Next Lecture :

Sequential Circuits

# Asynchronous / Ripple Counters


GO CLASSES
GATE OVERFLOW
GO Test Series
is now
GATE Overflow + GO Classes
2-IN-1 TEST SERIES
Most Awaited
GO Test Series
is Here
REGISTER NOW
http://tests.gatecse.in/
100+ More than 100 Quality Tests.
15 Mock Tests.
FROM
14th April
+91 - 6302536274
+91 9499453136
GO CLASSES
GATE OVERFLOW
LIVE
GATE 2023
Live + Recorded Lectures
Daily Home Work + Solution
Watch Any Time + Any Number of Times
Summary Lectures For Every Topic
Practice Sets From Standard Resources
Enroll Now
+91 - 6302536274
www.goclasses.in
linkedin.com/company/go-classes
instagram.com/goclasses_cs

Join **GO+ GO Classes Combined Test Series** for BEST quality tests, matching GATE CSE Level:

Visit www.gateoverflow.in website to join Test Series.

1. **Quality Questions:** No Ambiguity in Questions, All Well-framed questions.
2. Correct, **Detailed Explanation**, Covering Variations of questions.
3. **Video Solutions.**

# NOTE :

**Complete** Discrete Mathematics & C-Programming Courses,

by GO Classes, are **FREE** for ALL learners.

Visit here to watch : https://www.goclasses.in/s/store/

SignUp/Login on Goclasses website for free and start learning.

Join GO Classes Resources, Notes, Content, information Telegram Channel:

Public Username: goclasses_cse

Join GO Classes **Doubt Discussion** Telegram Group :

Username: GATECSE_Goclasses

(Any doubt related to Goclasses Courses can also be asked here.)

Join GATEOverflow Doubt Discussion Telegram Group :

Username: gateoverflow_cse

# Sequential Circuits:

1. Definition ✓
2. Memory element: ✓
   SR latch ; $\bar{S}\bar{R}$ latch
3. flipflop → 4 ffs (SR, JK, T, D)
   - Toggle
   - Data store
   - J≡S ; K≡R

## SR latch: (SR latch with NOR)

| S | R | $Q_n$ (next output/state) | $\overline{Q}_n$ |
|---|---|---|---|
| 0 | 0 | Q (present output/state) | $\overline{Q}$ |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 (forbidden) |

**Why S = R = 1 forbidden:**

11 ⟶ 00

↳ this Transition causes Indeterminate behaviour

# $\overline{S}\overline{R}$ latch:

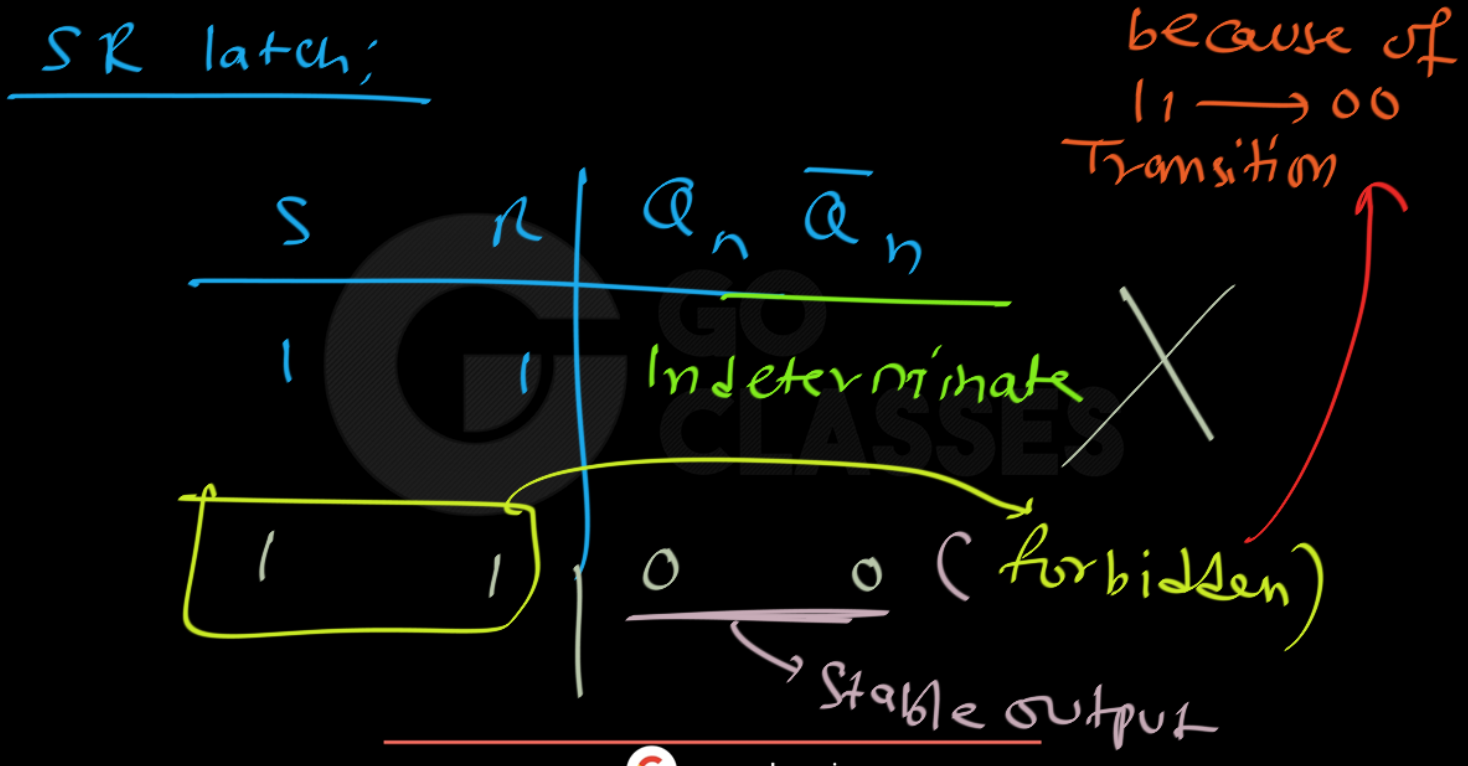
(SR latch with MAKIN)

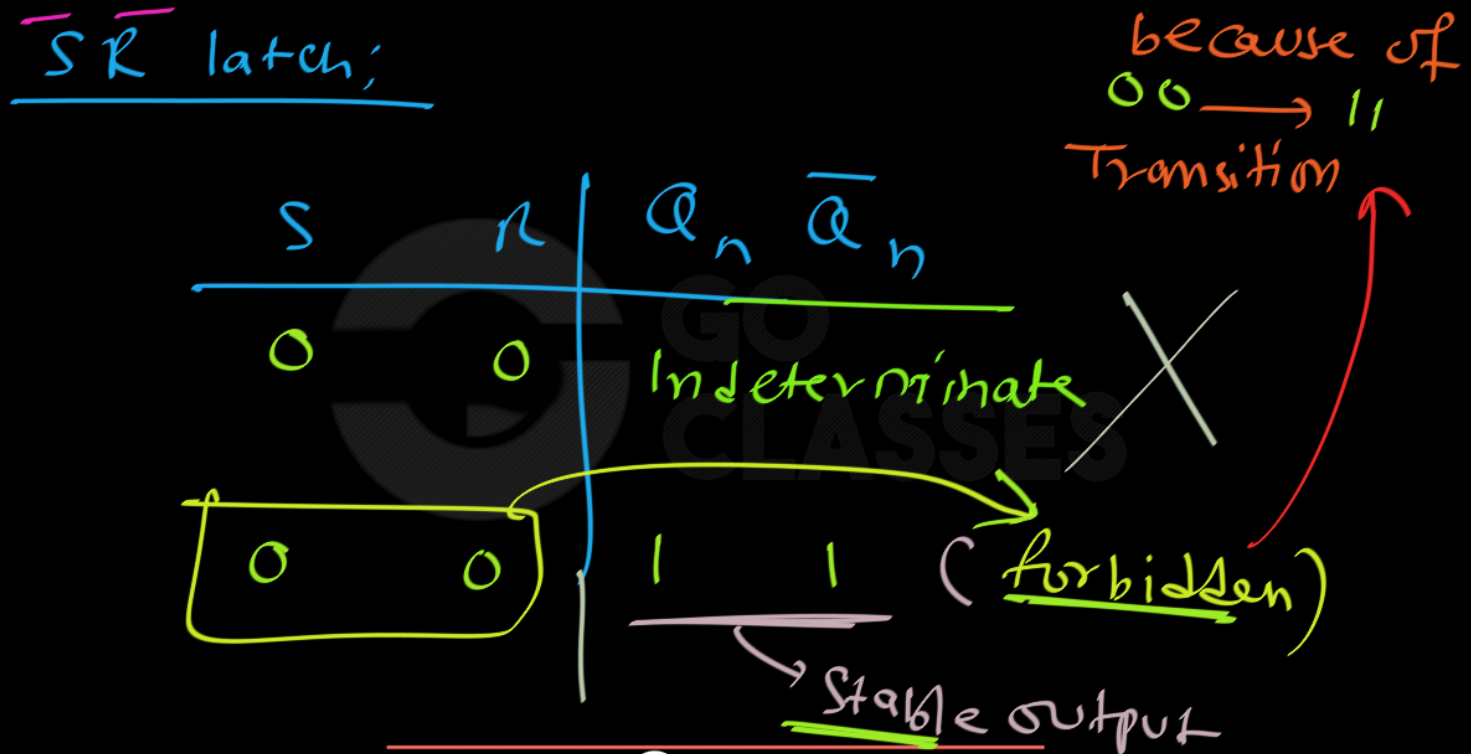
| S | R | $Q_n$ (next output/state) | $\overline{Q}_n$ |
|---|---|---|---|
| 1 | 1 | Q (present output/state) | $\overline{Q}$ |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 (forbidden) |

Why SR = 00 forbidden:

00 ⟶ 11

↳ this Transition causes Indeterminate behaviour


SR latch;
because of
11 ⟶ 00
Transition
S
R
$Q_n$
$\bar{Q}_n$
1
1
Indeterminate
1
1
0
0
(forbidden)
Stable output


$\overline{S}\,\overline{R}$ latch;
because of
00 → 11
Transition
S
R
$Q_n$ $\overline{Q}_n$
0 0 Indeterminate
0 0 1 1 (forbidden)
Stable output

# Sequential ckts:

4. Clock terminology
5. Timing Diagrams
6. Flipflop Triggering mechanisms
   - by default ⇒ Edge Triggered
7. ff Conversion

(8) Master Slave ff ≡ -Ve Edge Triggered ff

(9) Race Conditions in JK, T

↳ (1) level Triggered and
(2) Toggle Condition

10 Timing Constraints — Hold time, setup ti, prop. Delay

11 Registers

12 FSM — Mealy, Moore

13 Counters

- Synchronous — next lecture
- Asynch — this lecture

# Asynchronous / Ripple Counters

# Recap:

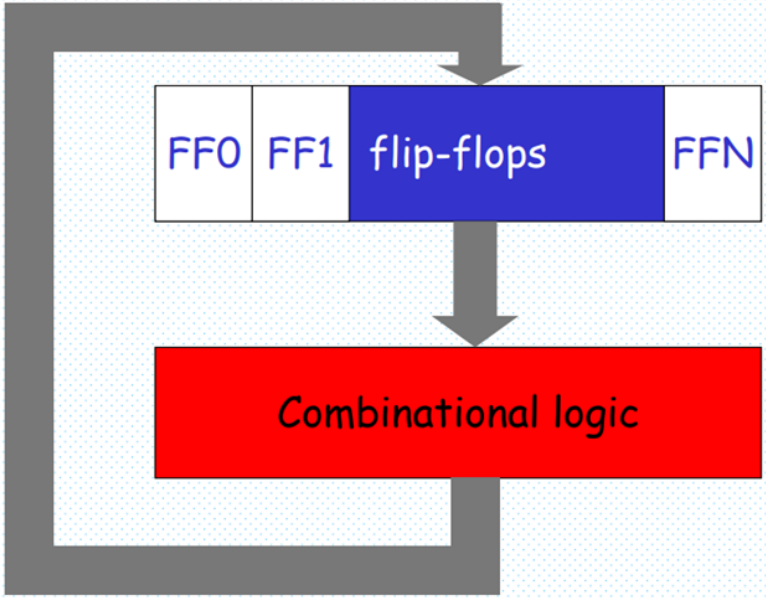
## Registers, Counters

# Registers

- Registers like counters are clocked sequential circuits
- A register is a group of flip-flops
  - Each flip-flop capable of storing one bit of information
  - An n-bit register
    - consists of n flip-flops
    - capable of storing n bits of information
  - besides flip-flops, a register usually contains combinational logic to perform some simple tasks
  - In summary
    - flip-flops to hold information
    - combinational logic to control the state transition

# Counters

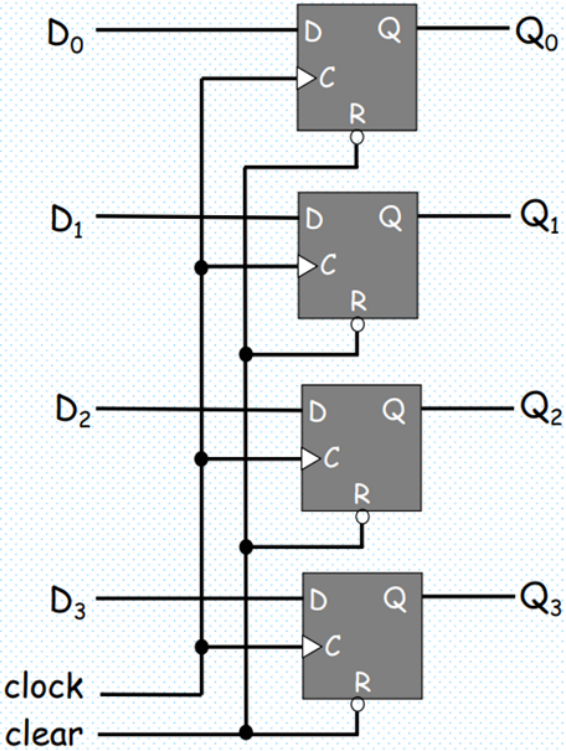
- A counter is essentially a register that goes through a predetermined sequence of states

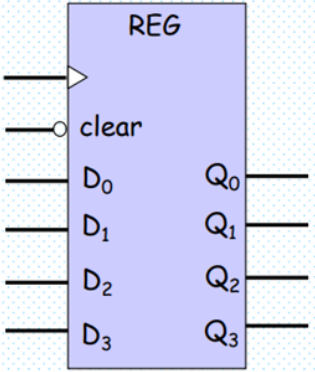


# Uses of Registers and Counters

- Registers are useful for storing and manipulating information
  - internal registers in microprocessors to manipulate data
- Counters are extensively used in control logic
  - PC (program counter) in microprocessors

# 4-bit Register



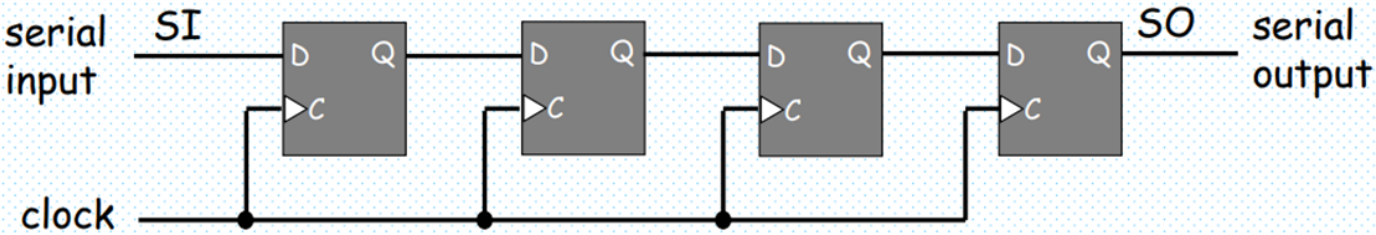




PIPO

Not Shift Register

# Shift Registers

- A register capable of shifting its information in one or both directions
  - Flip-flops in cascade

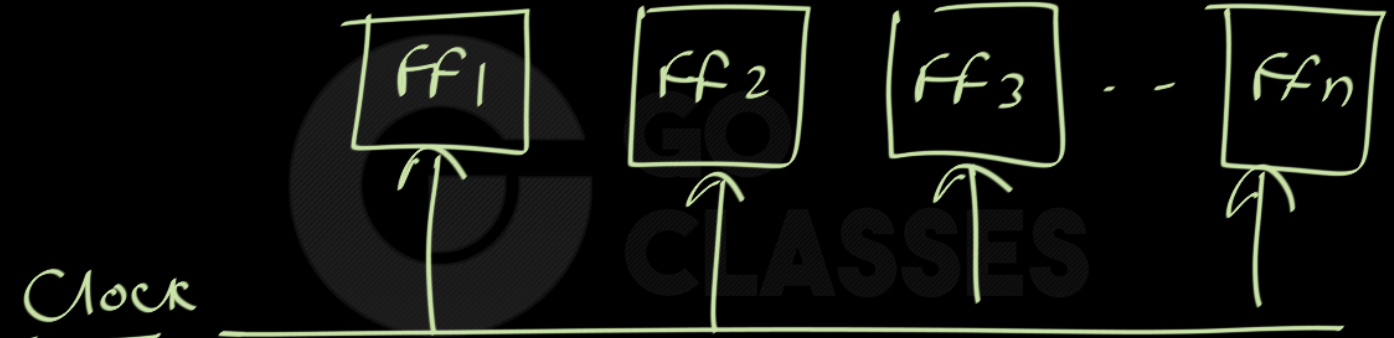



- The current state can be output in n clock cycles

# Counters

- A counter is basically a register that goes through a prescribed sequence of states upon the application of input pulses
  - input pulses are usually clock pulses
- Example: n-bit binary counter
  - count in binary from 0 to $2^n$-1
- Classification
  1. Ripple counters
     - flip-flop output transition serves as *the* pulse to trigger other flip-flops
  2. Synchronous counters
     - flip-flops receive the same common clock as *the* pulse

Every Register:



# Recap:
# Asynchronous Vs Synchronous Counters

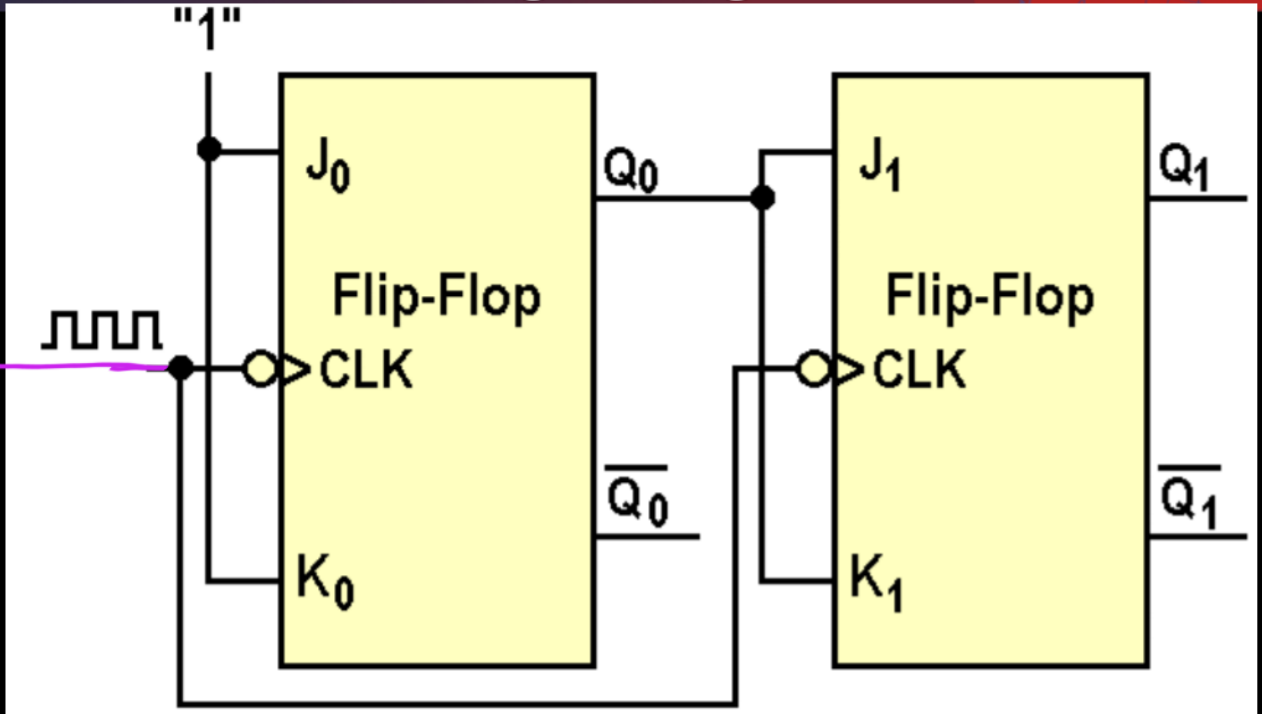



**Figure 9.1:** A modulus 4 2-bit synchronous counter

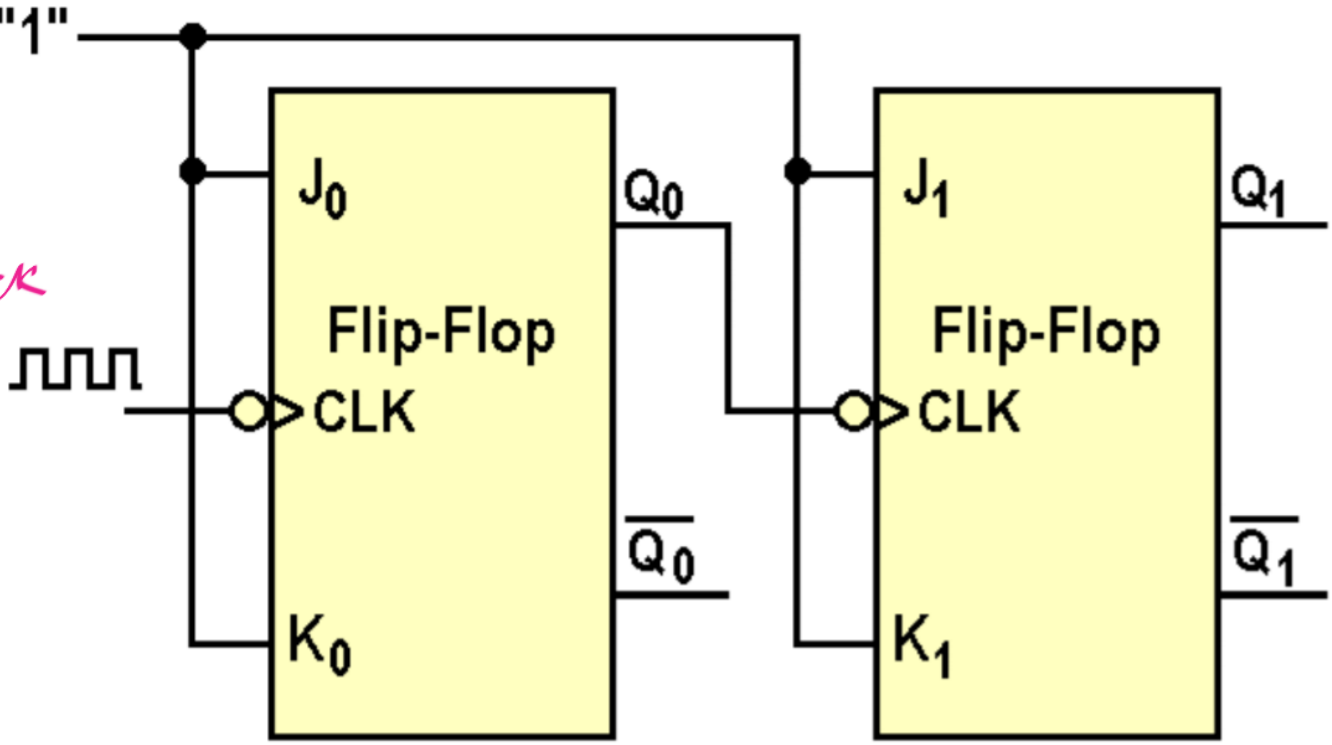



**Figure 9.2:** A modulus 4 2-bit asynchrounous counter

# Recap:

# Applications of FlipFlops

# FlipFlop

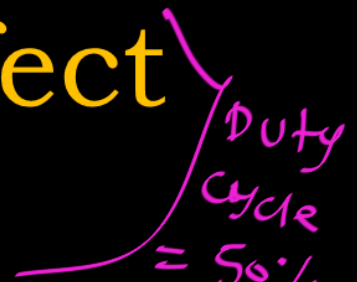
- The memory elements used in clocked sequential circuits are called flipflops. These circuits are binary cells capable of storing one bit of information.
- A flipflop circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit)until directed by an input signal to switch states
- A bi stable device
- Have two outputs one complement of another
- Applications of Flipflops
  - Counters
  - Shift Registers
  - Storage Registers ≡ Register
  - Frequency Dividers

# Recap:

## Applications of FlipFlops

# Frequency Division

## & Producing Perfect Square Wave



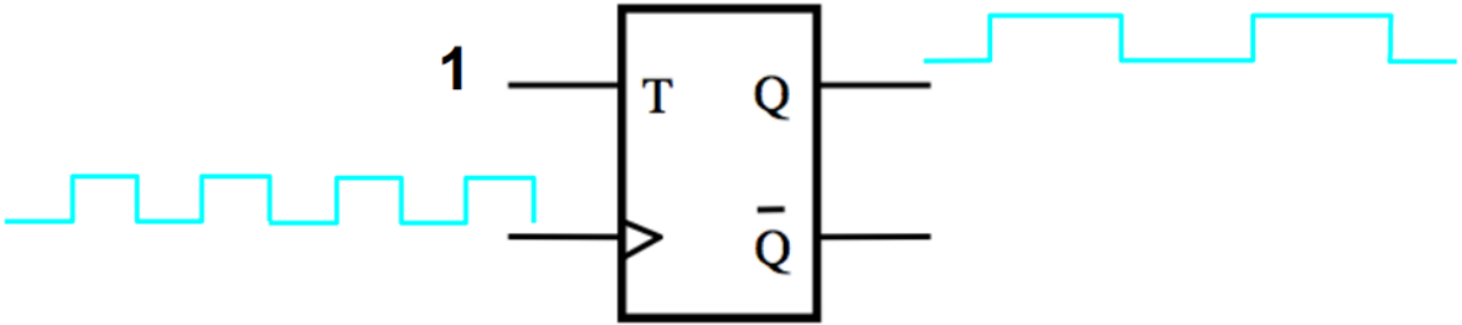
Q :

Given a clock signal of frequency f Hz, how to generate another clock of frequency f/2 Hz. i.e. Frequency division circuit.

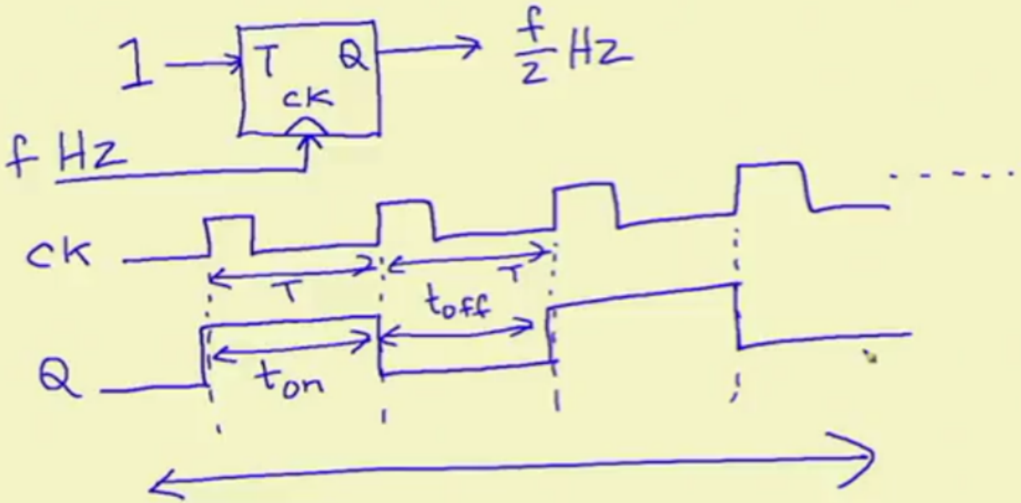
Per second

f Hz ≡ f oscillations per second

# The output of the T Flip-Flop
## divides the frequency of the clock by 2



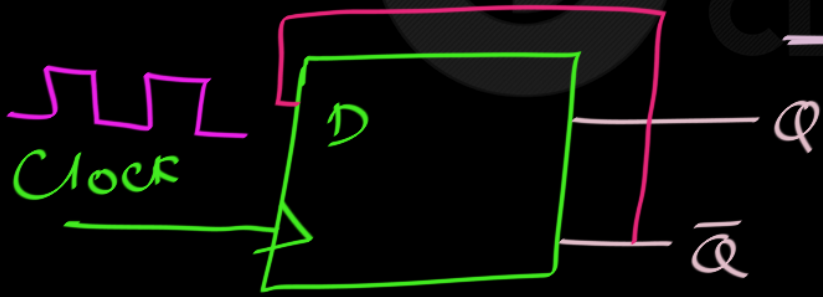
- Given a clock signal of frequency *f Hz*, how to generate another clock of frequency *f/2 Hz*.
  - Frequency division.



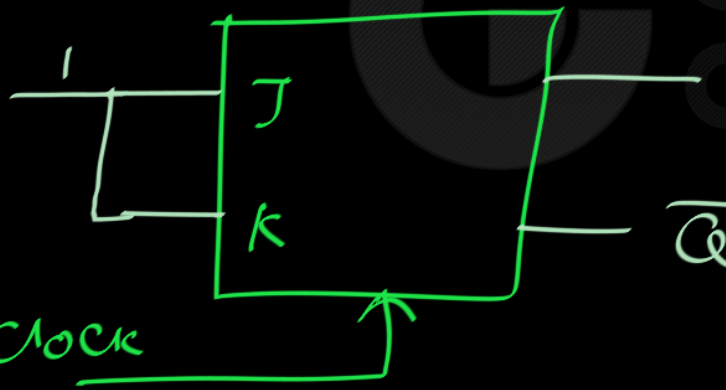
NOTE :

The output of the T Flip-Flop divides the frequency of the clock by 2, Hence, T-flipflop is also called Divide by 2 device.

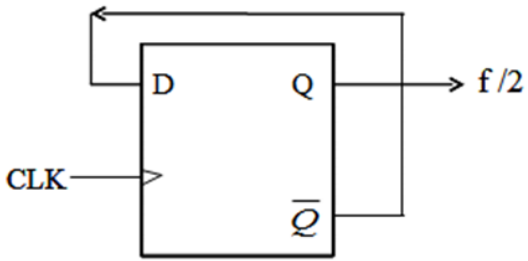
Q : Can we use D-ff for frequency Division? Yes. (but D-ff will uses as Toggle-ff)

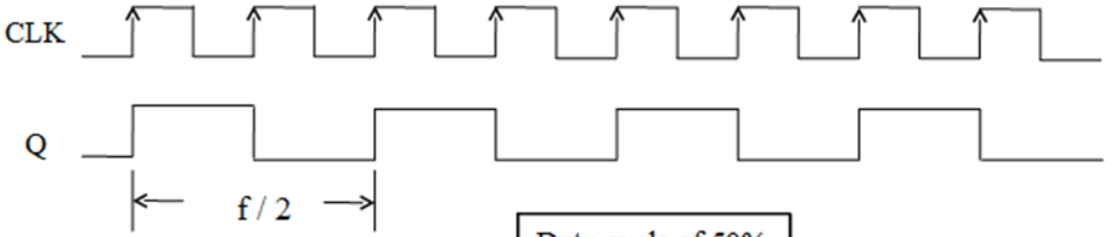


Q : Can we use JK-ff for frequency Division? Yes. (Toggle feature of JK used)

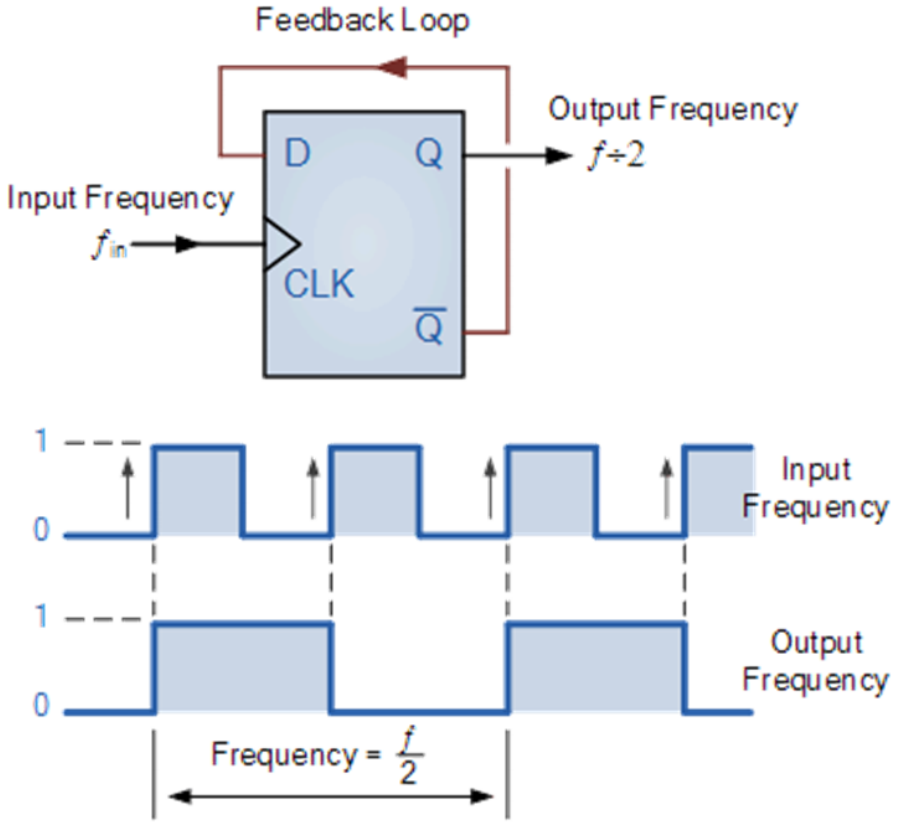


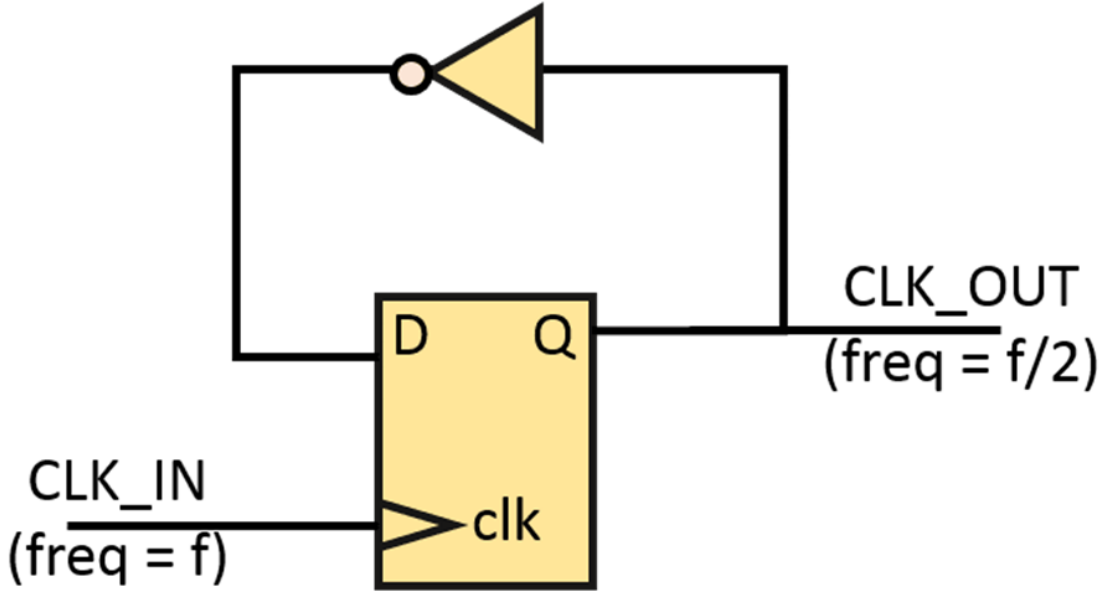
# Frequency Divide by 2 (f/2)

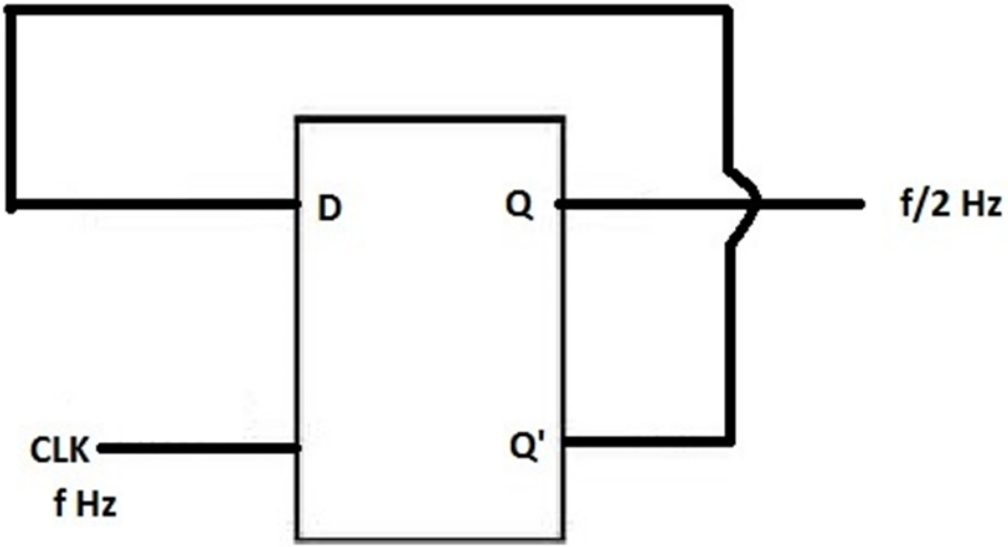



Duty cycle of 50%
(ON- 1, OFF- 1)

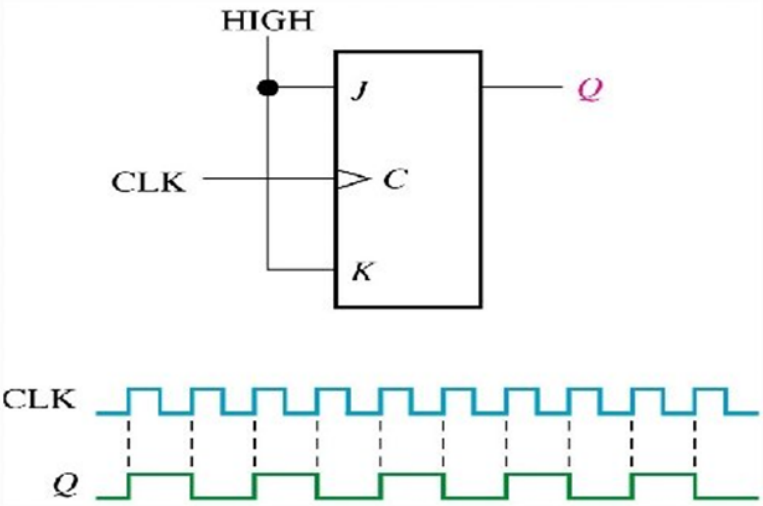

Feedback Loop
Output Frequency
$f \div 2$
D
Q
Input Frequency
$f_{in}$
CLK
$\overline{Q}$
1
0
Input Frequency
1
0
Output Frequency
Frequency = $\frac{f}{2}$


D
Q
clk
CLK_OUT
(freq = f/2)
CLK_IN
(freq = f)




FREQUENCY DIVIDER USING D FLIP FLOP

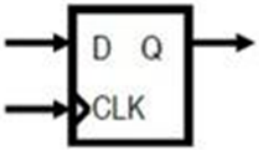
# J-K flip-flop as a divide-by-2 device. Q is one-half the frequency of CLK.



