# Logic Design

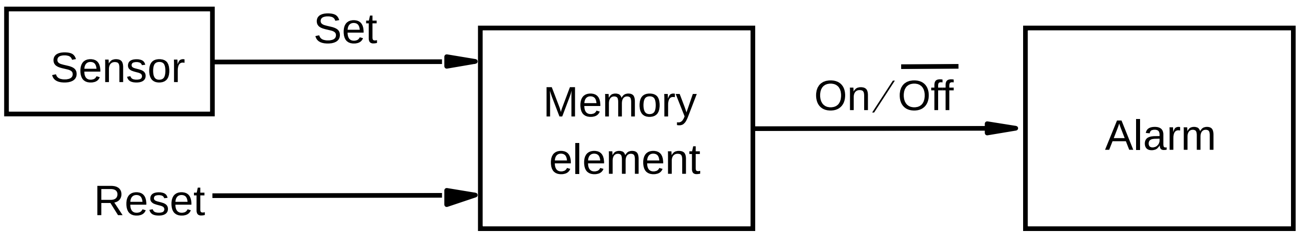
## Flip Flops, Registers and Counters

McMaster University

# Introduction

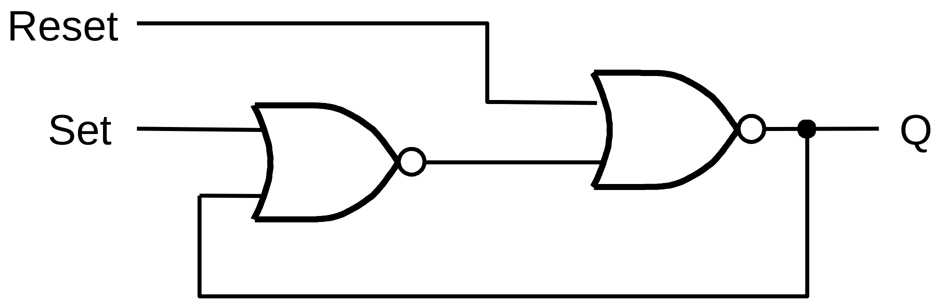
- Combinational circuits: value of each output depends only on the values of inputs
- Sequential Circuits: values of outputs depend on inputs and past behavior of the circuit
  - Circuit contains storage (memory) elements

- Example: an alarm system in which the alarm stays on when triggered even if the sensor output goes to zero

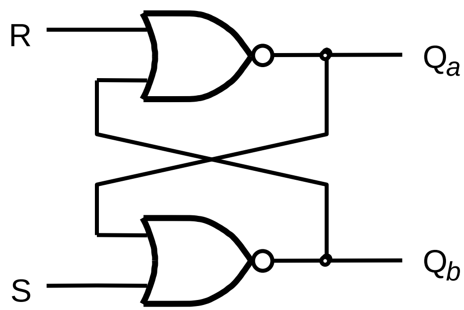


# Basic Latch

- Simplest memory element: basic latch
- Can be built with NAND or NOR gates



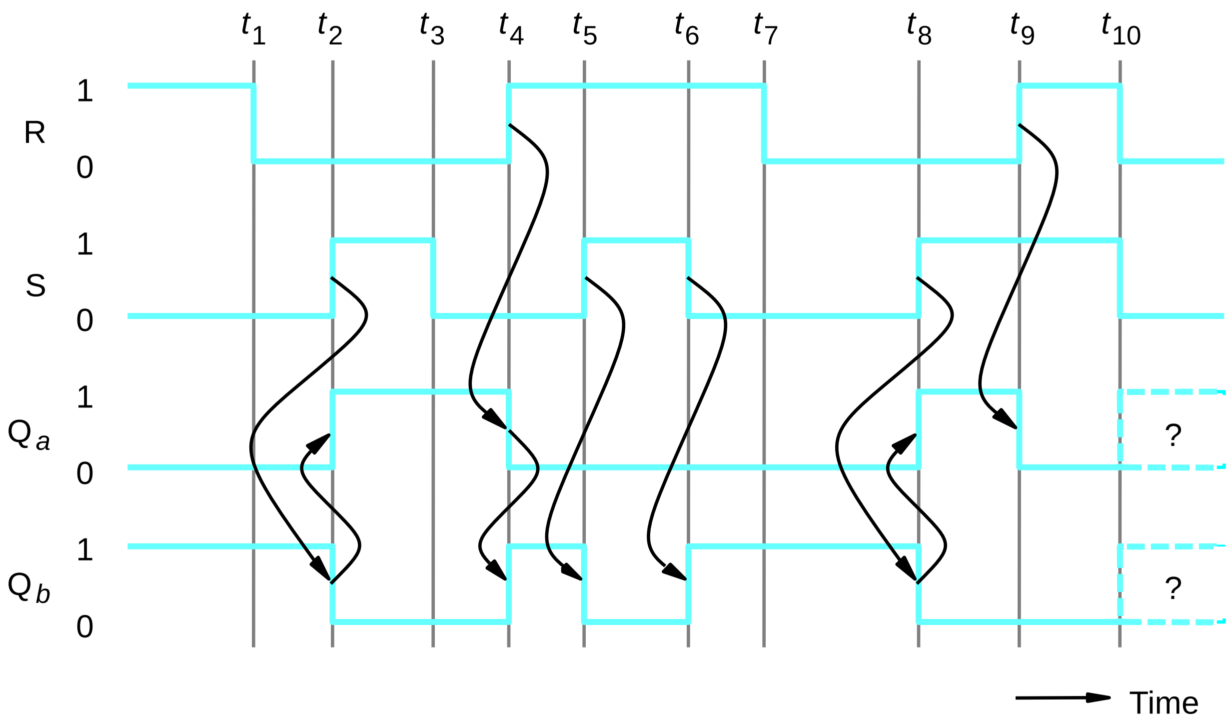
# Basic Latch




(a) Circuit

| S | R | $Q_a$ | $Q_b$ | |
|---|---|---|---|---|
| 0 | 0 | 0/1 | 1/0 | (no change) |
| 0 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |

(b) Truth table

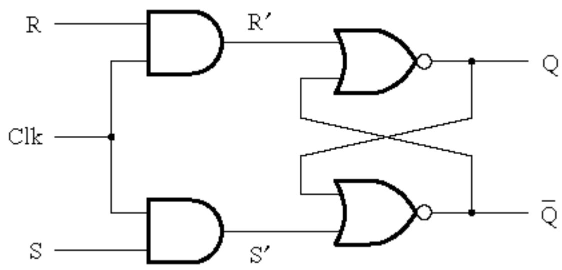



(c) Timing diagram

# Basic Latch

- In basic latch, the state changes when the inputs change
- In many circuits we cannot control when the inputs change but would like the change in state happens at particular times
- We add a clock (clk) signal to the basic latch

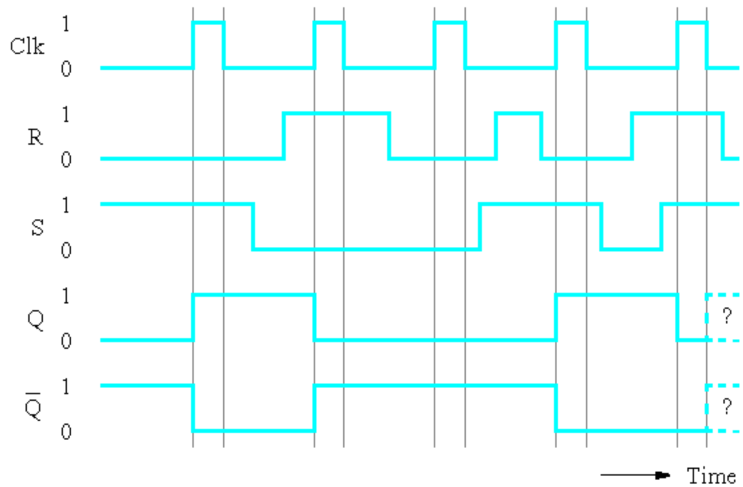
# Gated SR Latch



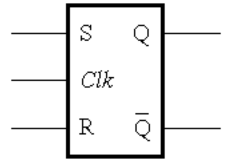

(a) Circuit

| Clk | S | R | Q(t + 1) |
|---|---|---|---|
| 0 | x | x | Q(t) (no change) |
| 1 | 0 | 0 | Q(t) (no change) |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | x |

