Sequential logic circuits

First Class

INTRODUCTION

- **Combinational Logic:**
  - Output depends only on current input
  - Has no memory
  - The design depends mainly on **Logic Gates**
- No way of remembering or storing information after inputs have been removed.
- To handle this, we need sequential logic capable of storing intermediate (and final) results.
- **Sequential Logic:**
  - Output depends not only on current input but also on past input values, e.g., design a counter
  - Need some type of memory to remember the past input values
  - The design depends mainly on **Flip-Flop Circuit**

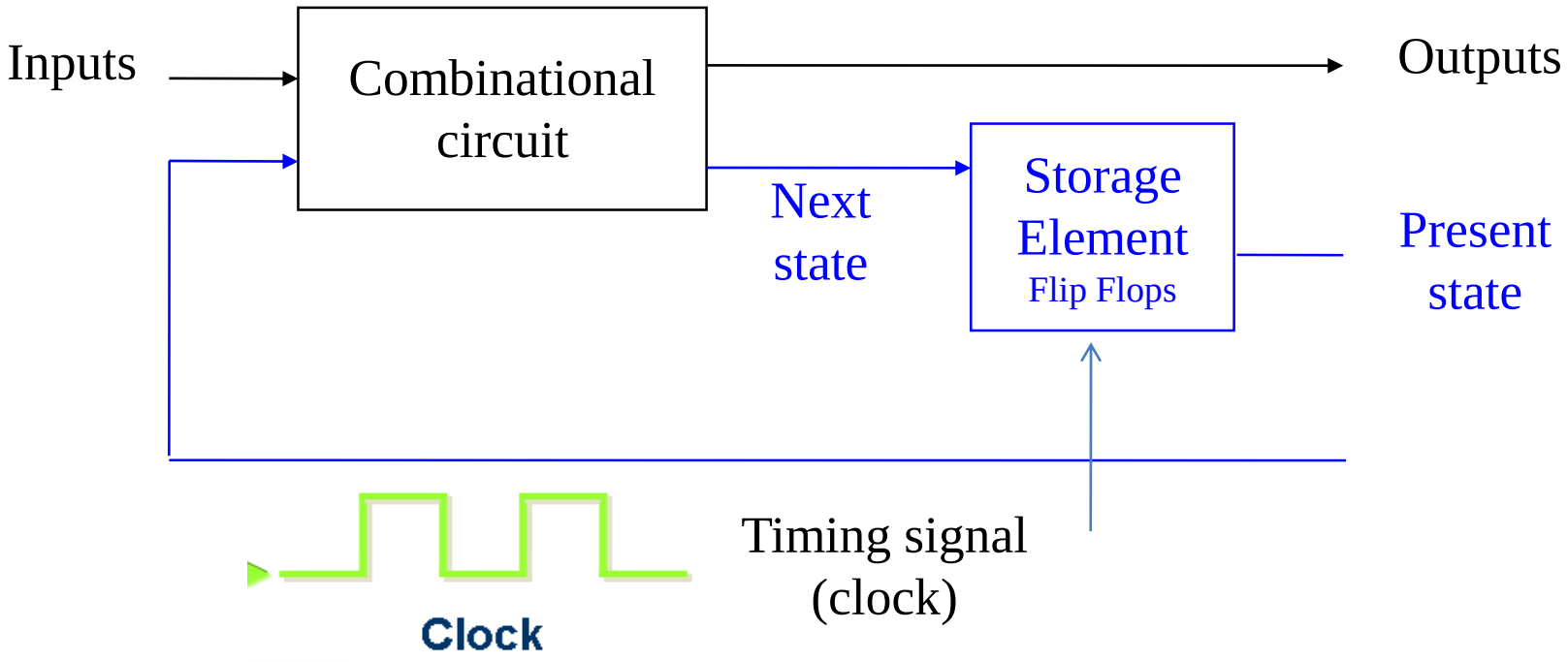
Sequential Logic Circuits

Sequential circuits can be Asynchronous or synchronous.

Asynchronous sequential circuits change their states and output values whenever a change in input values occurs.

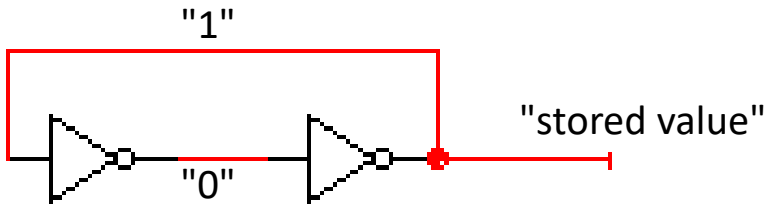
Synchronous sequential circuits change their states and output values at fixed points of time, i.e. clock signals.

Sequential Logic Circuits

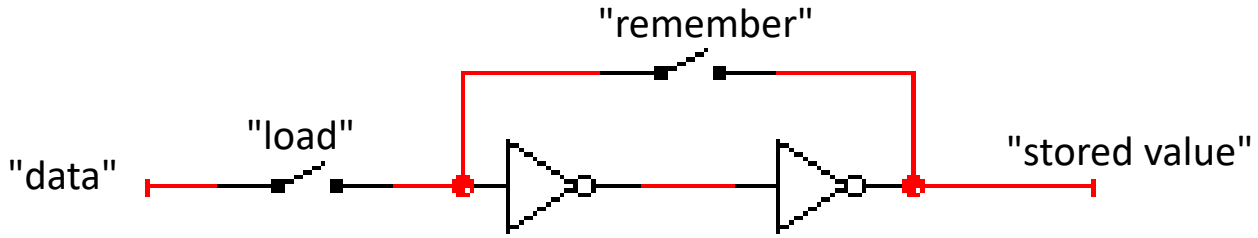


Storage Element

- Two inverters form a static memory cell
  - will hold value as long as it has power applied



- How to get a new value into the memory cell?
  - selectively break feedback path
  - load new value into cell



Sequential Logic Circuits

- ***Multivibrators Logic Devices:***

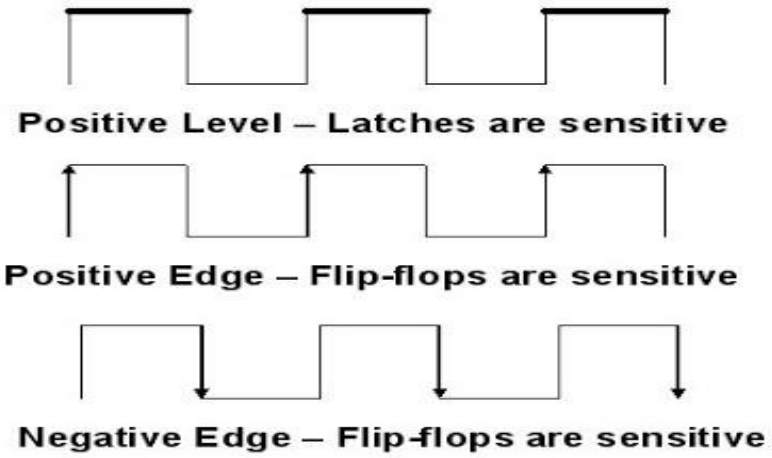
✓ **Bistable device:** have two stable states, called SET and RESET.

✓ **Monostable device:** commonly known as the One-shot, it has only one stable state.

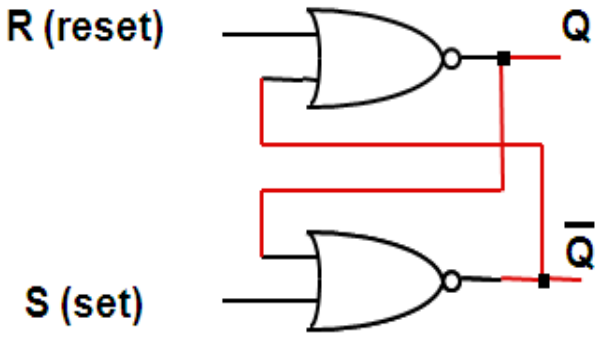
✓ **Astable device:** has no stable state and is used primarily as an oscillator, which is a self-sustained waveform generator.

Storage Cells

- The two most popular varieties of storage cells used to build sequential circuits are: **latches** and **flip-flops**.
  - **Latch:** level sensitive storage element
  - **Flip-Flop:** edge triggered storage element
- Common examples of latches:
  S-R latch, \S-\R latch, D latch
  (= gated D latch)
  Common examples of flip-flops:
  D-FF, D-FF with enable,
  Scan-FF, JK-FF, T-FF

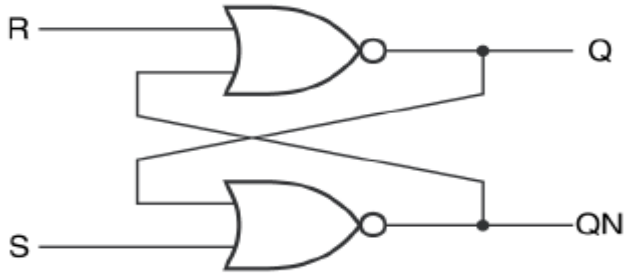


S-R Latch with NORs

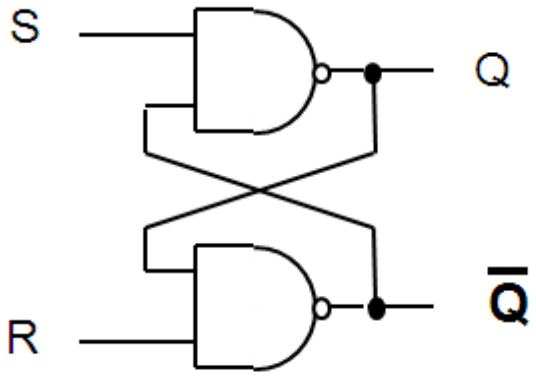


S	R	Q	Q'	
1	1	0	0	**Undefined - Not allowed**
1	0	1	0	**Set**
0	1	0	1	**Reset**
0	0	0 1	1 0	**Stable - No change**

- S-R latch made from cross-coupled NORs
- If Q = 1, set state
- If Q = 0, reset state
- Usually S=0 and R=0
- S=1 and R=1 generates unpredictable results