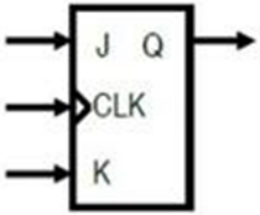
Build a frequency divider, divide-by-2 and divide-by-4 circuits using

1. D Flip Flops
2. JK Flip Flops

D Flip-Flop



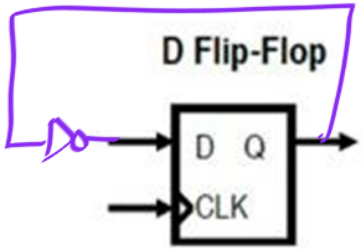

JK Flip-Flop

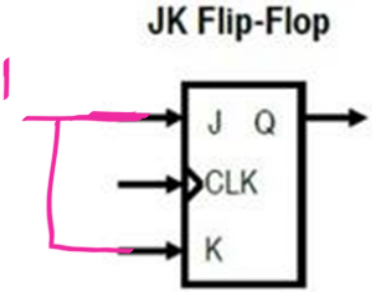



You will build four circuits in total. Draw the truth table and diagram of this design and observe the output waveform.

Build a frequency divider, divide-by-2 and divide-by-4 circuits using

1. D Flip Flops
2. JK Flip Flops

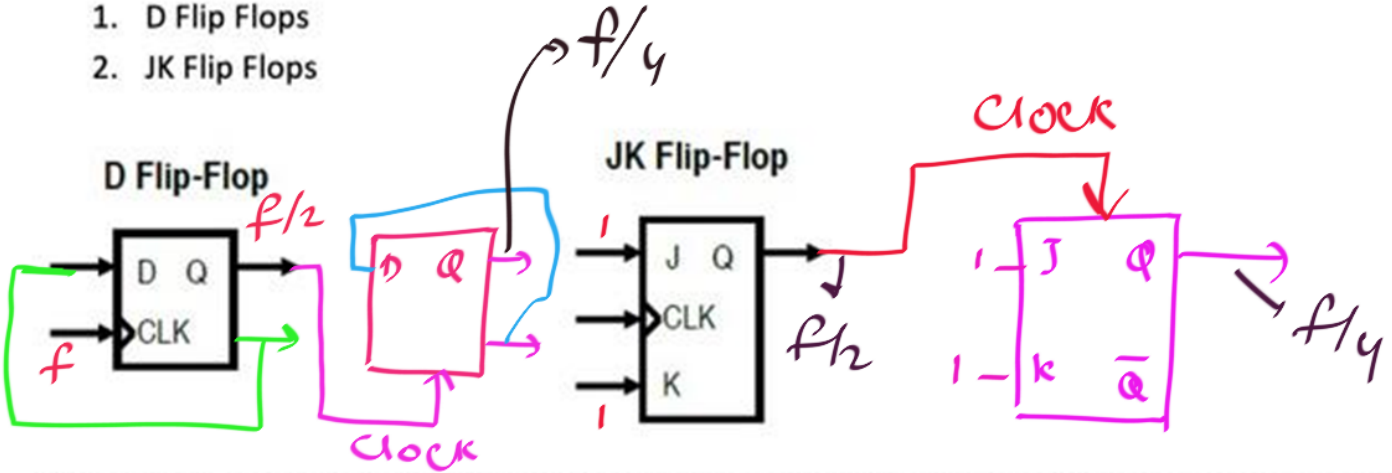



You will build four circuits in total. Draw the truth table and diagram of this design and observe the output waveform.

Build a frequency divider, divide-by-2 and divide-by-4 circuits using

1. D Flip Flops
2. JK Flip Flops




You will build four circuits in total. Draw the truth table and diagram of this design and observe the output waveform.

NOTE :

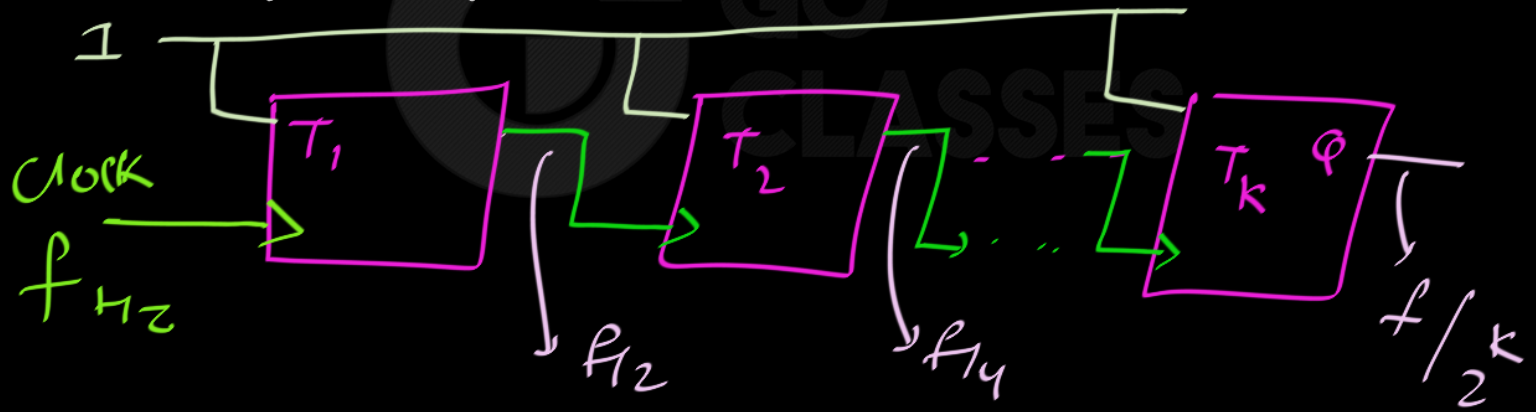
The output of the T Flip-Flop divides the frequency of the clock by 2, Hence, T-flipflop is also called Divide by 2 device.

Q :

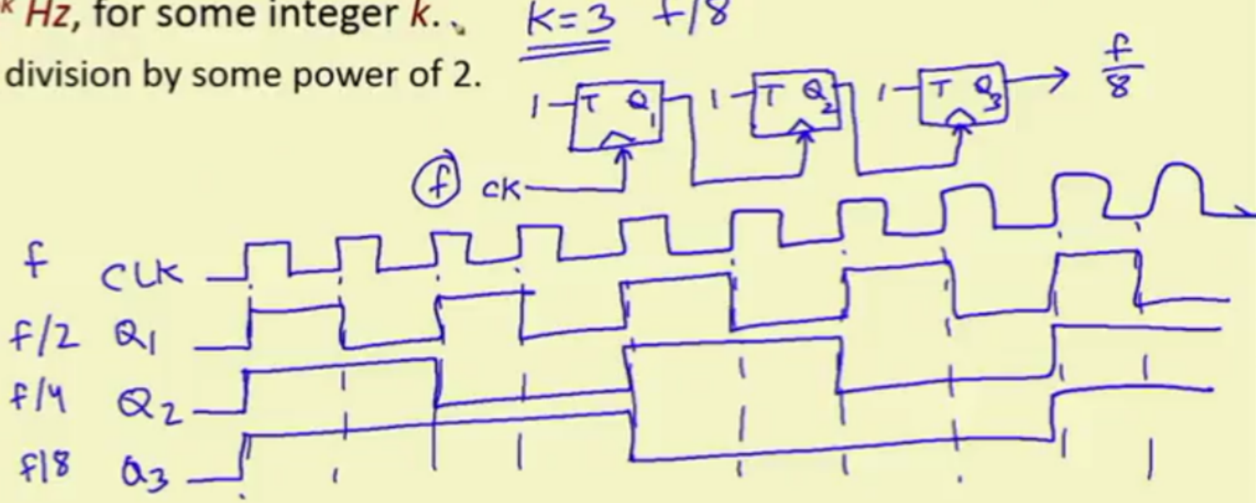
Given a clock signal of frequency f Hz, how to generate another clock of frequency f/2^k Hz. i.e. Frequency division by some power of 2.

Q :

Given a clock signal of frequency f Hz, how to generate another clock of frequency f/2^k Hz. i.e. Frequency division by some power of 2.



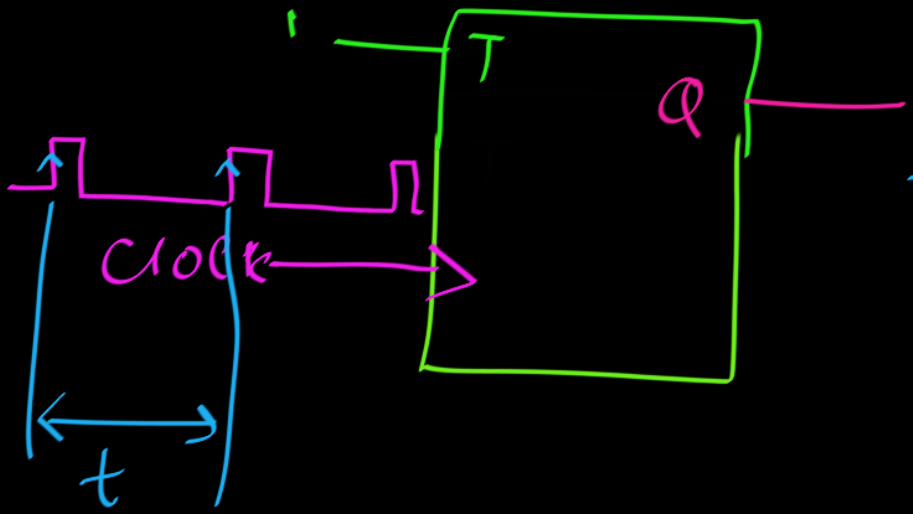
- Given a clock signal of frequency $f$ Hz, how to generate another clock of frequency $f/2^k$ Hz, for some integer $k$.
  - Frequency division by some power of 2.



# flip flops

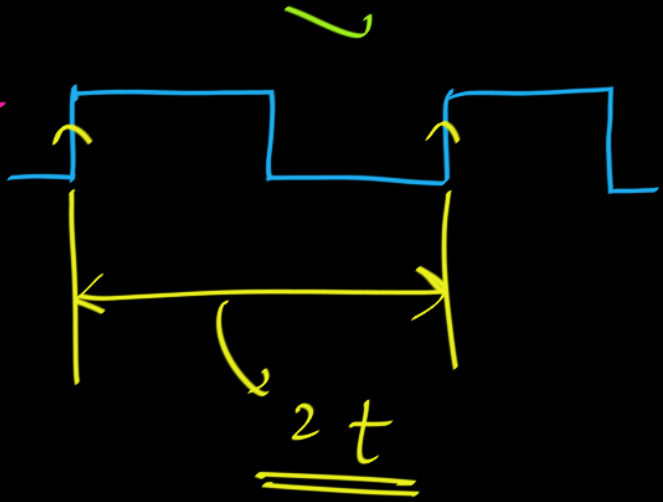
Application: ① frequency Divisor

② produce Perfect Square wave (Duty cycle = 50%)




time Period = t

f = 1/t




freq : $\frac{1}{2t} = \frac{f}{2}$

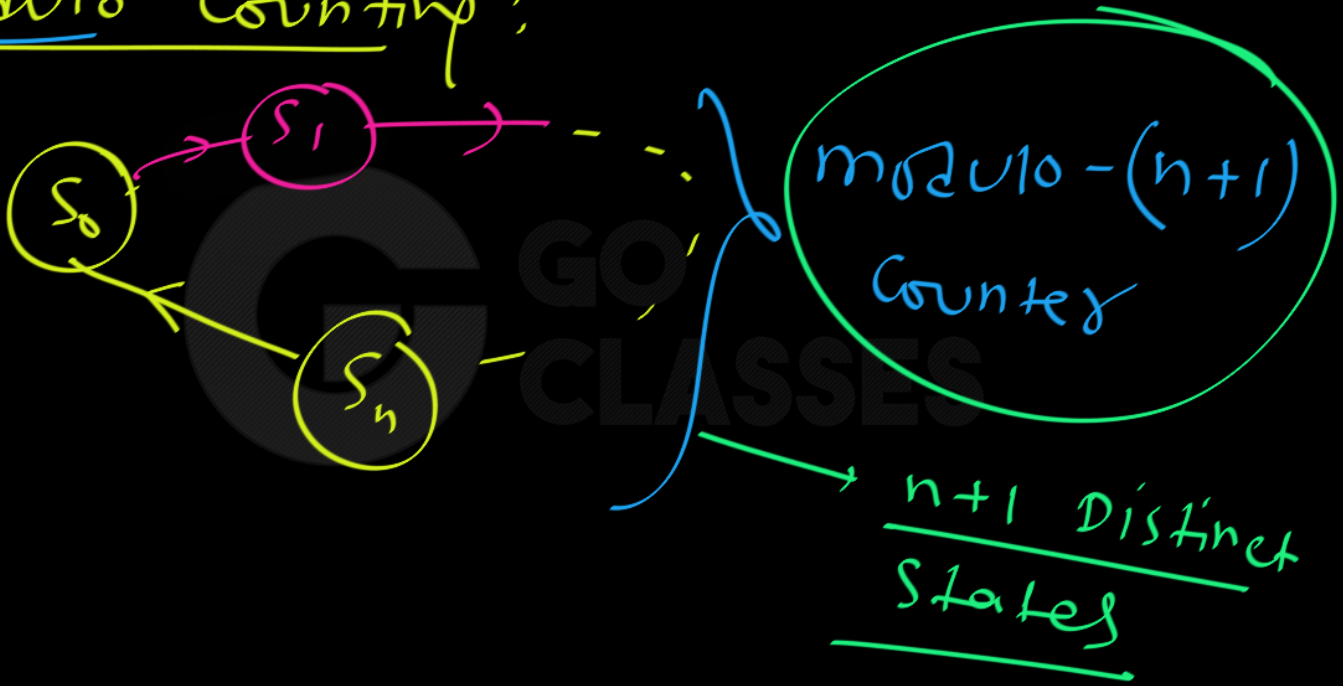
# Recap:

## Applications of FlipFlops

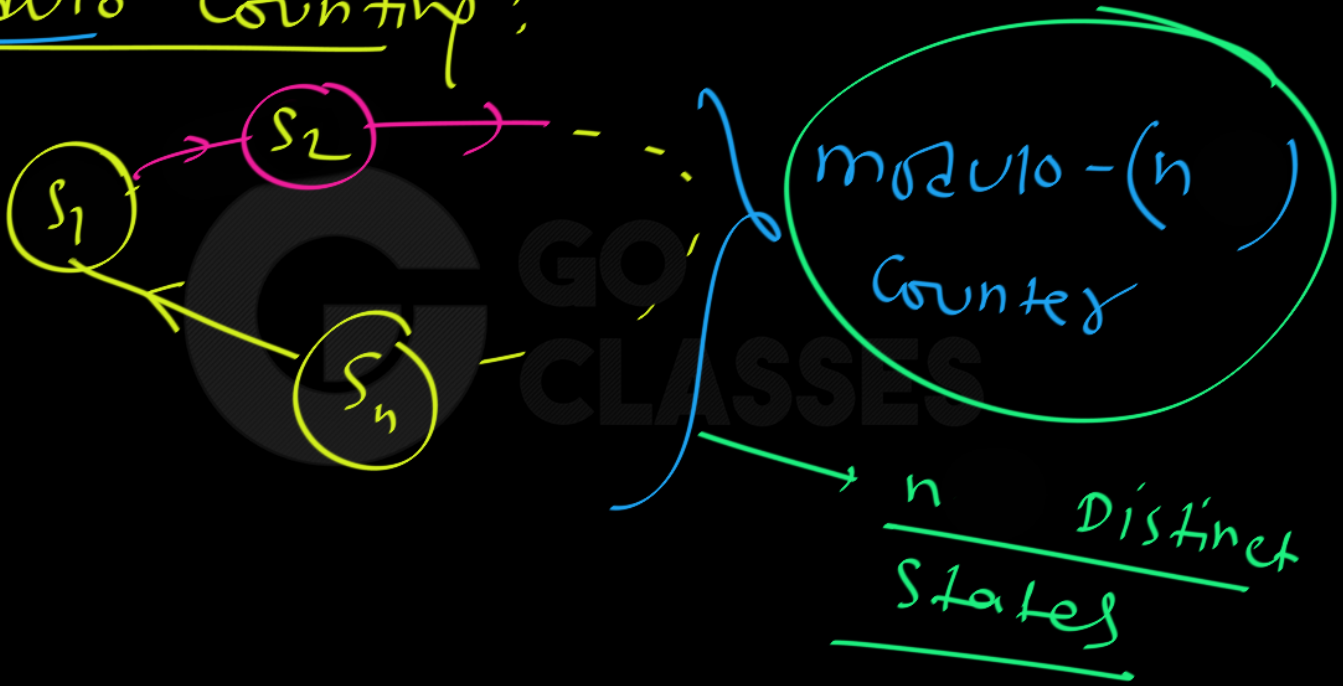
## (Modulo)Counting

## Asynchronous Counters

## Modulo Counting :

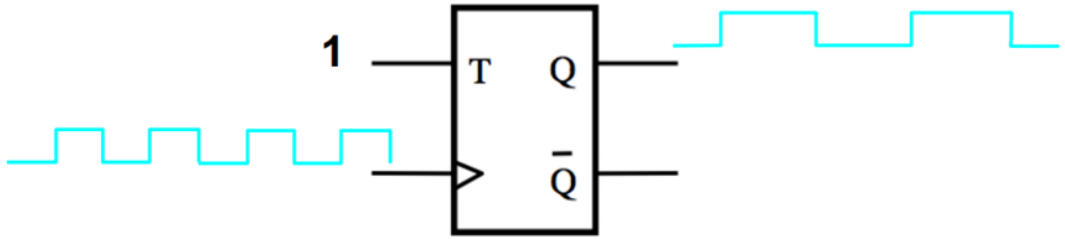


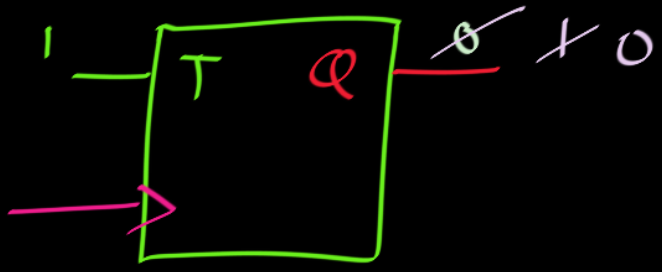
## Modulo Counting :



So, clearly there's some division going on.
But how do we interpret this as counting?

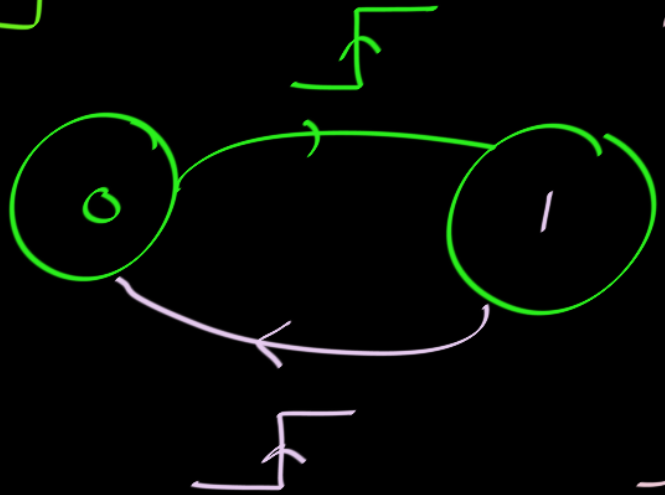
## The output of the T Flip-Flop
### divides the frequency of the clock by 2






output ≡ State

+ve Edge Triggered




Modulo-2 Counter

So, the previous circuit is : → T-ff

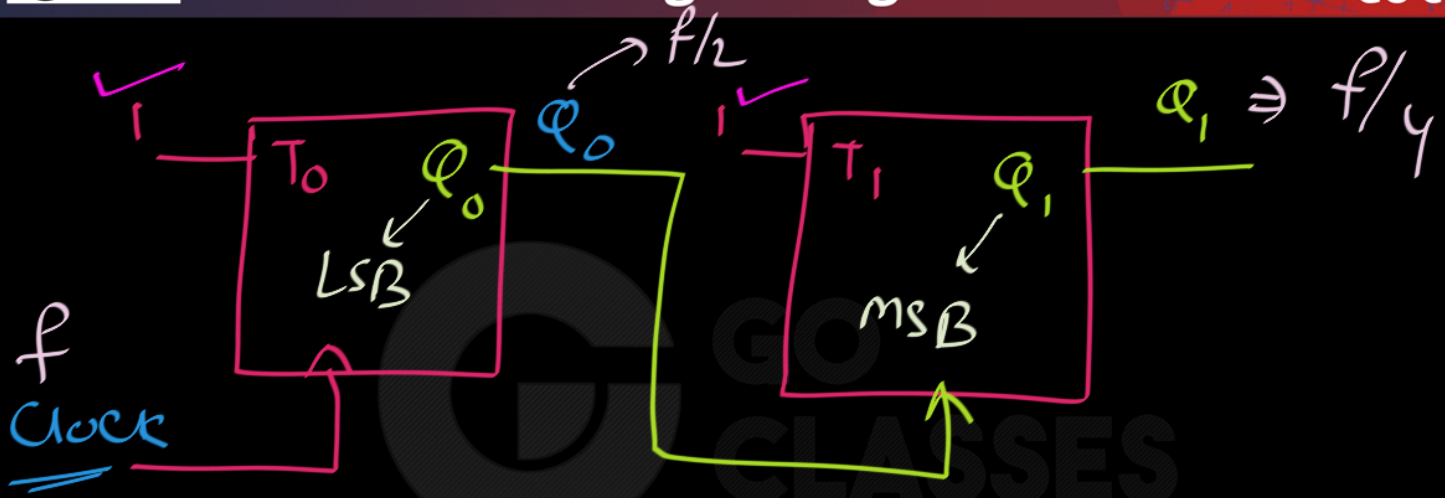
Frequency Divider by 2

Modulo 2 Counter

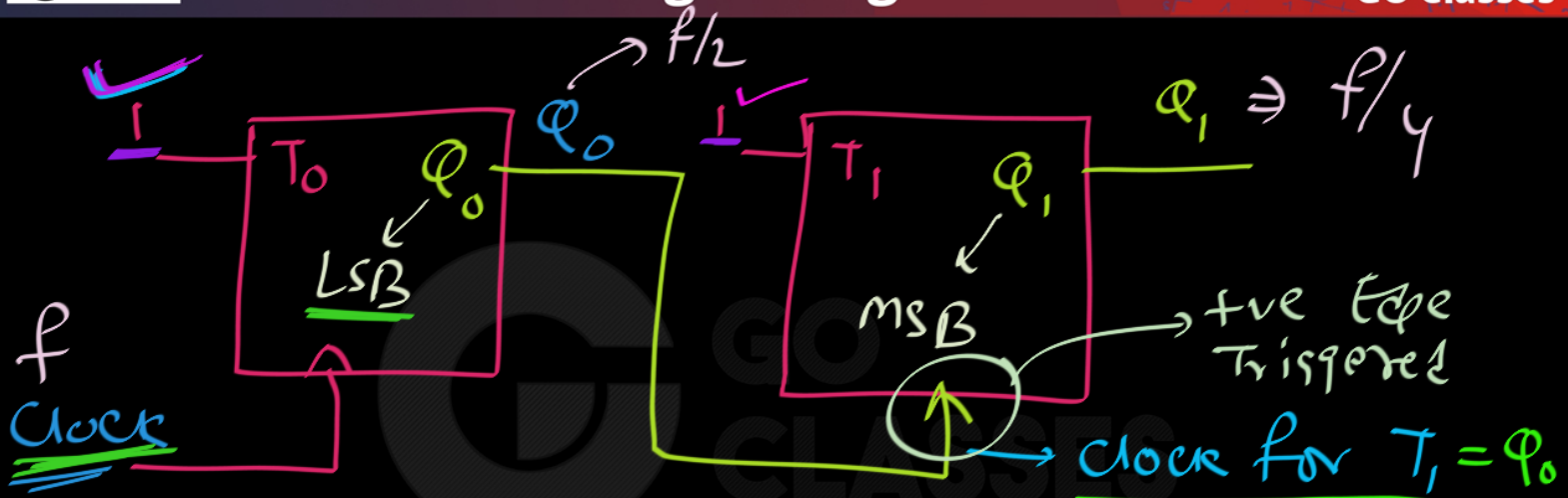
Divide-by-2 Counter

→ frequency Divide by 2

Modulus of a counter is defined as the number of unique states that a counter will sequence through.

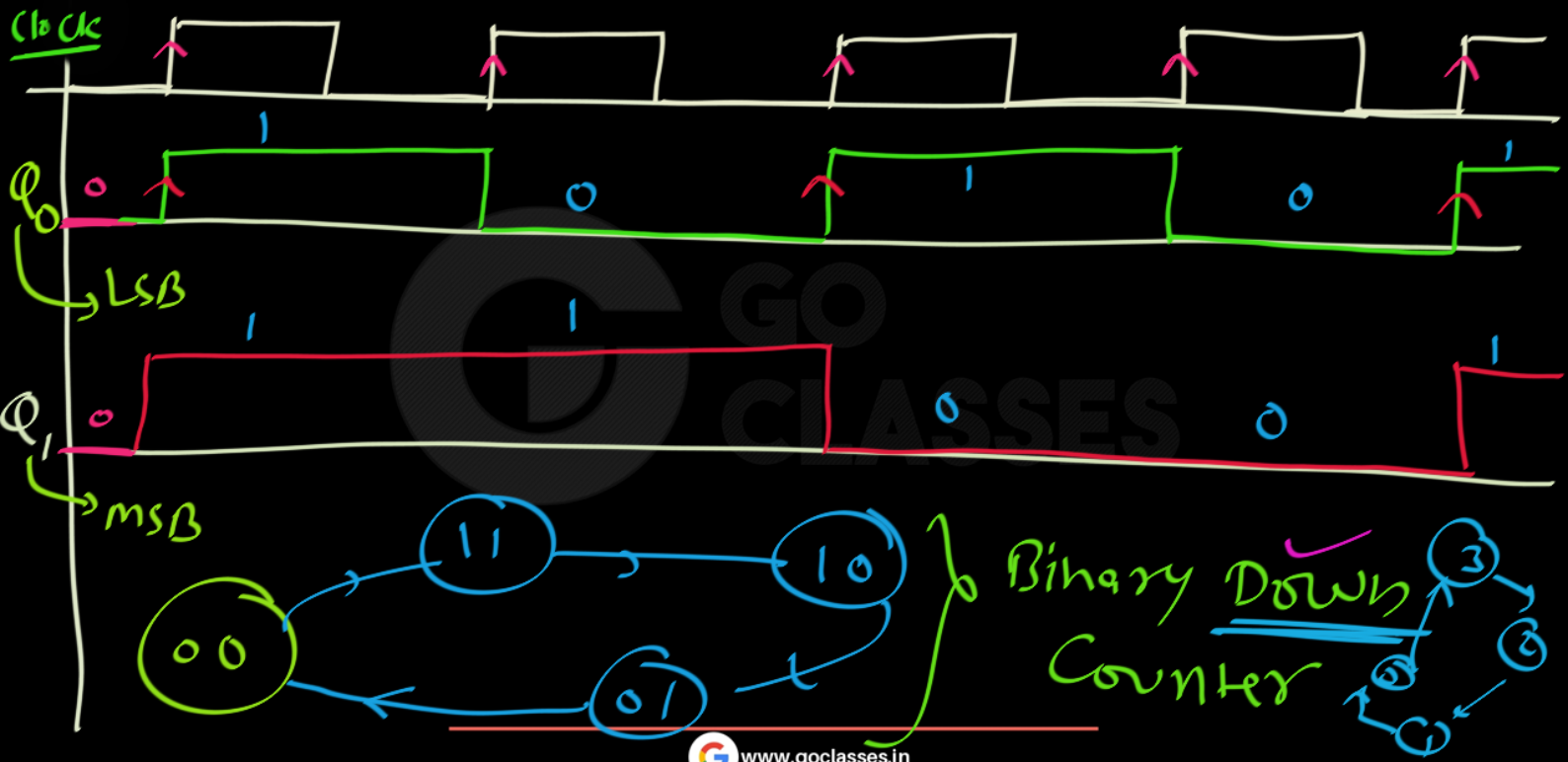


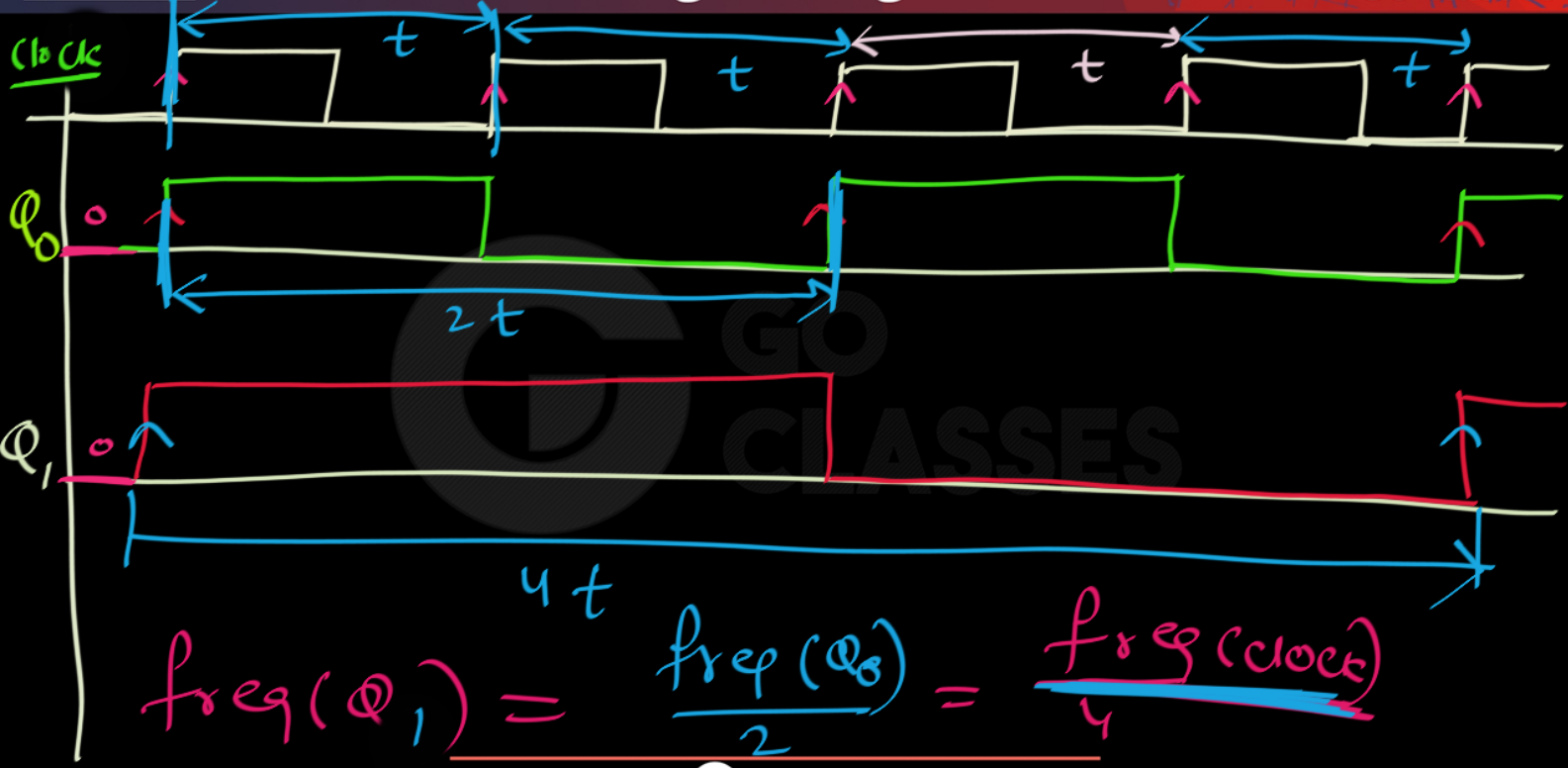

find the Counting Sequence?



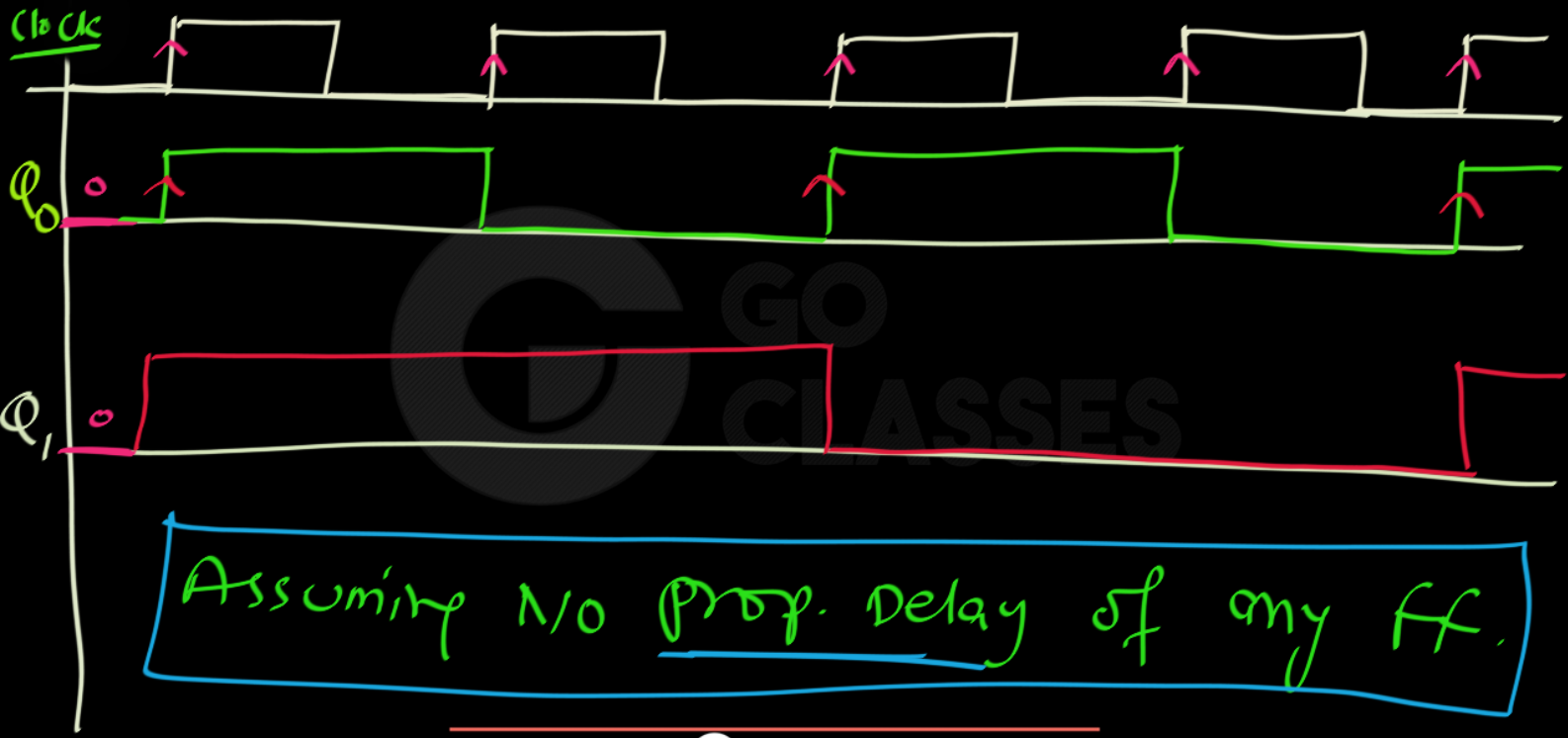

When clock ↑ then $Q_0$ Toggles

" $Q_0$ ↑ " $Q_1$ " .