(b) Characteristic table



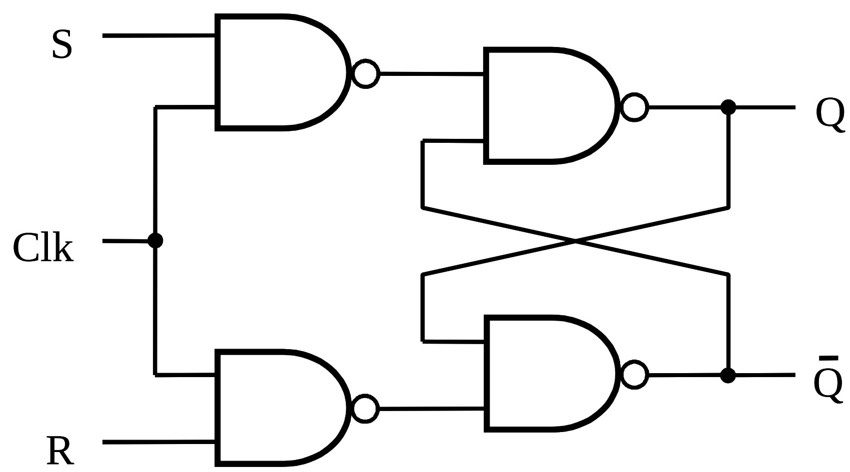

(c) Timing diagram




(d) Graphical symbol

# Gated Latch with NAND

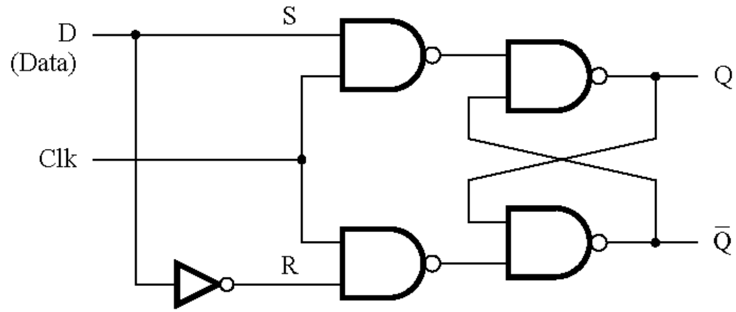
- Behavior of the circuit is the same as the one with NOR
- Clock is gated by NAND gates rather than AND gates
- S and R inputs are reversed



# D latch

- D latch is based on gated SR latch
- Instead of two inputs, has one input

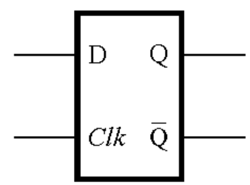
# D Latch



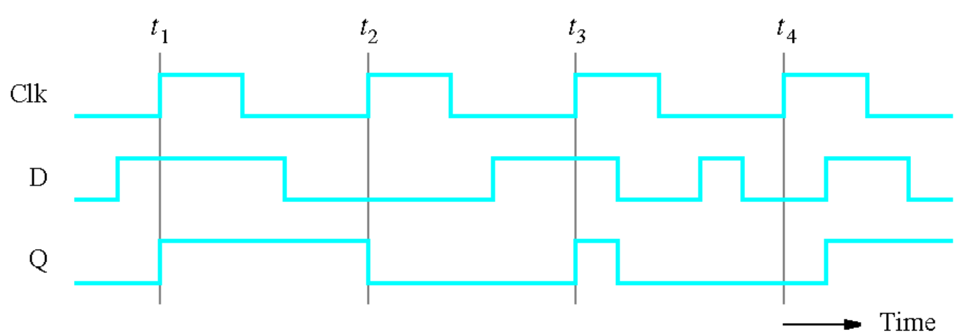

(a) Circuit

| Clk | D | $Q(t+1)$ |
|---|---|---|
| 0 | x | $Q(t)$ |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(b) Characteristic table




(c) Graphical symbol



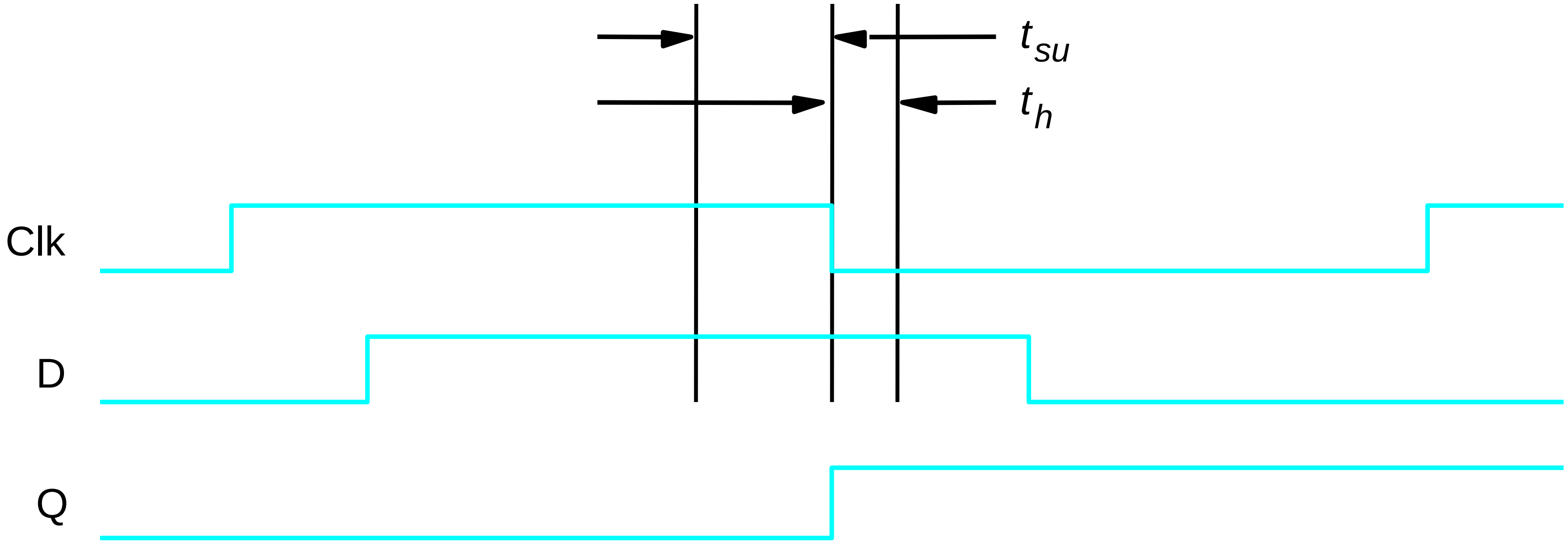

(d) Timing diagram

# D Latch

- Since the output of gated D latch is controlled by the level of clock, it is called level sensitive
  - In the window of clk=1, the output Q tracks the changes of input D. This is undesirable.
- It is possible to design storage elements for which the output changes only when clock changes from one value to the other.
- Those circuits are called edge triggered

# Propagation delay

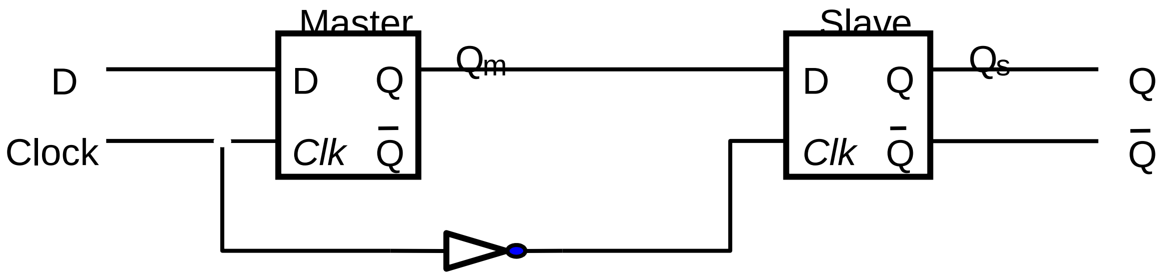
- D latch: stores the value of D input at the time clock goes from 1 to 0.
- It operates properly if input is stable (not changing) at the time clk goes from 1 to 0.