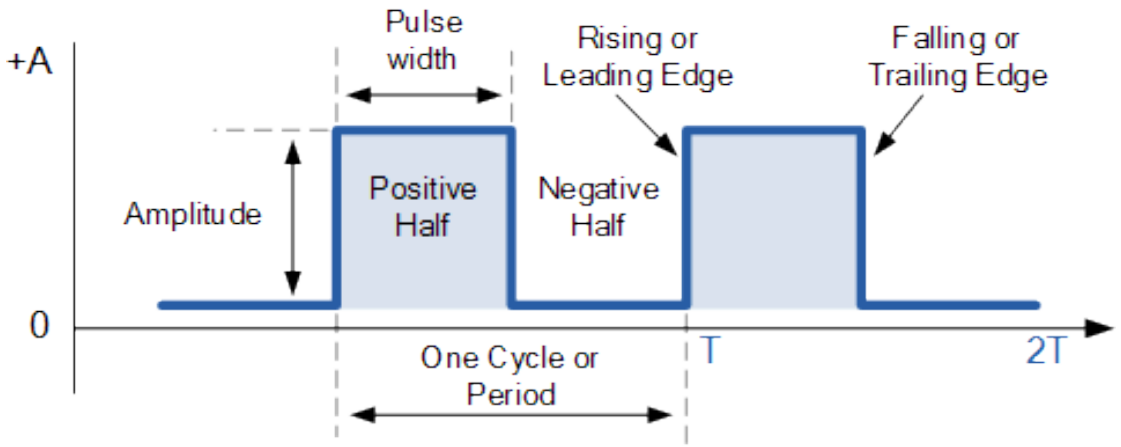
S-R Latch with NANDs

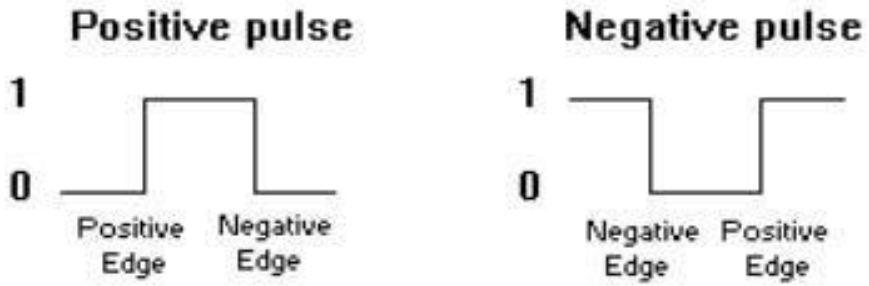


S	R	Q	Q'	
0	0	1	1	**Not allowed**
0	1	1	0	**Set**
1	0	0	1	**Reset**
1	1	0	1	**Store - No change**
		1	0	

- Latch made from cross-coupled NANDs
- Sometimes called S'-R' latch
- Usually S=1 and R=1
- S=0 and R=0 generates unacceptable result.

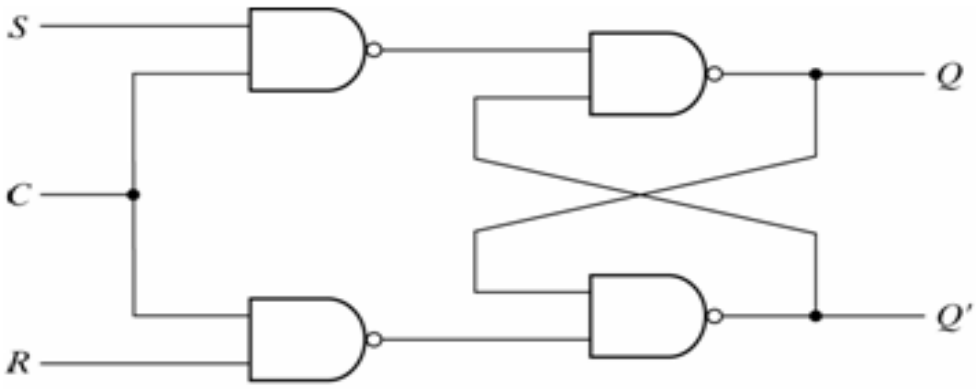
Clock Pulse Definition





Definition of clock pulse transition

S-R Latch with Control Input



(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; set state
1	1	1	Indeterminate

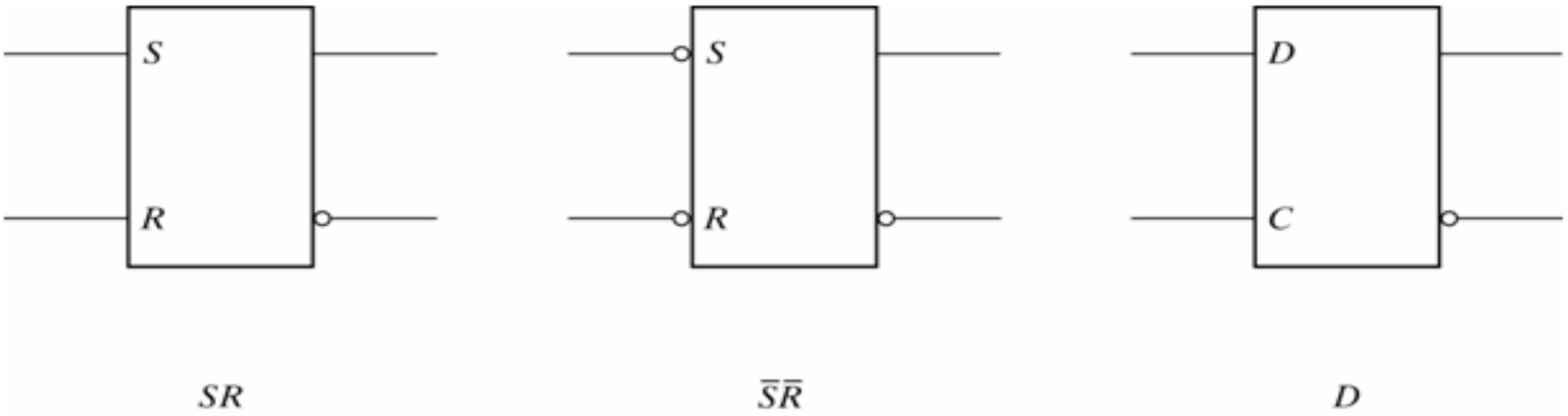
- Not allowed

(b) Function table

SR Latch with Control Input

- Occasionally, desirable to avoid latch changes
- C = 0 disables all latch state changes
- Control signal enables data change when C = 1
- Right side of circuit same as ordinary S-R latch.

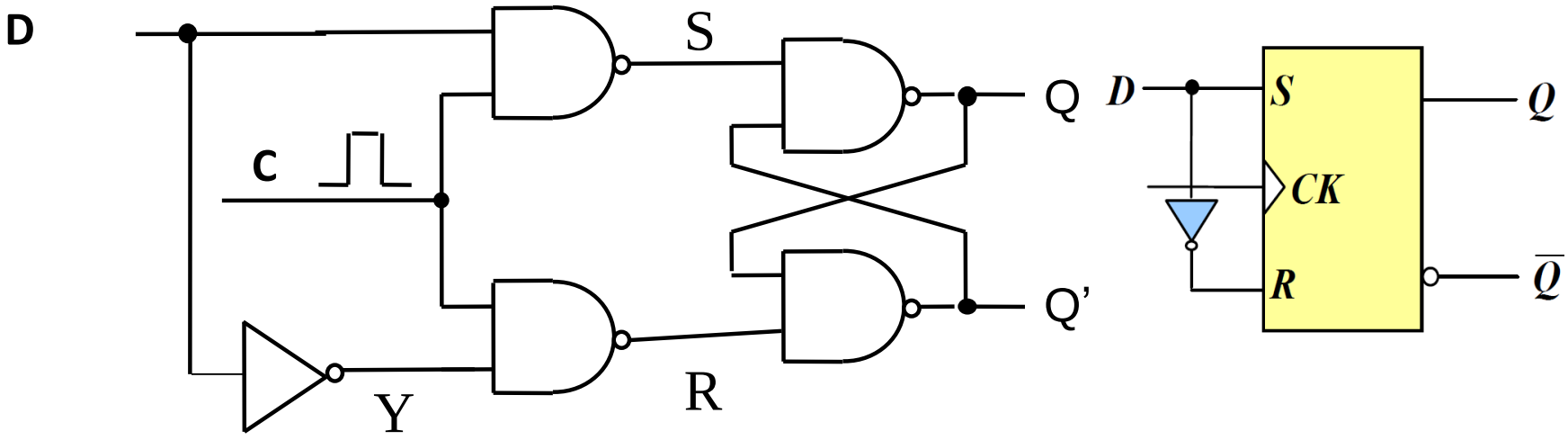
Symbols for Latches



Graphic Symbols for Latches

- SR latch is based on NOR gates
- S'R' latch based on NAND gates
- D latch can be based on either.
- D latch sometimes called transparent latch

D Latch

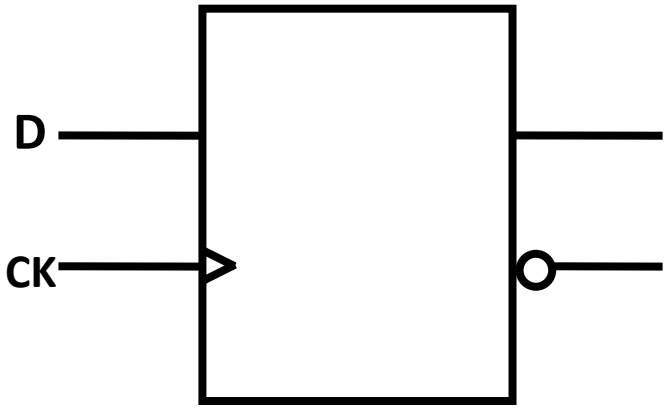


D	C	Q	Q'
0	1	0	1
1	1	1	0
X	0	Q_0	Q_0'

D Flip-Flop

- Stores a value on the positive edge of C
- Input changes at other times have no effect on output

Positive edge triggered

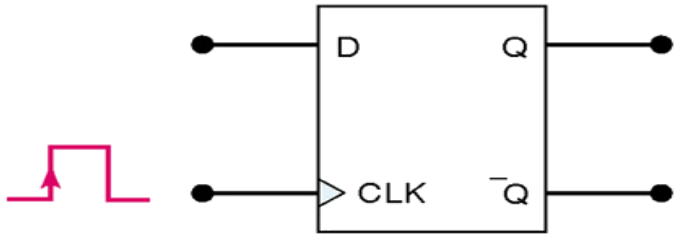


D	C	Q	Q'
0	↑	0	1
1	↑	1	0
X	0	Q_0	Q_0'

D gets latched to Q on the rising edge of the clock.