Clock
1
0
1
0
1
$Q_0$
LSB
1
1
0
0
1
$Q_1$
MSB
11
10
00
01
Binary Down Counter
3
2
1
0

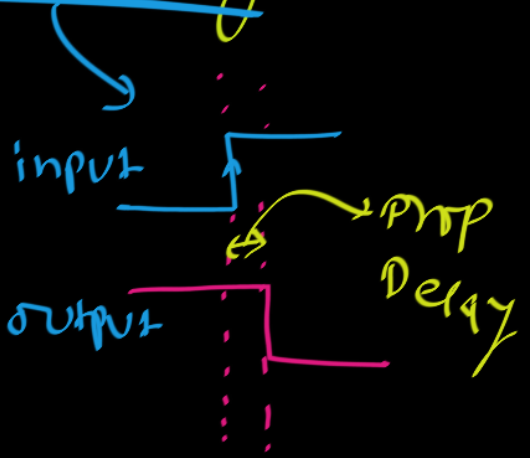


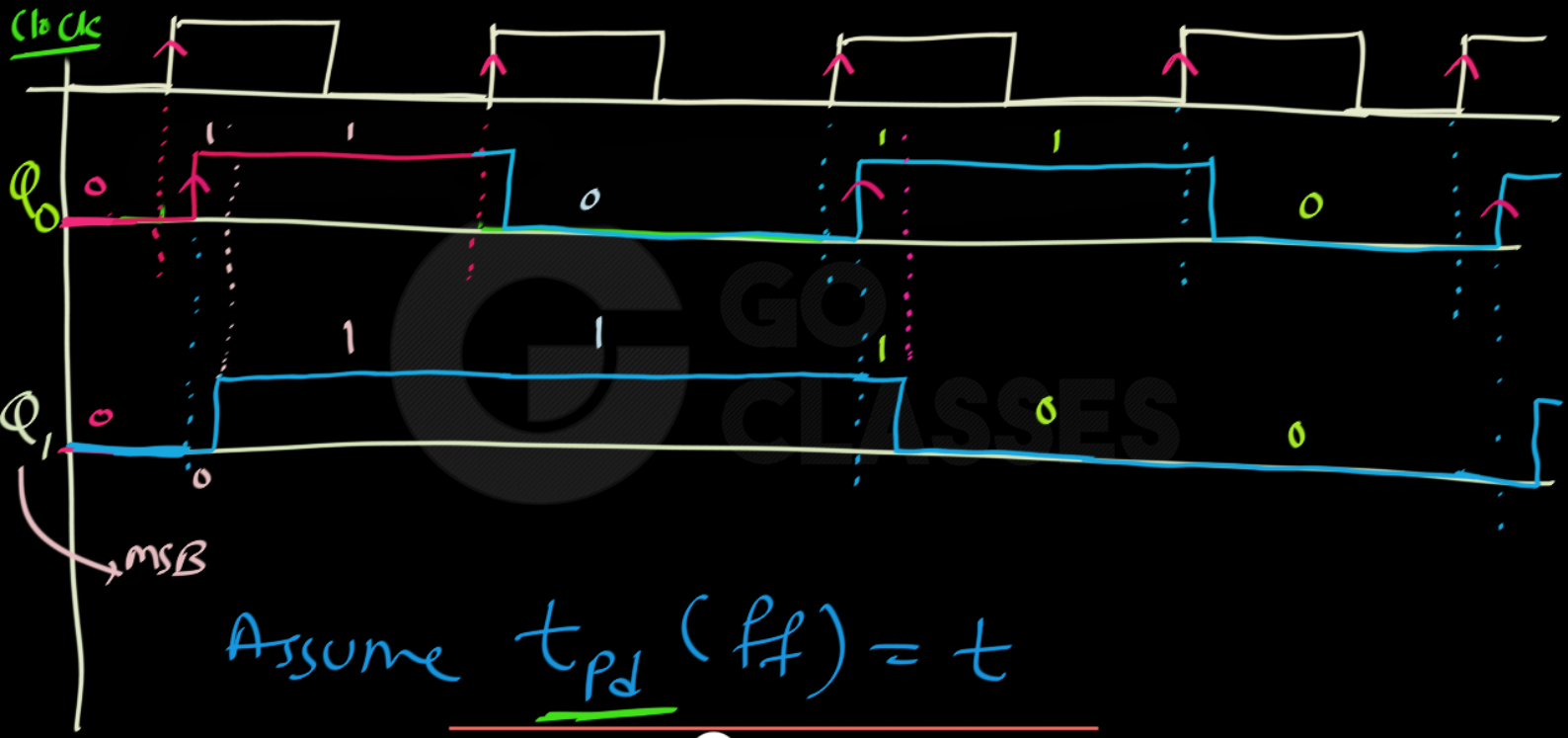

$$freq(Q_1) = \frac{freq(Q_0)}{2} = \frac{freq(clock)}{4}$$

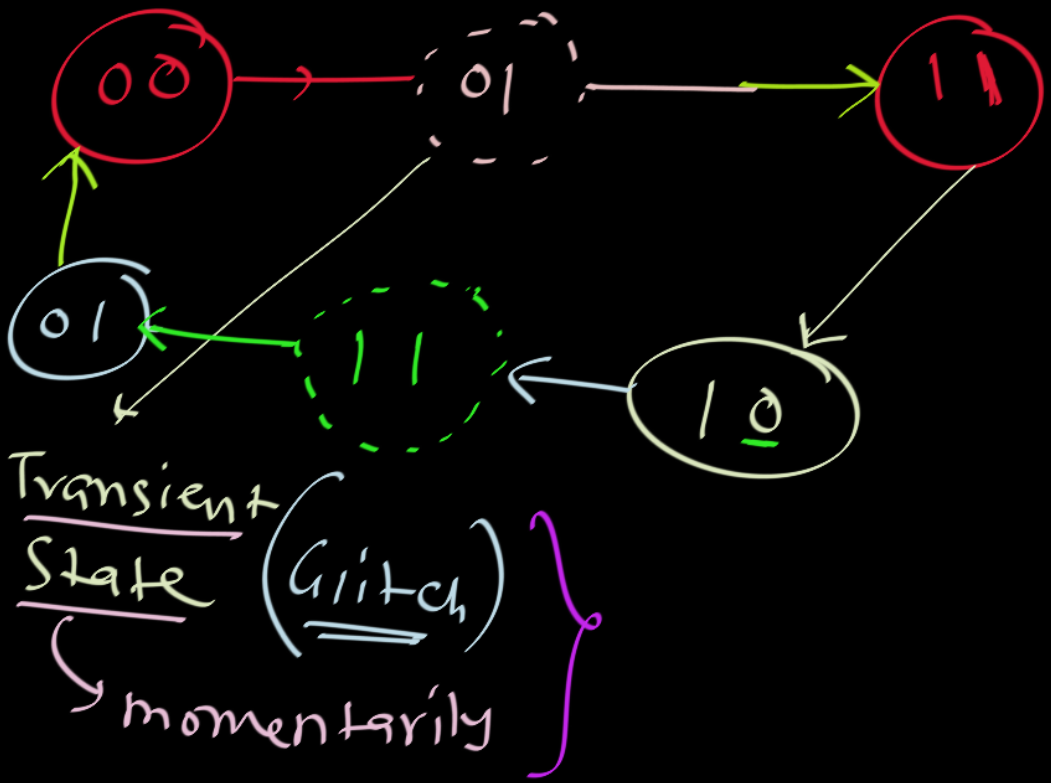

Clock
$Q_0$
0
$Q_1$
0
Assuming N/O Prop. Delay of any FF.

But in reality,

Every ff has a prop. delay




Clock
$Q_0$
$Q_1$
MSB
Assume $t_{pd}$ (ff) = t

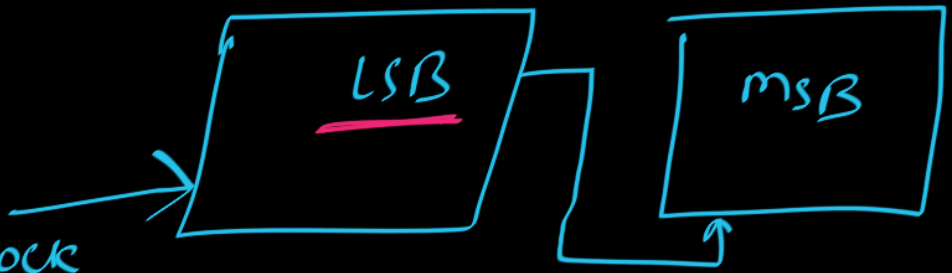

00
01
11
01
11
10
Transient
State
(Glitch)
momentarily

# Why Transient States in Asynchronous Counters ?

(1) Due to Propagation Delay of ff

AND

(2)



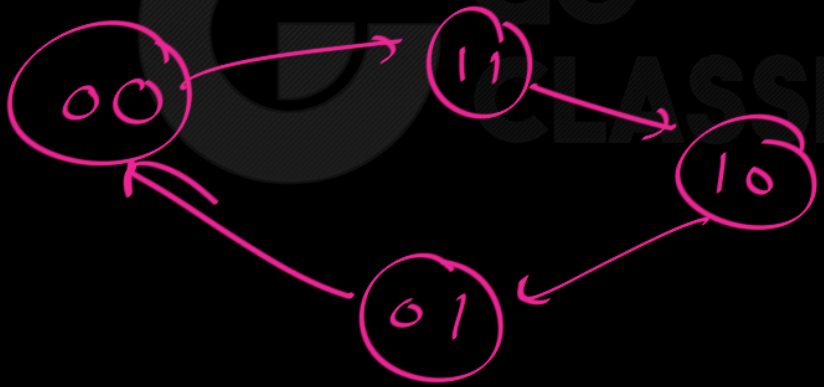
Because of clock ⇒ first LSB changes

//

Because of the change in LSB ← then MSB changes

# Binary Down Ripple Counter :

## If No Prop. Delay :



# Binary Down Ripple Counter :

If Prop. delay of ff = t

# Next Topic:

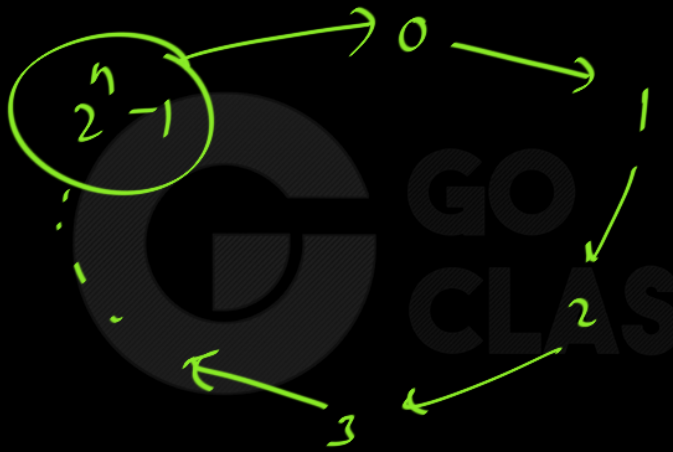
## Asynchronous Counters

## 1. Binary Ripple counter

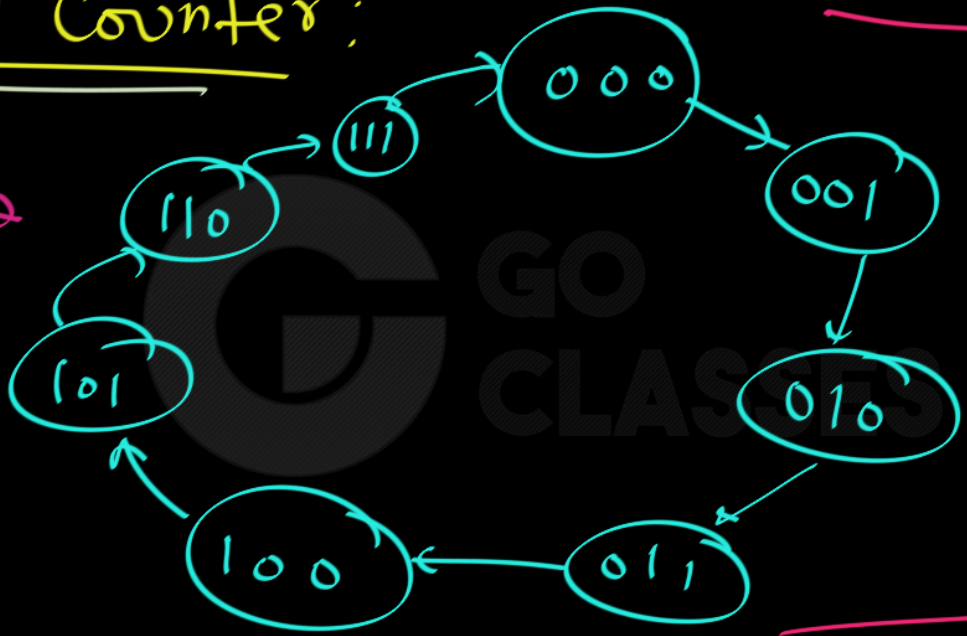
# Binary Counter

- A binary counter is a set of flip-flops whose states change in response to pulses applied at the input to the counter.
  - The combined state of the flip-flops at any time is the binary equivalent of the total number of pulses that have been applied.
- Counters are of two types:
  a) Asynchronous counter
  b) Synchronous counter

# Binary Counter : ( Binary Up Counter )



# 3-bit Binary Counter :




Counting sequence

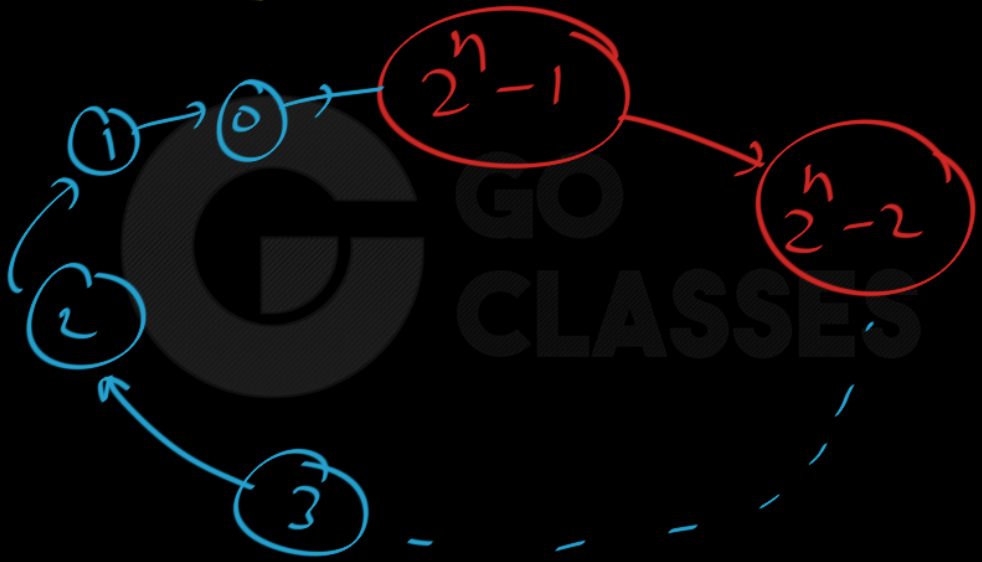
Implement

① Asyn. Counter

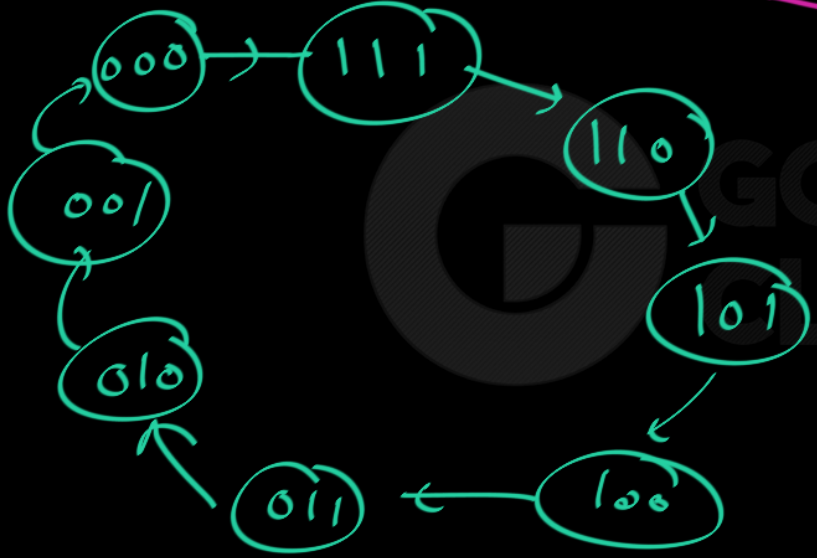
OR

② Synch Counter

# Binary Down Counter:



# 3 bit Binary Down Counter:




## Implementations:

1. Async Counter (Ripple ")
2. Sync Counter

# Binary Down Counter :

## Next Topic :

- Asynchronous Binary Down Counter
- OR
- Binary Down Ripple Counter

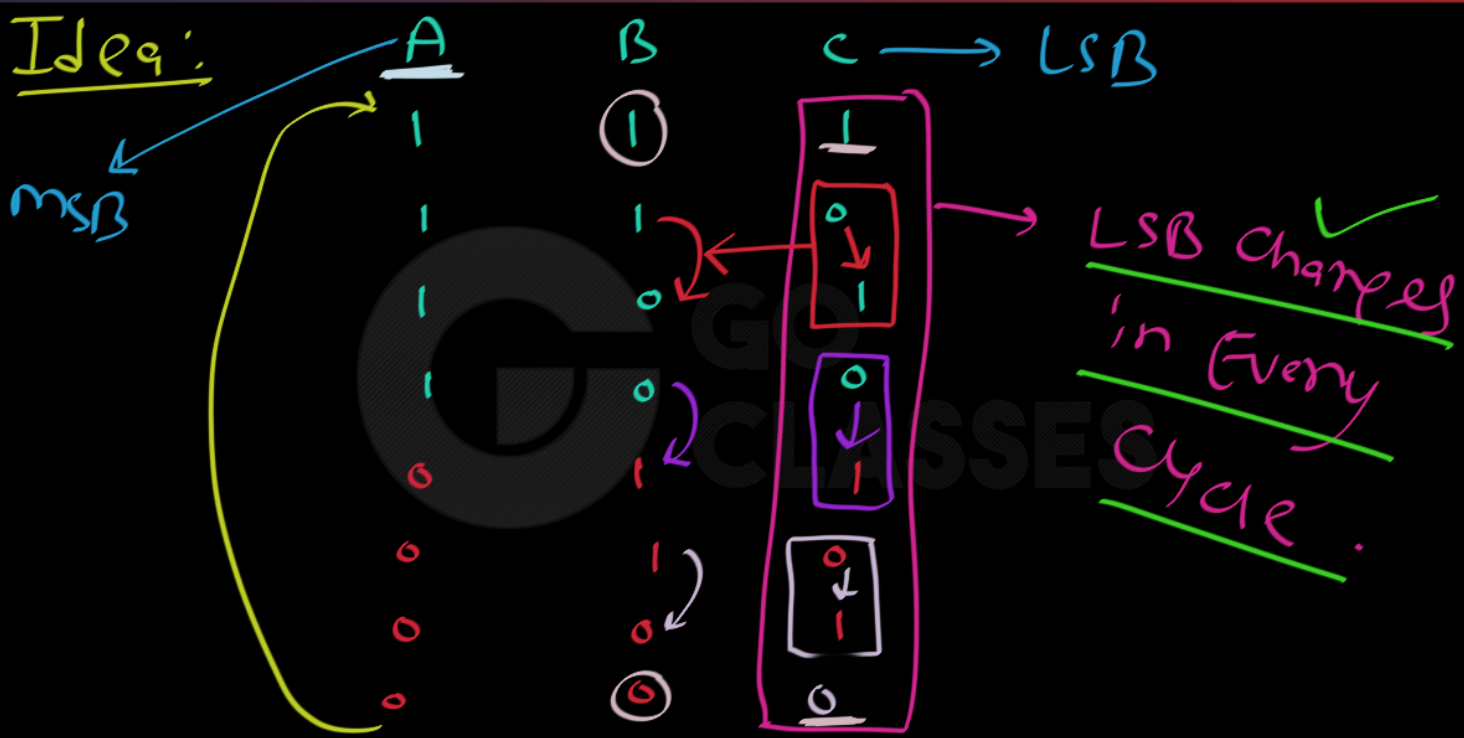
# Asynchronous Counter (Ripple Counter)

- This type of counter is the easiest to design, and requires the least amount of hardware.
- The counter is called *asynchronous* because the flip-flops are not driven by the same clock signal, and hence do not change state simultaneously.
  - Constructed using T flip-flops.
- It is called *ripple counter* because when the counter, for example, goes from *1111* to *0000*, it temporarily goes through a number of intermediate states:
  *1111* → *0111* → *0011* → *0001* → *0000*.
  - Can lead to unwanted transitions if the outputs drive some other circuit.
  - Glitches are possible.

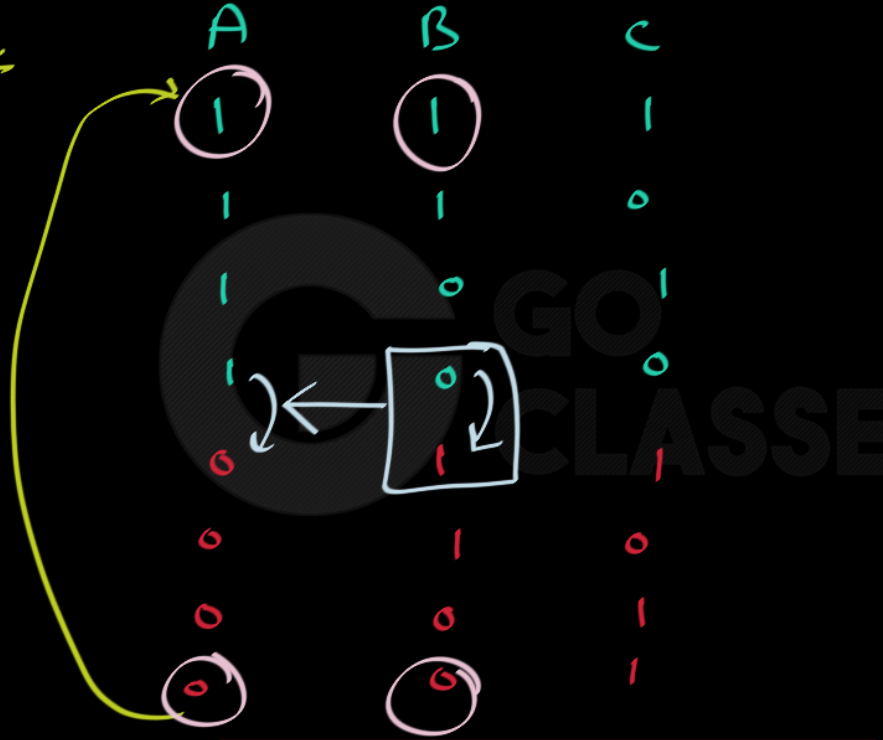
# Next Topic:

# Asynchronous Counters

## 1.1: Binary Down counter


Idea:
A
B
C → LSB
MSB
LSB changes in Every Cycle.

Idea:



Idea: → Binary Down Counter

(1) LSB Changes in Every cycle.
↳ Connect to clock. $Q_0$

(2) $2^{n_2}$ bit B Changes when LSB 0 → 1

↳ If ffs are +ve Edge Triggered : $Q_0$ to Clock of B.

↳ If ffs are -ve " " : $\overline{Q_0}$ to Clock of B.

Idea: → Binary Down Counter

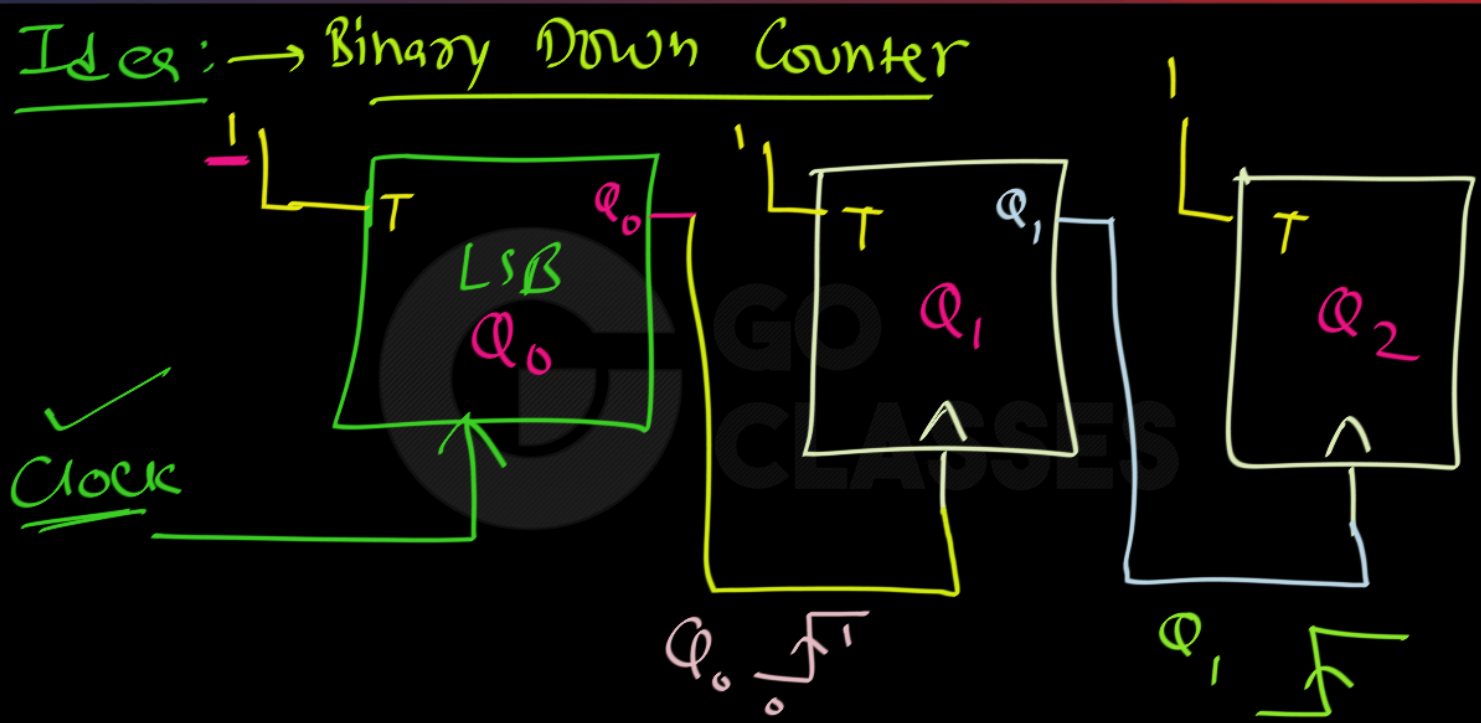
(1) LSB changes in Every cycle. $Q_1 = B$

↳ Connect to clock.

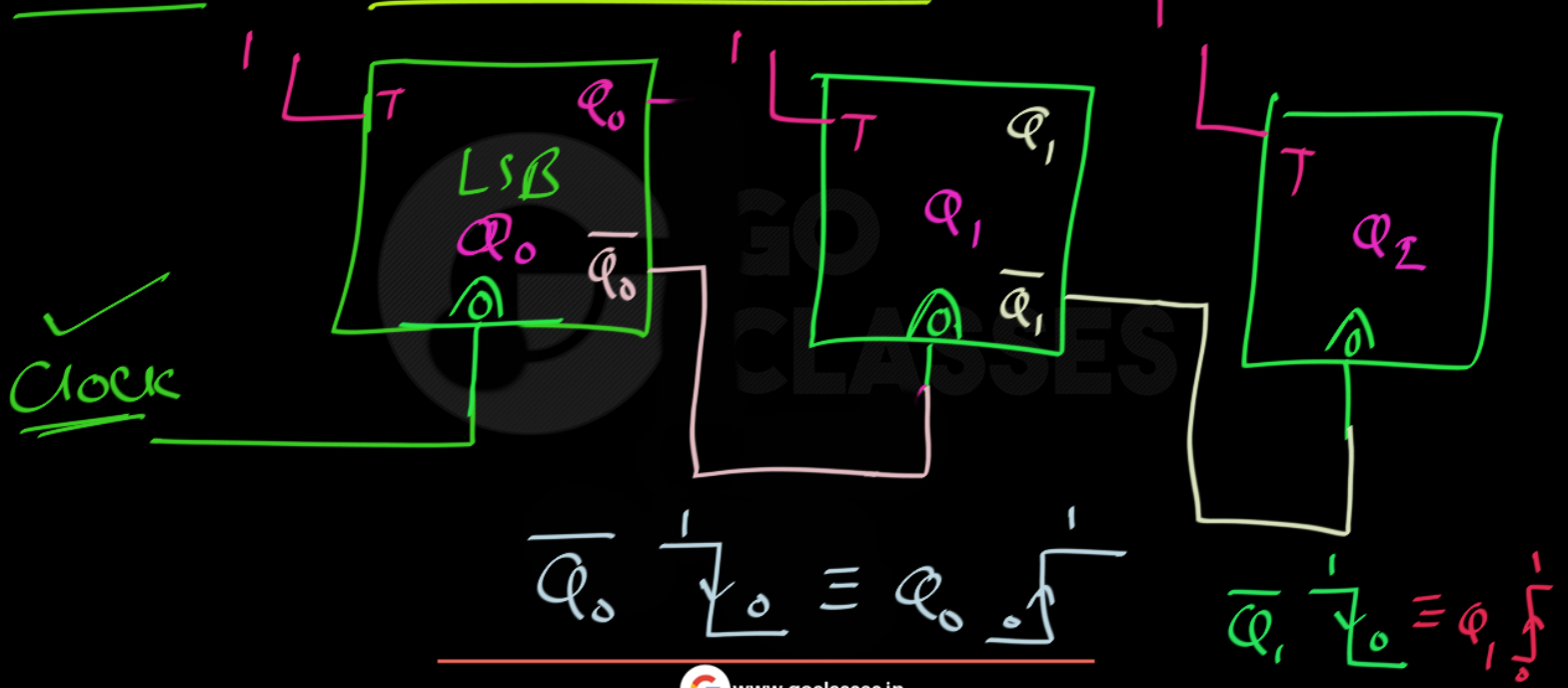
(2) $3^n$ bit changes when A B 0 → 1

↳ If ffs are +ve Edge Triggered : $Q_1$ to Clock of A

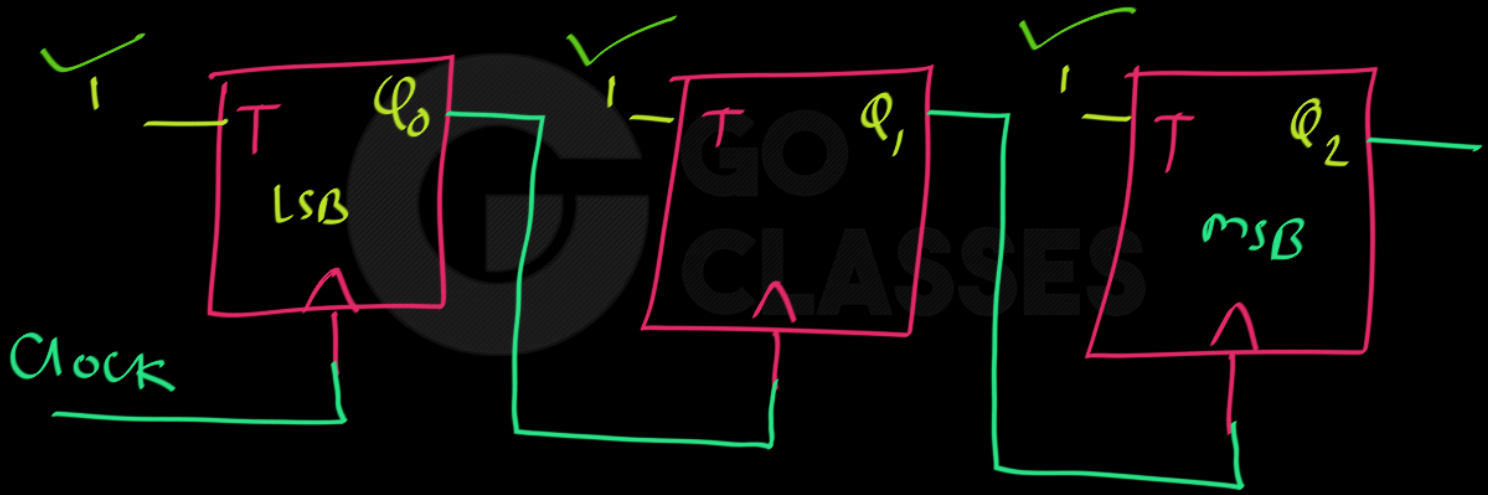
↳ If ffs are -ve " " : $\overline{Q_1}$ to Clock of A


Idea :→ Binary Down Counter
1
T
$Q_0$
LSB
$Q_0$
1
T
$Q_1$
$Q_1$
1
T
$Q_2$
Clock
$Q_0$
$Q_1$