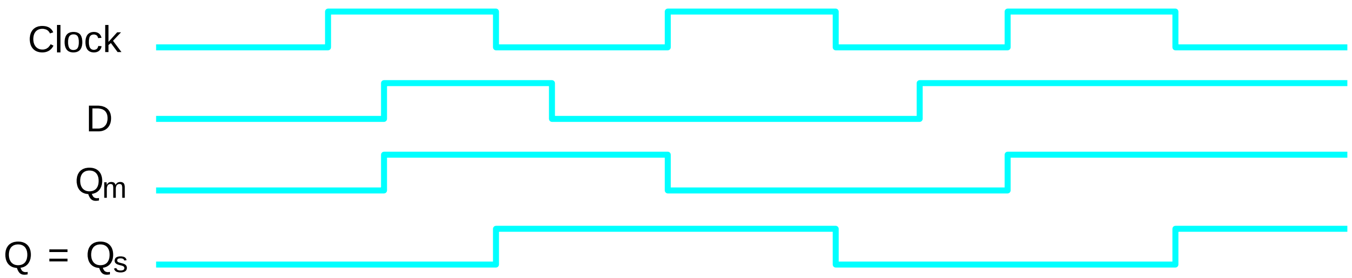
# Master-slave D flip-flop

- Master-slave D flip-flop: two gated D latches
- First one, called master, changes its state when clk=1
- Second one, called slave, changes its state when clk=0
- From external point of view, master-slave flip-flop changes its state at the negative edge of clock

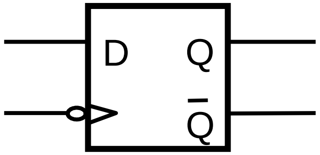
# Master-slave D flip-flop




(a) Circuit




(b) Timing diagram




(c) Graphical symbol

# Edge-triggered D flip-flop

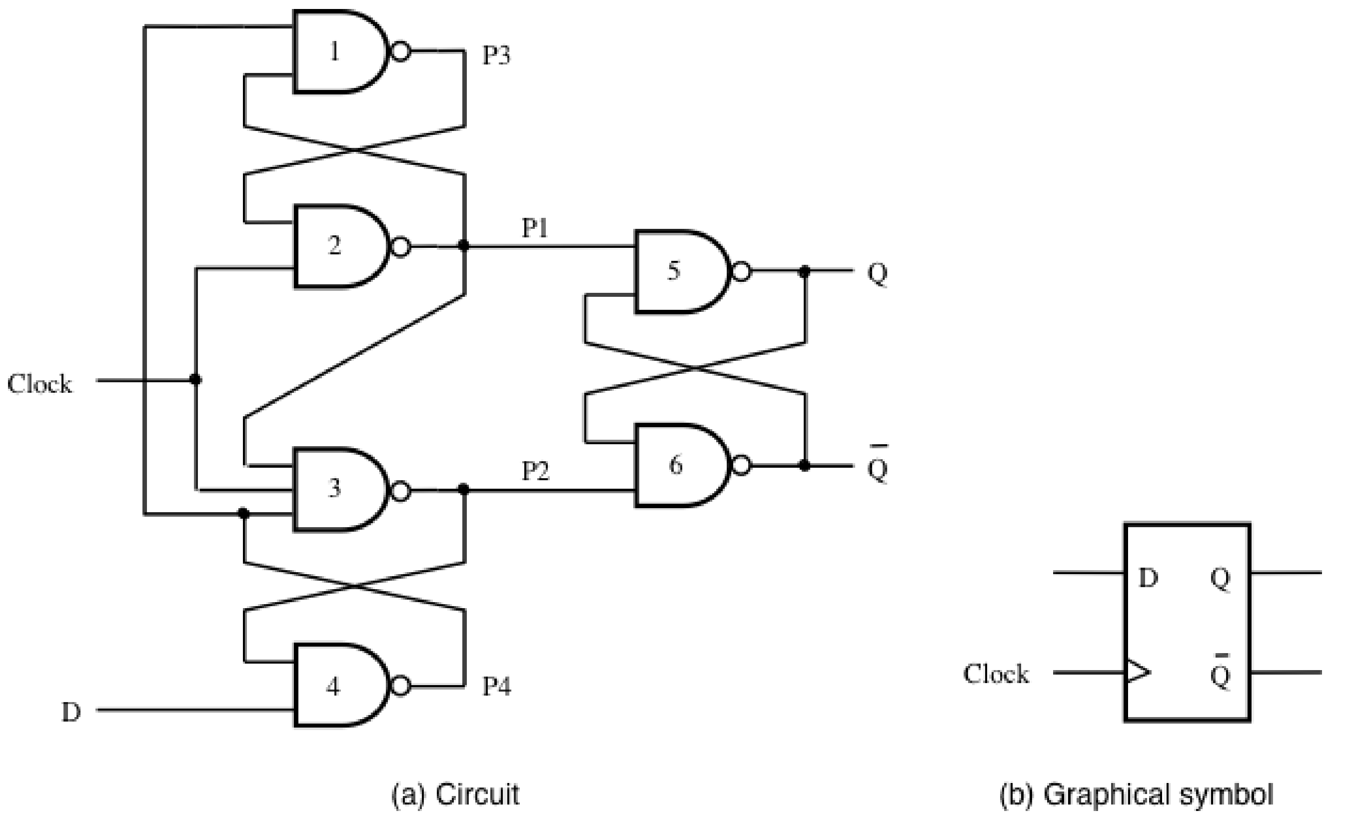
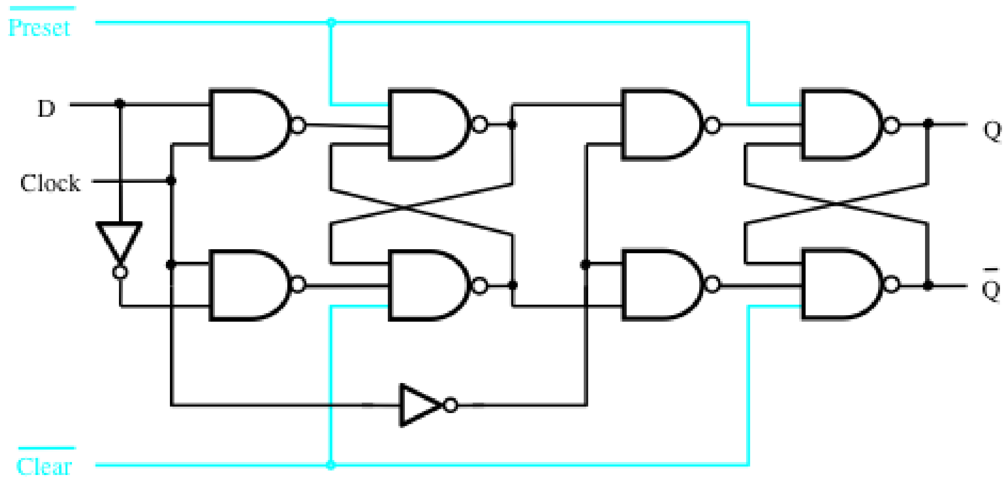


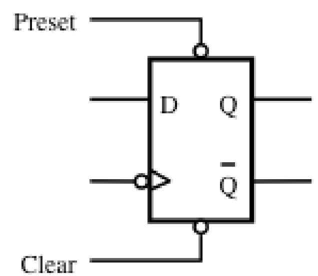

Figure 7.11. A positive-edge-triggered D flip-flop.

# D flip-flop with clear and preset

- An example of application of flip-flops: counters
- We should be able to clear the counter to zero
- We should be able to force the counter to a known initial count
- Clear: asynchronous, synchronous
- Asynchronous clear: flip-flops are cleared without regard to clock signal
- Synchronous clear: flip-flops are clear with the clock signal



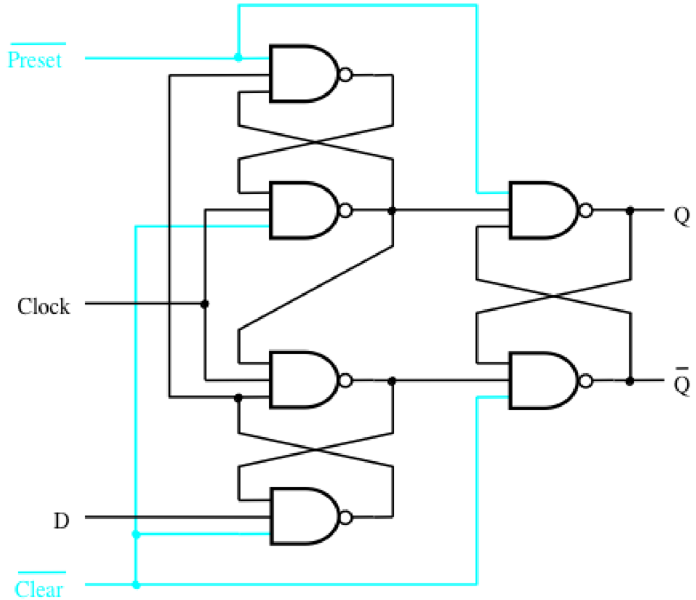

(a) Circuit



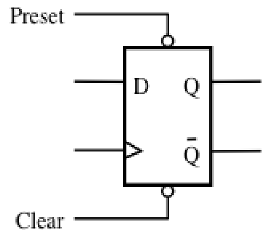

(b) Graphical symbol

Figure 7.13. Master-slave D flip-flop with *Clear* and *Preset*.