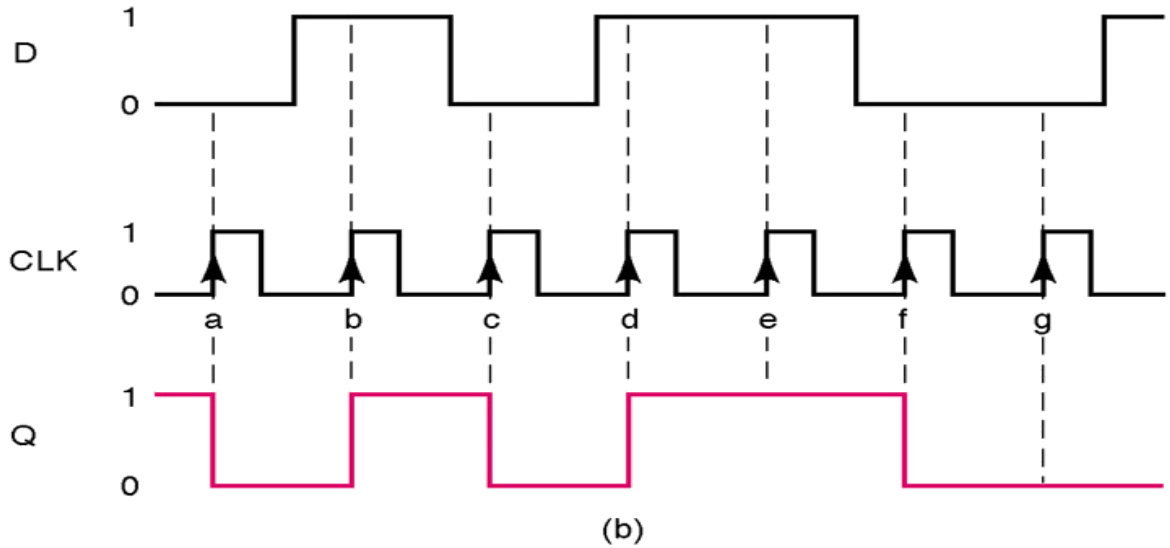
D Flip-Flop

$Q_{n+1} = D_n$



D	CLK	Q
0	↑	0
1	↑	1

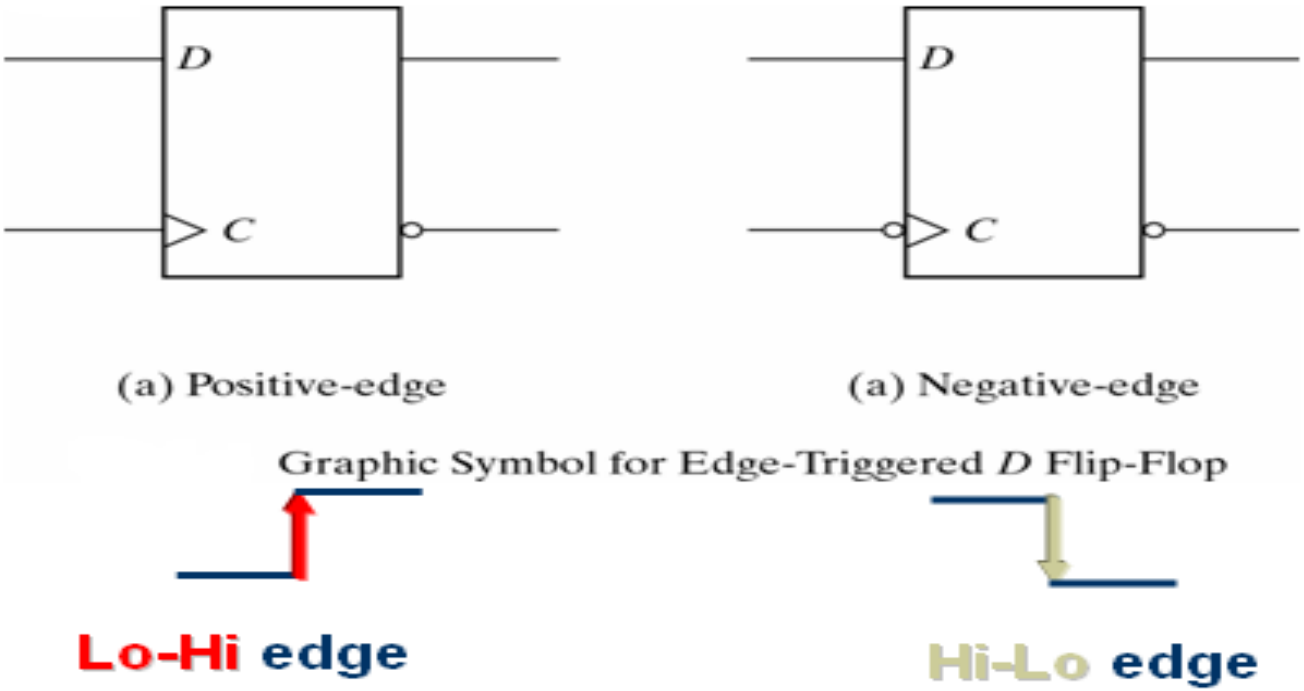
(a)



(b)

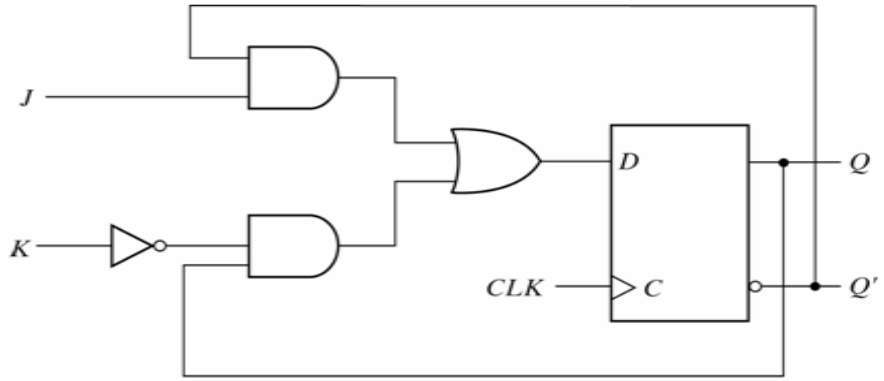
D Flip-Flop
Positive and Negative Edge

- D flops can be triggered on positive or negative edge. Bubble before *Clock (C)* input indicates negative edge trigger

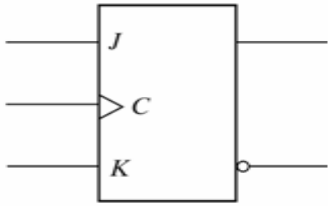


Other Types of Flip-Flops

J-K Flip Flop



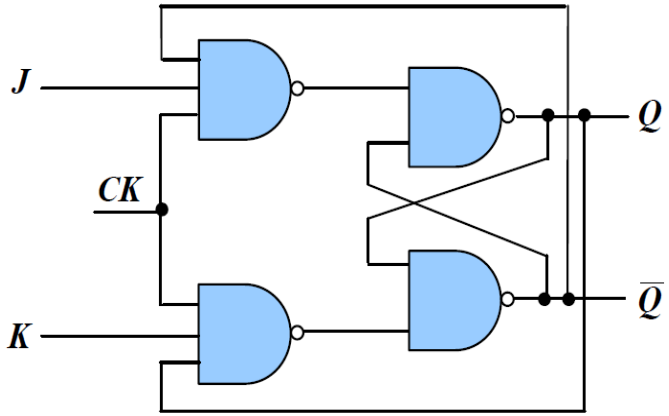
(a) Circuit diagram

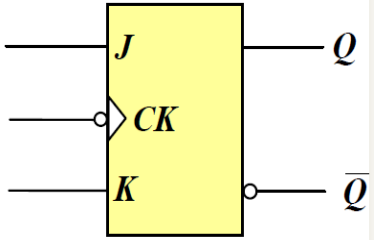


(b) Graphic symbol

$D = J\overline{Q} + \overline{K}Q$

JK Flip-Flop

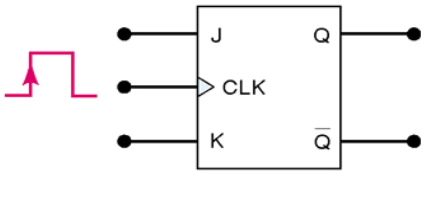




Clocked

- Two data inputs, J and K
- J -> set, K -> reset, if J=K=1 then toggle output

J K differs from RS, because there is **NO** not allowed state (11)

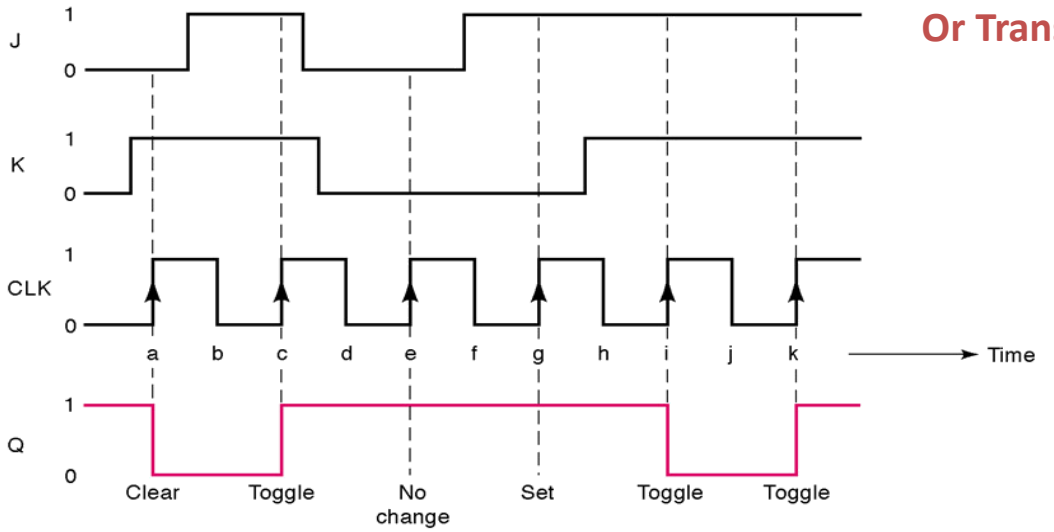


(a)

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

Characteristic Table

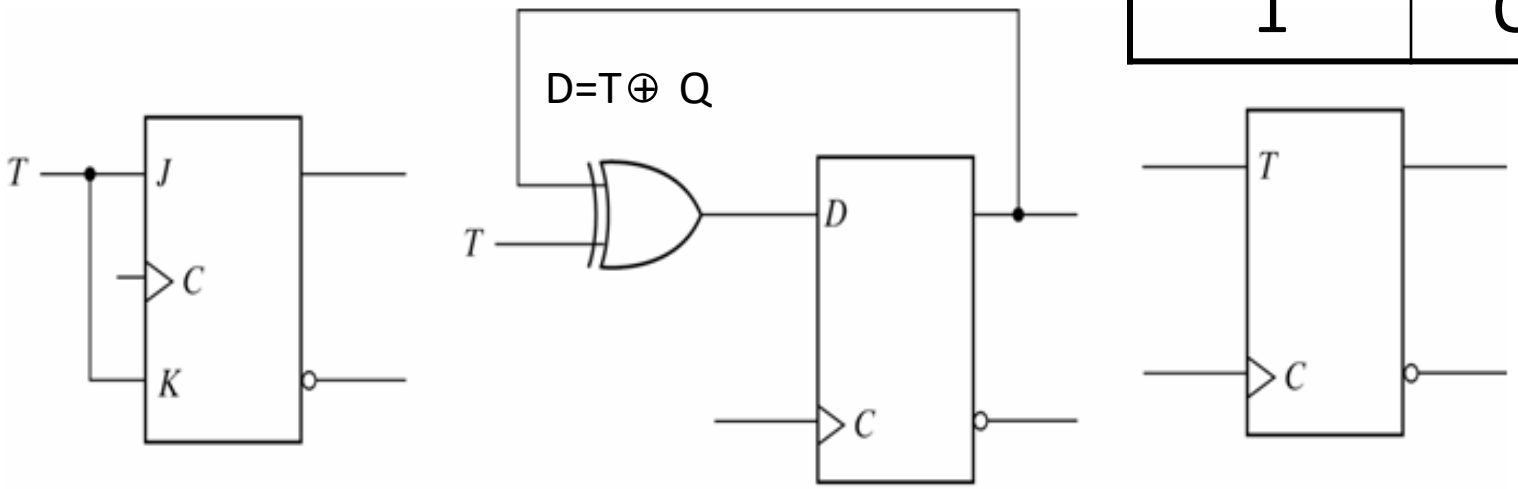
Or Transition Table



(b)