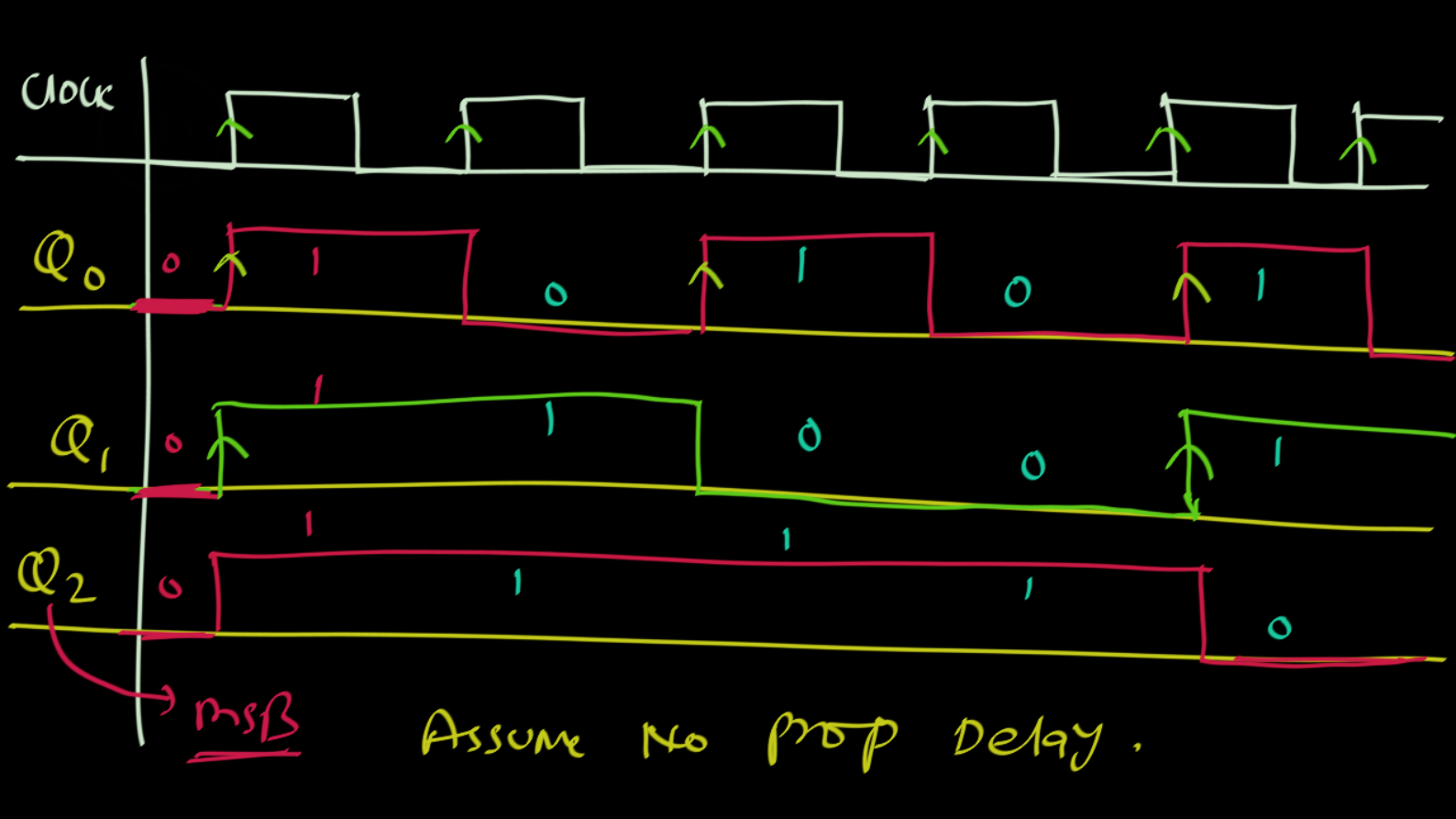
Idea :→ Binary Down Counter

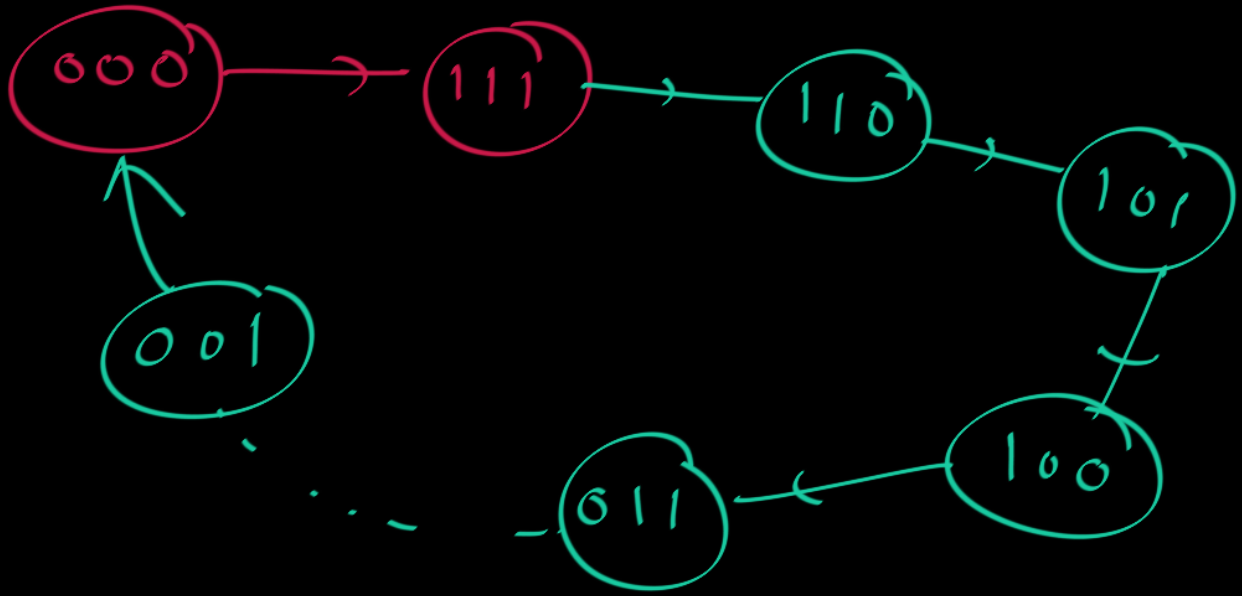


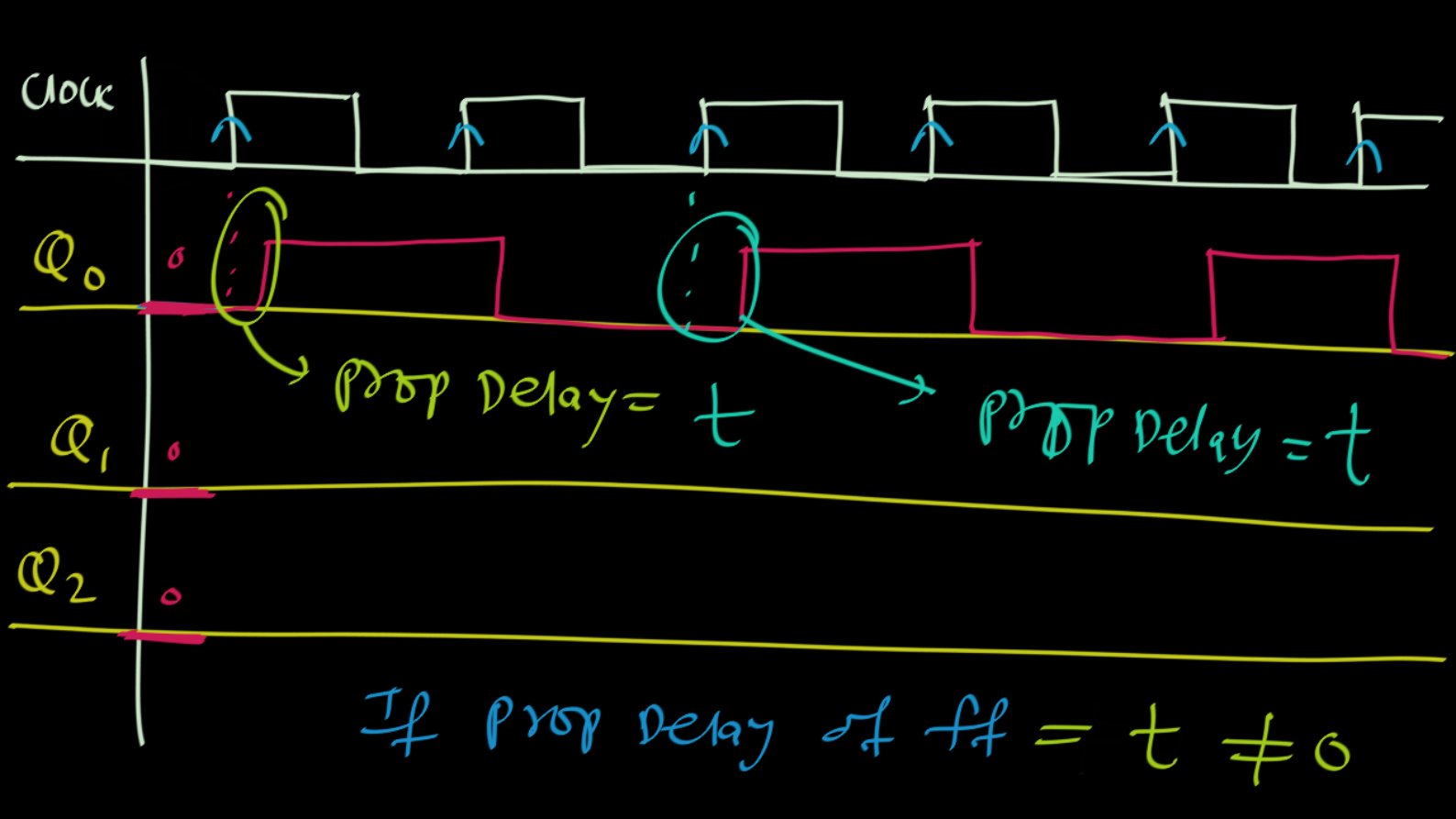
# 3-bit Binary Down Counter:

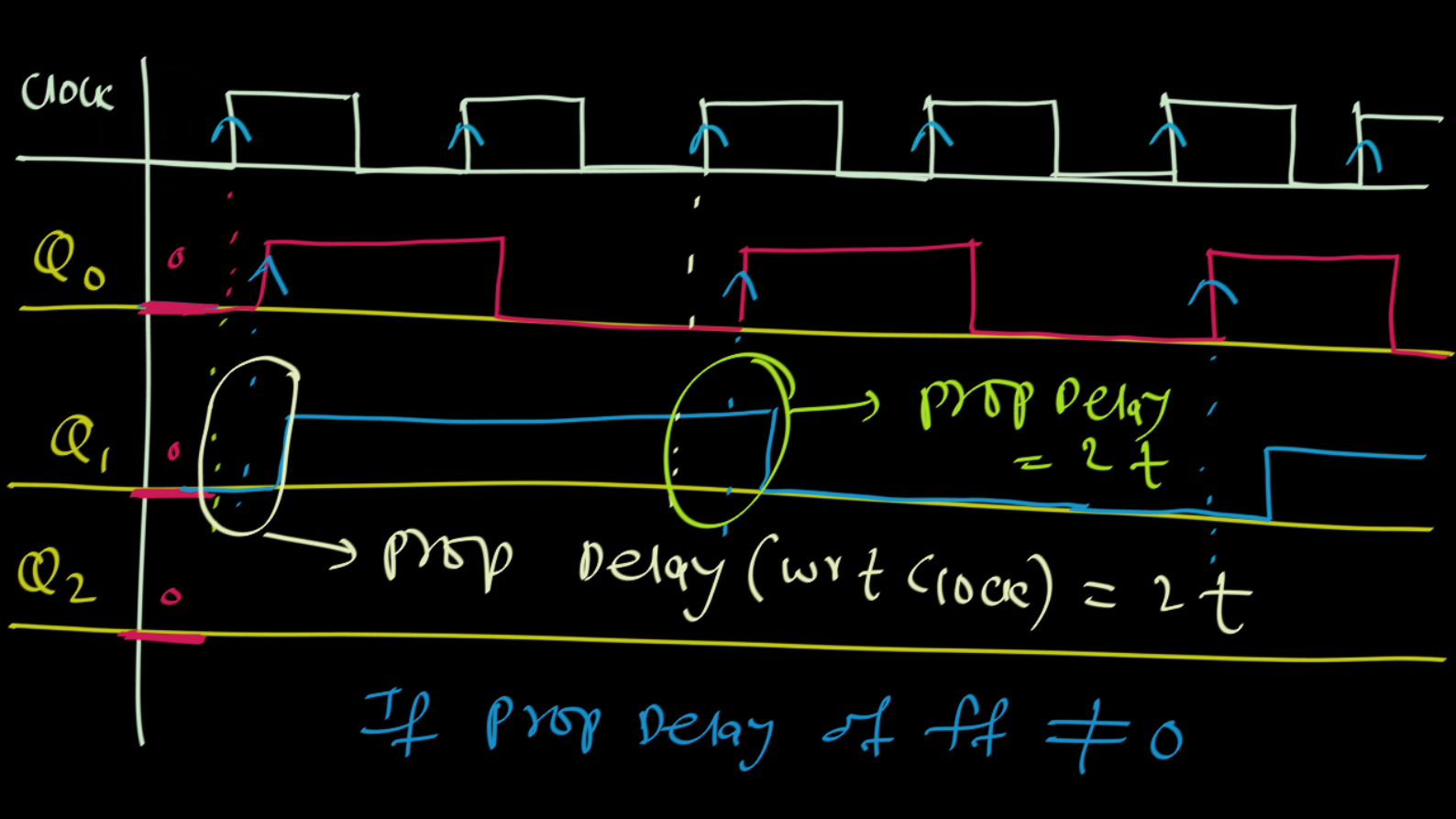


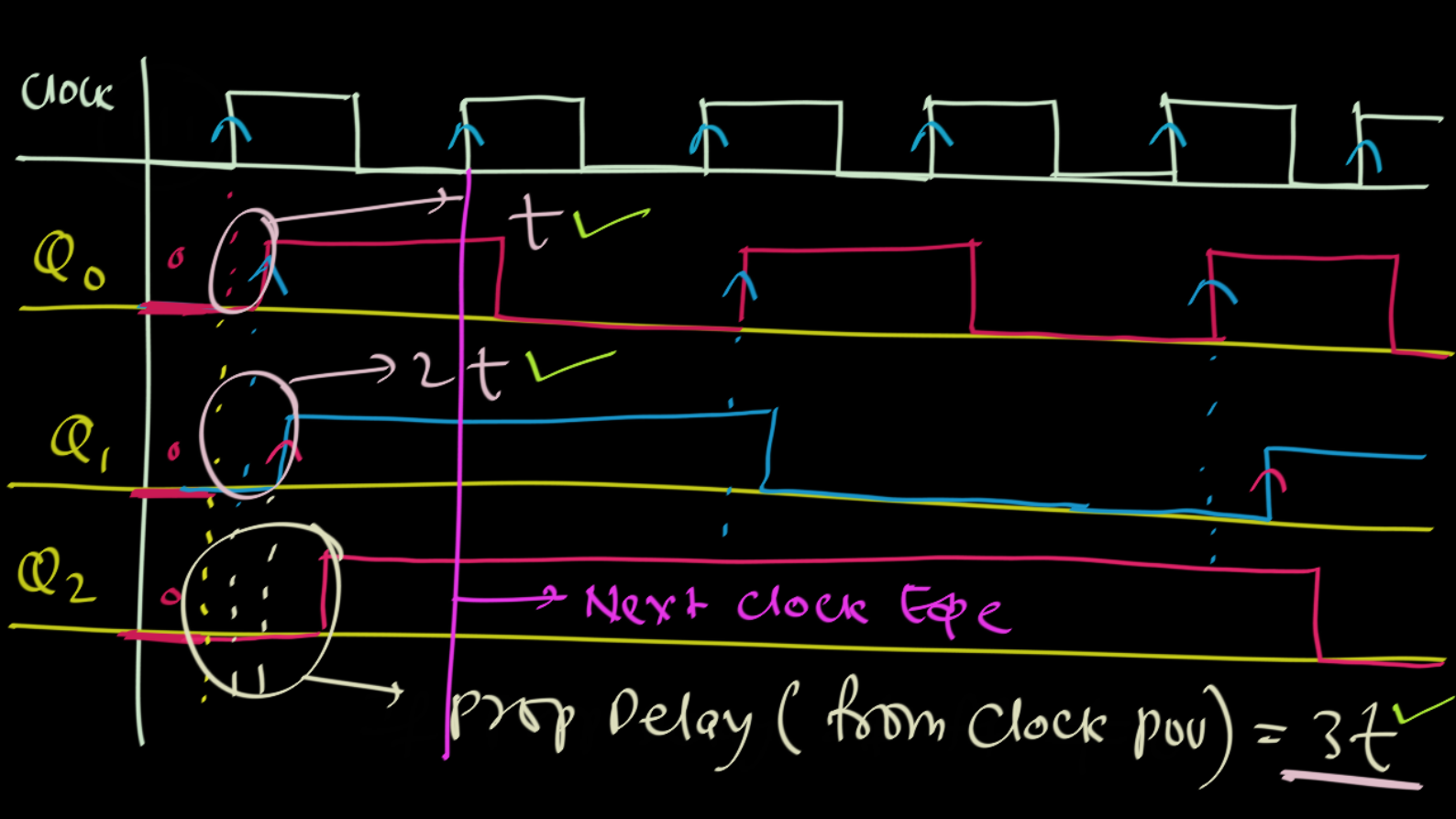

Positive Edge Triggered FFs

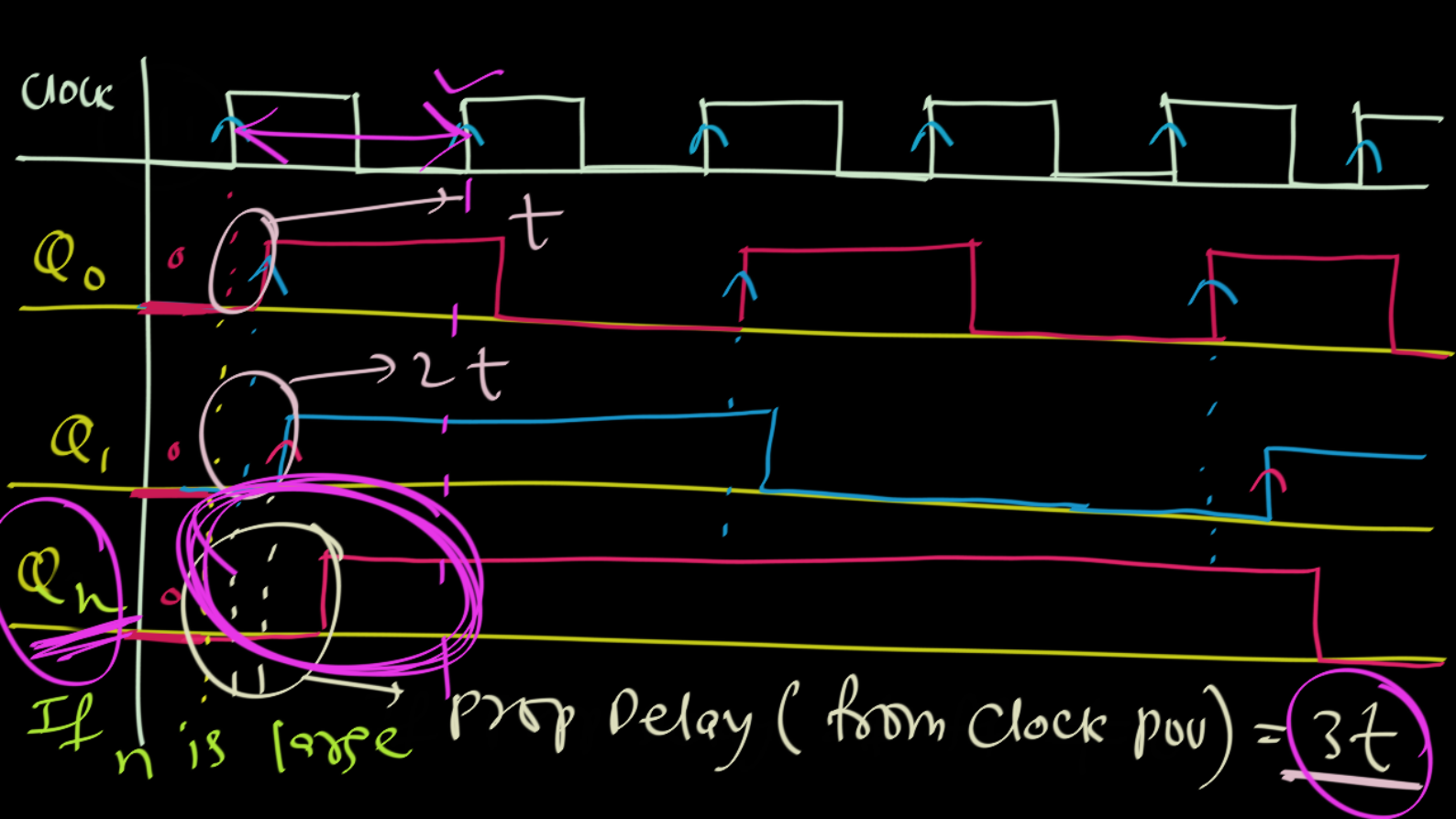

Clock
$Q_0$
$Q_1$
$Q_2$
MSB
Assume No Prop Delay.


000
111
110
101
001
011
100

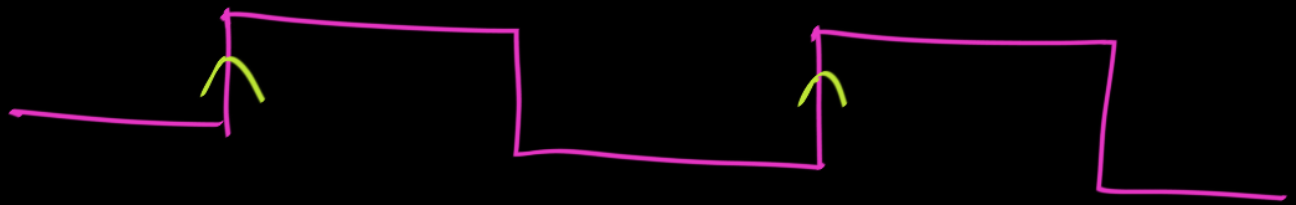

Clock
$Q_0$
0
$Q_1$
0
$Q_2$
0
Prop Delay = t
Prop Delay = t
If Prop Delay of ff = t ≠ 0


Clock
$Q_0$
$Q_1$
$Q_2$
0
0
0
Prop Delay = 2t
Prop Delay (wrt Clock) = 2t
If Prop Delay of ff ≠ 0


Clock
$Q_0$
$t$
$Q_1$
$2t$
$Q_2$
Next clock Edge
Prop Delay ( from Clock pov) = $3t$

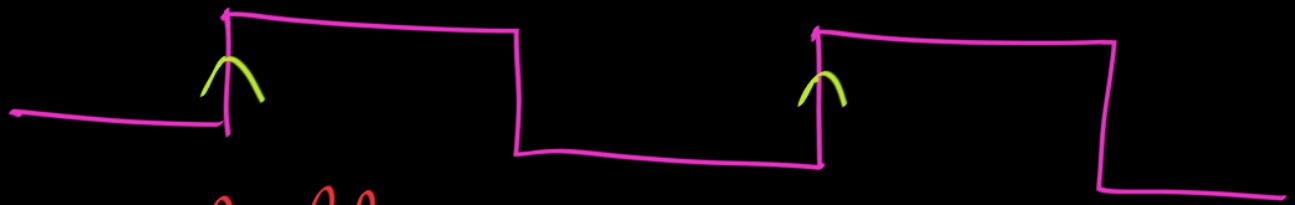

Clock
t
$Q_0$
2t
$Q_1$
$Q_n$
If n is large
Prop Delay (from Clock pov) = 3t

Note: n-bit Ripple Counter :




msb → Prop Delay = $n \cdot t_{pd}$ → one ff prop delay

Note: n-bit Ripple Counter :




max. no. of ffs we can have in Ripple Counter

Time Period of clock $\geq n.t_{pd}$

one ff prop delay

In Any Ripple Counter on n-ffs :

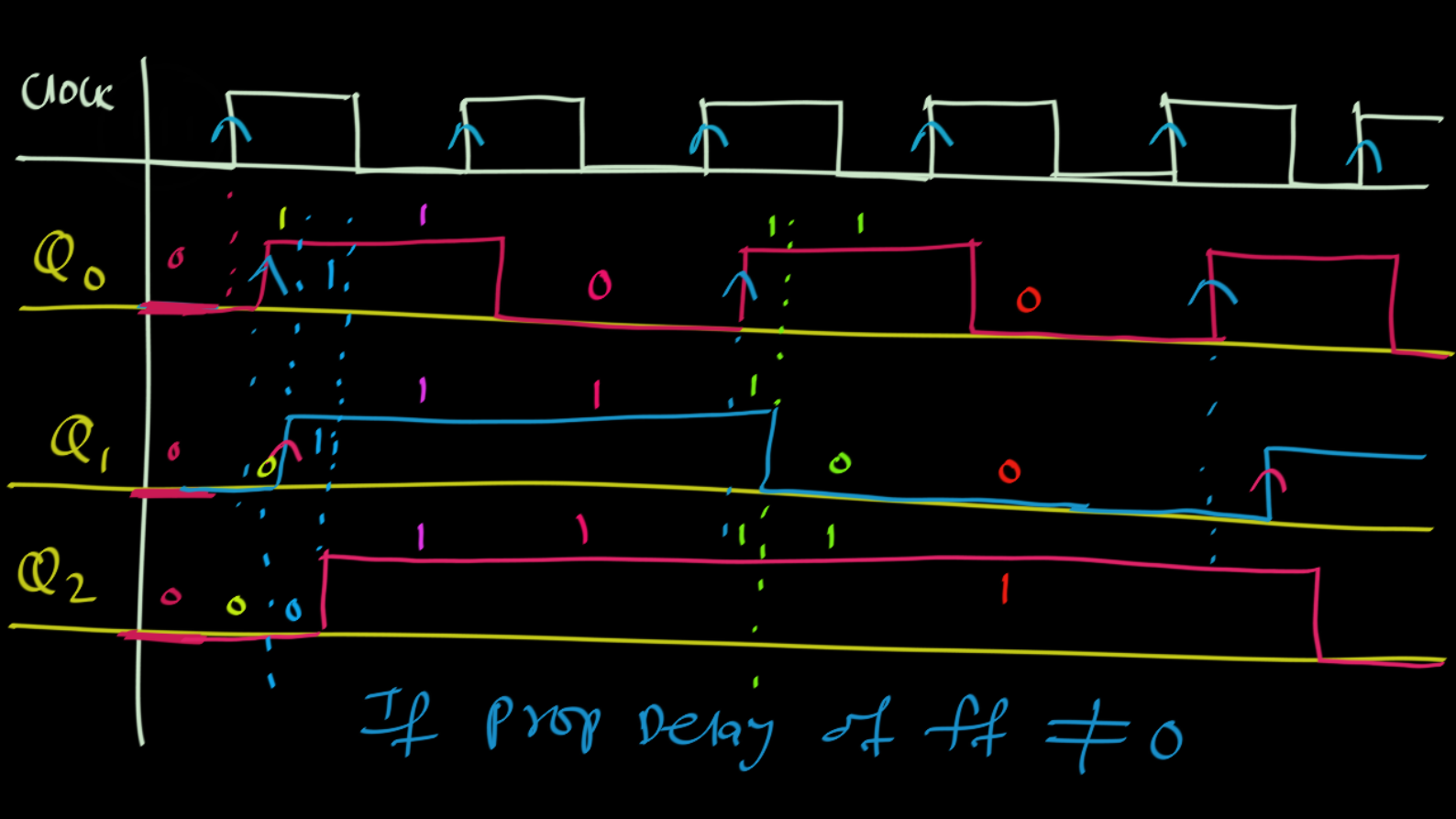
for proper functioning :

$n t_{pd} \leq$ Clock Period time

In Any Ripple Counter on n-ffs :

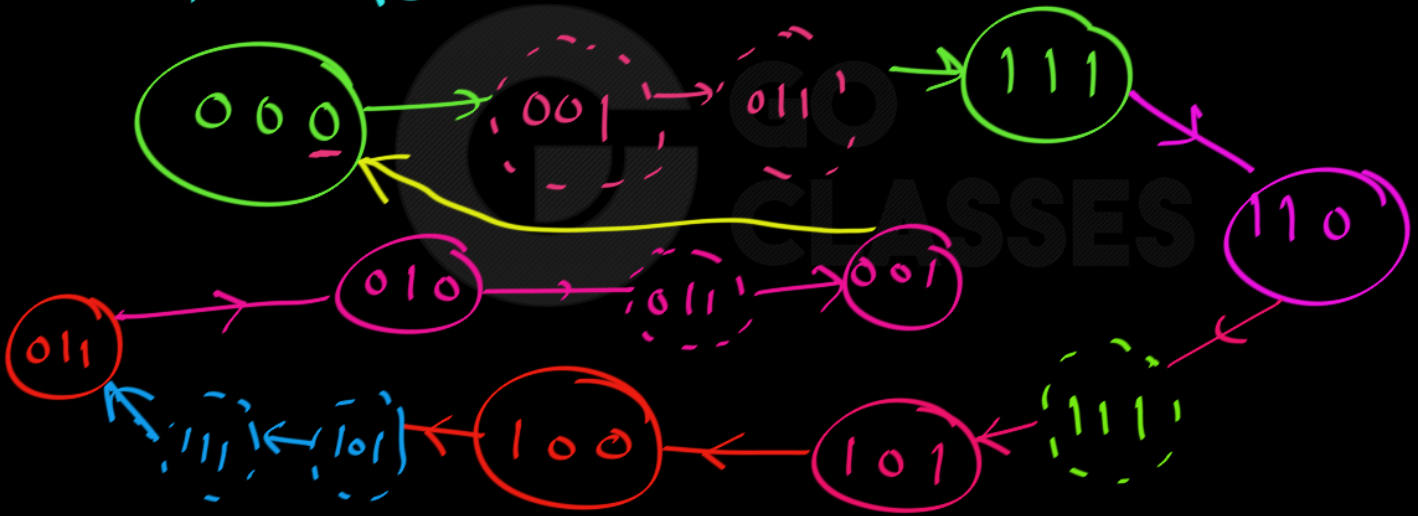
for proper functioning :

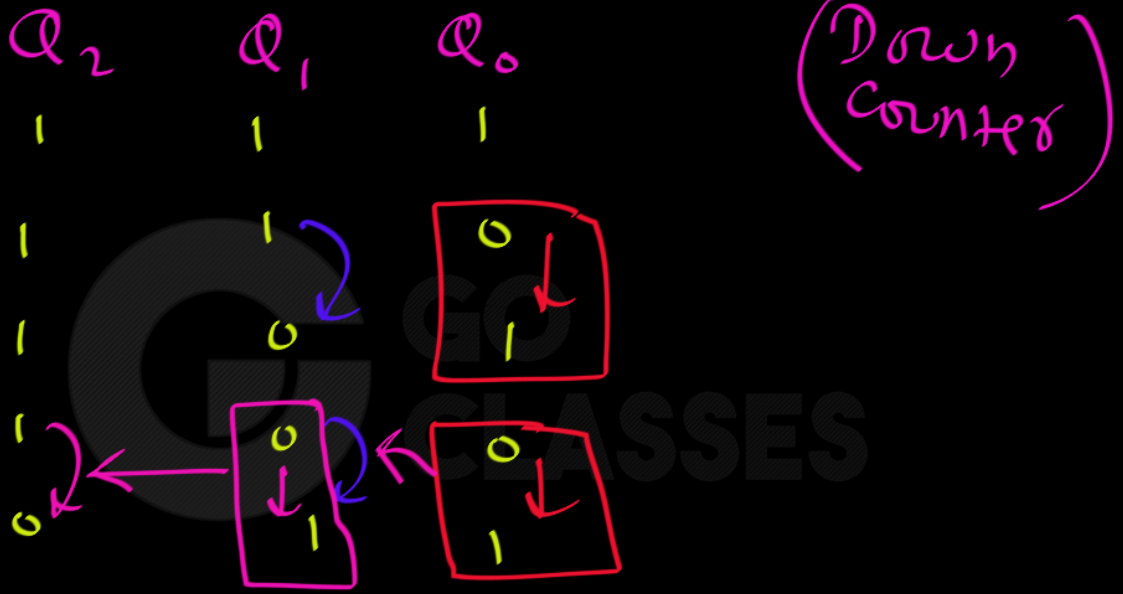
$$\frac{1}{n\, t_{pd}} \geq \text{frequency of clock}$$


Clock
$Q_0$
$Q_1$
$Q_2$
If prop delay of ff ≠ 0

# 3-bit Binary Down Counter:

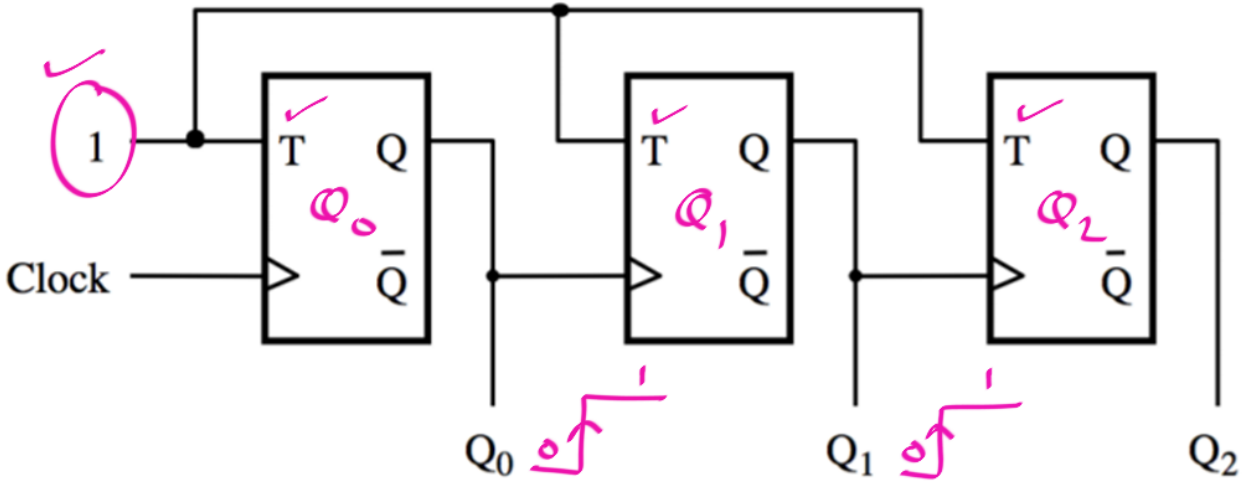
If $t_{PD} \neq 0$



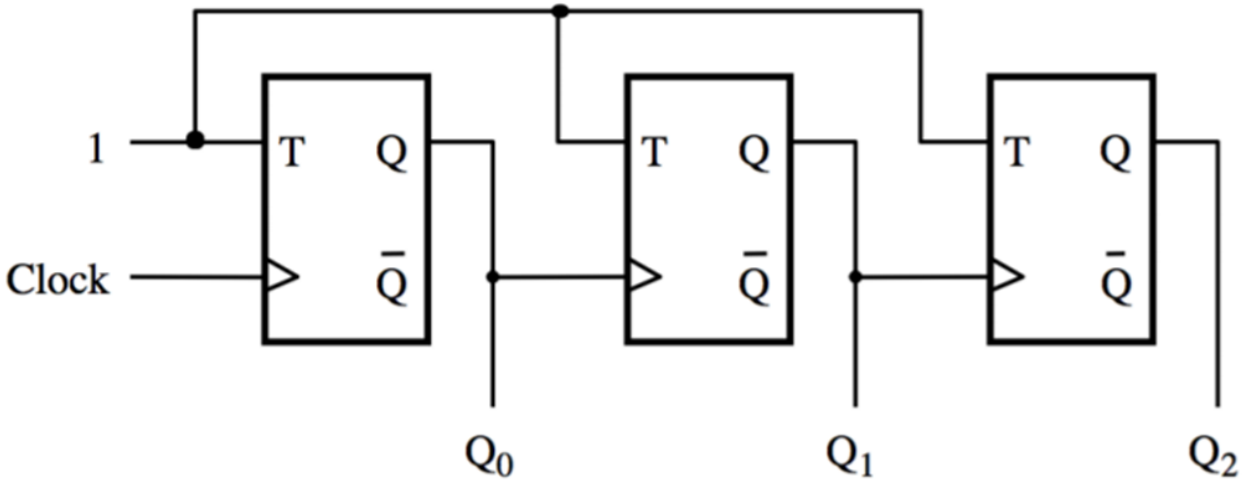

$Q_2$
$Q_1$
$Q_0$
(Down Counter)

# A three-bit down-counter

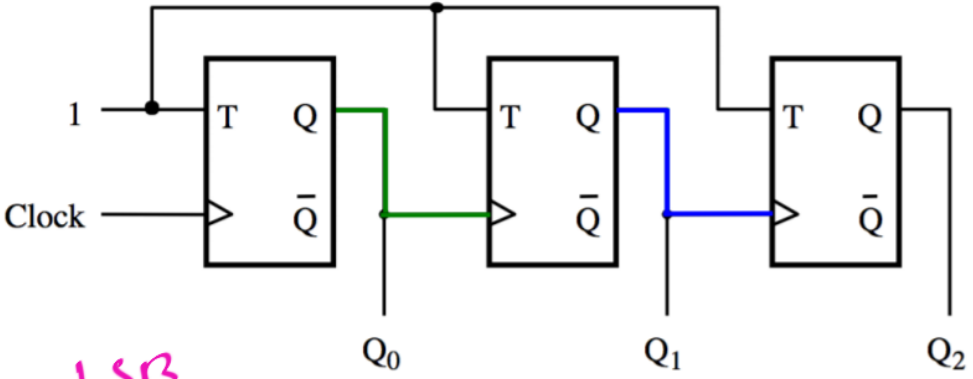
$Q_2$ $Q_1$ $Q_0$ → Change in every cycle



# A three-bit down-counter



# A three-bit down-counter



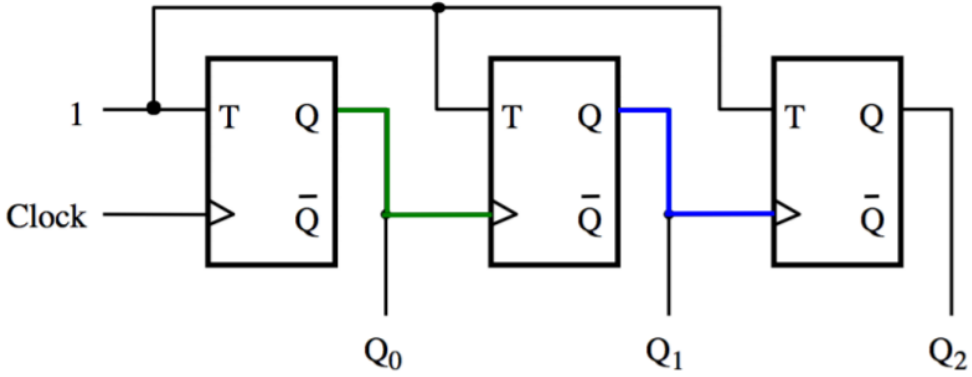

LSB

The first flip-flop changes on the positive edge of the clock

The second flip-flop changes on the positive edge of $Q_0$

The third flip-flop changes on the positive edge of $Q_1$

# A three-bit down-counter

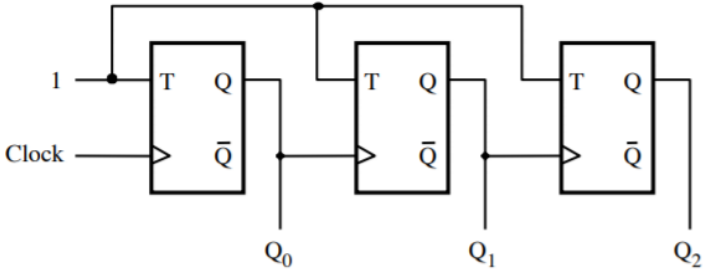



The first flip-flop changes on the positive edge of the clock

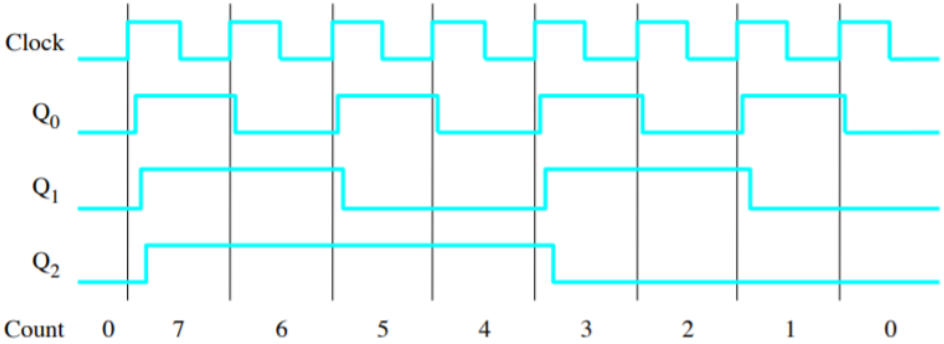
The second flip-flop changes on the positive edge of $Q_0$