# Edge triggered D flip flop with clear & preset



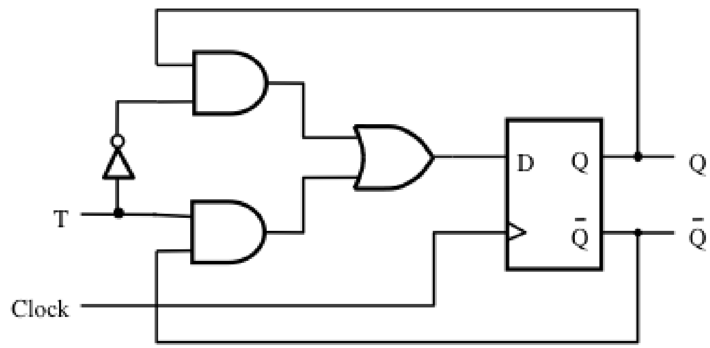

(a) Circuit




(b) Graphical symbol

Figure 7.14. Positive-edge-triggered D flip-flop with *Clear* and *Preset*.

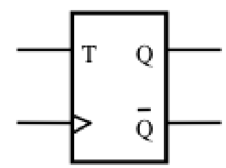
# T Flip-Flop



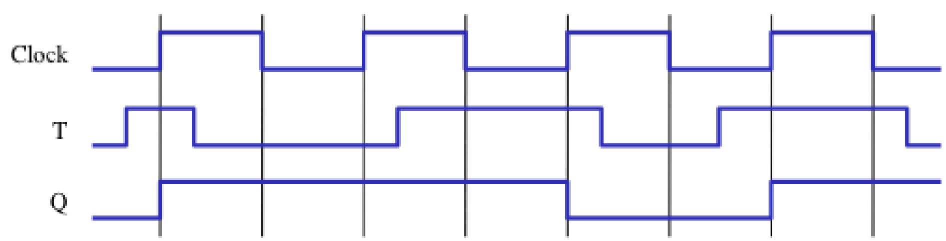

(a) Circuit

| T | Q(t + 1) |
|---|---|
| 0 | Q(t) |
| 1 | $\overline{Q}(t)$ |

(b) Truth table




(c) Graphical symbol

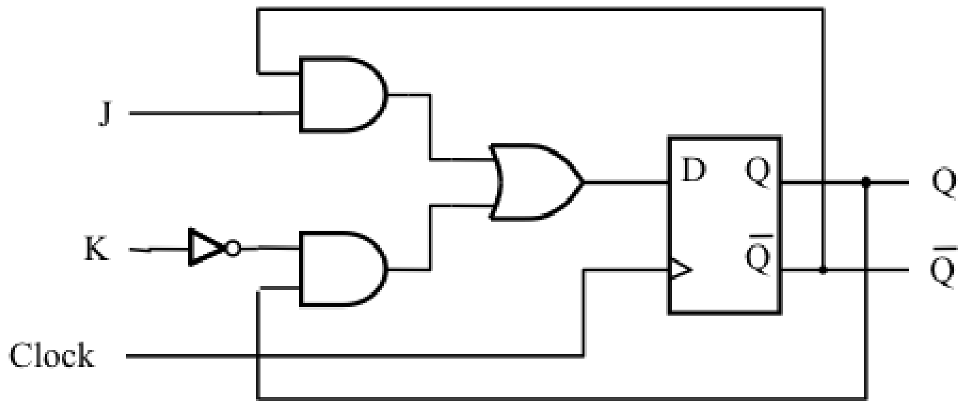



(d) Timing diagram

Figure 7.16. T flip-flop.

# JK flip flop

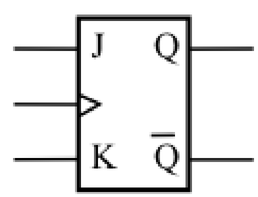
- D=JQ'+K'Q
- When J=S and K=R it will behave like a SR flip-flop




(a) Circuit

| J | K | Q (t+1) |
|---|---|---|
| 0 | 0 | Q (t) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}$ (t) |

(b) Truth table




(c) Graphical symbol

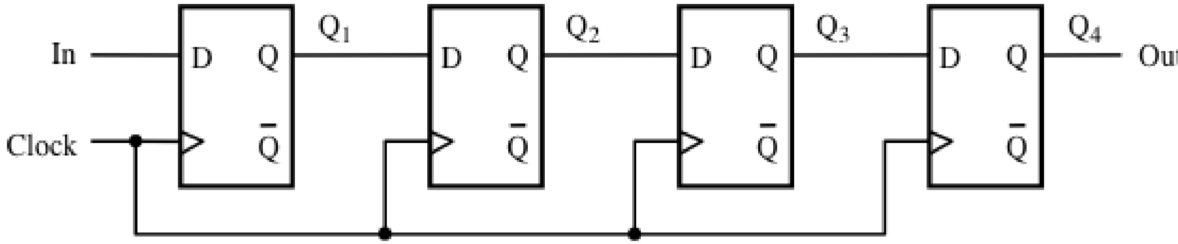
# Summary

- Basic latch- a feedback connection of two NOR or NAND gates to store 1-bit information. S »1; R »0.
- Gated latch- a basic latch with a control (clk). clk =0: the existing state maintains; clk=1: the existing state may change
  - Gated SR latch. S »1; R »0.
  - Gated D latch. The D input forces the state to be the same as D
- Flip-flop- a storage element. Its output state changes only on the edge of clk.
  - Edge-triggered flip-flop
  - Master-slave flip-flop. The master is active in $1^{st}$ half of a clock cycle; The slave active in $2^{nd}$ half.
  - Regardless how many times the D input to the master changes, the slave output can only change at the negative edge of clk.

# Registers

- Register: a set of n flip-flops used to store n bits of information
- A common clock is used for all the flip-flops
- A register that provides the ability to shift its contents is called a shift register
- To implement a shift register, it is necessary to use edge-triggered or master-slave flip-flops

# Shift register




(a) Circuit

| | In | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ = Out |
|---|---|---|---|---|---|
| $t_0$ | 1 | 0 | 0 | 0 | 0 |
| $t_1$ | 0 | 1 | 0 | 0 | 0 |
| $t_2$ | 1 | 0 | 1 | 0 | 0 |
| $t_3$ | 1 | 1 | 0 | 1 | 0 |
| $t_4$ | 1 | 1 | 1 | 0 | 1 |
| $t_5$ | 0 | 1 | 1 | 1 | 0 |
| $t_6$ | 0 | 0 | 1 | 1 | 1 |
| $t_7$ | 0 | 0 | 0 | 1 | 1 |

(b) A sample sequence

Figure 7.18. A simple shift register.

- In computer systems it is often necessary to transfer n-bit data
- Using n separate wires: parallel transmission
- Using a single wire and performing the transfer one bit at a time in n consecutive cycles: serial transmission