T Flip-Flop (*Toggle)*

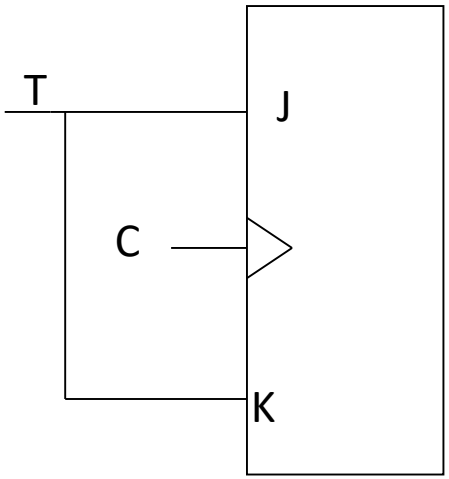
T	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$



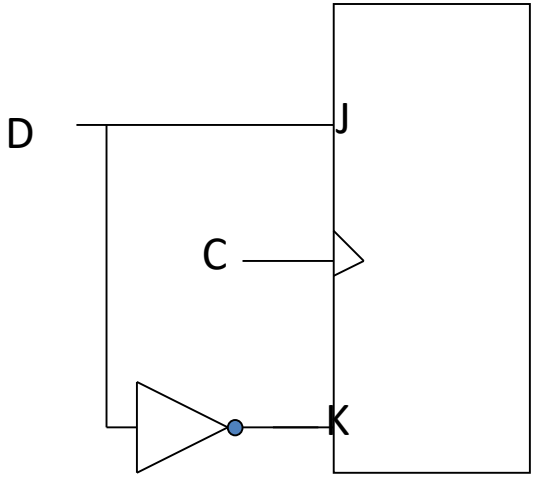
(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

T Flip-Flop

JK Converted to Other FFs



T from JK



D from JK

Summary

- Latches are based on combinational gates (e.g. NAND, NOR)
- Latches store data even after data input has been removed
- S-R latches operate like cross-coupled inverters with control inputs (S = set, R = reset)
- With additional gates, an S-R latch can be converted to a D latch (D stands for data)
- D latch is simple to understand conceptually
  - When C = 1, data input D is stored in latch at the output as Q.
  - When C = 0, data input D is ignored and previous latch value is at the output Q.

Summary

Latch

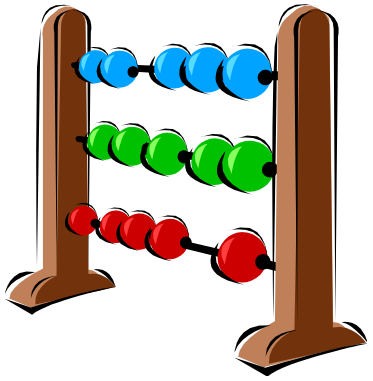
- – Clock input is level sensitive.
- – Output can change multiple times during a clock cycle.
- – Output changes while clock is active.

Flip-Flop

- – Clock input is edge sensitive.
- – Output can change only once during a clock cycle.
- – Output changes on clock transition.

Sequential Circuit

- Introduction to Counter
- ❖ Asynchronous Counter
- Asynchronous Up/Down Counter
- ❖ Synchronous Counter
- Synchronous Counter Up/Down



Sequential Circuit: Counter

Introduction

- Counter is a circuit which cycle through state sequence
- Two types of counter
  - Asynchronous counter (e.g. **ripple**)
  - Synchronous counter (e.g. **parallel**)
- **Ripple** counter let some flip-flop output, to be used as clock signal source for other flip-flop
- Synchronous counter use the same clock signal for all flip-flop

Sequential Circuit: Counter

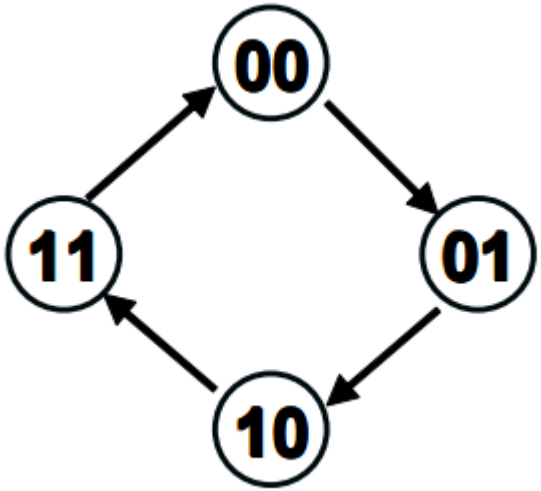
Asynchronous Counter (Ripple)

- Asynchronous Counter : flip-flop doesn’t change condition simultaneously because it doesn’t use single clock signal
- Also known as ripple counter because clock signal input as ripple through counter.
- Modulus (MOD) – the number of states it counts in a complete cycle before it goes back to the initial state.
- Thus, the number of flip-flops used depends on the MOD of the counter (ie; MOD-4 use 2 FF (2-bit), MOD-8 use 3 FF (3-bit), etc.

Asynchronous (Ripple) UP Counters

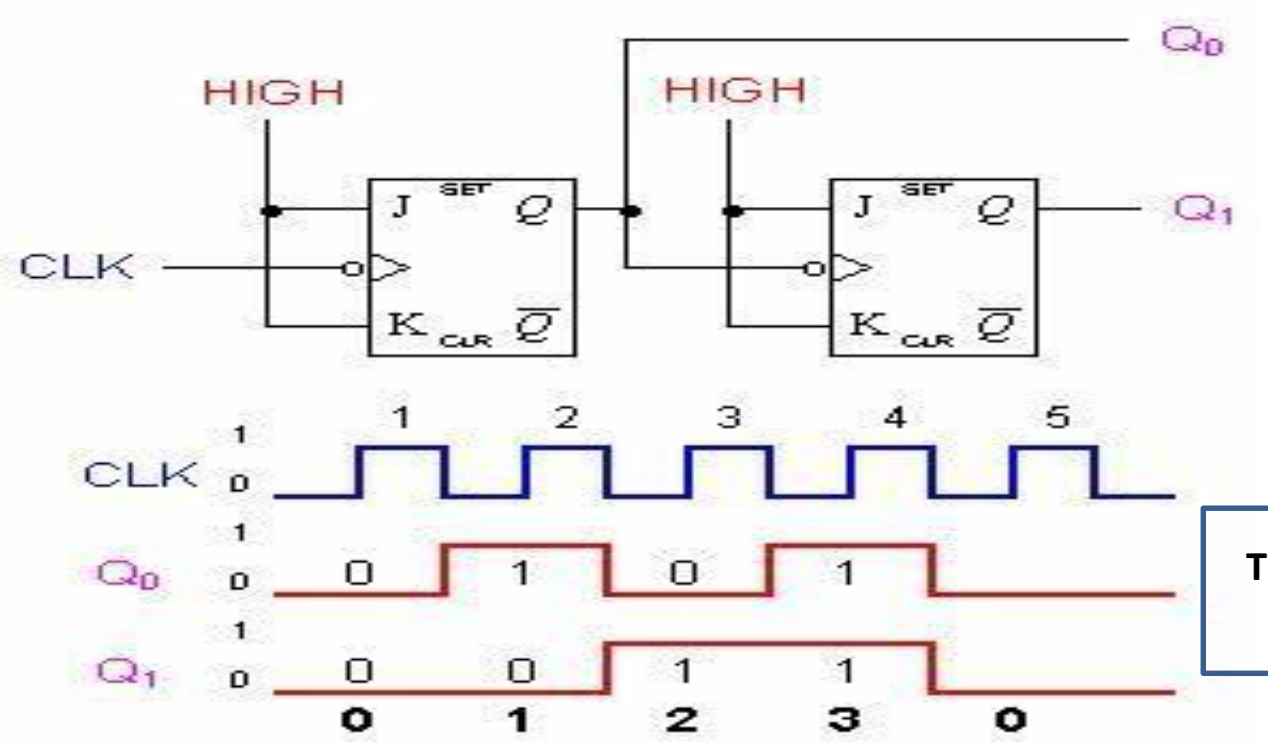
- The Asynchronous Counter that counts 4 number starts from 00→01→10→11 and back to 00 is called MOD-4 Ripple (Asynchronous) Up-Counter.
- **Next state table** and **state diagram**

Present State	Next State
Q_1Q_0	Q_1Q_0
00	01
01	10
10	11
11	00



Asynchronous Counter (Ripple) MOD-4

- **Example**: 2-bit ripple counter (UP COUNTER)
- Output from one flip-flop is connected to clock input for the next flip-flop.



Timing diagram
00 01 10 11

Asynchronous Counter (Ripple) MOD-8

- Example: 3-bit ripple counter (UP COUNTER)

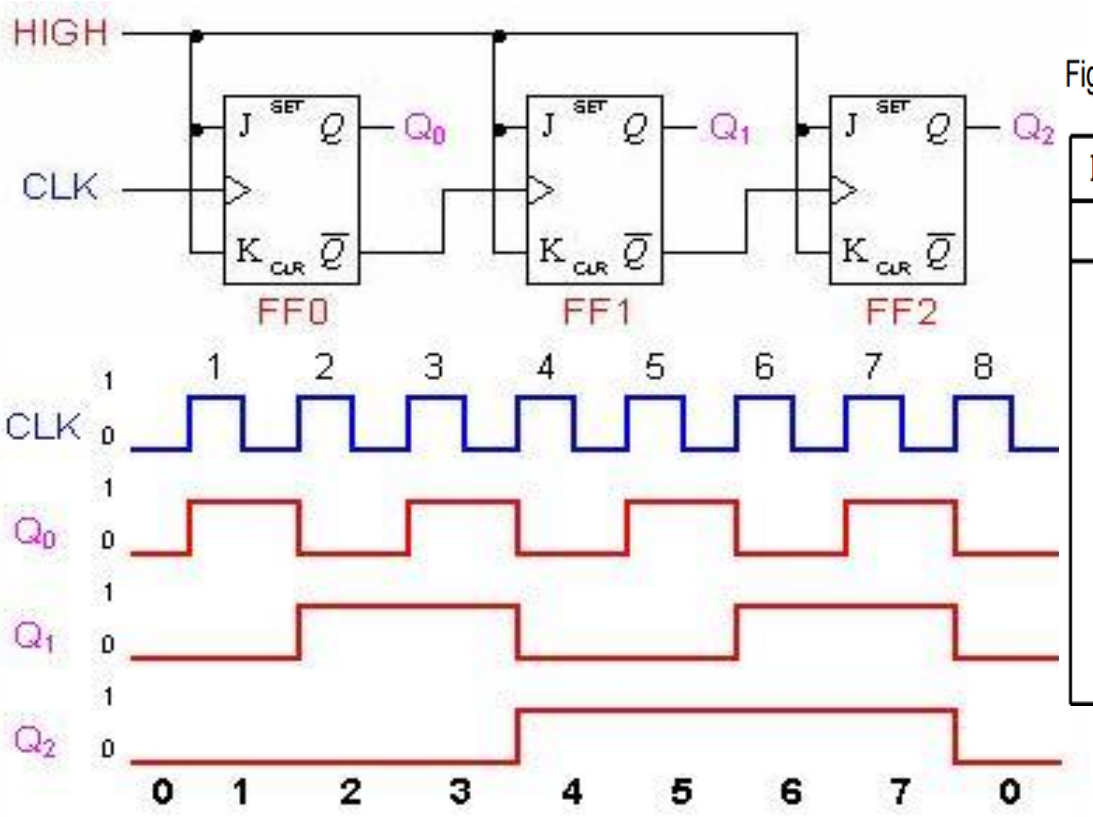
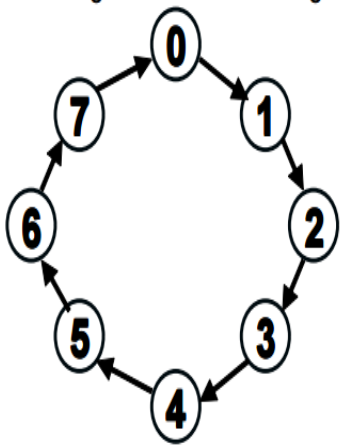