The third flip-flop changes on the positive edge of $Q_1$

# A three-bit down-counter

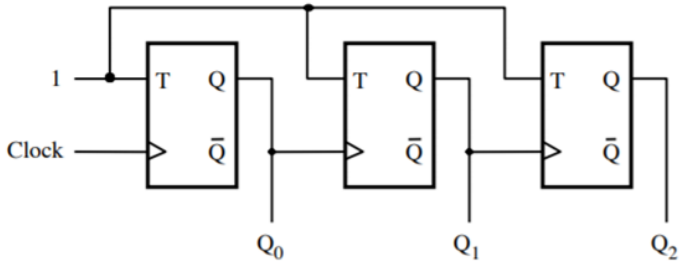



(a) Circuit

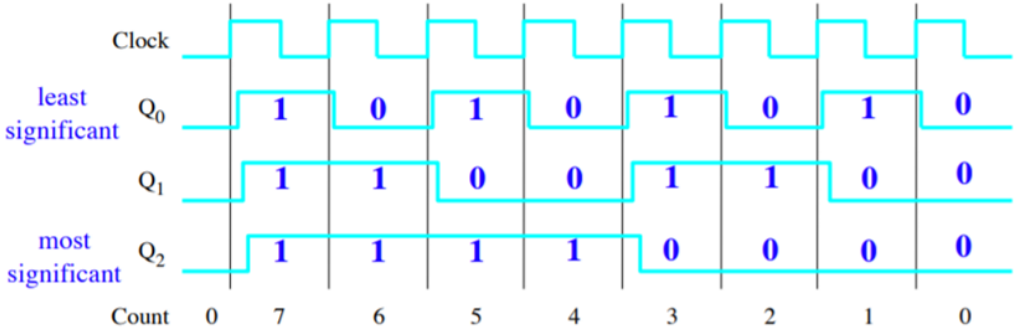


# A three-bit down-counter





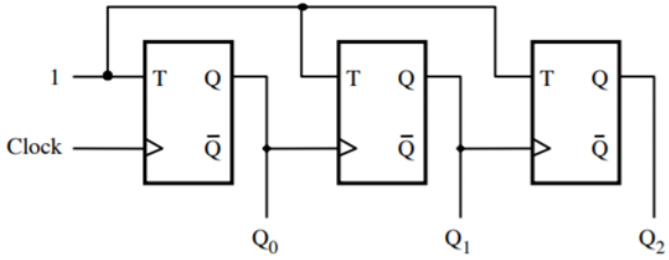

(a) Circuit



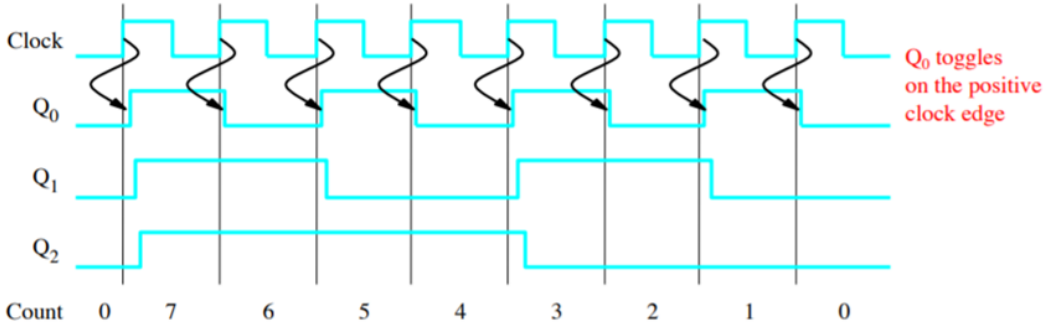

(b) Timing diagram

# A three-bit down-counter



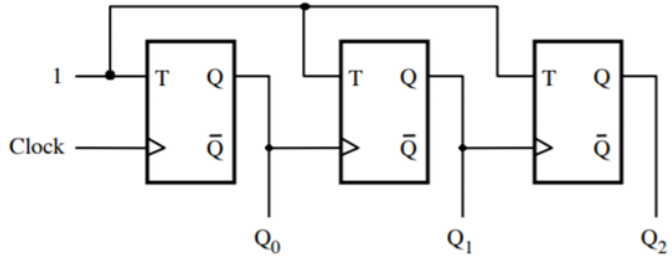



(a) Circuit

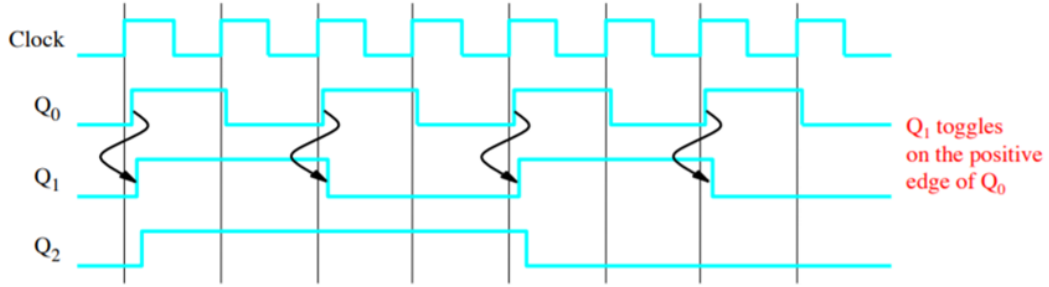



(b) Timing diagram

# A three-bit down-counter



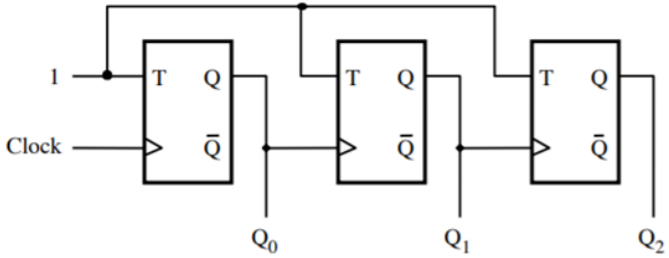

(a) Circuit



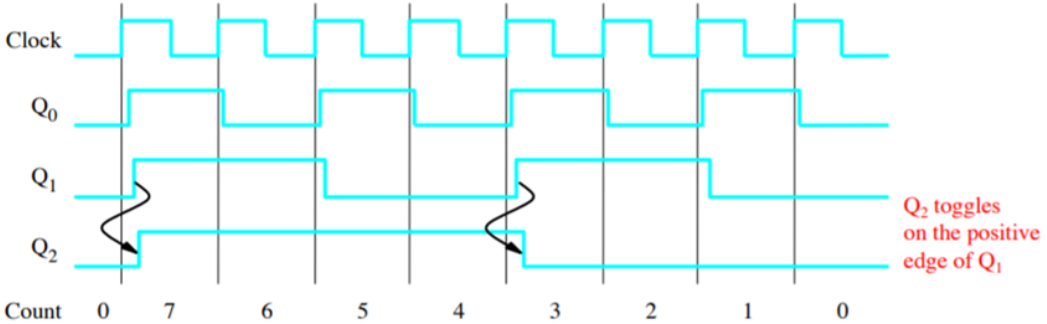

(b) Timing diagram

# A three-bit down-counter



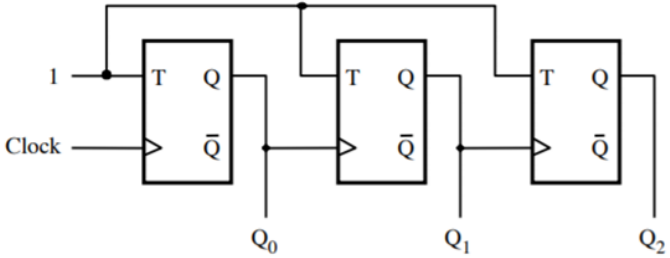



(a) Circuit



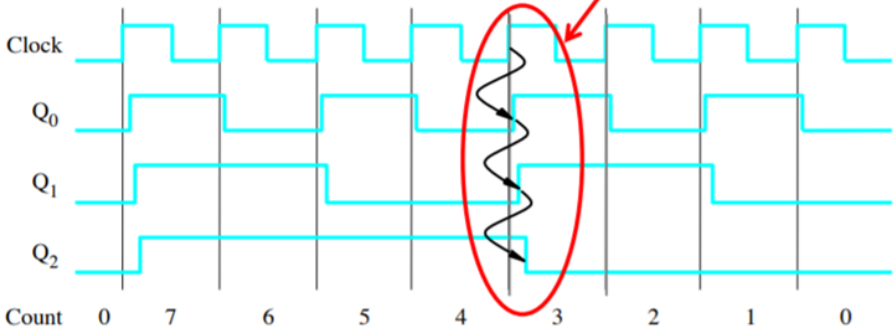

(b) Timing diagram

# A three-bit down-counter




(a) Circuit

The propagation delays get longer



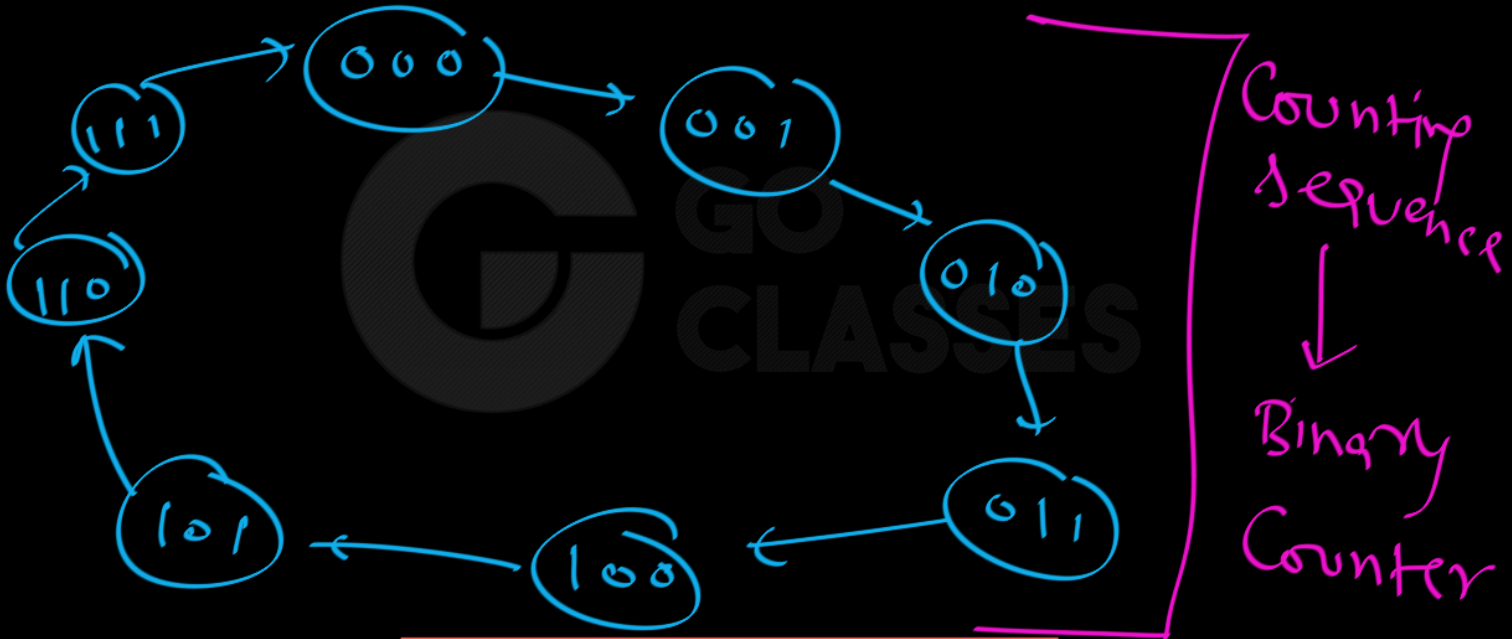

(b) Timing diagram

# Next Topic:

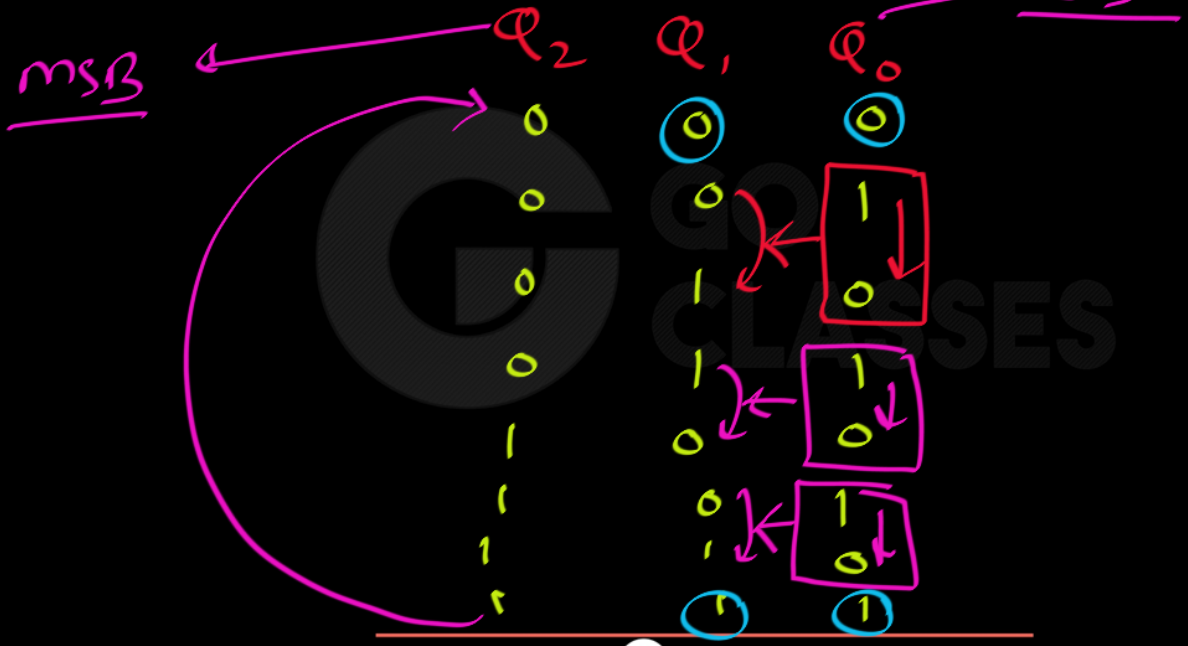
## Asynchronous Counters

## 1.2: Binary (up) counter

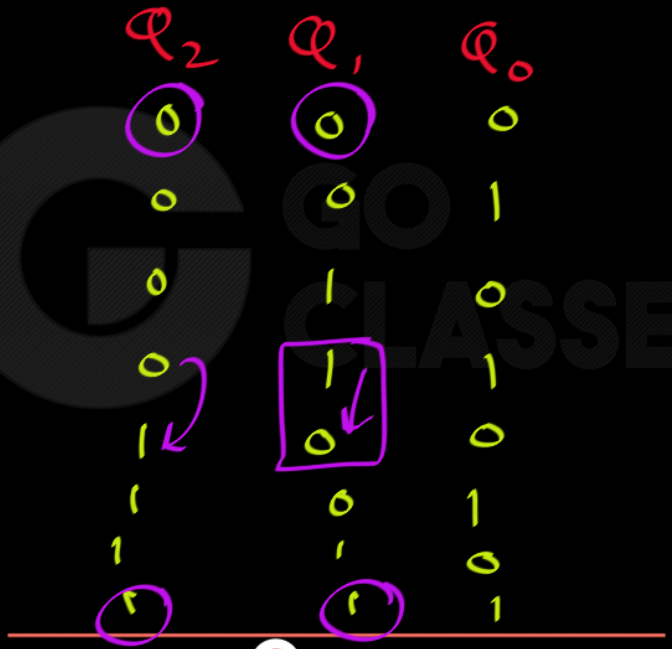
# 3-bit Binary (UP) Counter :



# 3-bit Binary (UP) Counter :



# 3-bit Binary (UP) Counter :



# 3-bit Binary (UP) Counter :

(1) LSB changes in every clock cycle.
↳ Clock to LSB-FF.

(2) $Q_1$ changes when $Q_0 = 1 \longrightarrow 0$

{ (1) If FFs are +ve Edge Trig : $\overline{Q_0}$ to clock of $Q_1$
{ (2) If FFs are -ve Edge Trig : $Q_0$ to clock of $Q_1$

# 3-bit Binary (UP) Counter :

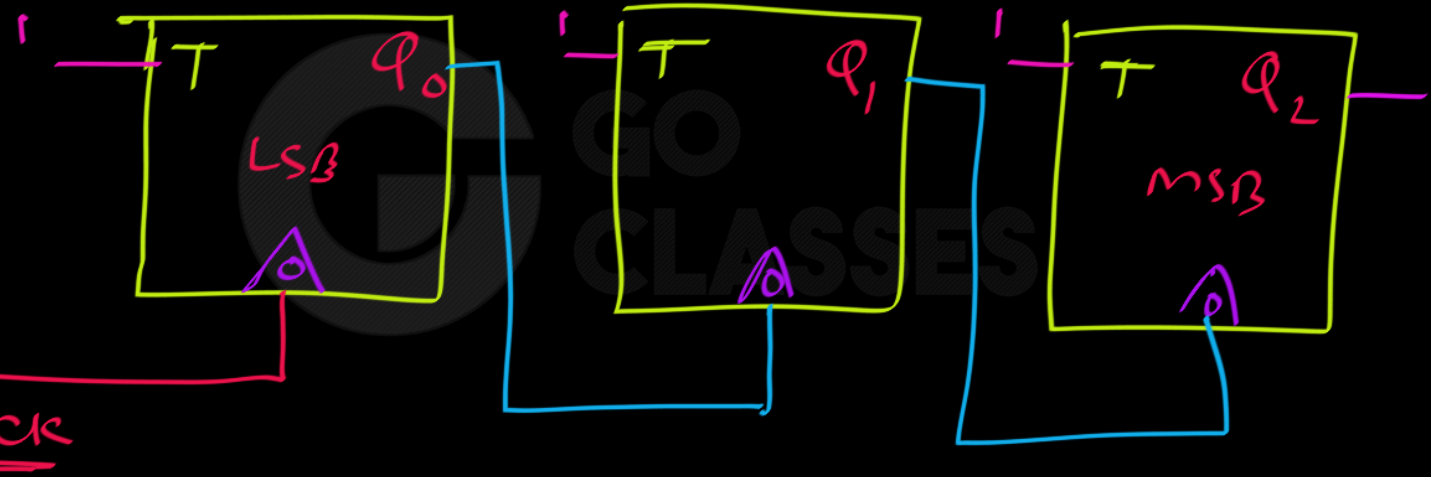
(1) LSB changes in every clock cycle.
↳ Clock to LSB-ff.

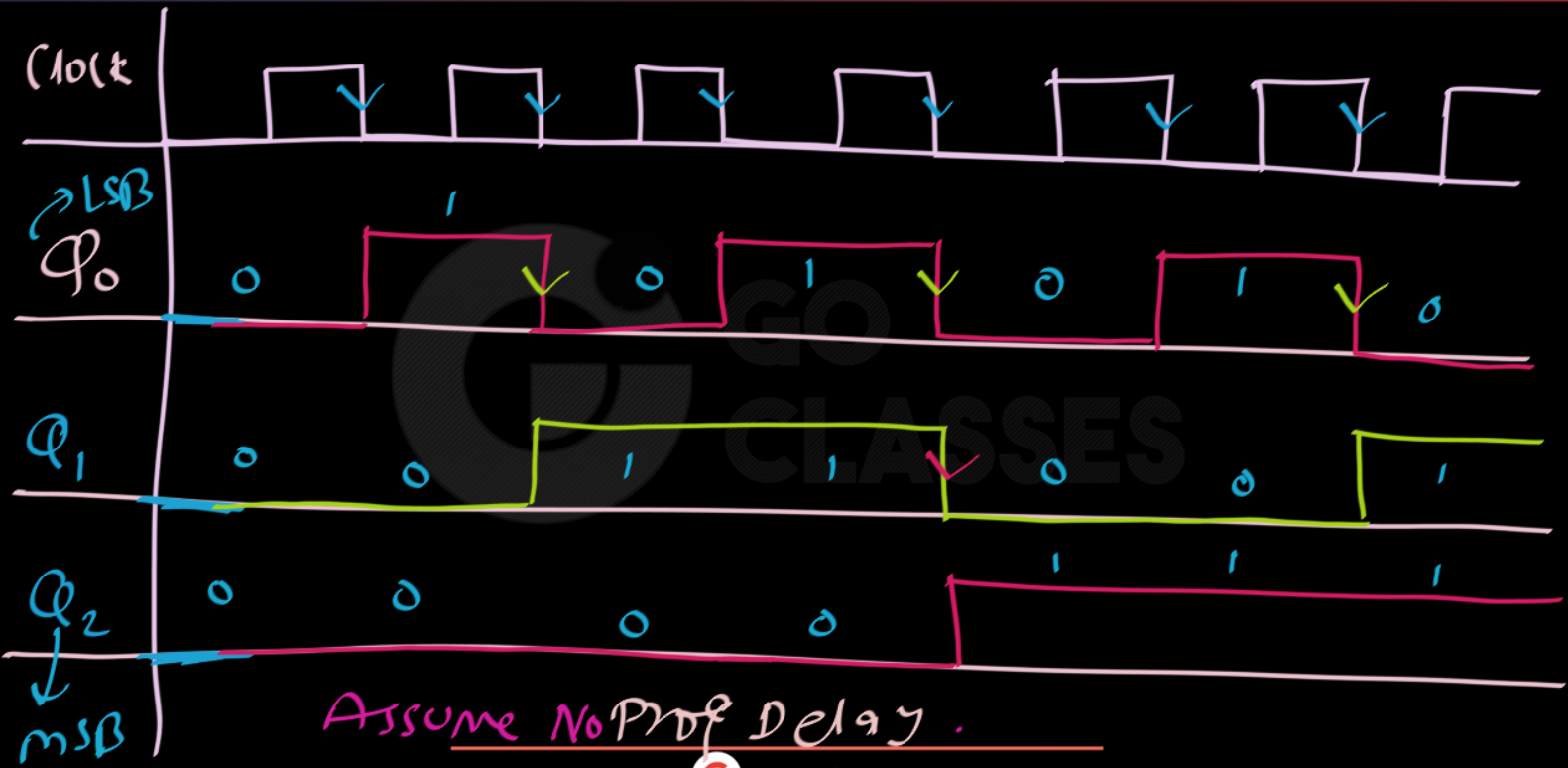
(3) $Q_2$ changes when $Q_1 = 1 \longrightarrow 0$

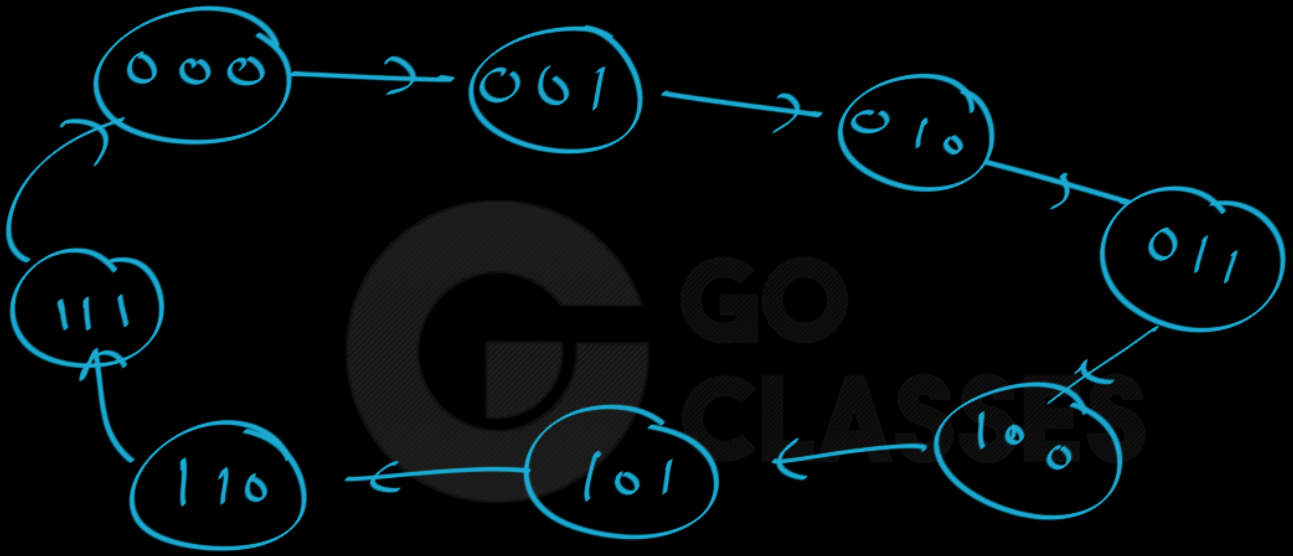
{ (1) If ffs are +ve Edge Trig : $\overline{Q_1}$ to clock of $Q_2$

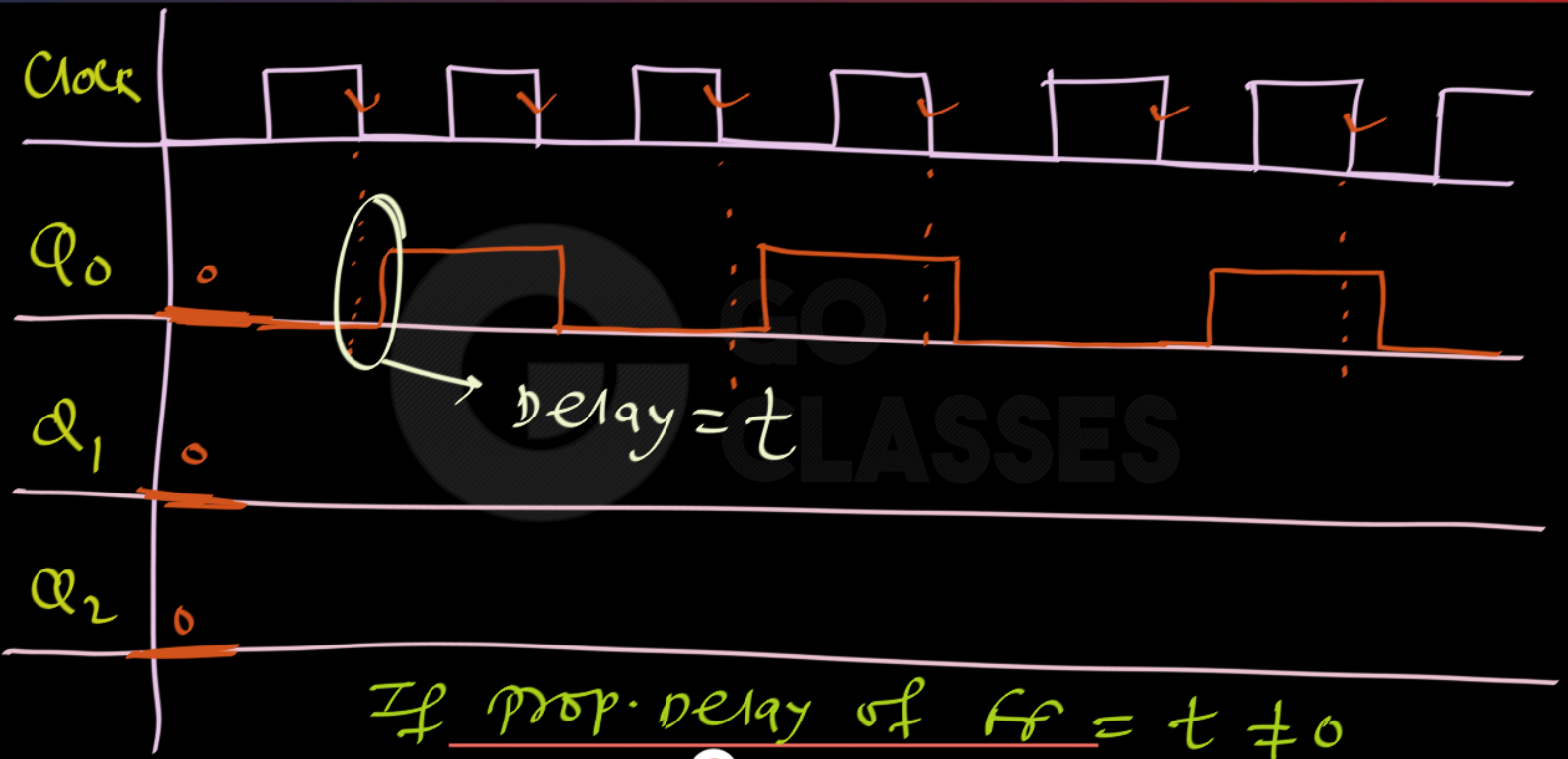
(2) If ffs are -Ve Edge Trig : $Q_1$ to clock of $Q_2$

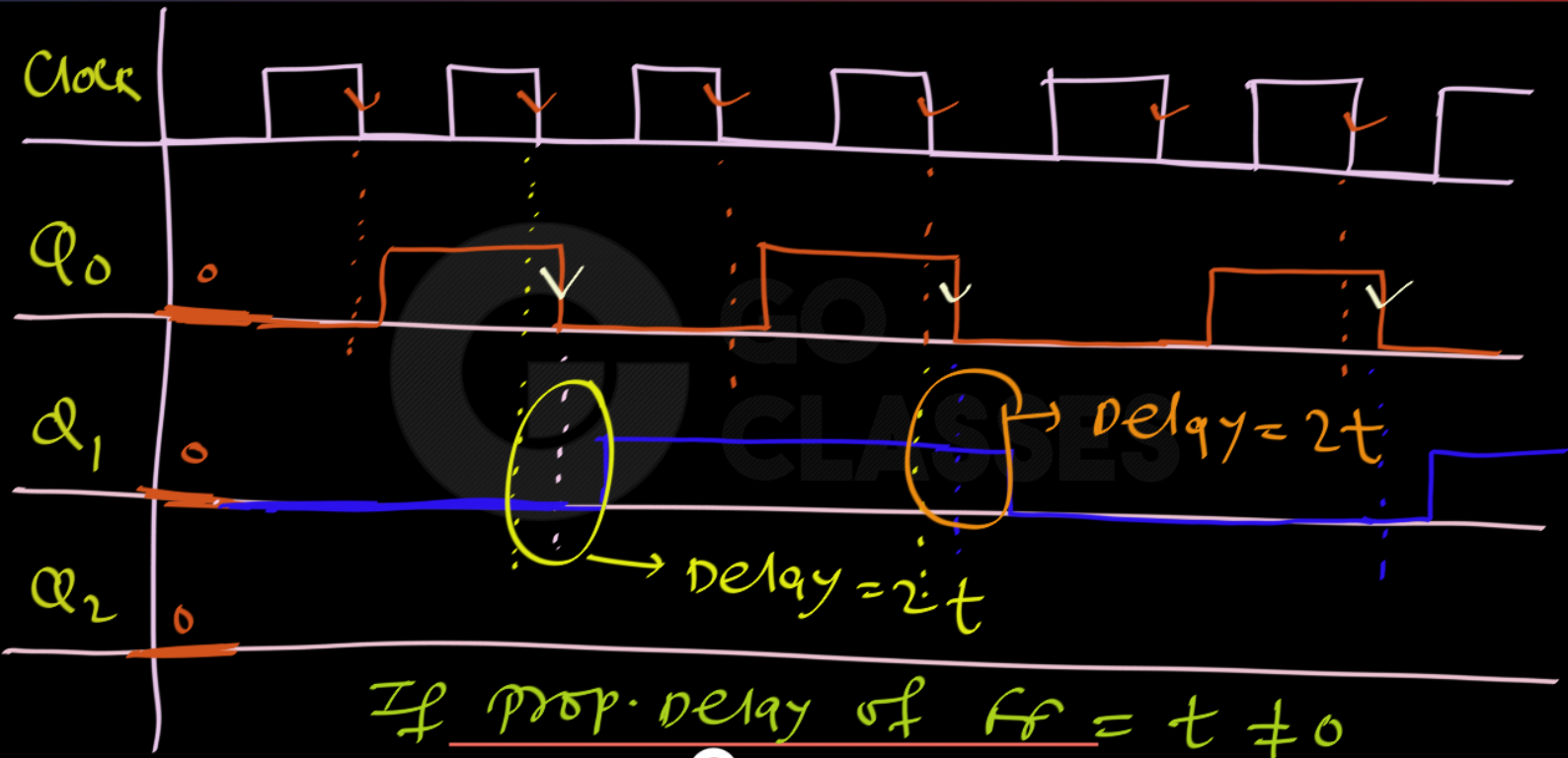
# 3-bit Binary Counter :

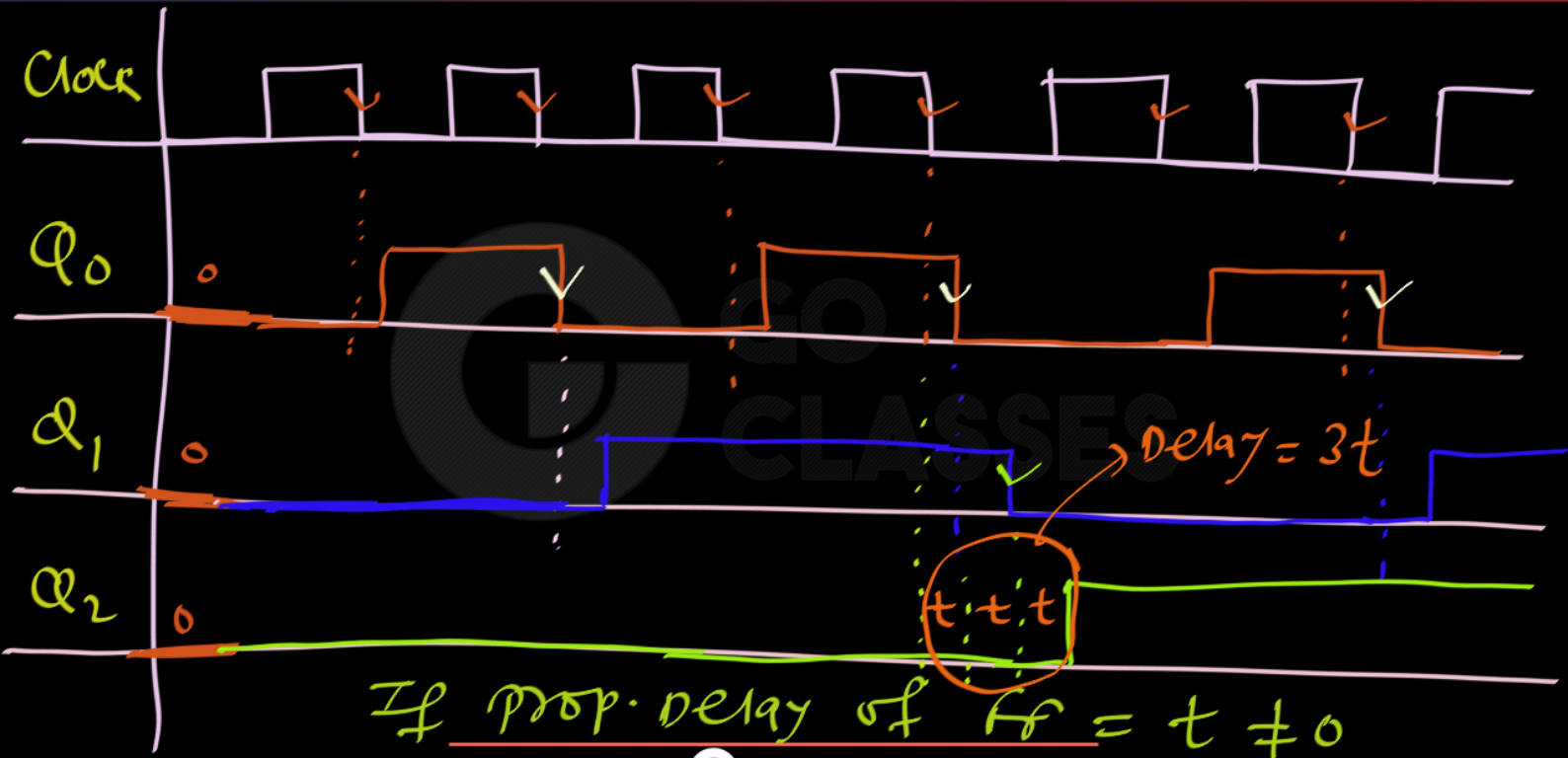


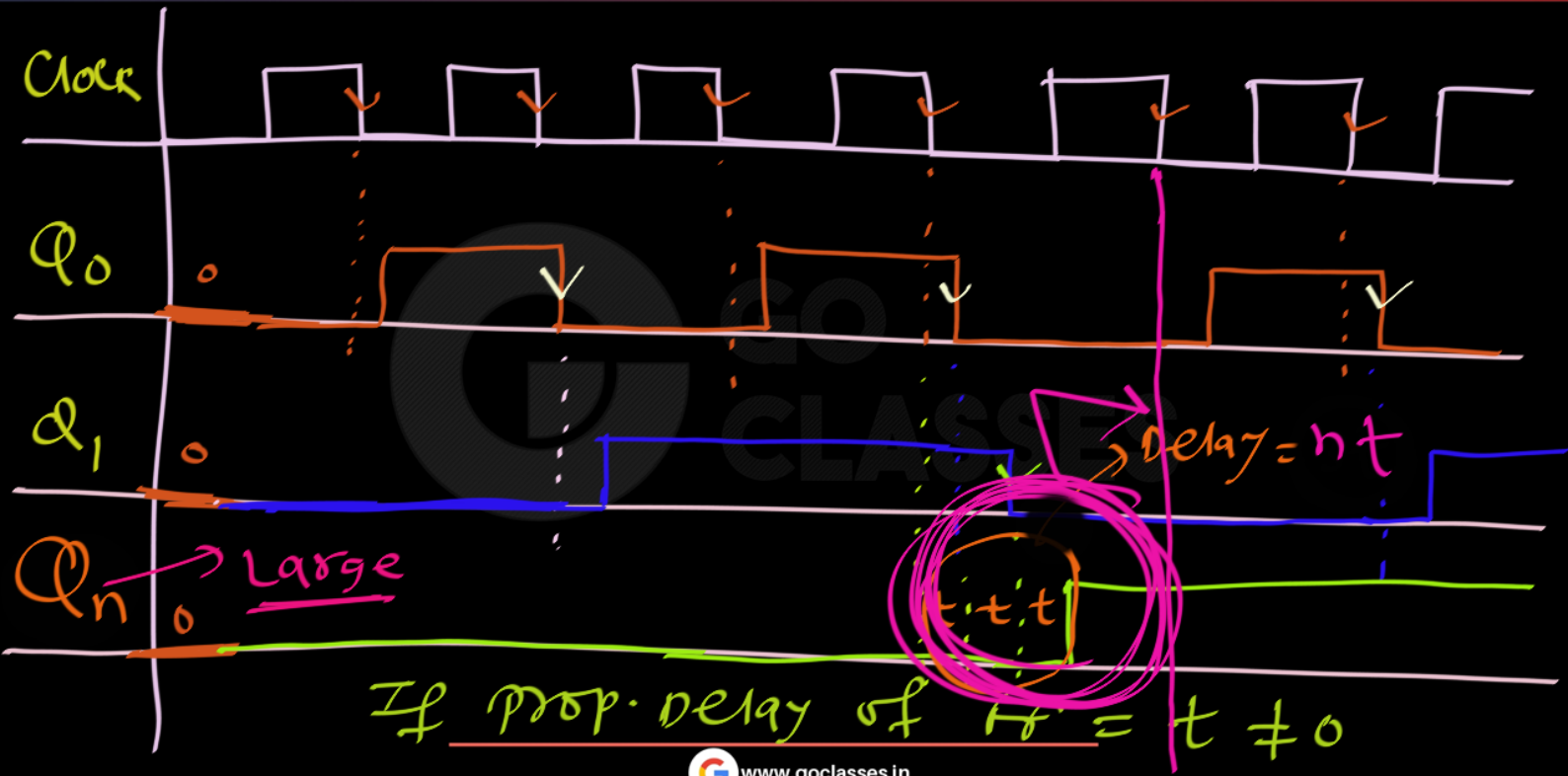

Clock
LSB
$Q_0$
$Q_1$
$Q_2$
MSB
Assume No Prop Delay.