# Parallel access shift register

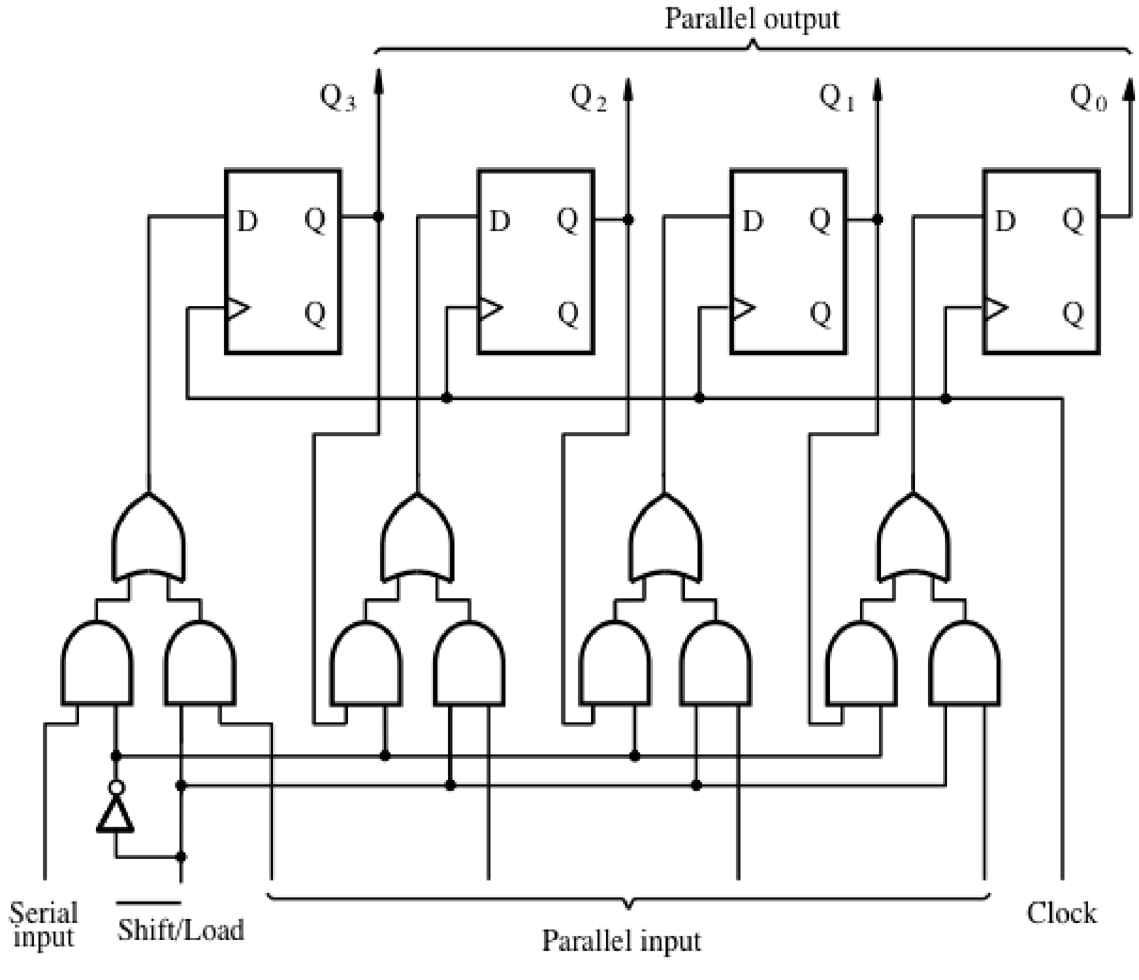
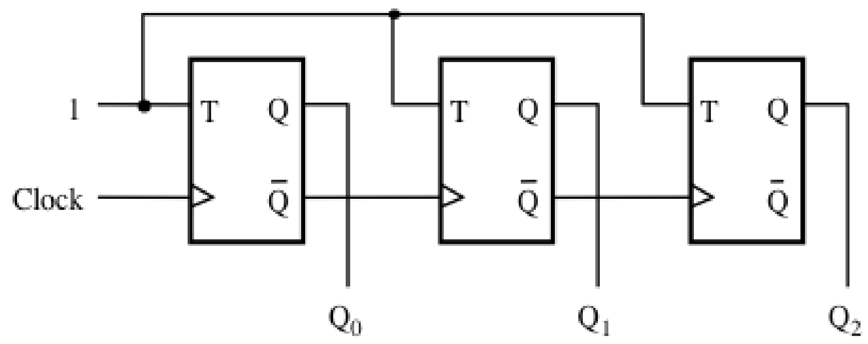



Figure 7.19. Parallel-access shift register.

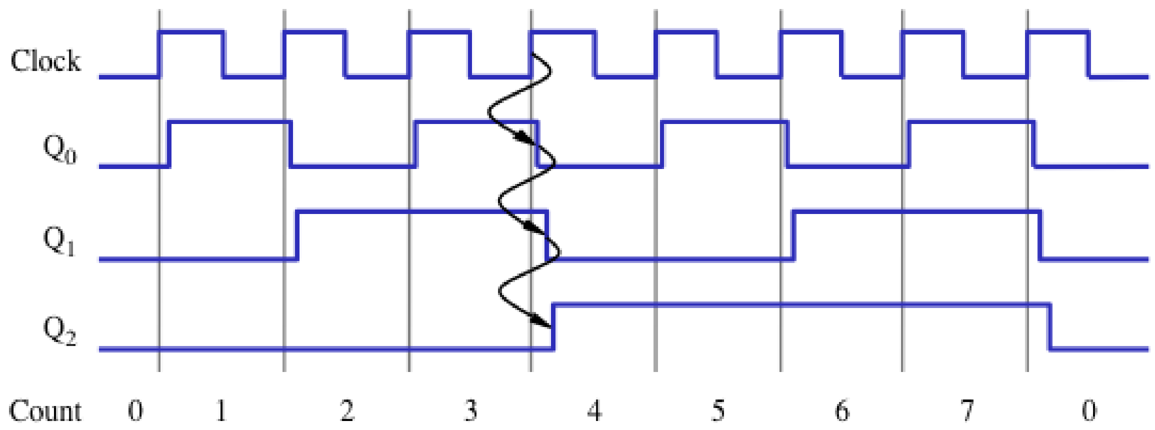
# Counters

- Counter: a circuit that can increment or decrement a count by 1
- Applications: generating time intervals, count the number of occurrence of an event, ….
- Counters can be build using T and D flip-flops

# Up counter with T flip-flop



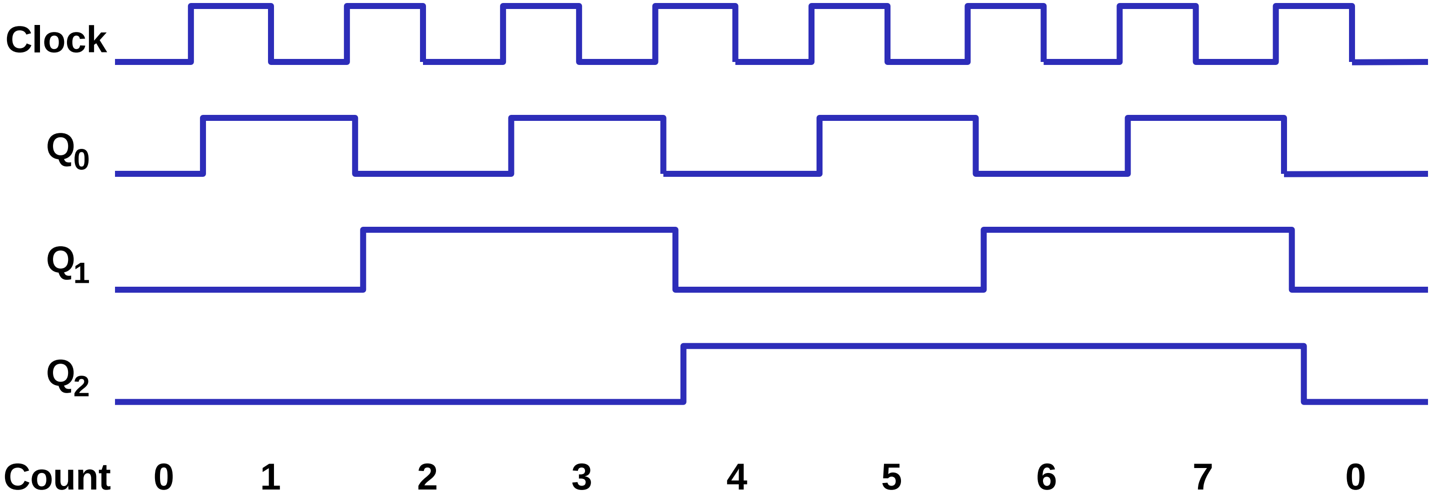

(a) Circuit




(b) Timing diagram

Figure 7.20. A three-bit up-counter.

# Up counter with T flip-flop



- The counter has three flip flops
- Only the first one is directly connected to the clock
- The other two respond after a delay
- For this reason it is called an asynchronous counter

