d |
| 011 | ddd | ddd | d |
| 100 | ddd | ddd | d |
| 101 | ddd | ddd | d |
| 110 | 001 | 001 | 0 |
| 111 | 110 | 110 | 0 |

# Example

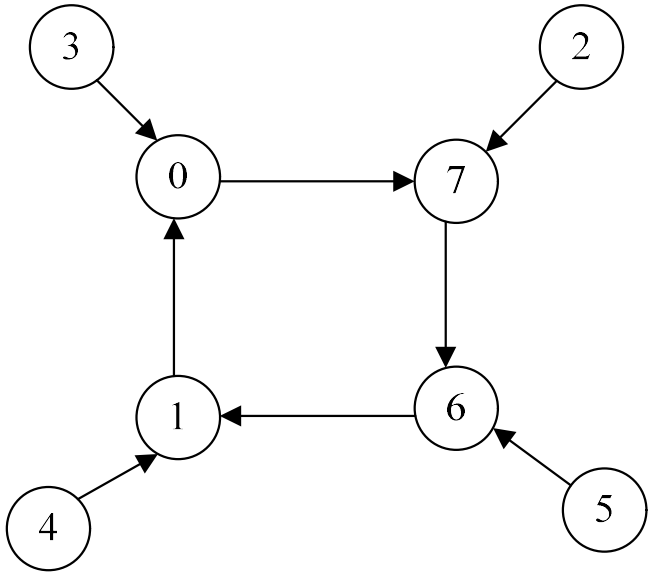
$D_3=Q_3'Q_1'+Q_3Q_1$

$D_2=Q_3'Q_1'+Q_3Q_1$

$D_1=Q_1'$

$Z= Q_3'Q_1$

| $Q_3Q_2Q_1$ | $Q_3+Q_2+Q_1+$ |
|---|---|
| 010 | 111 |
| 011 | 000 |
| 100 | 001 |
| 101 | 110 |



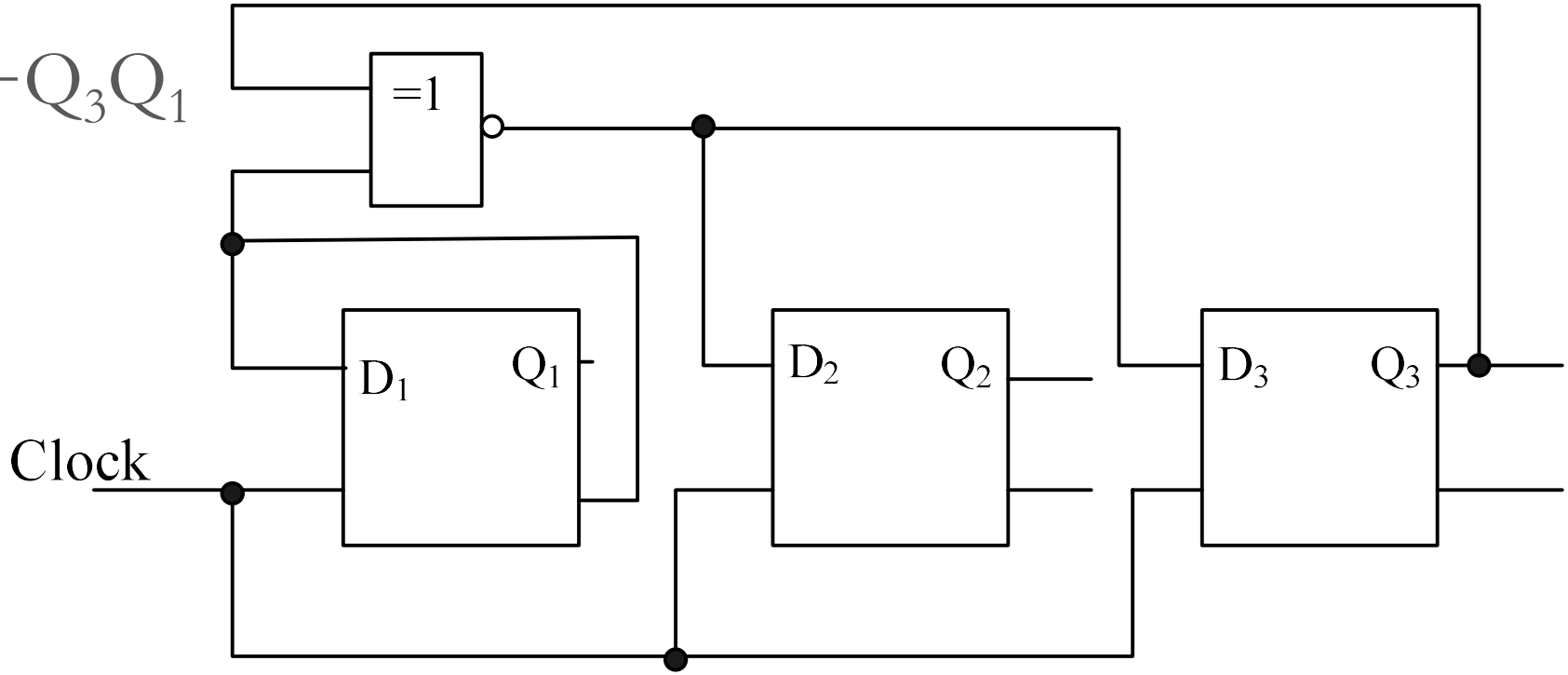
# Example

$D_3 = Q_3'Q_1' + Q_3Q_1$

$D_2 = Q_3'Q_1' + Q_3Q_1$

$D_1 = Q_1'$

$Z = Q_3'Q_1$



# Homework

# Thanks