000
001
010
011
100
101
110
111


Clock
$Q_0$
0
Delay = t
$Q_1$
0
$Q_2$
0
If prop. delay of ff = t ≠ 0


Clock
$Q_0$
0
$Q_1$
0
$Q_2$
0
Delay = 2t
Delay = 2t
If prop. delay of ff = t ≠ 0


Clock
$Q_0$
0
$Q_1$
0
$Q_2$
0
Delay = 3t
t, t, t
If prop. delay of ff = t ≠ 0


Clock
$Q_0$
0
$Q_1$
0
$Q_n$
0
Large
Delay = nt
t, t, t
If prop. delay of FF = t ≠ 0

# n-ff Ripple Counter: (n-bit Asyn. Counter)

for proper functioning:

Clock time Period $\geq n t$

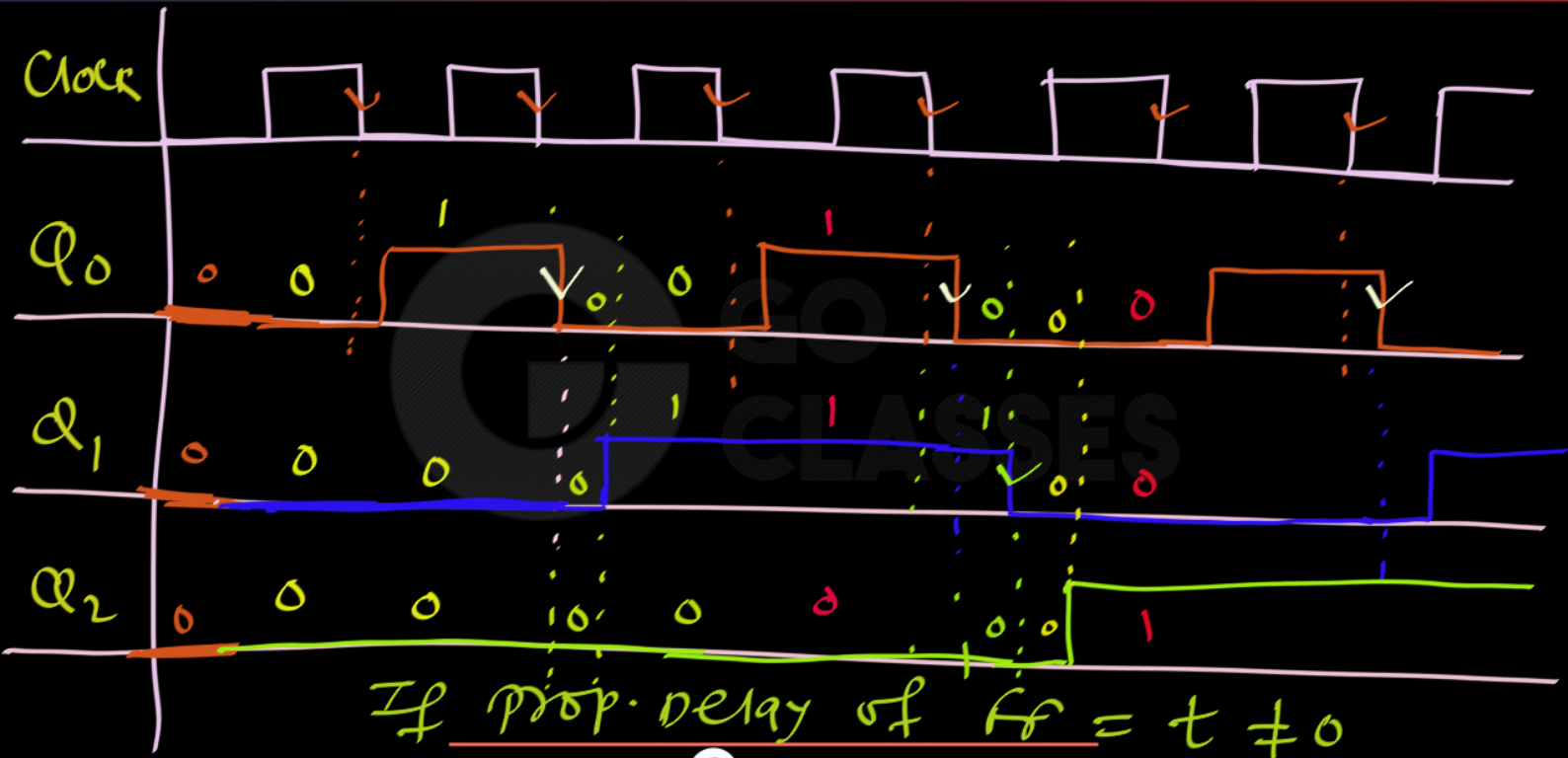
$t$ → Prop. Delay of a ff

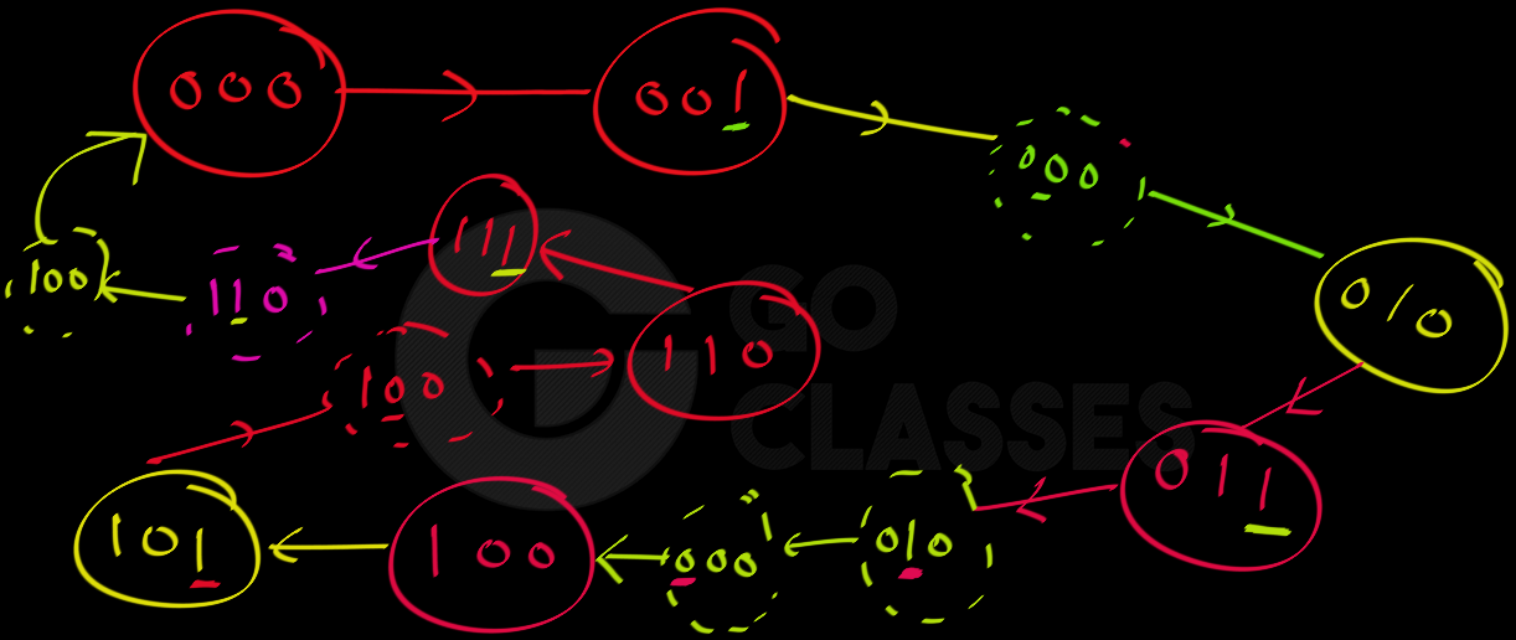
# n-ff Ripple Counter: (n-bit Asyn. Counter)

for proper functioning:

Clock frequency $\leq \frac{1}{nt}$

$t$ → Prop. Delay of a ff


Clock
$Q_0$
$Q_1$
$Q_2$
If prop. delay of ff = t ≠ 0


000
001
000
111
100
110
010
110
100
011
101
100
000
010

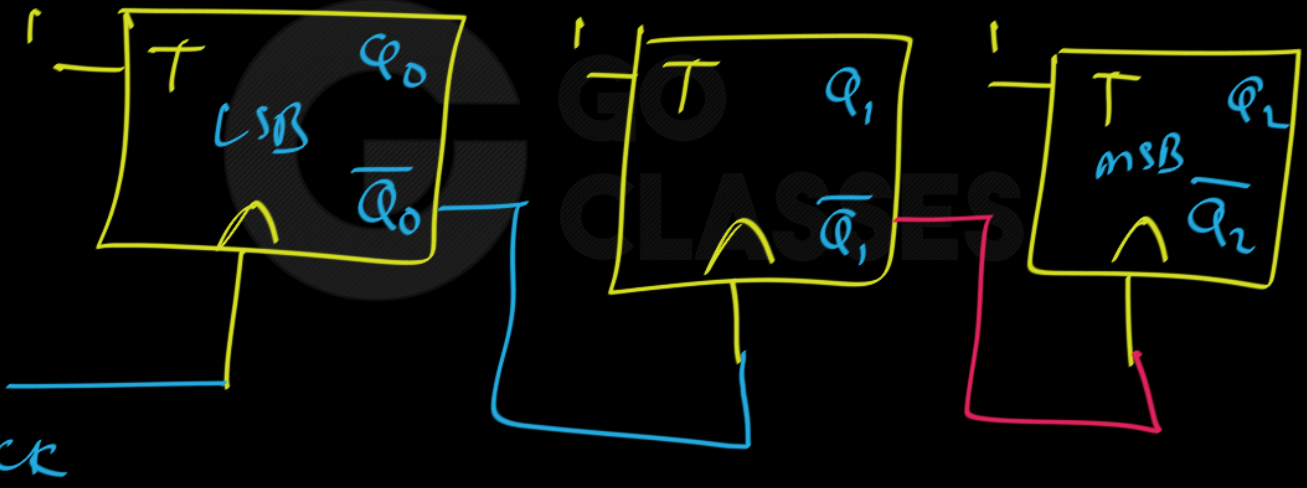
Q: why Asynchronous Counter ?
Call it

⇒ Same Clock not operating all ffs.

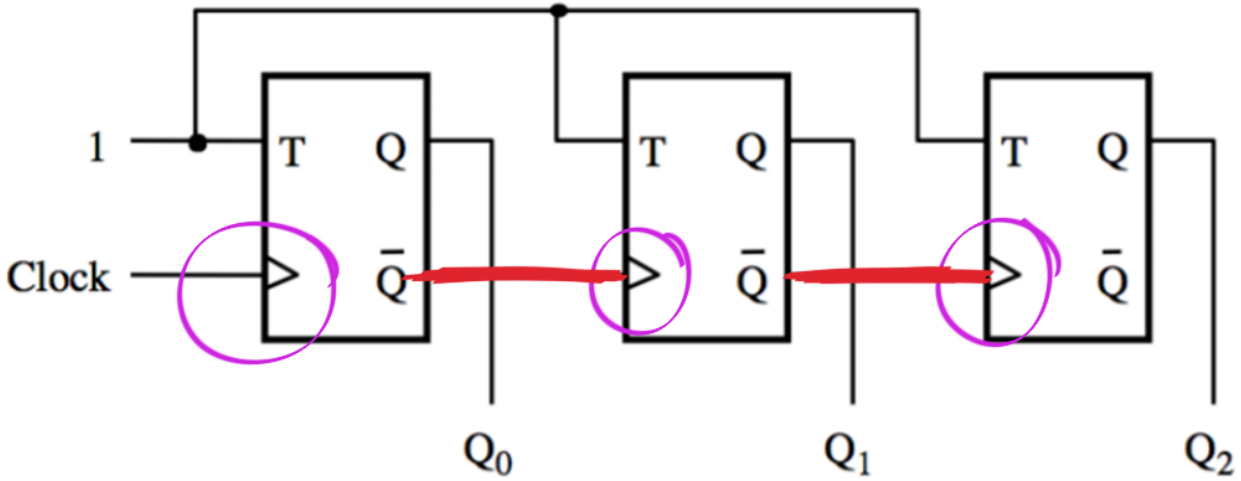
Q: Why Asynchronous Counter is called Ripple Counter ?

Q: Why Asynchronous Counter is Called Ripple Counter ?

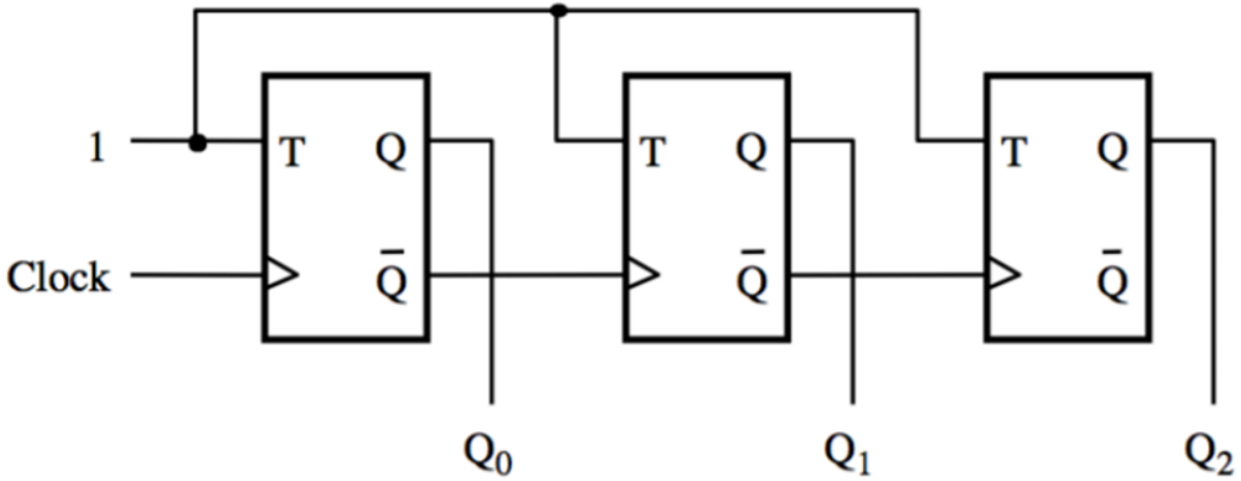
# 3-bit binary Counter;



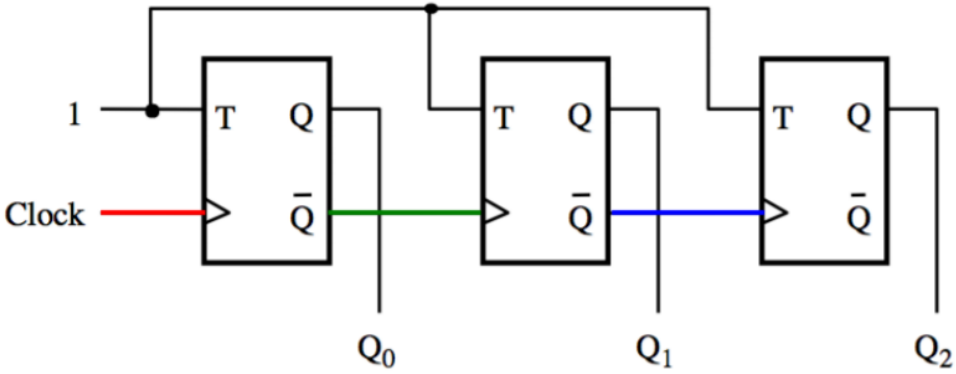
# A three-bit up-counter



# A three-bit up-counter



# A three-bit up-counter

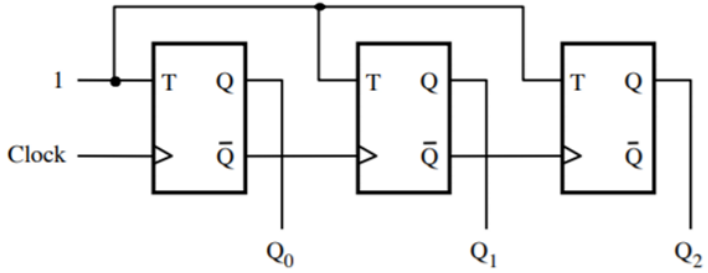



The first flip-flop changes on the positive edge of the clock

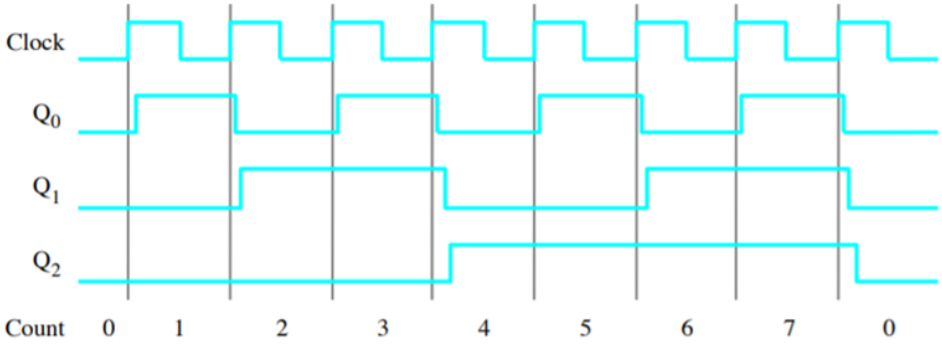
The second flip-flop changes on the positive edge of $\overline{Q}_0$

The third flip-flop changes on the positive edge of $\overline{Q}_1$

# A three-bit up-counter

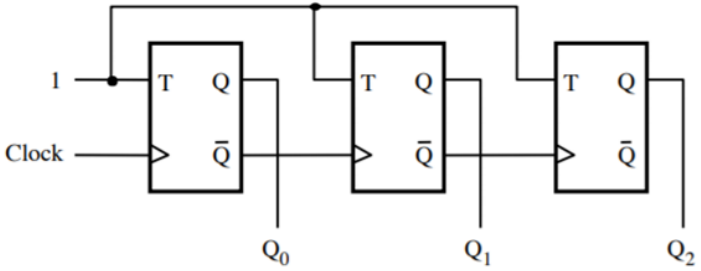



(a) Circuit

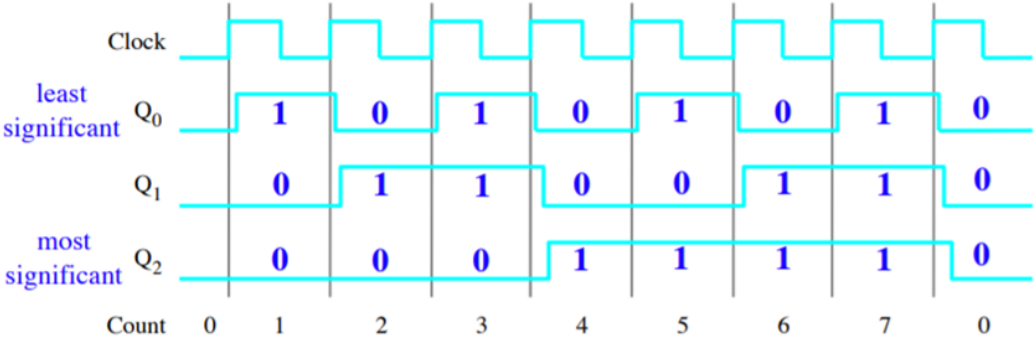



(b) Timing diagram

# A three-bit up-counter

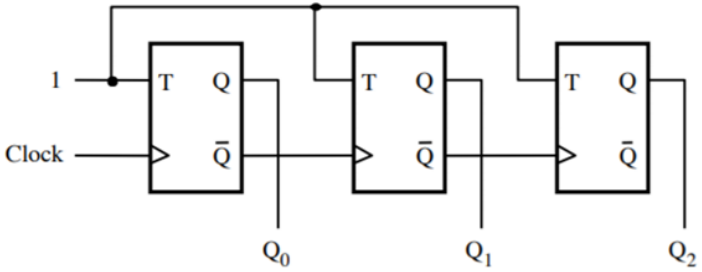



(a) Circuit

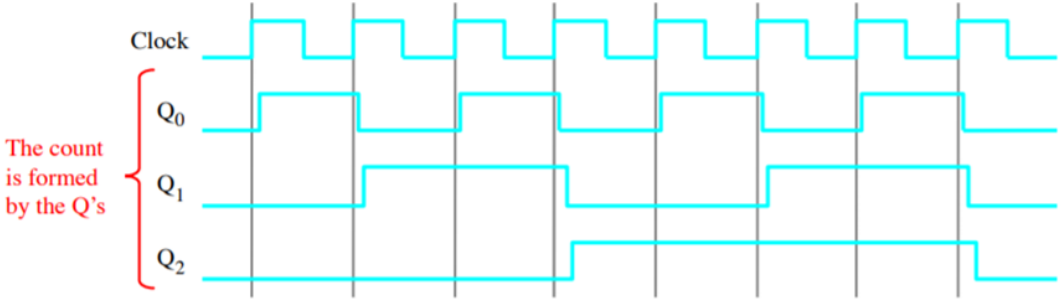



(b) Timing diagram

# A three-bit up-counter

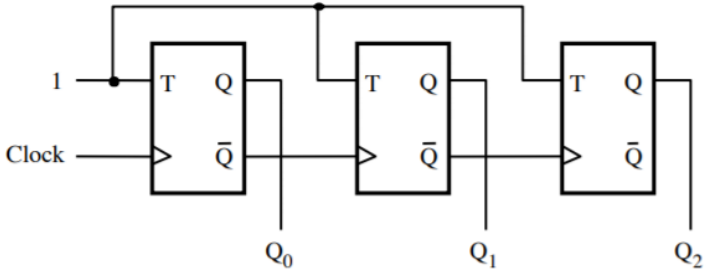



(a) Circuit

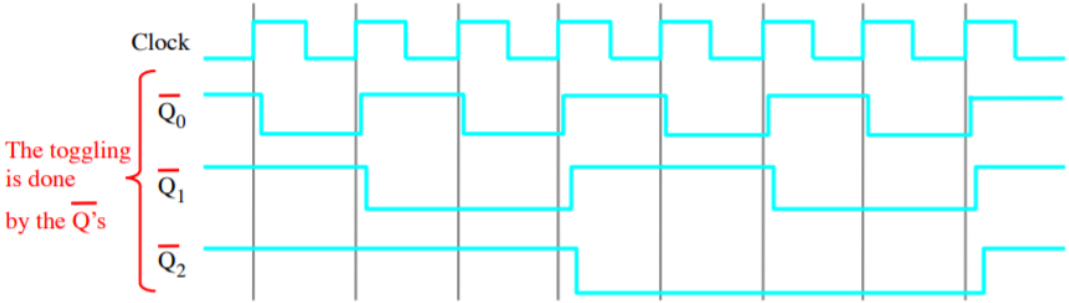



(b) Timing diagram

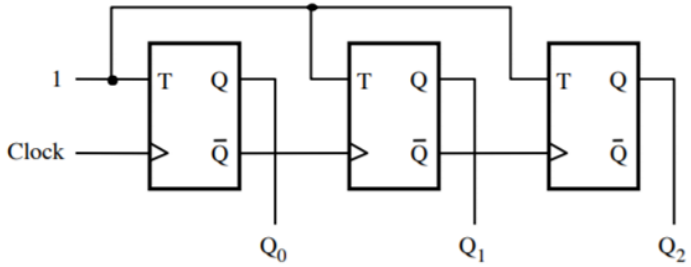
# A three-bit up-counter



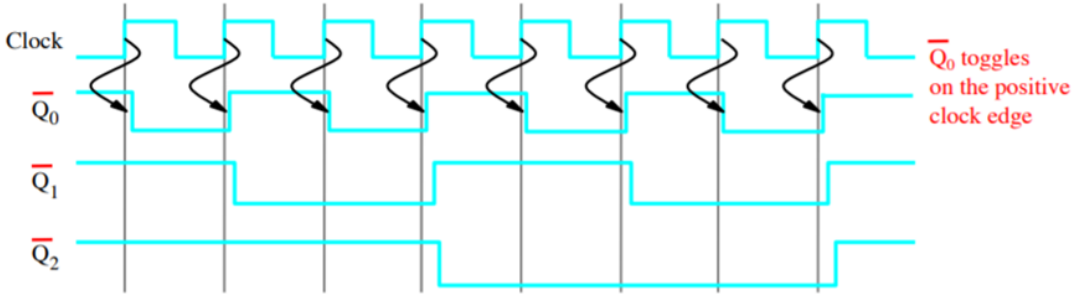

(a) Circuit



# A three-bit up-counter

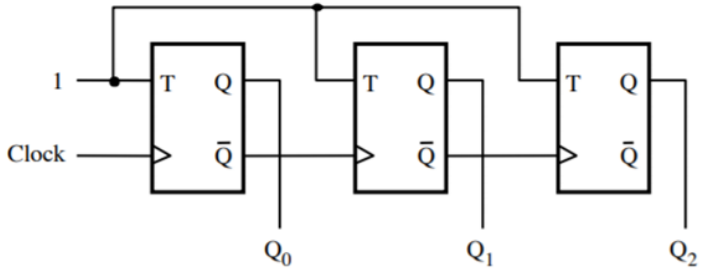



(a) Circuit

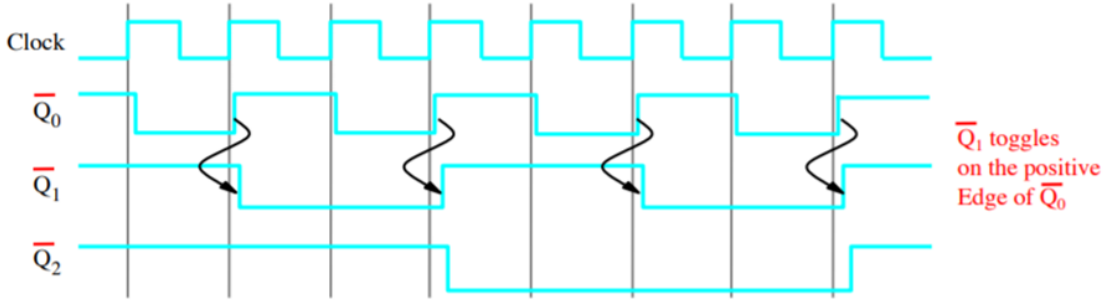



(b) Timing diagram

# A three-bit up-counter

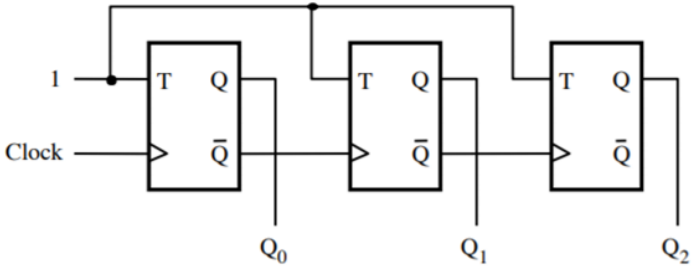



(a) Circuit

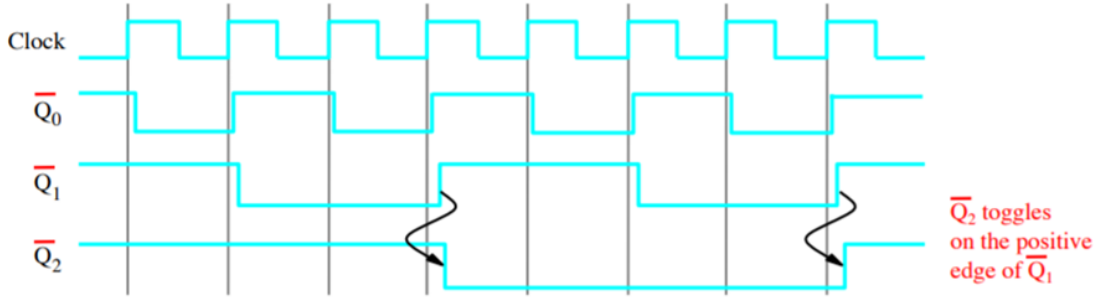



(b) Timing diagram

# A three-bit up-counter

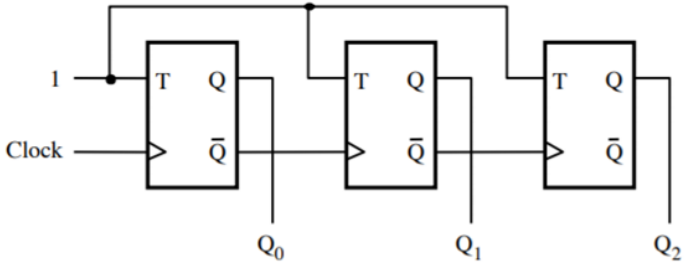



(a) Circuit



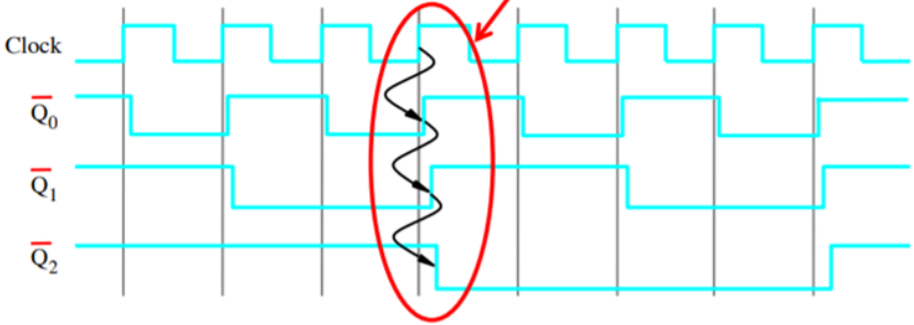

(b) Timing diagram

# A three-bit up-counter



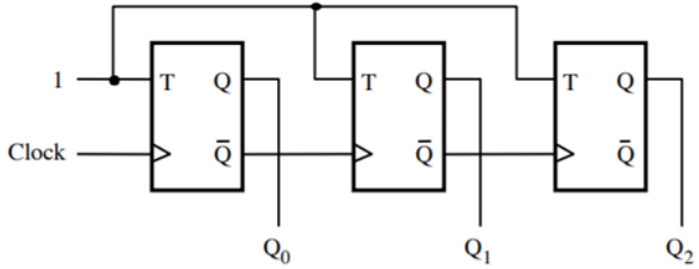

(a) Circuit

The propagation delays get longer



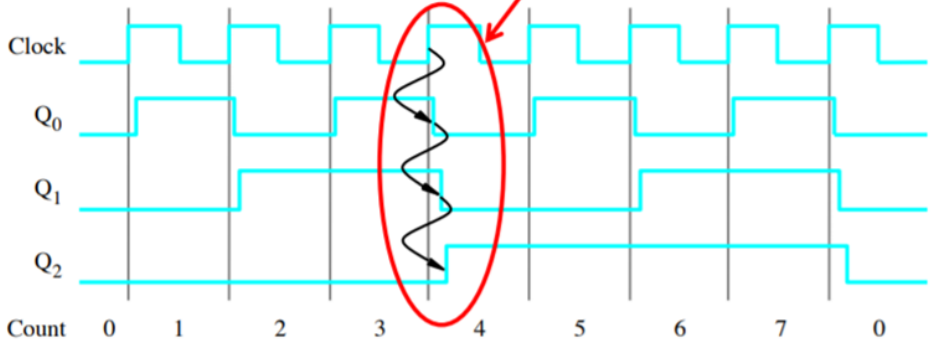

(b) Timing diagram

# A three-bit up-counter




(a) Circuit

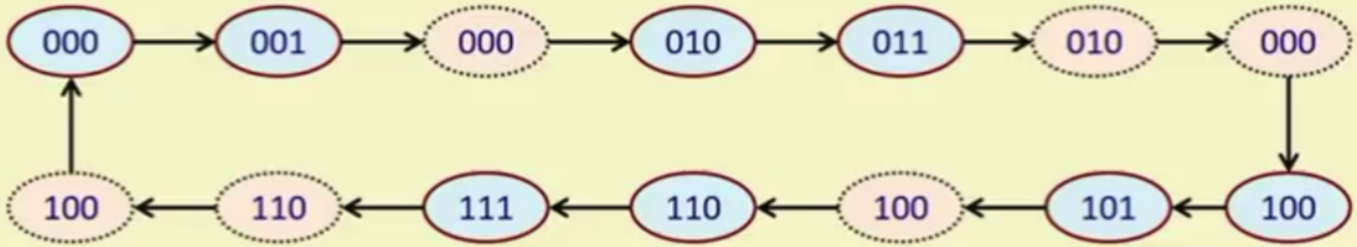
The propagation delays get longer




(b) Timing diagram

# Transient States During Counting

- Consider a 3-bit ripple counter.
- The normal counting sequence is: 000, 001, 010, 011, 100, 101, 110,110, 000, ...
- However, in practice, because of the delays of the T flip-flops, there are several transient states (shown in pink) that appear.



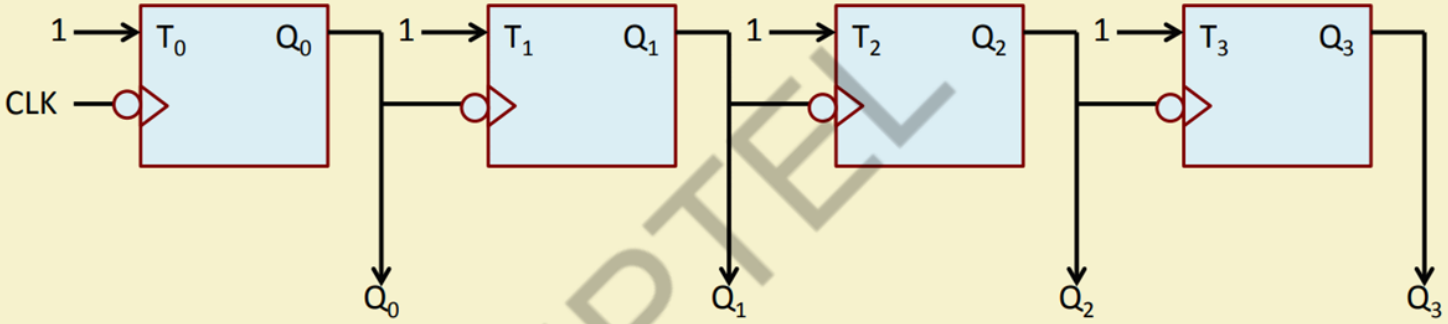
# Asynchronous Counter (Ripple Counter)

- This type of counter is the easiest to design, and requires the least amount of hardware.
- The counter is called *asynchronous* because the flip-flops are not driven by the same clock signal, and hence do not change state simultaneously.
  - Constructed using T flip-flops.
- It is called *ripple counter* because when the counter, for example, goes from *1111* to *0000*, it temporarily goes through a number of intermediate states:
  *1111* → 1110 → 1100 → 1000 → *0000*.
  - Can lead to unwanted transitions if the outputs drive some other circuit.
  - Glitches are possible.