# Down counter with T flip-flops

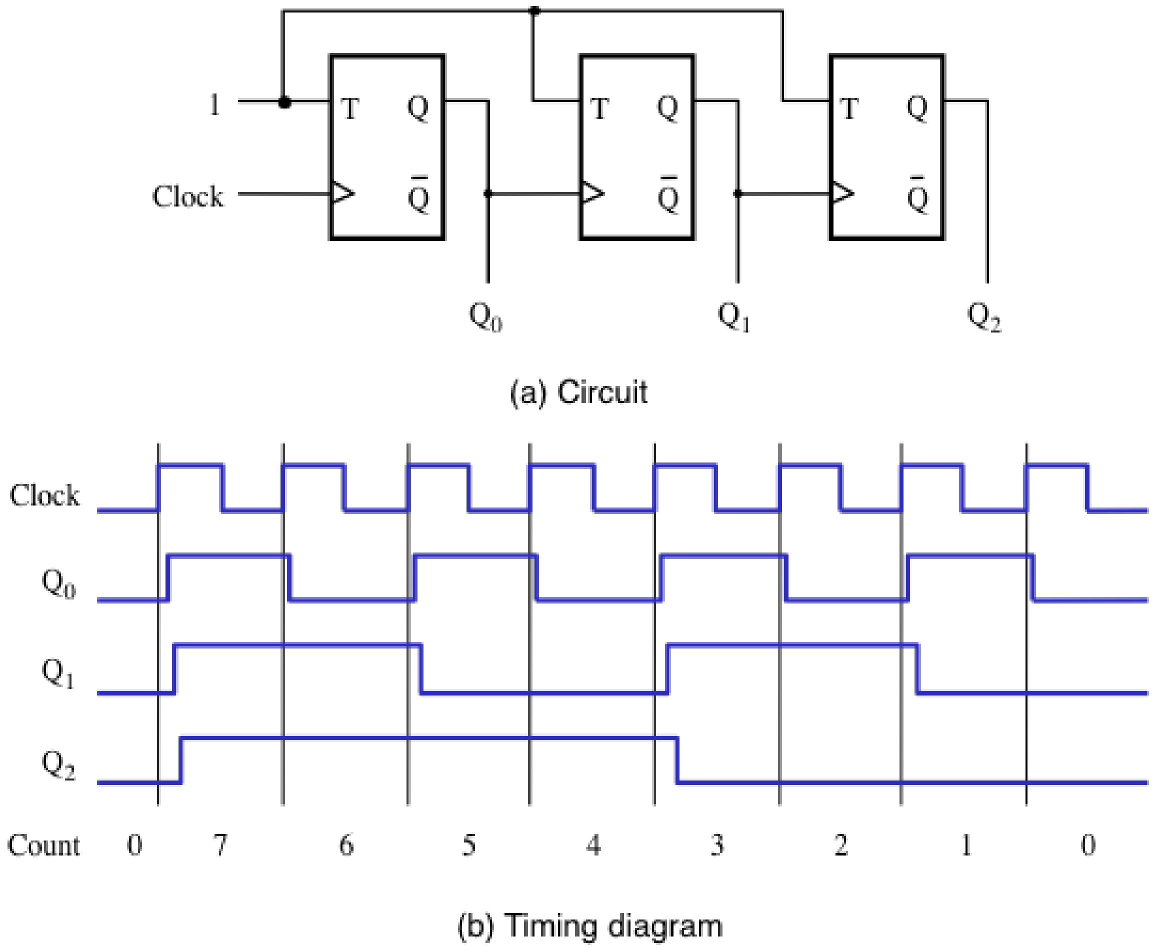



Figure 7.21. A three-bit down-counter.

# Synchronous counters

- Problem with asynchronous counters: long delays for large number of bits
- Solution: clock all the flip-flops at the same time (synchronous counter)

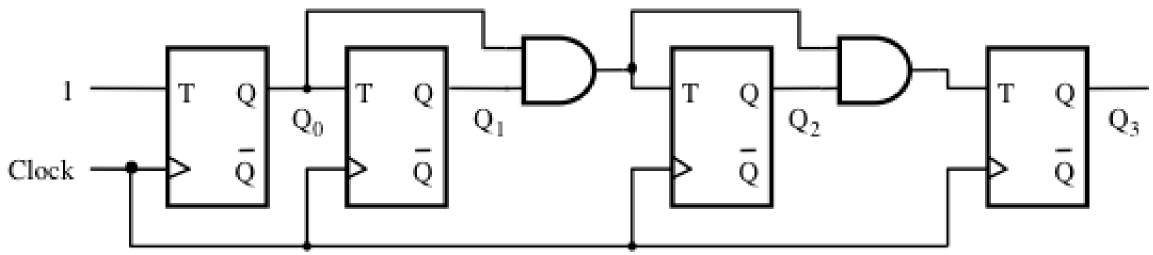
| Clock cycle | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 |

$Q_1$ changes

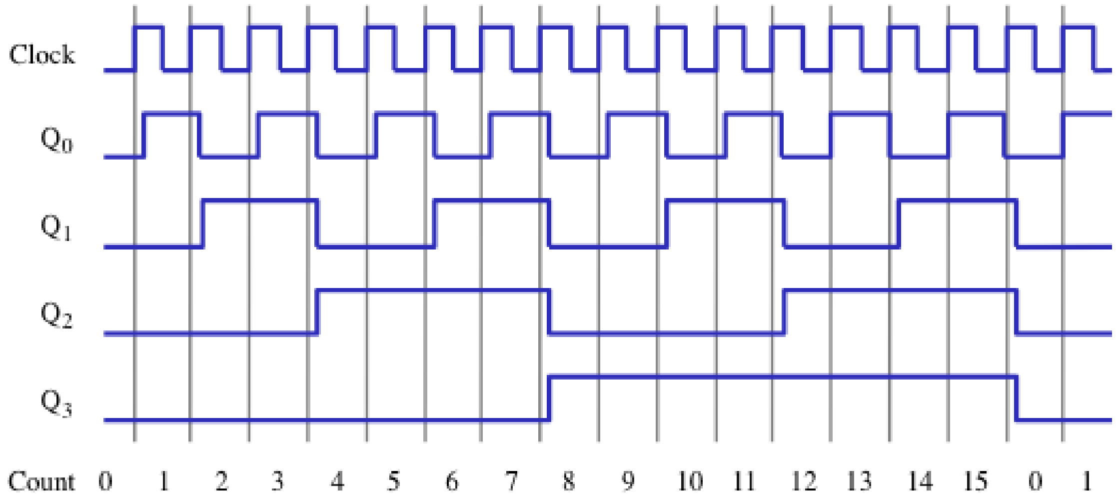
$Q_2$ changes

- Q0 changes on each clock cycle
- Q1 changes only when Q0=1
- Q2 changes only when Q1=1 and Q0=1

- $T_0=1$;
- $T_1=Q_0$
- $T_2=Q_0Q_1$
- $T_3=Q_0Q_1Q_2$




(a) Circuit




(b) Timing diagram

Figure 7.22. A four-bit synchronous up-counter.

# Enable and clear capability

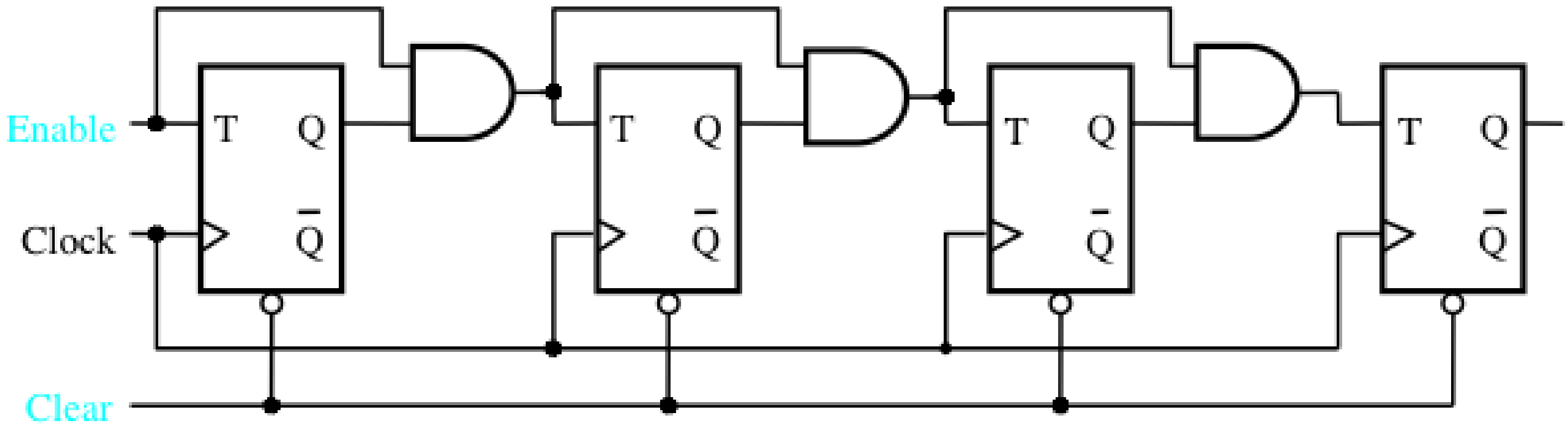



Figure 7.23. Inclusion of Enable and Clear capability.