Figure 1.3a Next State Table

Present State	Next State
CBA	CBA
000	001
001	010
010	011
011	100
100	101
101	110
110	111
111	000

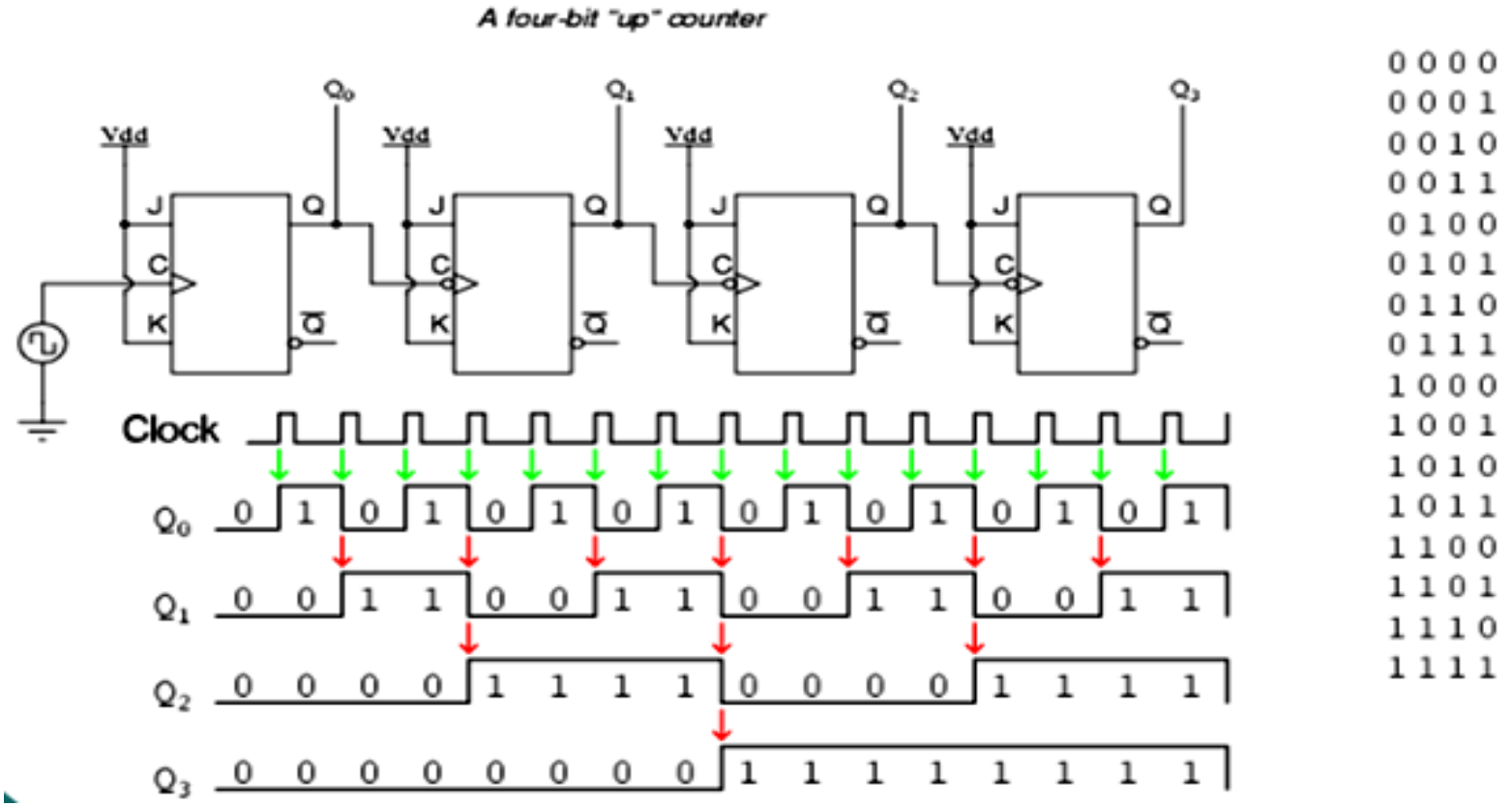
Figure 1.3b State Diagram



Asynchronous Counter (Ripple) MOD-16

Asynchronous Ripple Counter ((UP COUNTER))