# Synchronous counter with D flip flop

- Formal method: chapter 8

$$D_0 = Q_0 \oplus Enable$$

$$D_1 = Q_1 \oplus Q_0 . Enable$$

$$D_2 = Q_2 \oplus Q_1 . Q_0 . Enable$$

$$D_3 = Q_3 \oplus Q_2 . Q_1 . Q_0 . Enable$$

# Synchronous Counter with D Flip Flop

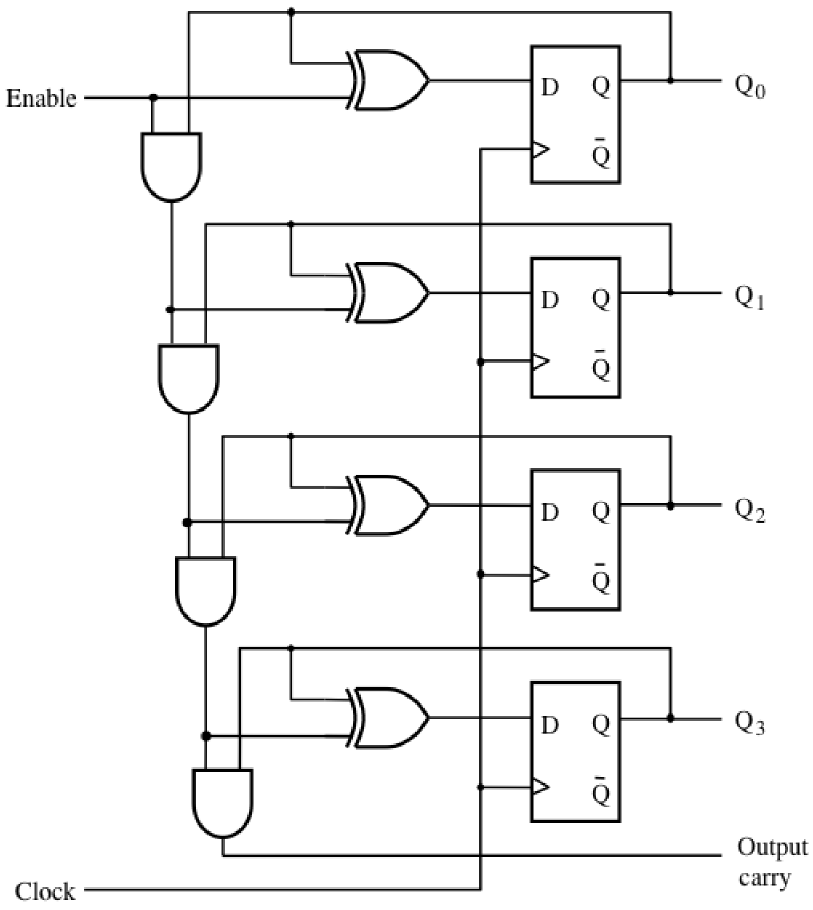



Figure 7.24. A four-bit counter with D flip-flops.

# Counter with parallel load

- Sometimes it is desirable to start the counter with an initial value

# Counter with parallel load

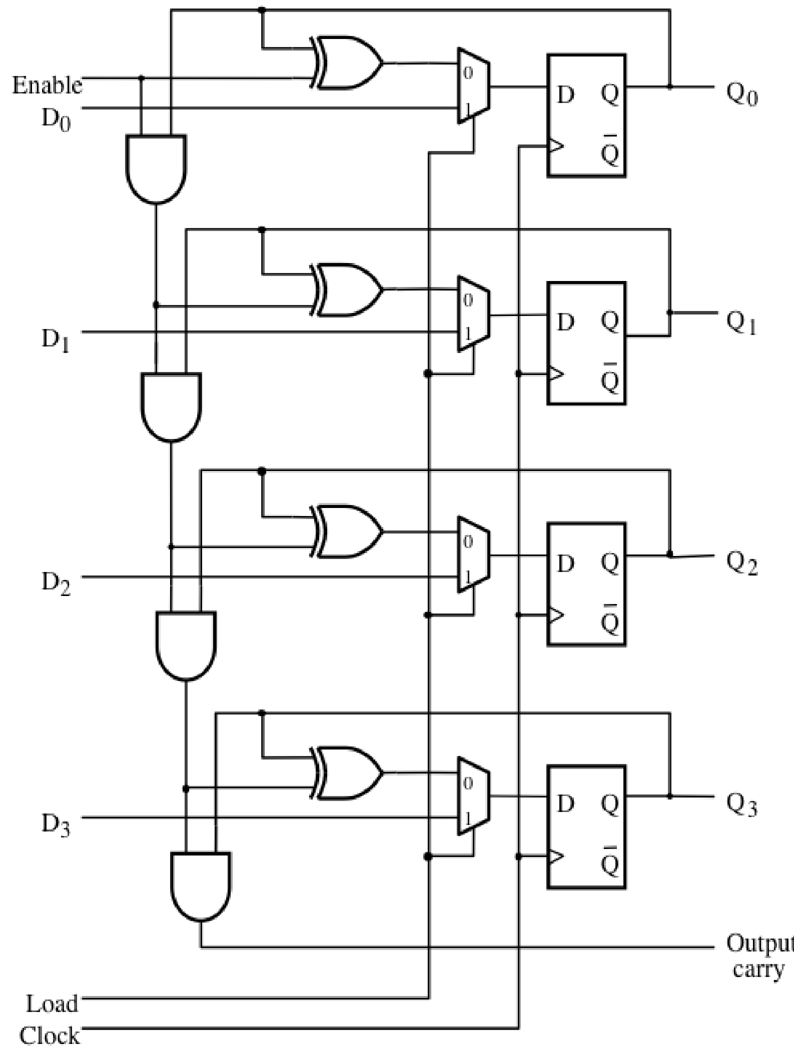
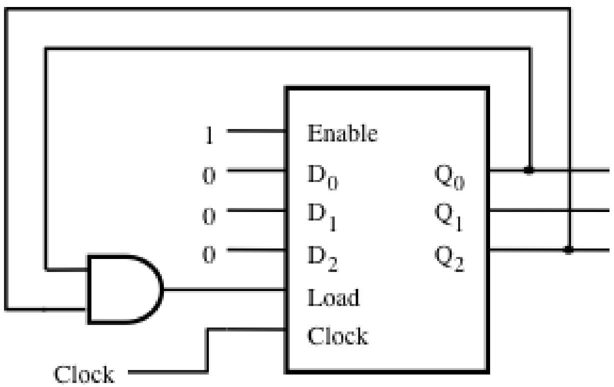


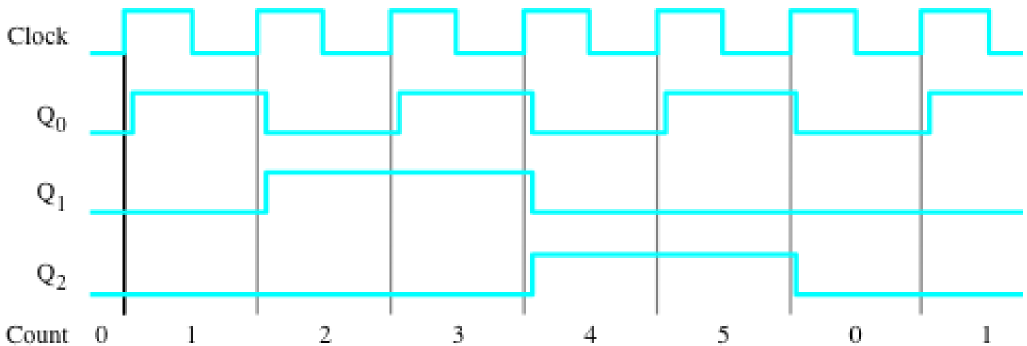

Figure 7.25. A counter with parallel-load capability.

# Reset Synchronization

- How can we design a counter that counts modulo some base that is not a power of 2 (e.g., modulo-6 counter counting 0, 1, 2, 3, 4, 5, 0, 1, ….)
- Detect 5 and then load zero into the counter

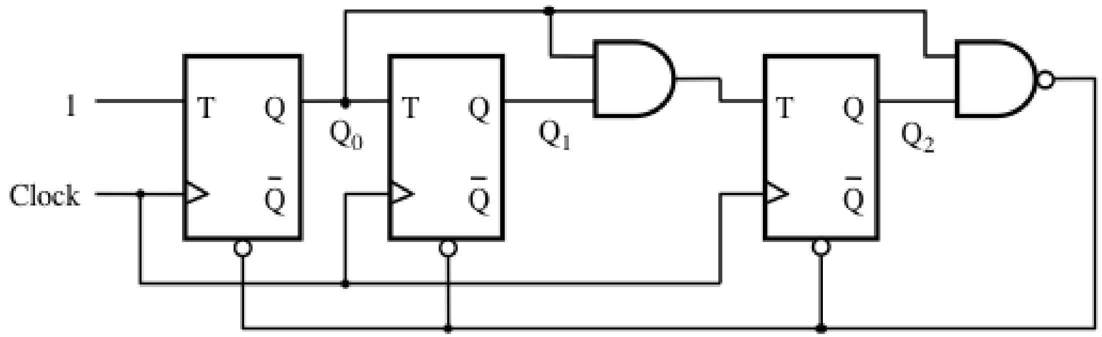



(a) Circuit

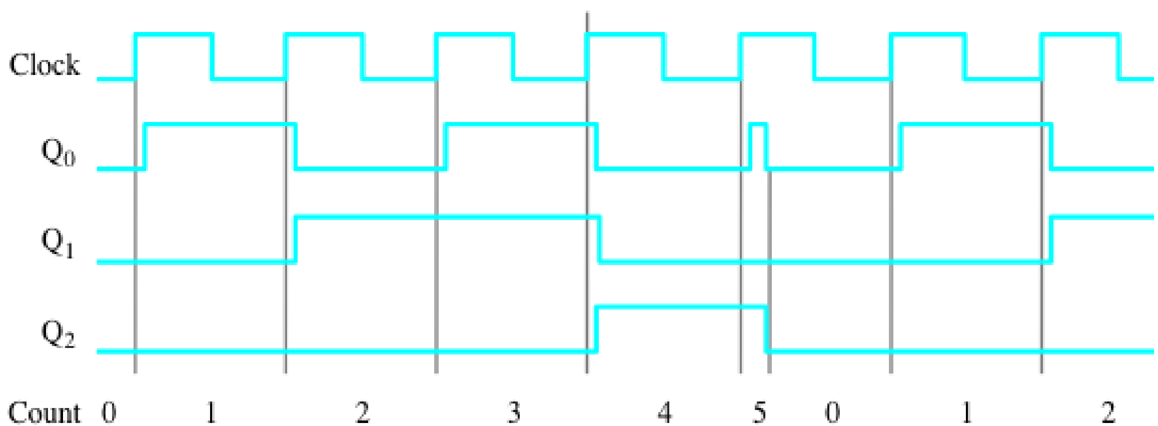



(b) Timing diagram

Figure 7.26. A modulo-6 counter with synchronous reset.