- Example: 4-bit ripple counter **(negative edge triggered)**

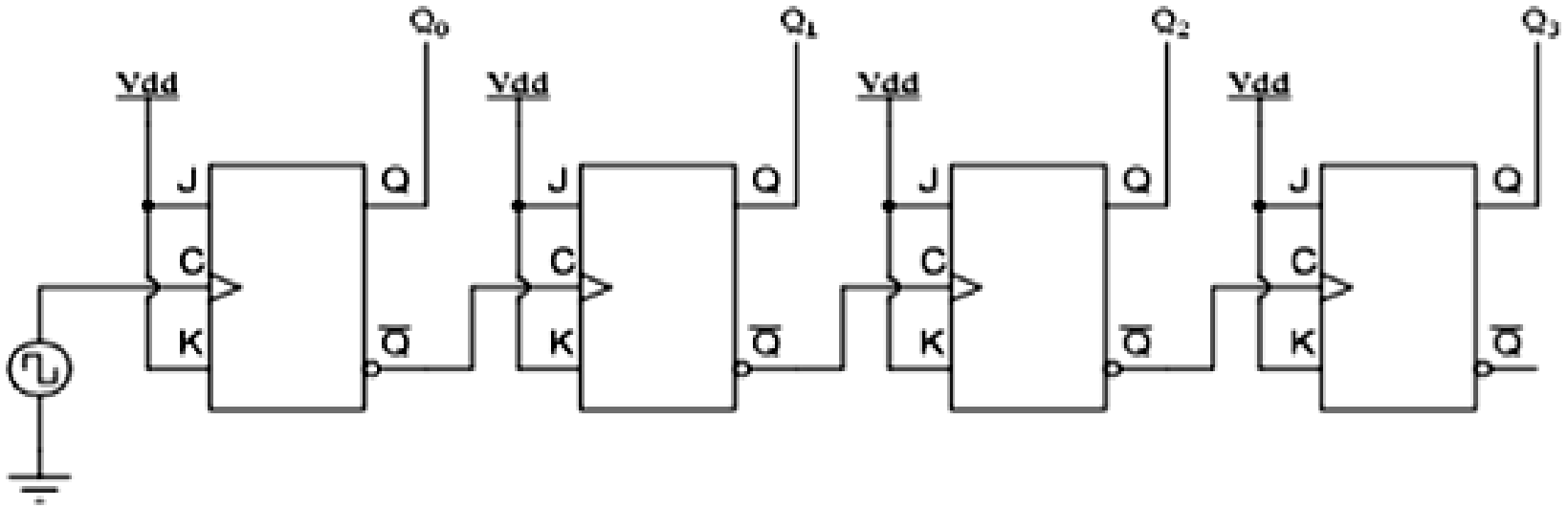


0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1

Asynchronous Counter (Ripple) MOD-16

- **(Positive edge triggered)**

Figure 1.6 : MOD 16 Asynchronous Up Counter



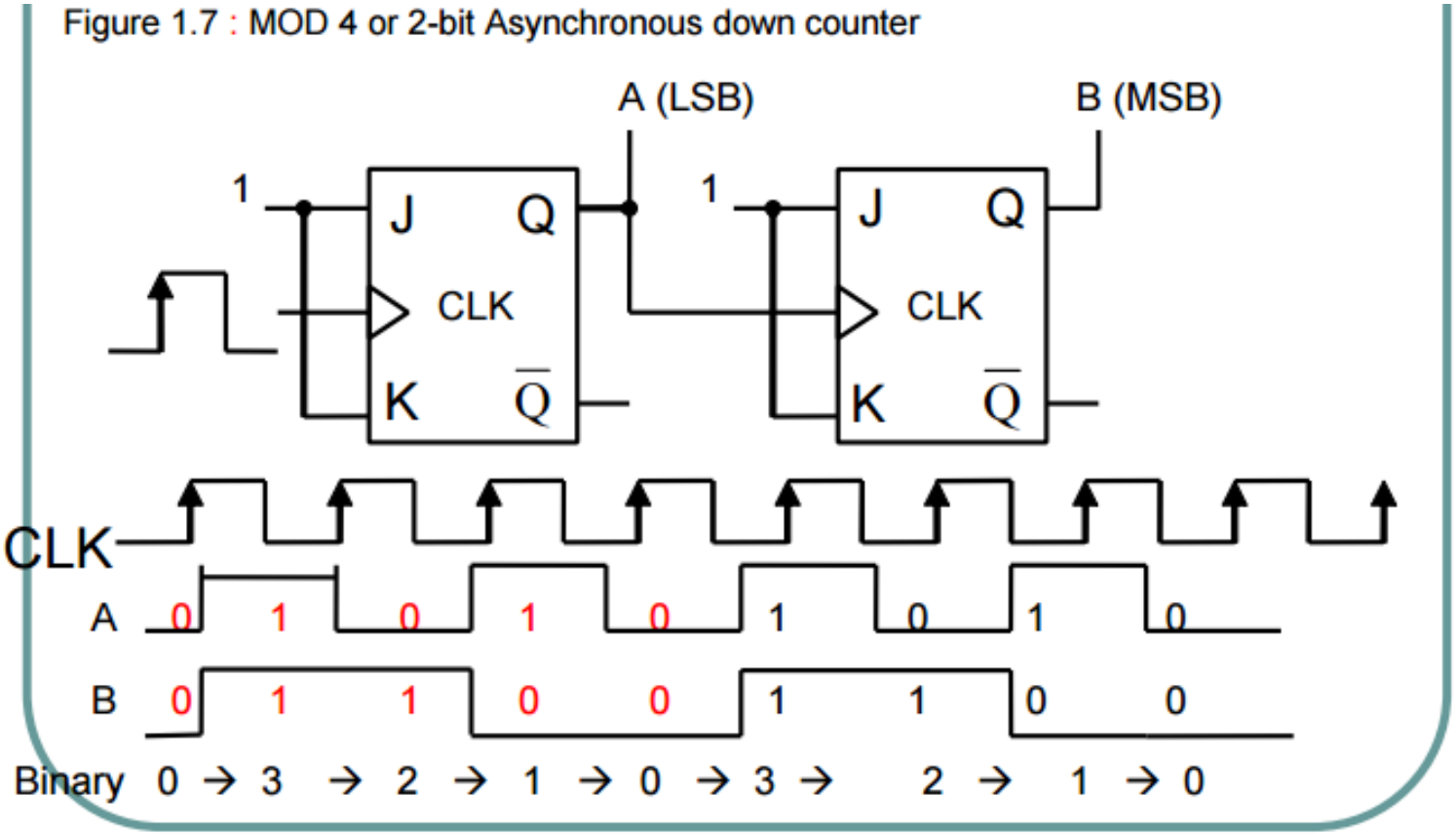
Asynchronous Down Counter

Binary Ripple down counter

- For positive edge triggered flip-flops the counter count down:
- e.g start from15 to 14 to 13 to…….
- The diagram is same as the count up binary counter except that the flip-flop trigger on the positive edge of the clock.
- If negative edge triggered flip-flops are used then the CLK input of each flip-flop must be connected to the complement output of the previous flip-flop. So, when the true output goes from 0 to 1, the complement will go from 1 to o and complement the next flip flop as required

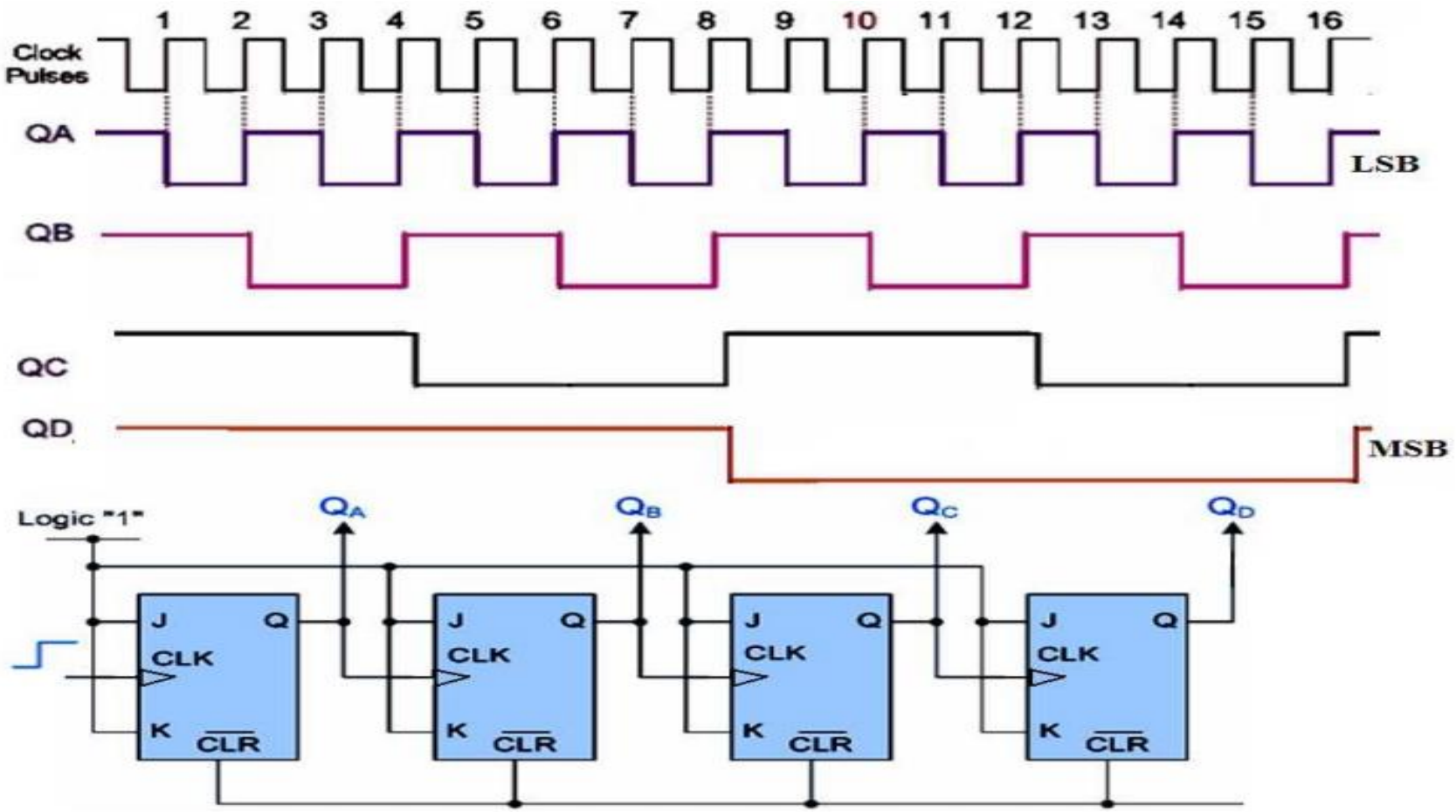
Asynchronous Down Counter

- **2-Bit Binary Down Counter**



Asynchronous Down Counter

- **4- Bit Binary Down Counter**



Asynchronous Counters (MOD $\neq 2^N$)

- So far, we have design the counters with MOD number equal to 2^N, where N is the number of bit (N = 1,2,3,4….) (also correspond to number of FF)
- Thus, the counters are limited on for counting MOD-2, MOD4, MOD-8, MOD-16 etc..
- The question is how to design a MOD-5, MOD-6, MOD-7, MOD-9 which is not a MOD-2^N **(MOD $\neq 2^N$)** ?
- MOD-6 counters will count from 0_{10} (000_2) to 5_{10}(101_2) and after that will recount back to 0_{10} (000_2) continuously.

MOD-6 ripple up-counter (MOD ≠ 2^N)

Present St.	Next St.
CBA	CBA
000	001
001	010
010	011
011	100
100	101
101	000(110)

Figure 1.8a :Next State Table

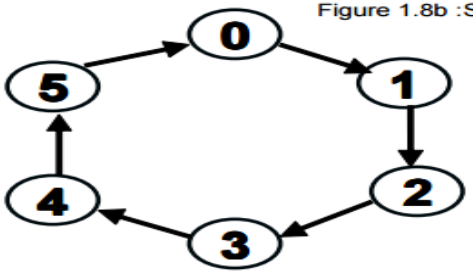
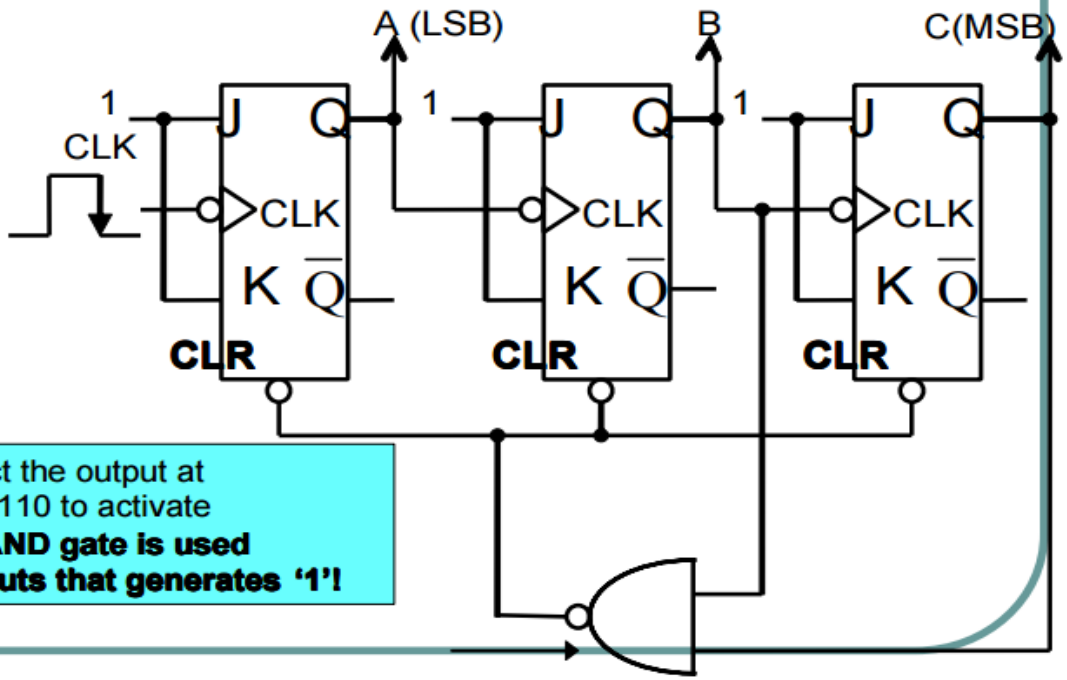


Figure 1.8b :State Diagram

Reset the state to 000_2 when 110_2 is detected

Circuit diagram for MOD-6 ripple up-counter (MOD ≠ 2^N)

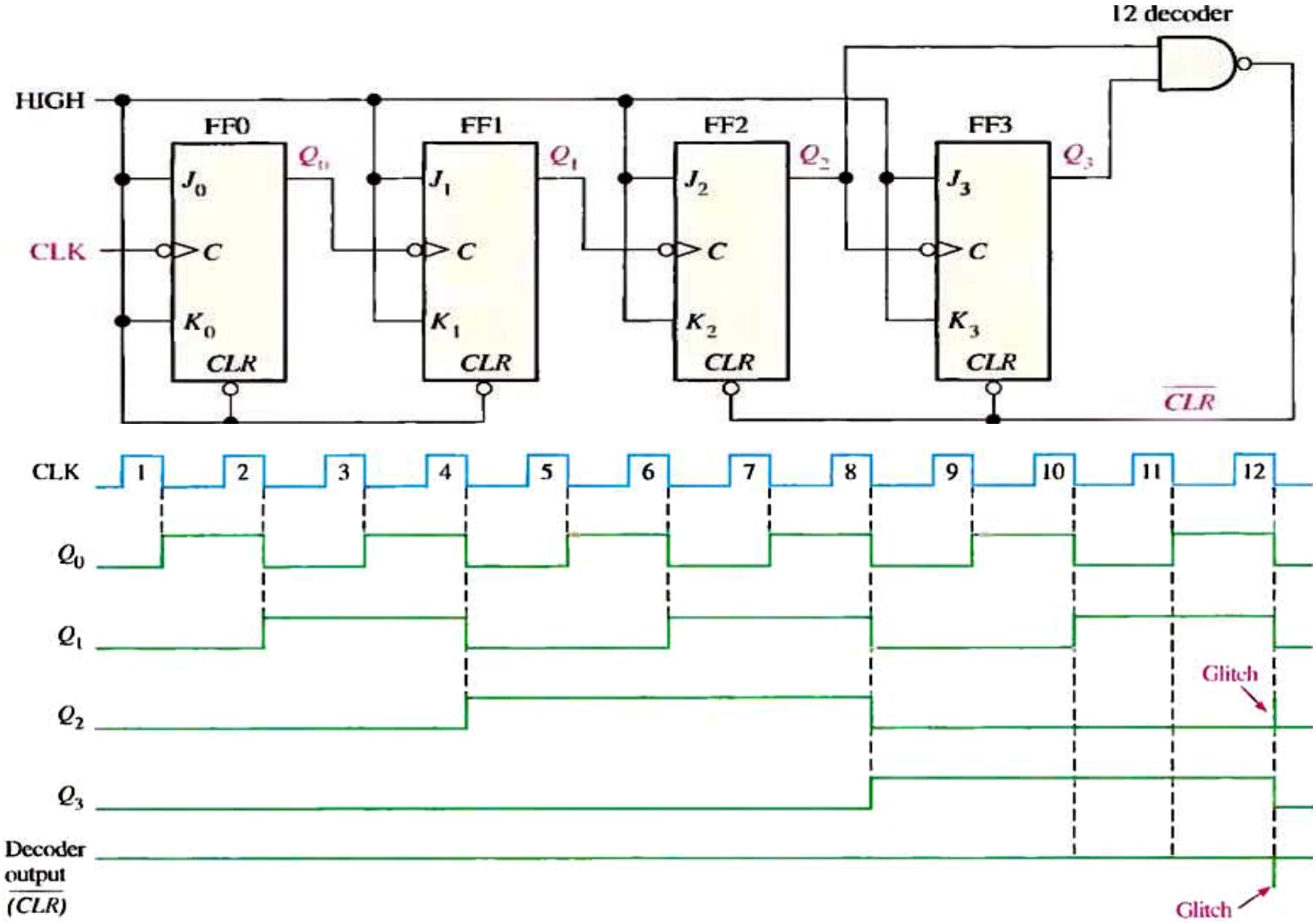
Present St.	Next St.
CBA	CBA
000	001
001	010
010	011
011	100
100	101
101	000(110)



Detect the output at CBA=110 to activate CLR. **NAND gate is used to detect outputs that generates '1'!**

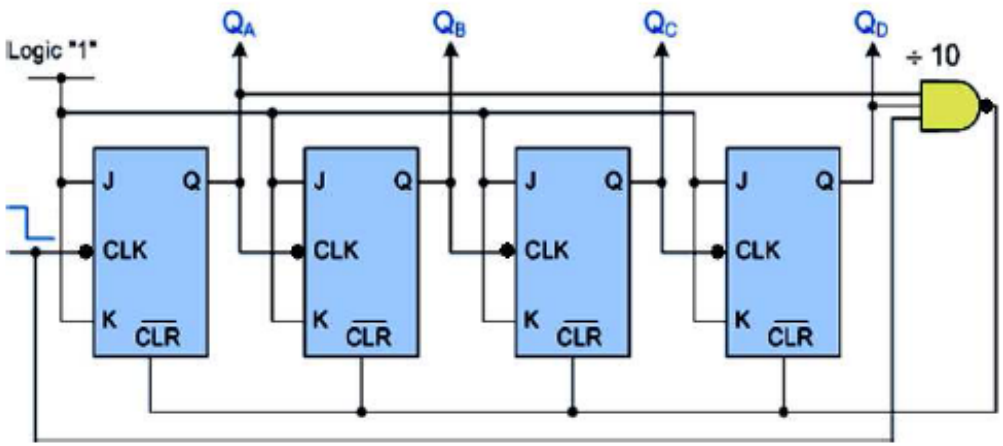
Asynchronous Counters (MOD $\neq 2^N$)

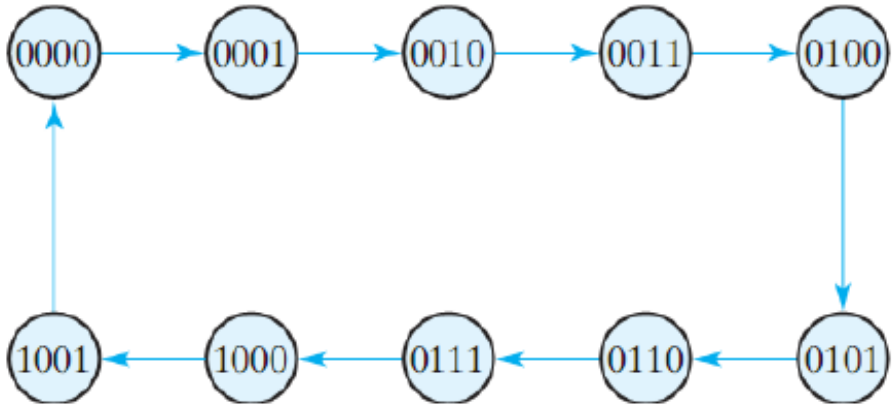
- **Example**: Show how an asynchronous counter can be implemented having a modulus of twelve with a straight binary sequence from 0000 to 1011.
- **Solution**: Four FF are required. When the counter gets to its last state 1011, it must recycle back to 0000 rather than going to its normal next state (1100)


12 decoder
HIGH
FF0
FF1
FF2
FF3
CLK
J_0
K_0
J_1
K_1
J_2
K_2
J_3
K_3
C
CLR
Q_0
Q_1
Q_2
Q_3
$\overline{CLR}$
CLK
1
2
3
4
5
6
7
8
9
10
11
12
Q_0
Q_1
Q_2
Q_3
Glitch
Decoder output $\overline{(CLR)}$
Glitch

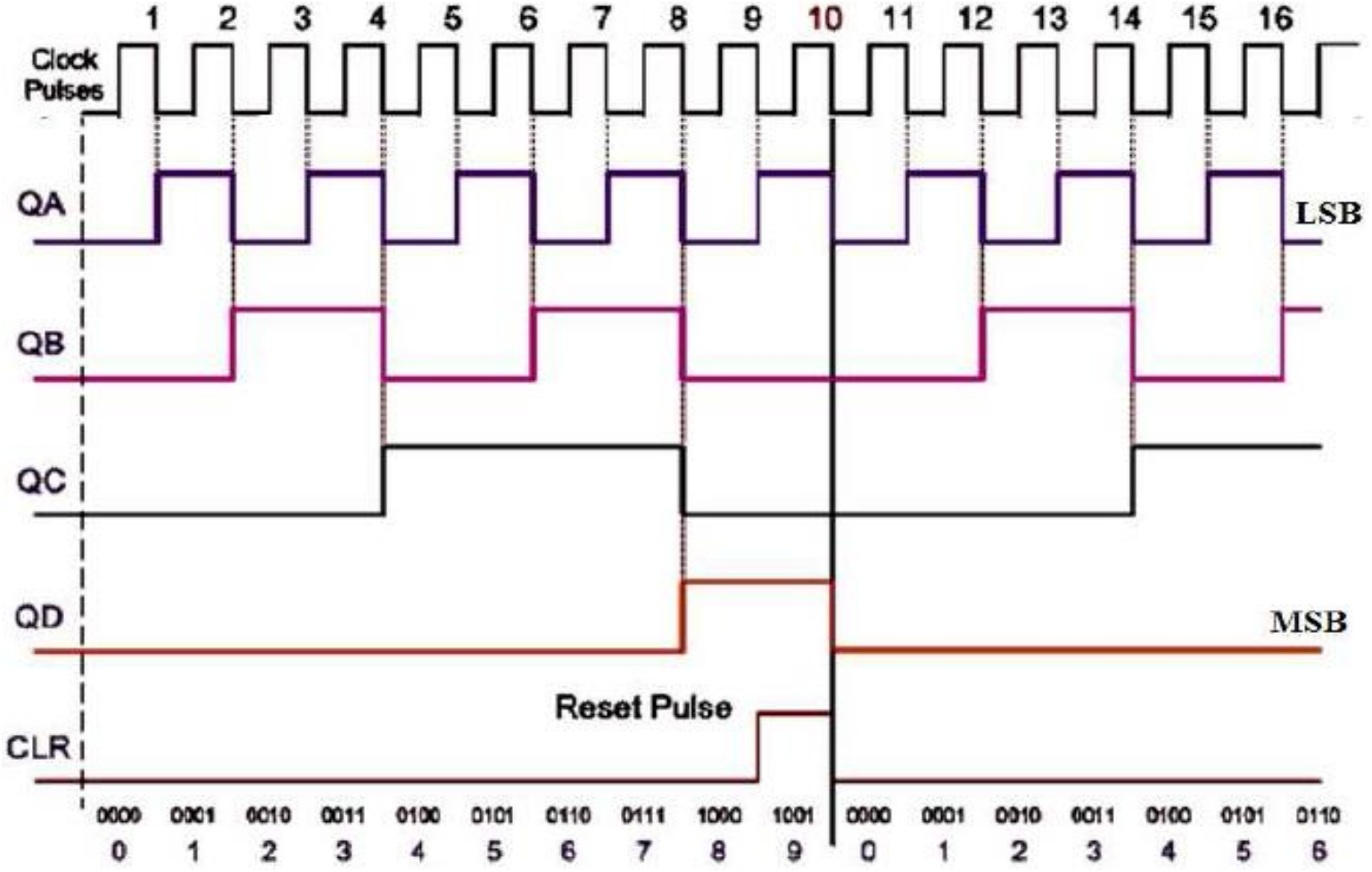
BCD Ripple Counter, Decade Counter

This counter counts upwards on each negative edge of the input clock signal starting from "0000" until it reaches an output "1001". Both outputs Q_A and Q_D are now equal to logic "1" and the output from the NAND gate changes state from logic "1" to a logic "0" level when the clock goes to level one and whose output is also connected to the CLEAR (CLR) inputs of all the J-K Flip-flops