(a) Circuit




(b) Timing diagram

Figure 7.27. A modulo-6 counter with asynchronous reset.

# BCD counter

- In a BCD counter, the counter should be reset after the count of 9 has been obtained

# BCD Counter

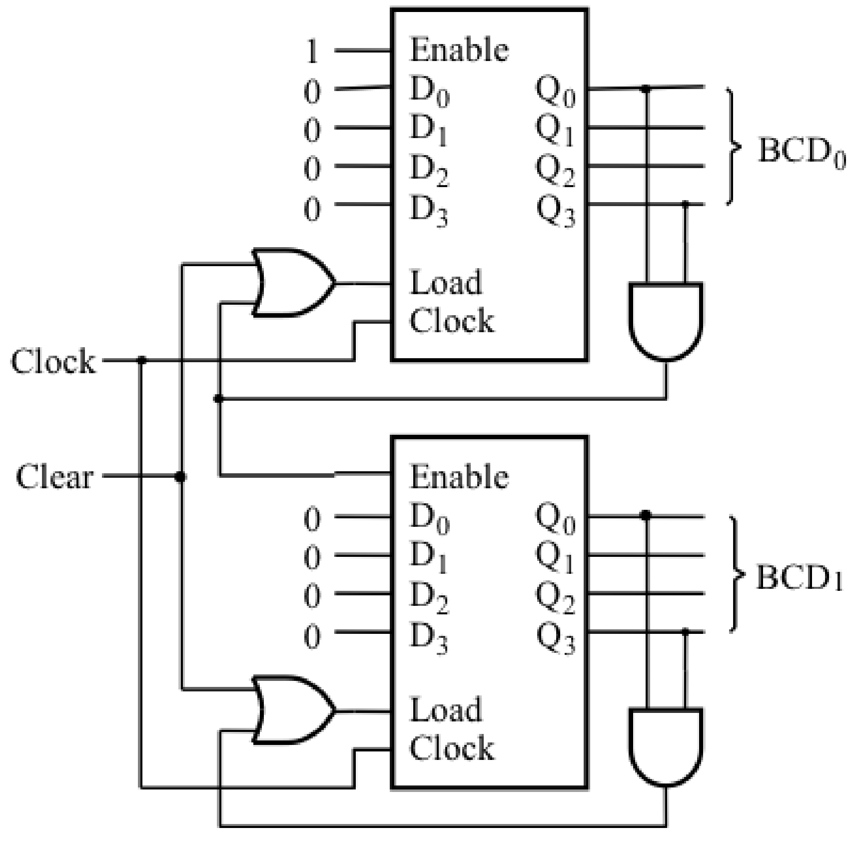
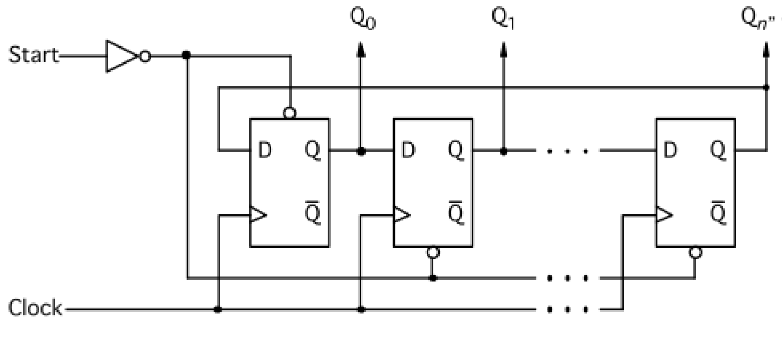



Figure 7.28. A two-digit BCD counter.

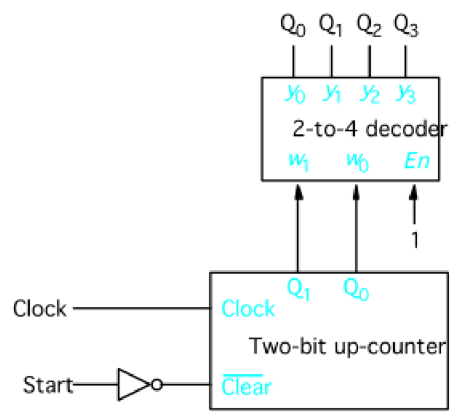
# Ring Counter

- In all the previous counters the count is indicated by the state of the flip-flops in the counter
- It is possible to design a counter in which each flip-flop reaches the state of Qi=0 for exactly one count while for other counts Qi=0
- This is called a ring counter and it can be built from a shift register

# Ring Counter




(a) An *n*-bit ring counter




(b) A four-bit ring counter

Figure 7.29. Ring counter.

# Johnson Counter

- If instead of Q output we take the Q' output of the last stage in a ring counter and feed it back to the first stage we get a Johnson counter.
- It counts to a sequence of length 2n
- For example for 4-bit the sequence would be: 0000, 0001, 1100, 1110, 1111, 0111, 0011, 0001, 0000.

# Johnson counter

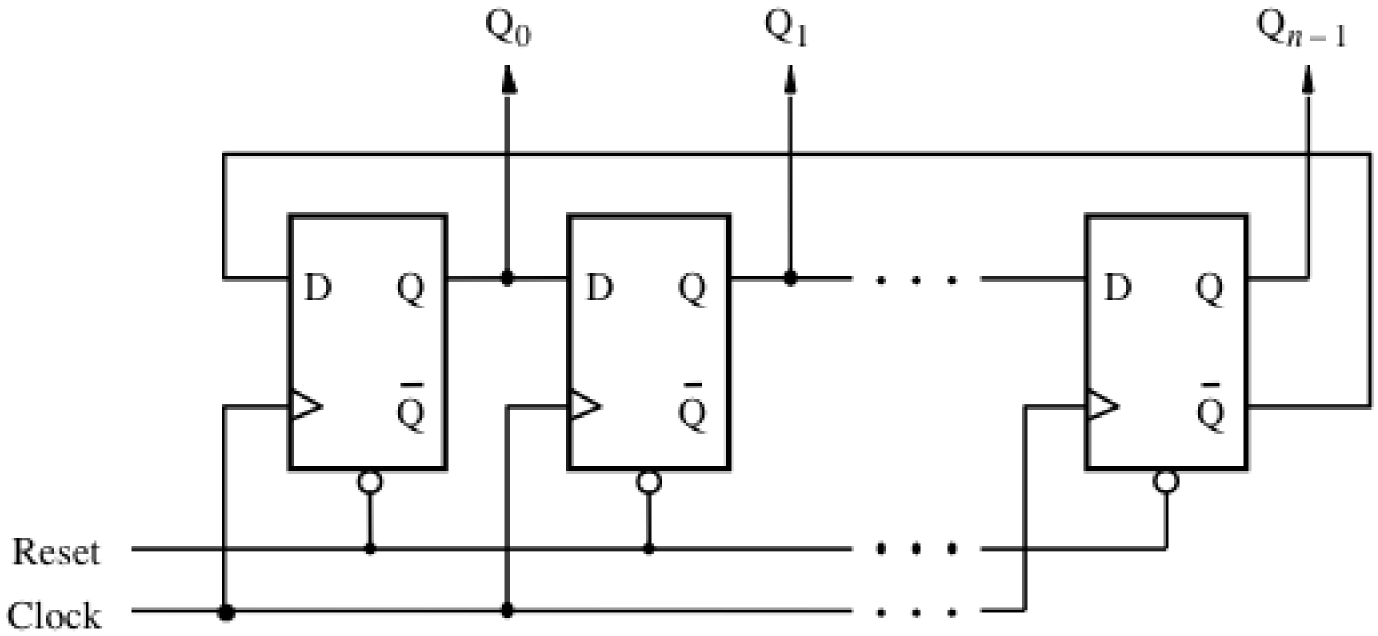



Figure 7.30. Johnson counter.