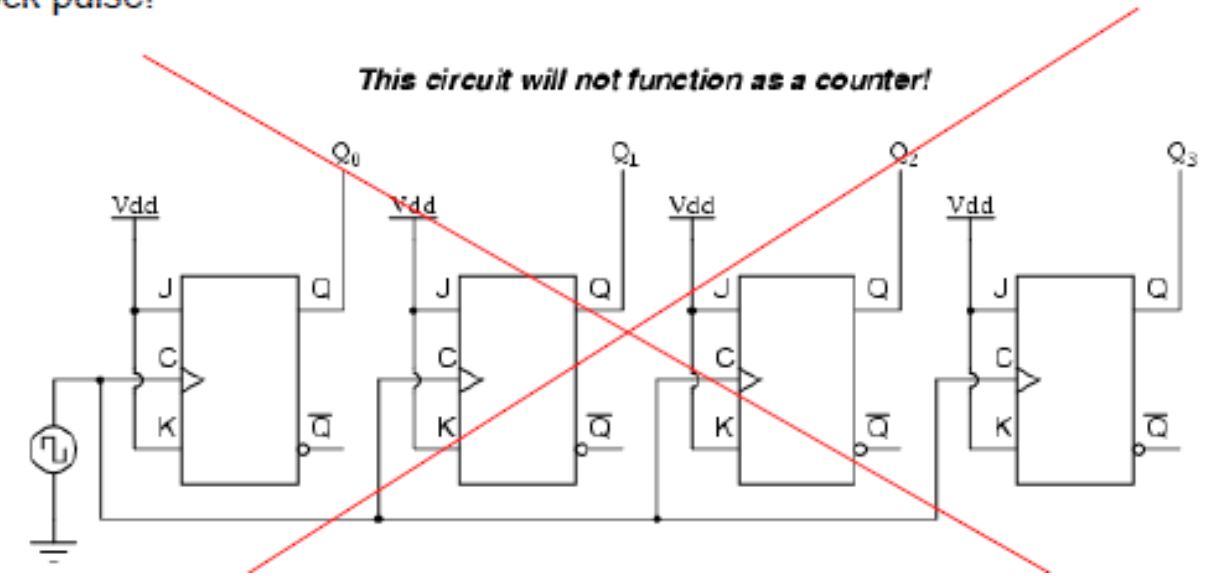
Decade Counter Timing Diagram



Synchronous Counters (Parallel)

- FFs in the counter are clocked at the same time by a **common clock pulse.**

Now, the question is, what do we do with the J and K inputs? We know that we still have to maintain the same divide-by-two frequency pattern in order to count in a binary sequence, and that this pattern is best achieved utilizing the "toggle" mode of the flip-flop, so the fact that the J and K inputs must both be (at times) "high" is clear. However, if we simply connect all the J and K inputs to the positive rail of the power supply as we did in the asynchronous circuit, this would clearly not work because all the flip-flops would toggle at the same time: with each and every clock pulse!



How To Design Synchronous Counter

- For synchronous counters, all the flip-flops are using the same CLOCK signal. Thus, the output would change synchronously.
- Procedure to design synchronous counter are as follows:-

  STEP 1: Obtain the State Diagram.

  STEP 2: Obtain the Excitation Table using state transition table for any particular FF (JK or D). Determine number of FF used.

  STEP 3: Obtain and simplify the function of each FF input using K-Map.

  STEP 4: Draw the circuit.

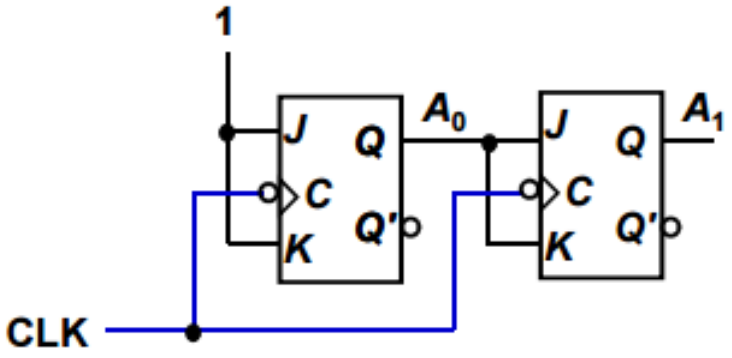
Synchronous (Parallel) Counters

- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

$TA_1 = A_0$

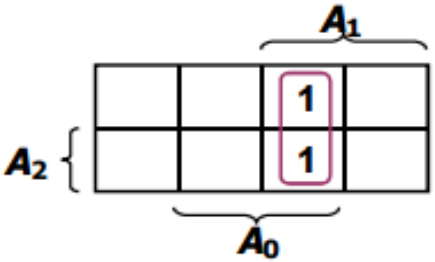
$TA_0 = 1$



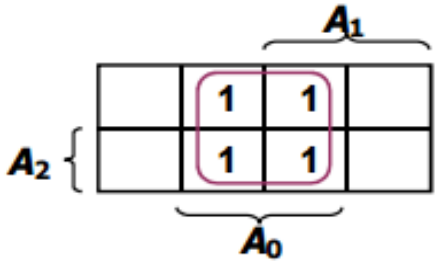
Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

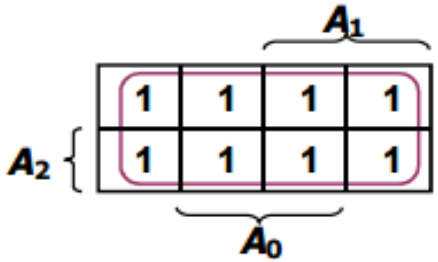
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$



$TA_1 = A_0$

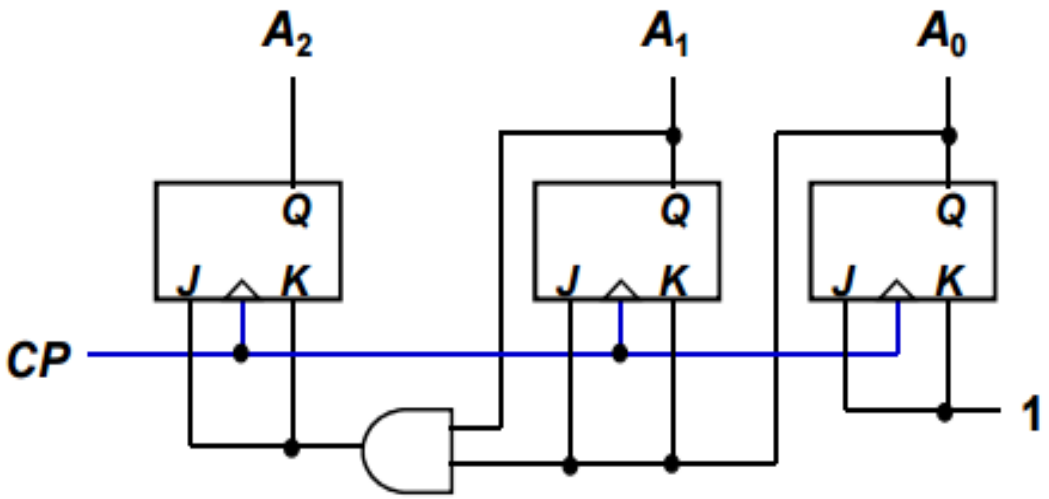


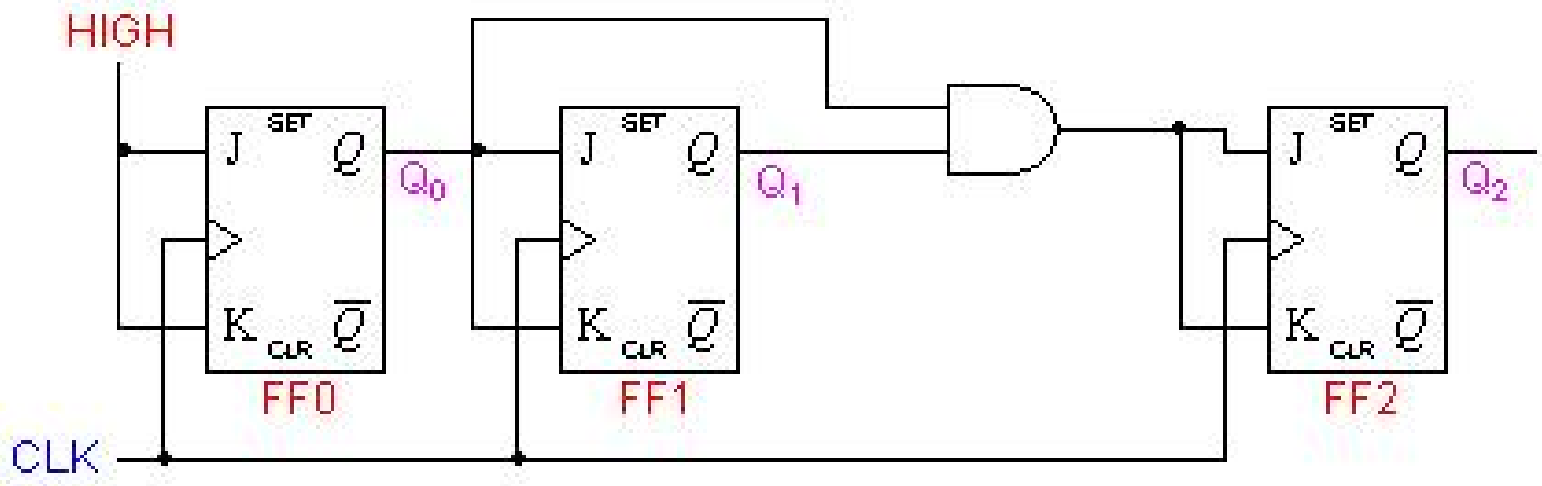
$TA_0 = 1$

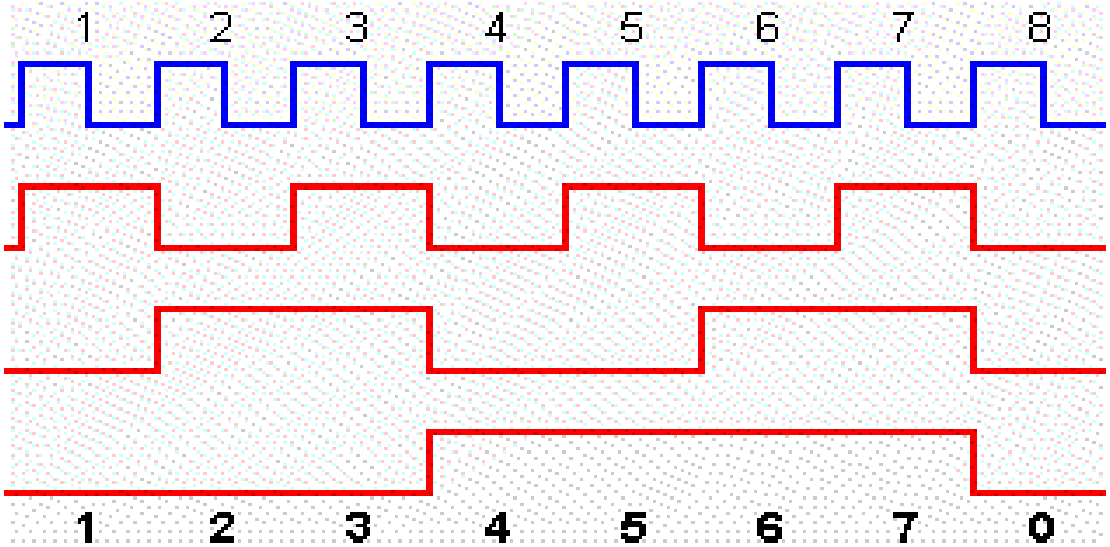
Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (cont'd).

$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$




HIGH
SET
J
Q
K
CLR
$\overline{Q}$
Q_0
FF0
Q_1
FF1
Q_2
FF2
CLK


1
2
3
4
5
6
7
8
1
2
3
4
5
6
7
0

Synchronous (Parallel) Counters

- Example: 4-bit synchronous binary counter.

$$TA_3 = A_2 \cdot A_1 \cdot A_0$$
$$TA_2 = A_1 \cdot A_0$$
$$TA_1 = A_0$$
$$TA_0 = 1$$