# Design of 4-bit Ripple Counter (Up Counting)

- We make an observation:
  - During counting, whenever a bit position changes from 1 to 0, the next higher bit position changes state.
  - This feature can be directly implemented using a T flip-flop with *negative-edge triggered clock*.

| | |
|---|---|
| 0: 0 0 0 0 | 8: 1 0 0 0 |
| 1: 0 0 0 1 | 9: 1 0 0 1 |
| 2: 0 0 1 0 | 10: 1 0 1 0 |
| 3: 0 0 1 1 | 11: 1 0 1 1 |
| 4: 0 1 0 0 | 12: 1 1 0 0 |
| 5: 0 1 0 1 | 13: 1 1 0 1 |
| 6: 0 1 1 0 | 14: 1 1 1 0 |
| 7: 0 1 1 1 | 15: 1 1 1 1 |






*If we are designing an up counter using positive edge-triggered T flip-flops, we simply connect the Q' output of a stage to the clock input of the next stage.*

- Some observations:
  - Suppose $f$ denotes the frequency of the pulse train applied at *CLK* input.
  - Then, frequency of pulse train at $Q_0 = f/2$
  - Frequency of pulse train at $Q_1 = f/4$
  - Frequency of pulse train at $Q_2 = f/8$
  - Frequency of pulse train at $Q_3 = f/16$
- An $n$-bit binary counter is also called *modulo-$2^n$* counter or *divide-by-$2^n$* counter.

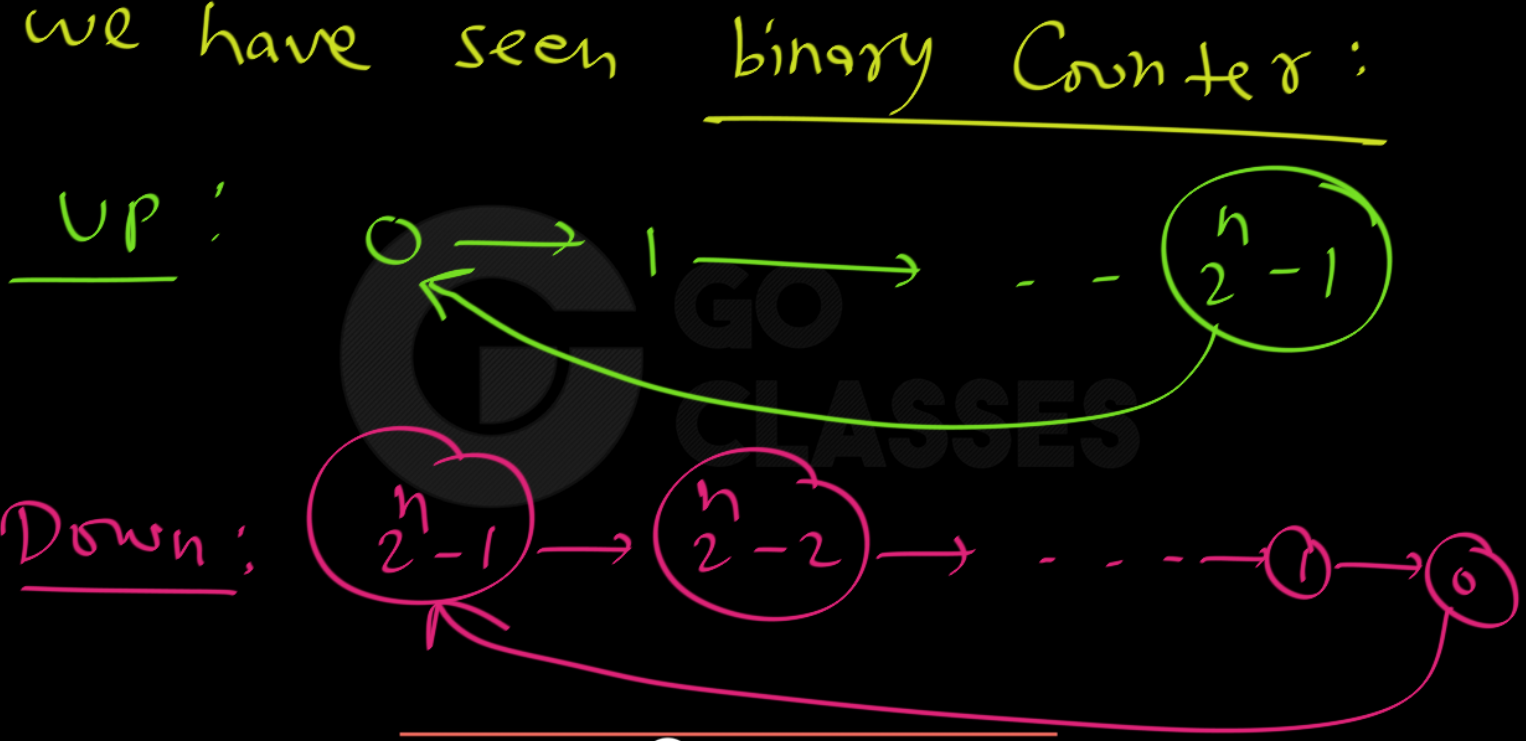
# Design of 4-bit Ripple Counter (Down Counting)

- We make an observation:
  - During counting, whenever a bit position changes from 0 to 1, the next higher bit position changes state.
  - This feature can be directly implemented using a T flip-flop with positive-edge triggered clock.

| | |
|---|---|
| 15: 1 1 1 1 | 7: 0 1 1 1 |
| 14: 1 1 1 0 | 6: 0 1 1 0 |
| 13: 1 1 0 1 | 5: 0 1 0 1 |
| 12: 1 1 0 0 | 4: 0 1 0 0 |
| 11: 1 0 1 1 | 3: 0 0 1 1 |
| 10: 1 0 1 0 | 2: 0 0 1 0 |
| 9: 1 0 0 1 | 1: 0 0 0 1 |
| 8: 1 0 0 0 | 0: 0 0 0 0 |

- If we use negative edge-triggered T flip-flops:
  - Connect *Q'* output of a stage to *CLK* input of next stage.
- If we use positive edge-triggered T flip flops:
  - Connect *Q* output of a stage to *CLK* input of next stage.

we have seen binary Counter :

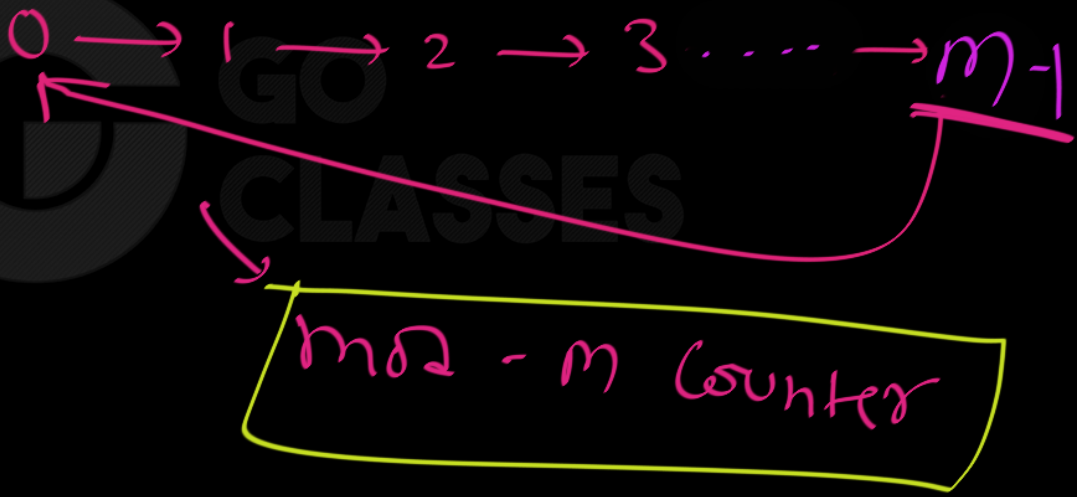


Q: How to create Ripple Counter for


6 Distinct States

Mod 6 Counter

Q: How to create Ripple Counter for



# Next Topic:

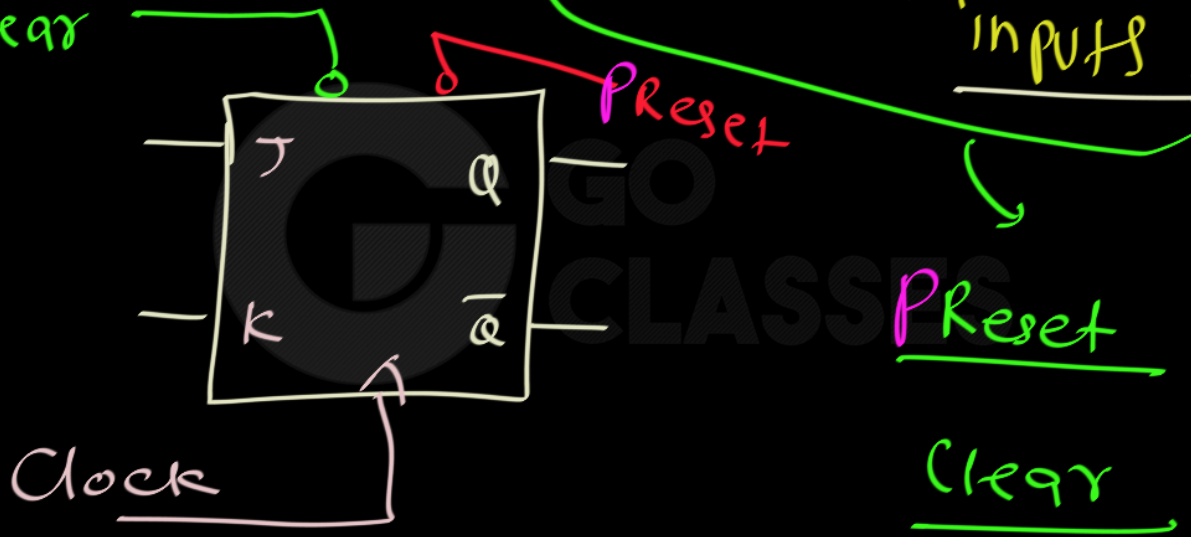
## Asynchronous Counters

## 2. Modulo M counter

# Motivation

- An n-bit counter counts from 0, 1, …, $2^n$-1
- For example a 3-bit counter counts up as follow
  - 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, …
- What if we want it to count like this
  - 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, …
- In other words, what is the cycle is not a power of 2?

Every ff has two Asynchronous inputs :



Every FF has two Asynchronous inputs :

Clear :
Reset :

Active low
||
when 0, they do their work

Do not wait for Clock Edge

Every FF has two Asynchronous inputs :

Clear : 0
Reset : 1

→ will clear the FF.

o/p becomes 0.

immediately

Every ff has two Asynchronous inputs :

Clear : 1
Reset : 0

→ will preset the ff.

o/p becomes 1.

immediately

Every FF has two Asynchronous inputs :

Clear : 0
Reset : 0

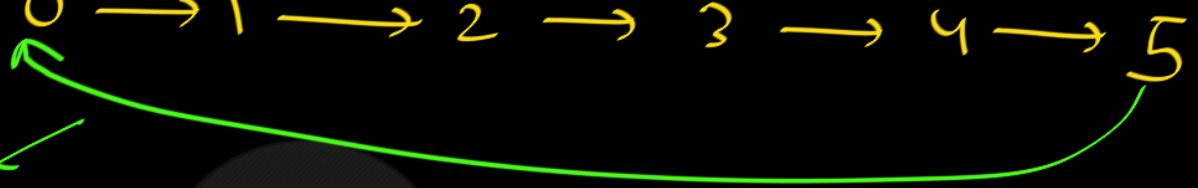
⟹ uncertain behaviour

→ forbidden

Every FF has two Asynchronous inputs :

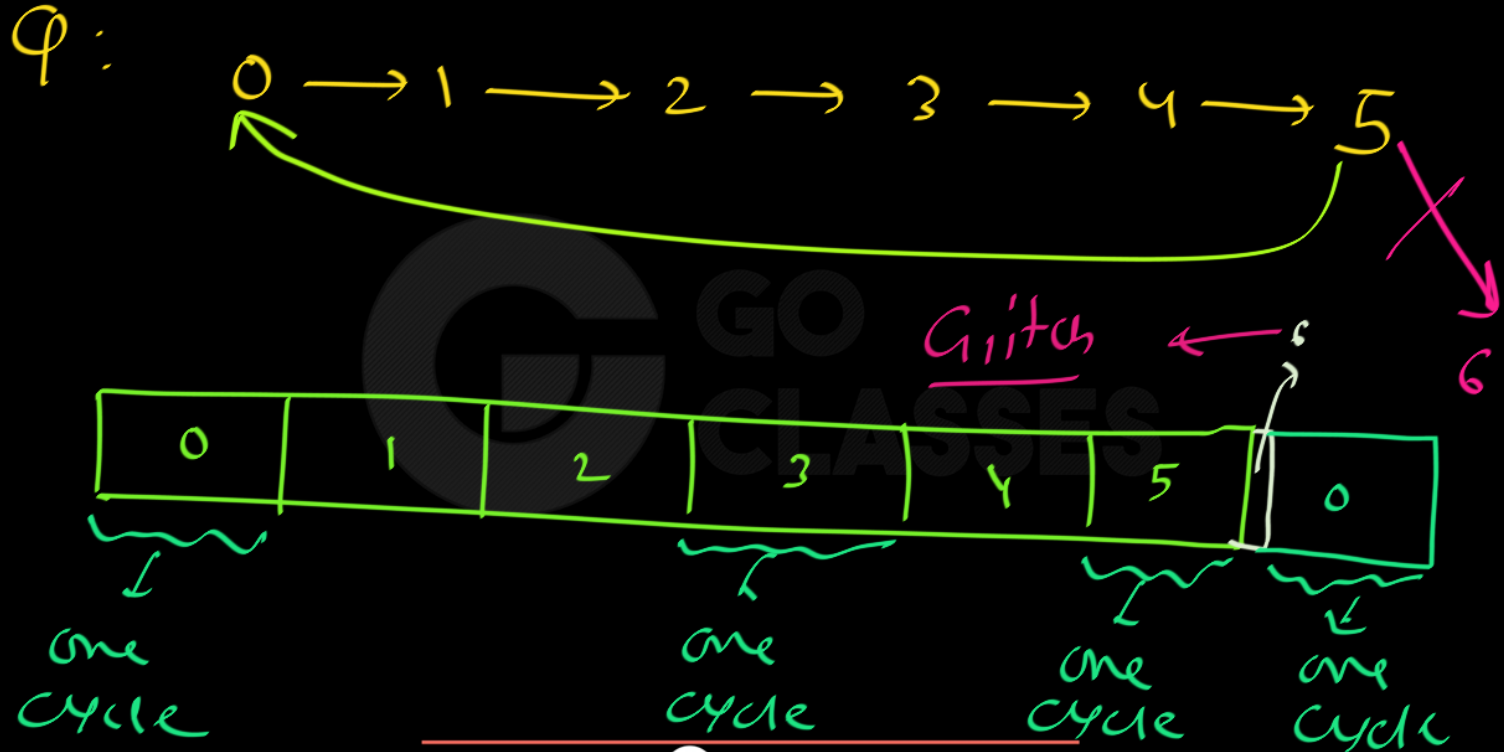
Clear : 1
Reset : 1

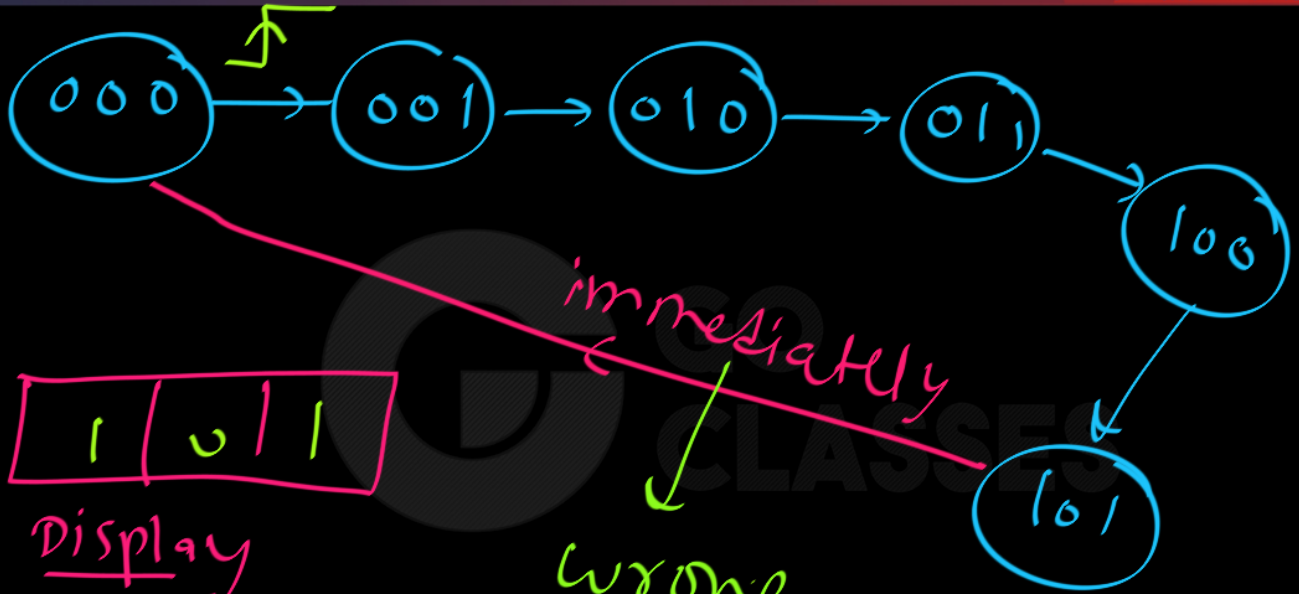
$\Rightarrow$ Normal FF behaviour happens

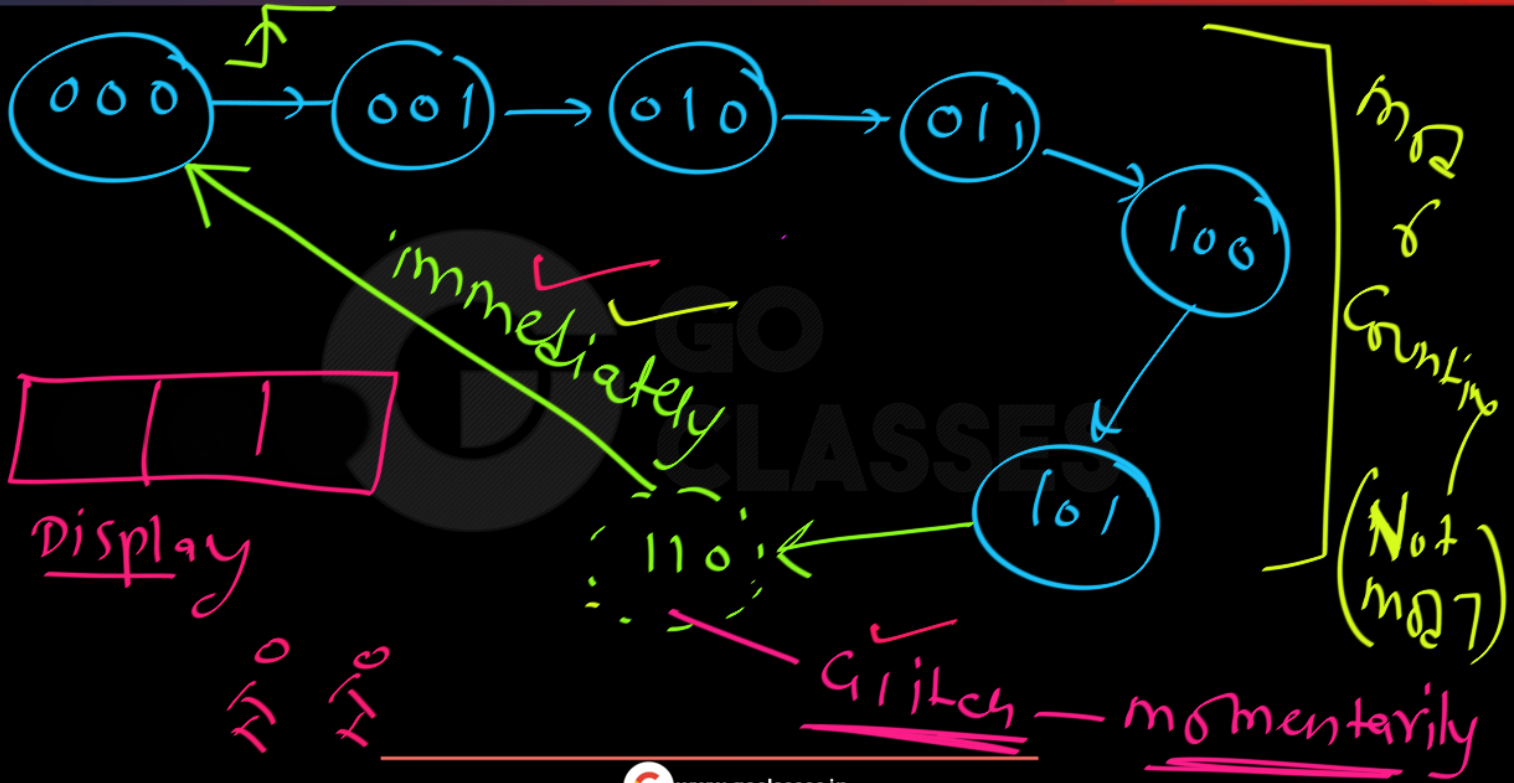
Q: 

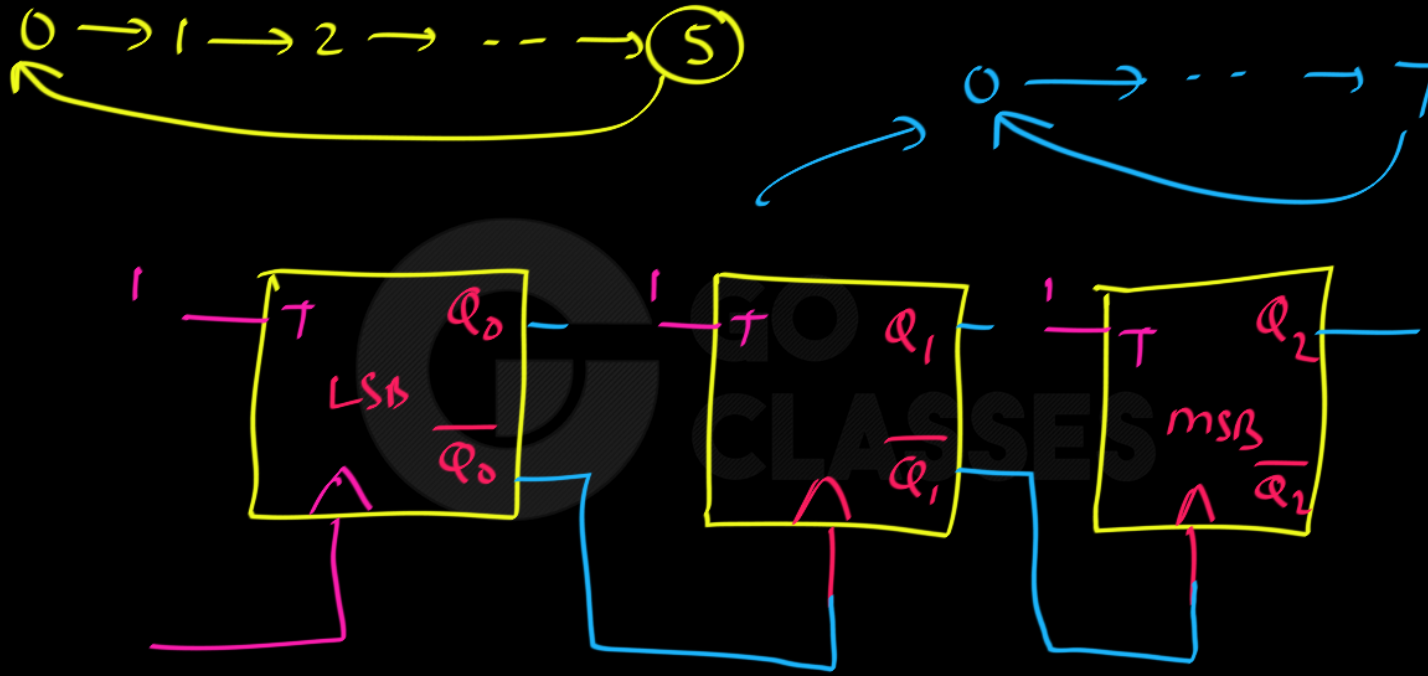

#States = 6

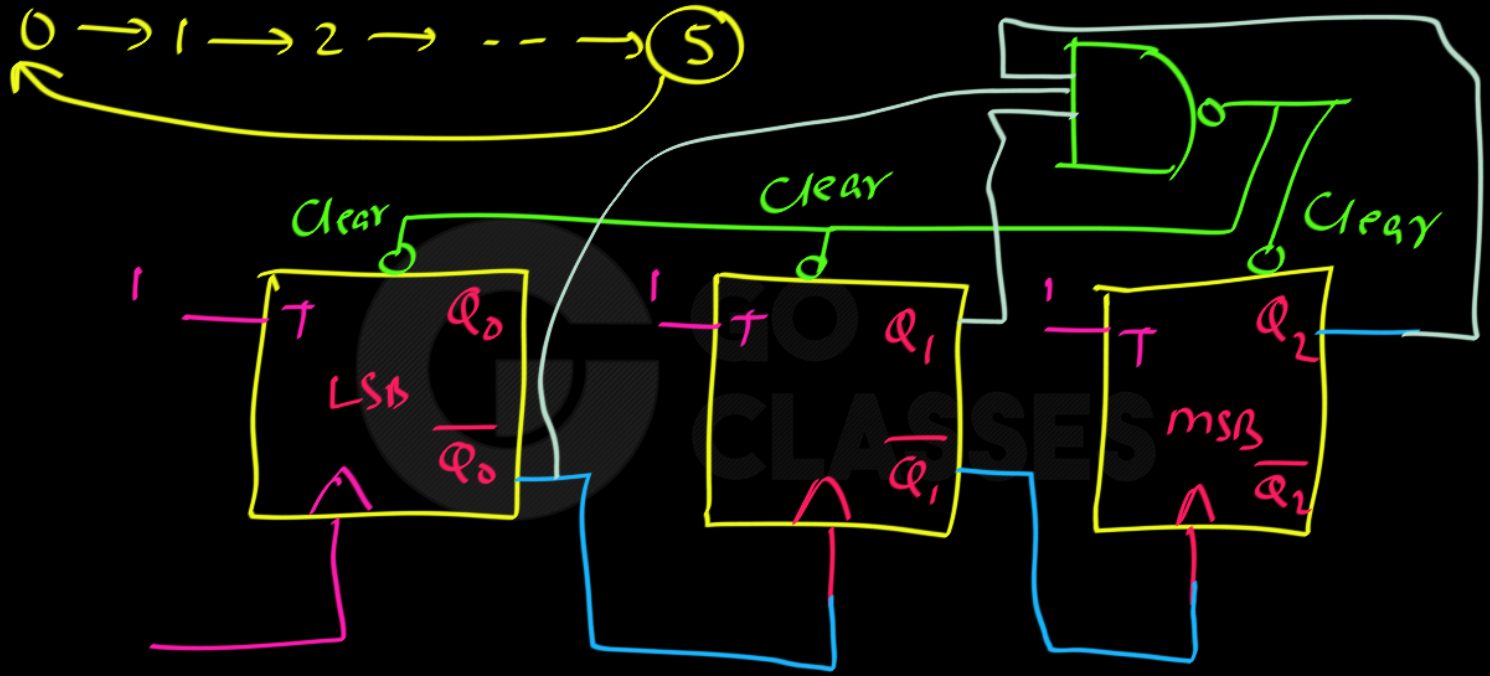
#ff needs = $\lceil \log_2 6 \rceil = 3$

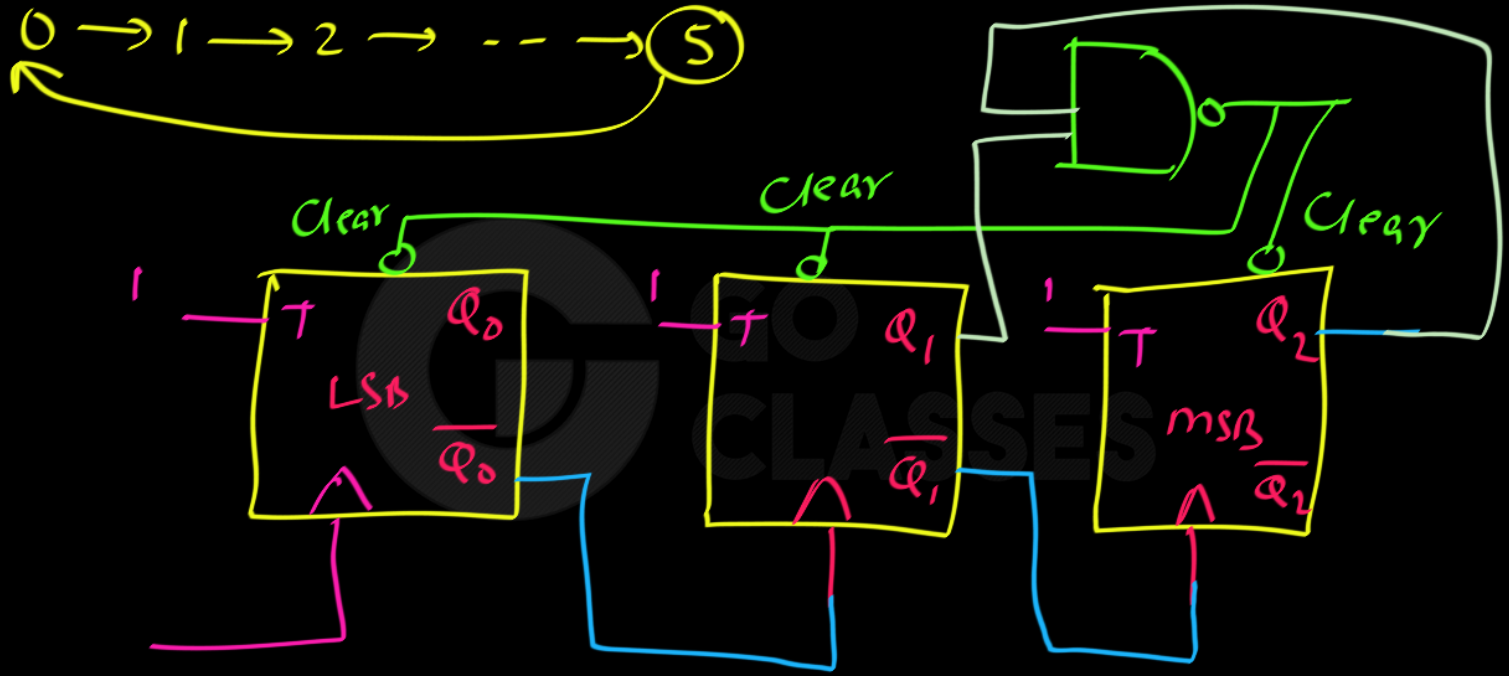

Q:
0 → 1 → 2 → 3 → 4 → 5
6
Glitch
0
1
2
3
4
5
0
1
One cycle
One cycle
One cycle
One cycle


000
001
010
011
100
101
immediately
1 0 1 1
Display
wrong


000
001
010
011
100
101
110
immediately
Display
Glitch — momentarily
Not MOD 7

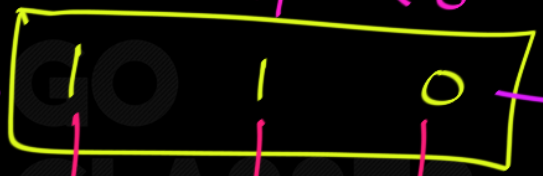

0 → 1 → 2 → - - → 5
0 → - - → 7
1
T
$Q_0$
LSB
$\overline{Q_0}$
1
T
$Q_1$
$\overline{Q_1}$
1
T
$Q_2$
MSB
$\overline{Q_2}$


0 → 1 → 2 → - - - → 5
Clear
Clear
Clear
1
T
$Q_0$
LSB
$\overline{Q_0}$
1
T
$Q_1$
$\overline{Q_1}$
1
T
$Q_2$
MSB
$\overline{Q_2}$


0 → 1 → 2 → - - - → 5
Clear
Clear
Clear
1
T
$Q_0$
LSB
$\overline{Q_0}$
1
T
$Q_1$
$\overline{Q_1}$
1
T
$Q_2$
MSB
$\overline{Q_2}$

Idea:

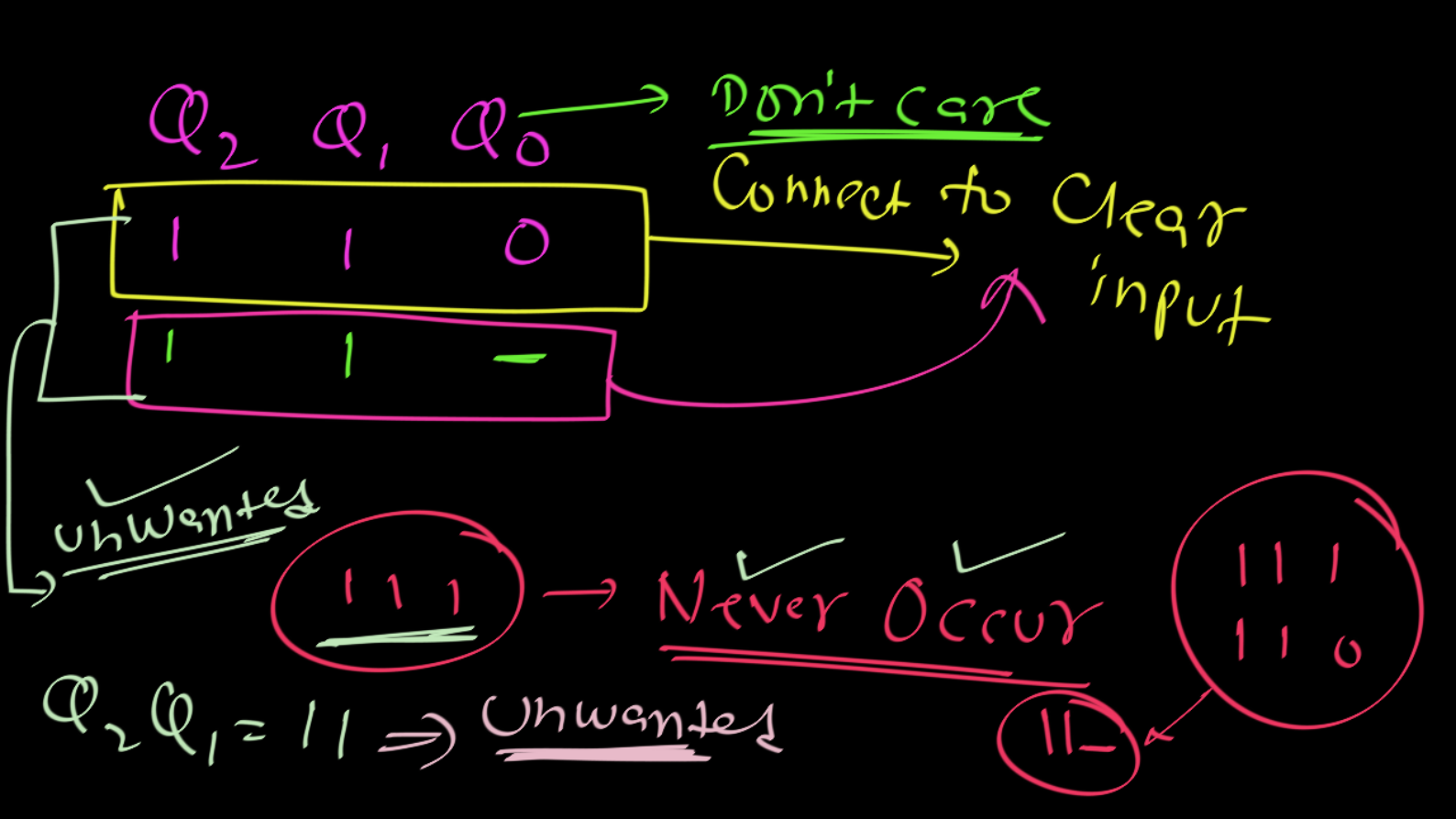
As soon as $Q_2\ Q_1\ Q_0 = 6$

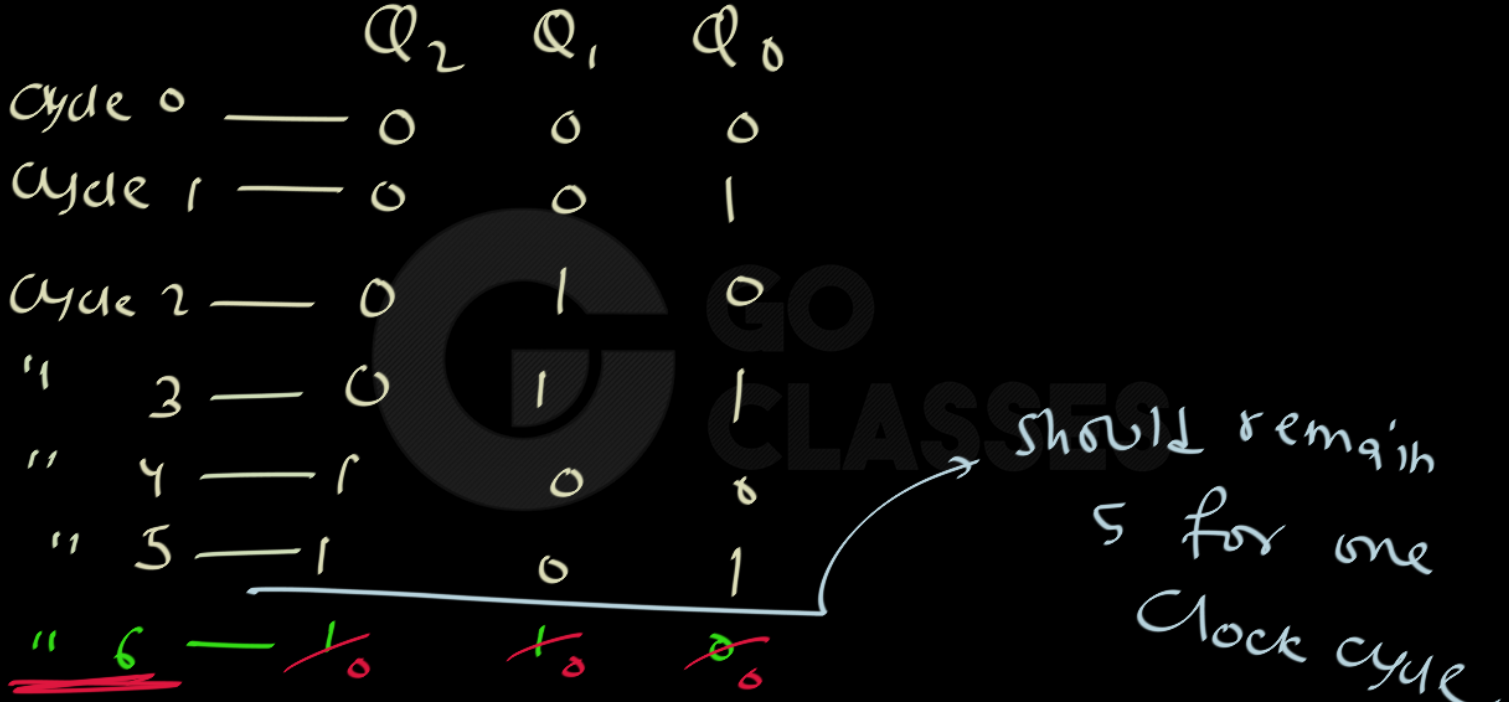



UnDesired

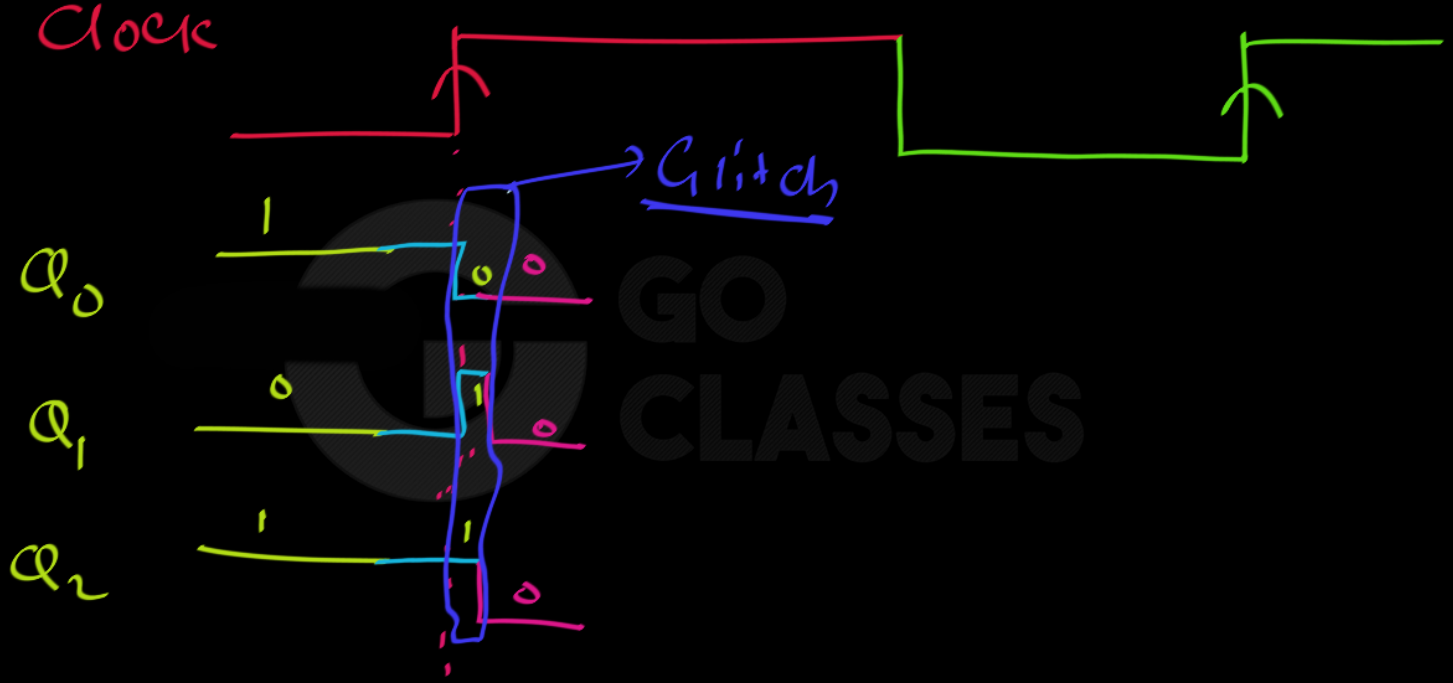
'immediately'


Clear of all ffs


$Q_2$ $Q_1$ $Q_0$
Don't care
1 1 0
Connect to Clear input
1 1 -
Unwanted
1 1 1
Never Occur
$Q_2Q_1 = 11 \Rightarrow$ Unwanted
1 1 1
1 1 0
1 1 -


$Q_2$ $Q_1$ $Q_0$
Cycle 0 — 0 0 0
Cycle 1 — 0 0 1
Cycle 2 — 0 1 0
" 3 — 0 1 1
" 4 — 1 0 0
" 5 — 1 0 1
" 6 — 1 1 0
should remain 5 for one clock cycle


Clock
$Q_0$
$Q_1$
$Q_2$


Clock
Glitch
$Q_0$
$Q_1$
$Q_2$
1
0
0
0
1
0
1
1
0

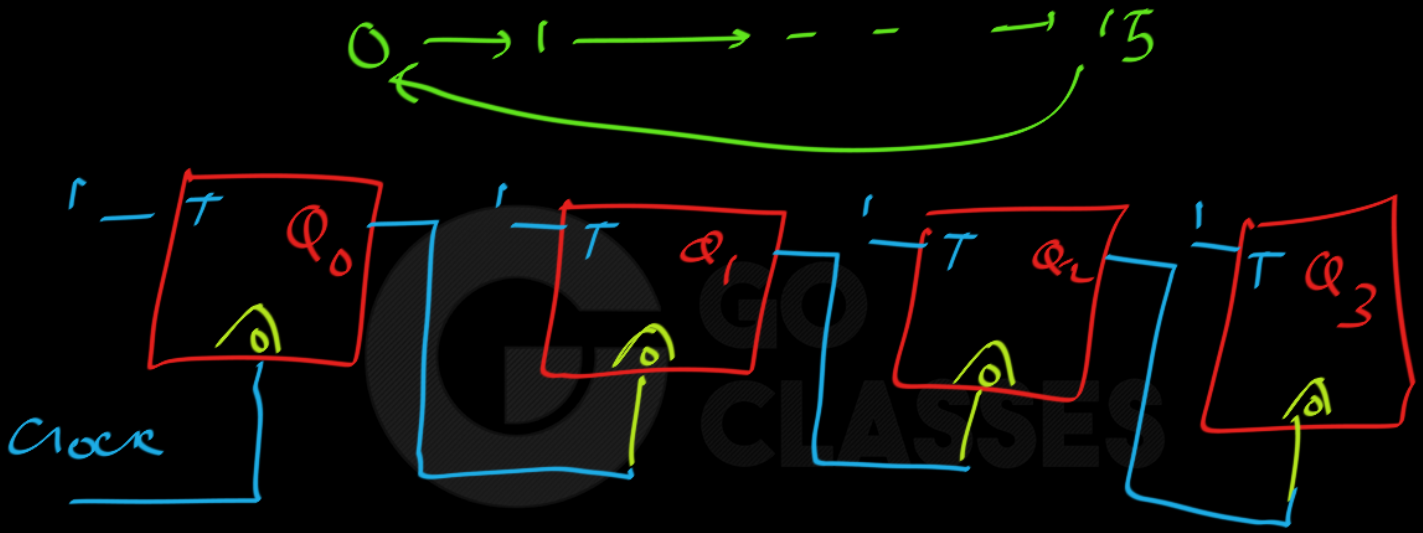
# Reset Synchronization

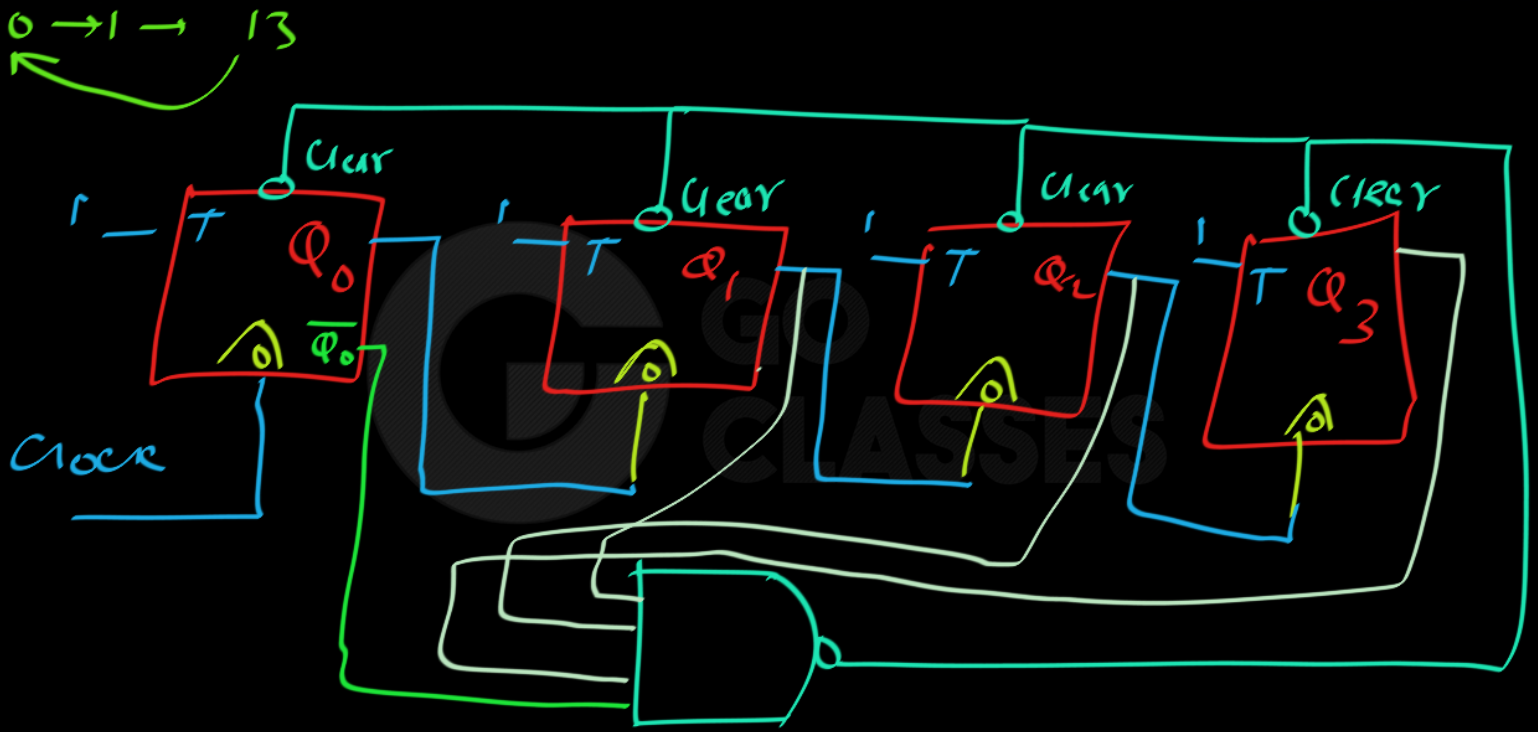
Q: 0 → 1 → 2 → ----- → 13

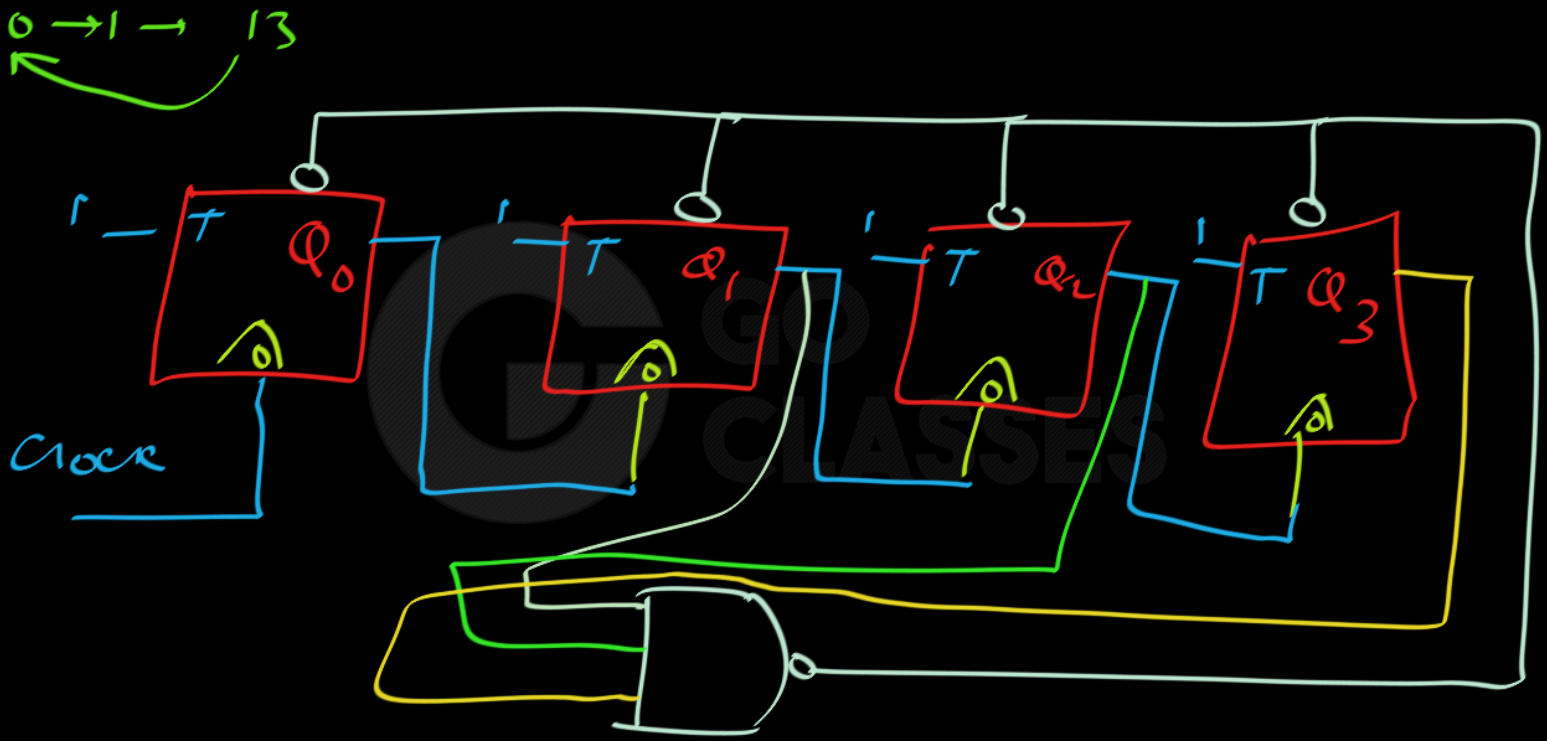
Ripple Counter

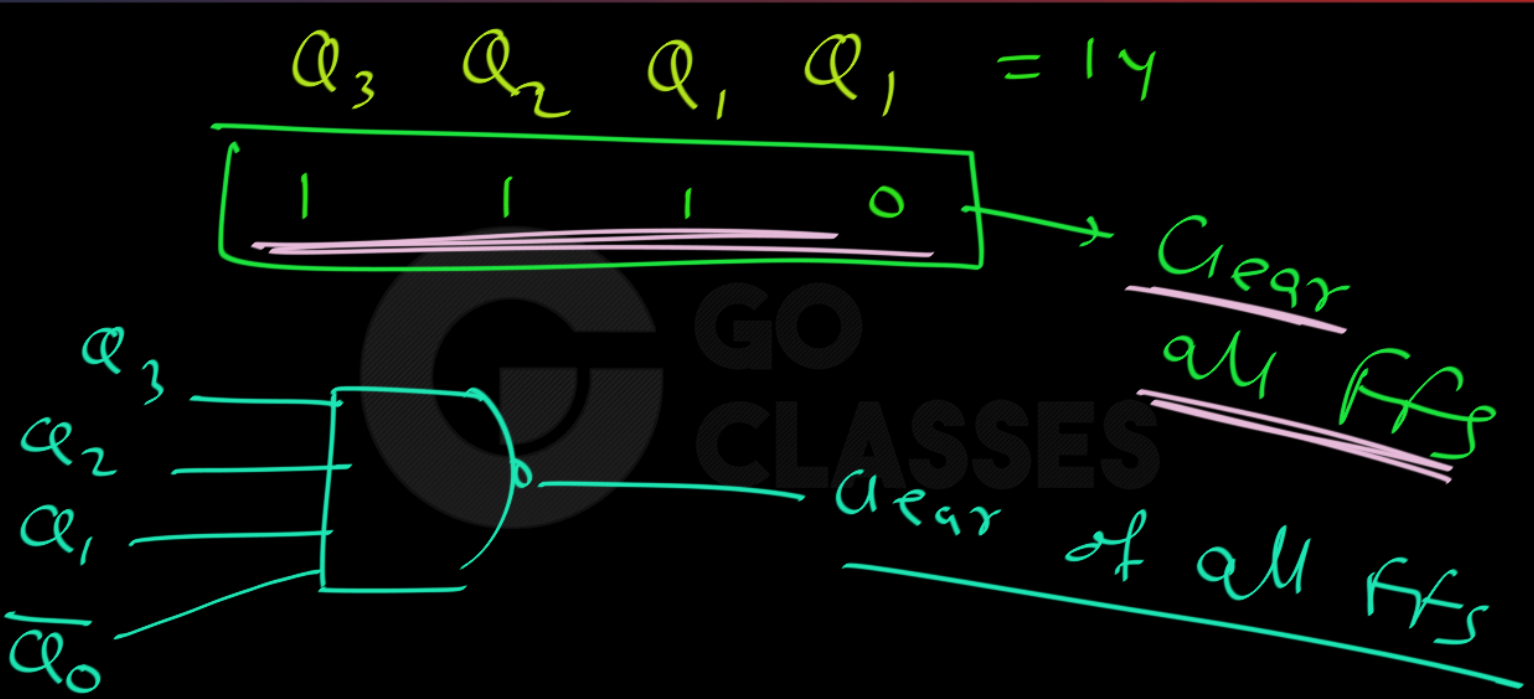
# ff needed = $\lceil \log_2 13 \rceil = 4$

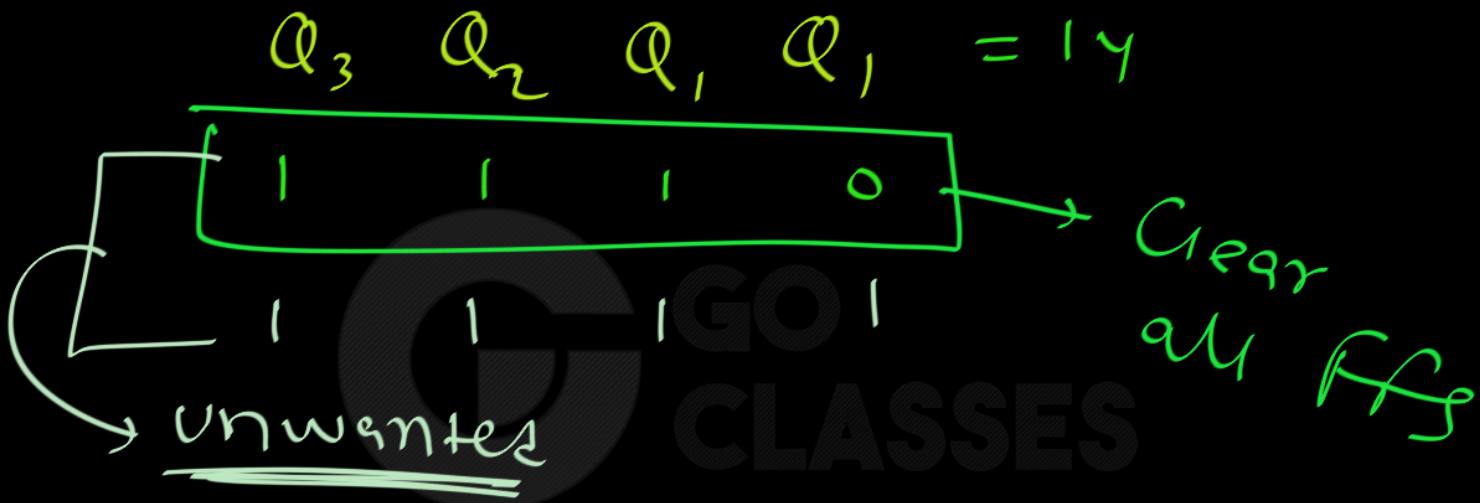

1 one cycle
one cycle
one cycle
14
one cycle


0
1
15
T
$Q_0$
$Q_1$
$Q_2$
$Q_3$
Clock


0 → 1 → 13
Clear
Clear
Clear
Clear
T
$Q_0$
$\overline{Q_0}$
$Q_1$
$Q_2$
$Q_3$
Clock


0 → 1 → 13
1
T
Q0
1
T
Q1
1
T
Q2
1
T
Q3
Clock


$Q_3$ $Q_2$ $Q_1$ $Q_1$ = 14
1 1 1 0
Clear all FFs
$Q_3$
$Q_2$
$Q_1$
$\overline{Q_0}$
Clear of all FFs



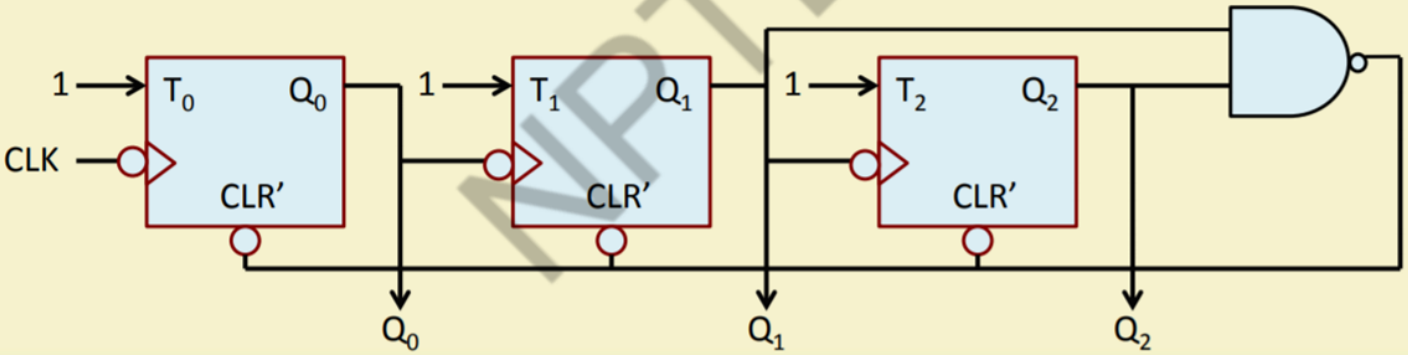

$Q_3 Q_2 Q_1 = 111 \Rightarrow$ Clear all ffs

# Design of Binary Ripple Counter of any Arbitrary Modulus

- Suppose we want to design a *modulo-M counter*, for any arbitrary value of *M*.
  - Counter will count up from *0* to *M-1*, and then back to *0*.
- Assume that the flip-flops have individual asynchronous *CLR'* inputs.
- Basic idea:
  - We first design a ripple counter with $\lceil log_2 M \rceil$ stages.
  - Then we design a gating circuit, which takes inputs from the counter outputs, and generates a *0* whenever the count value reaches *M*.
  - Connect the output of the gating circuit to the *CLR'* inputs of the flip-flops.

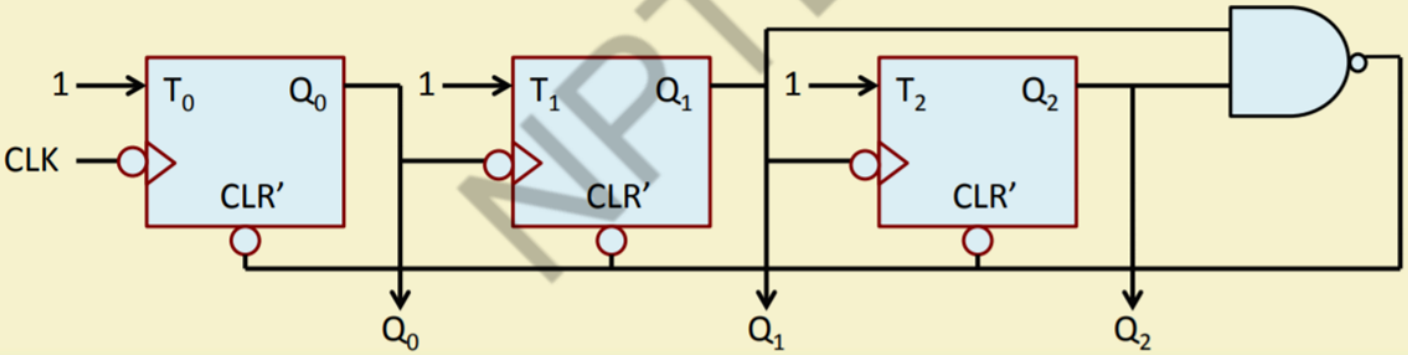
# Example: Design of a modulo-6 binary ripple counter

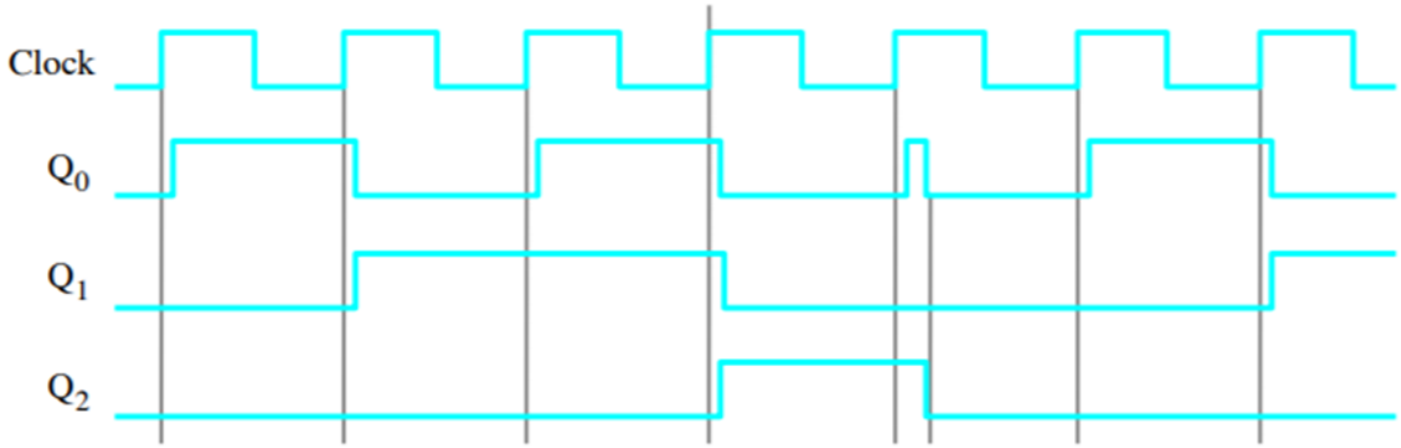
- Number of flip-flop stages required will be $\lceil \log_2 6 \rceil = 3$
- The desired count sequence will be: 000, 001, 010, 011, 100, 101, 000, …
  - From state 101, instead of going to the state 110, we have to bring the state back to 000.



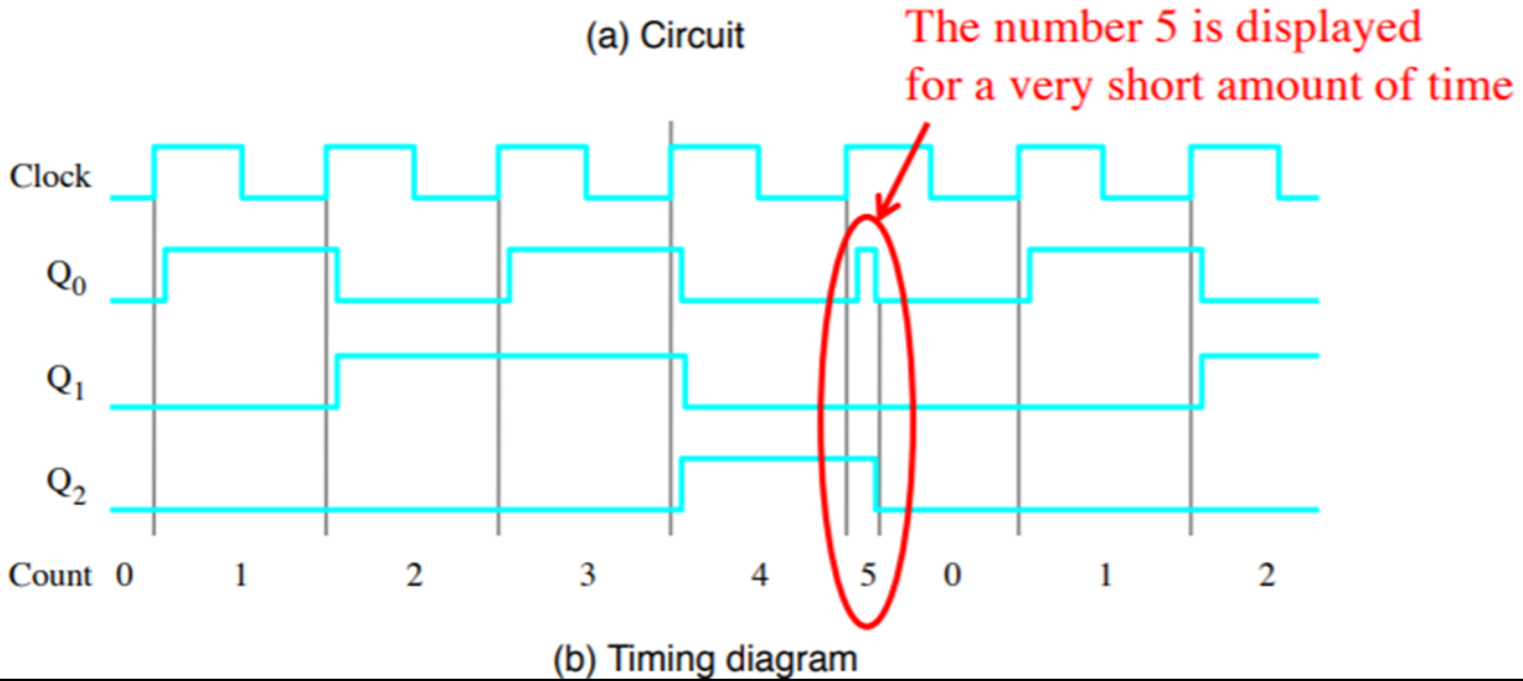
# Example: Design of a modulo-6 binary ripple counter

- Number of flip-flop stages required will be $\lceil \log_2 6 \rceil = 3$
- The desired count sequence will be: 000, 001, 010, 011, 100, 101, 000, …
  - From state 101, instead of going to the state 110, we have to bring the state back to 000.



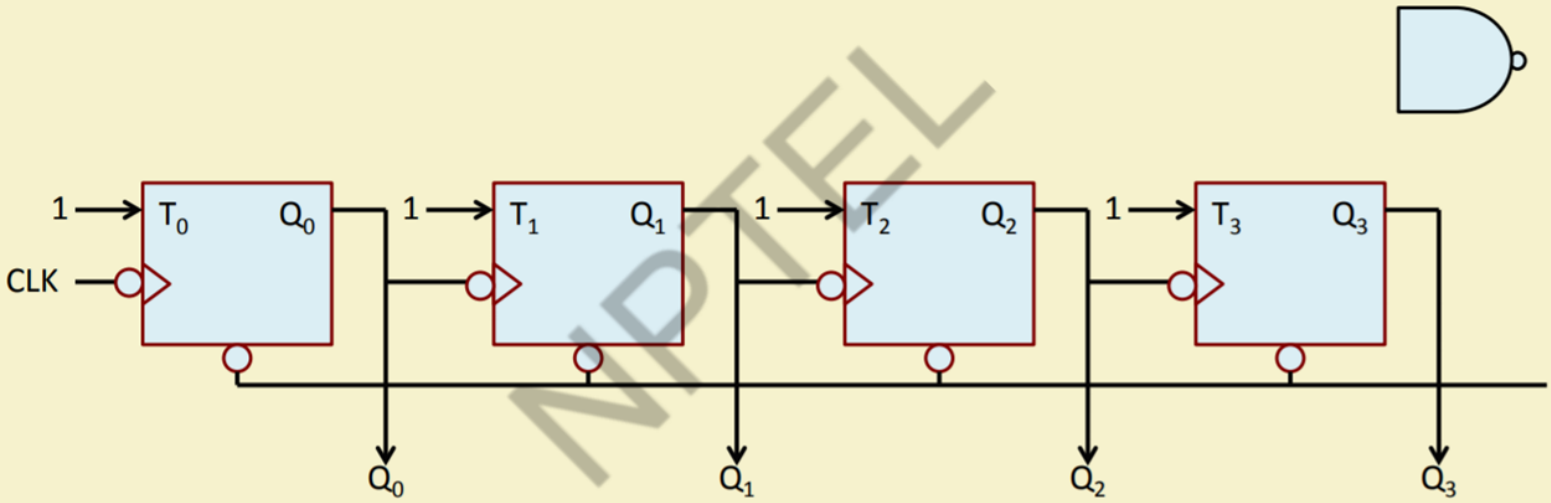



(b) Timing diagram

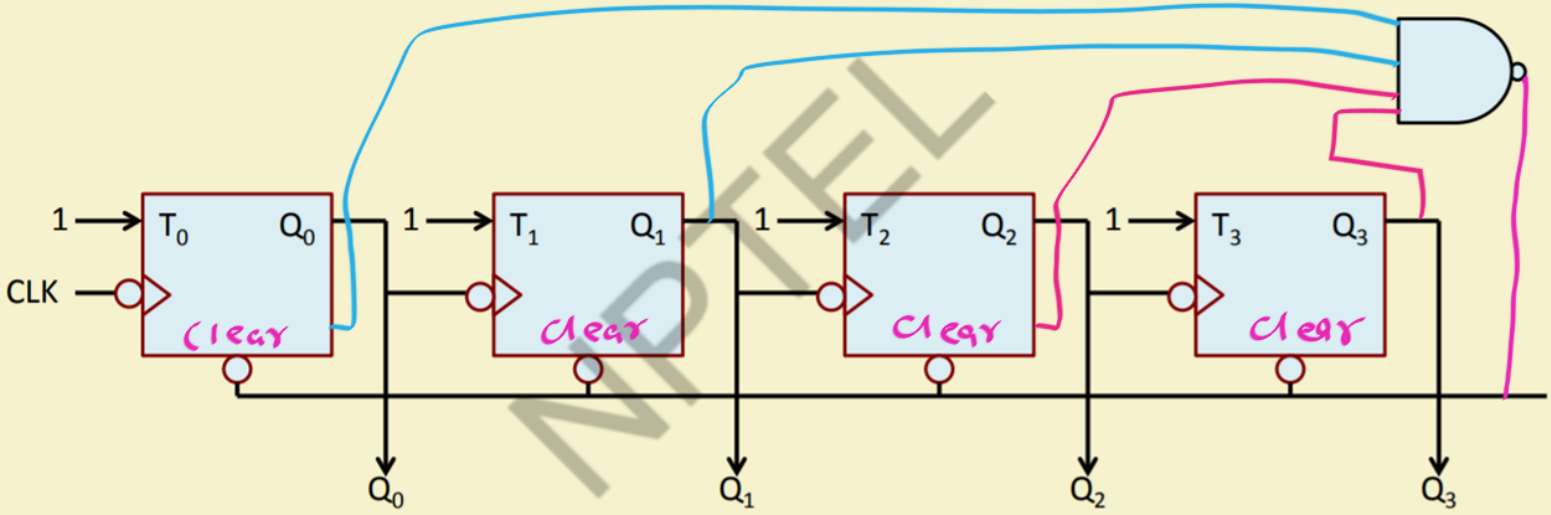

(a) Circuit
The number 5 is displayed
for a very short amount of time
Clock
$Q_0$
$Q_1$
$Q_2$
Count 0 1 2 3 4 5 0 1 2
(b) Timing diagram

# Decade Counter : ( BCD Counter)

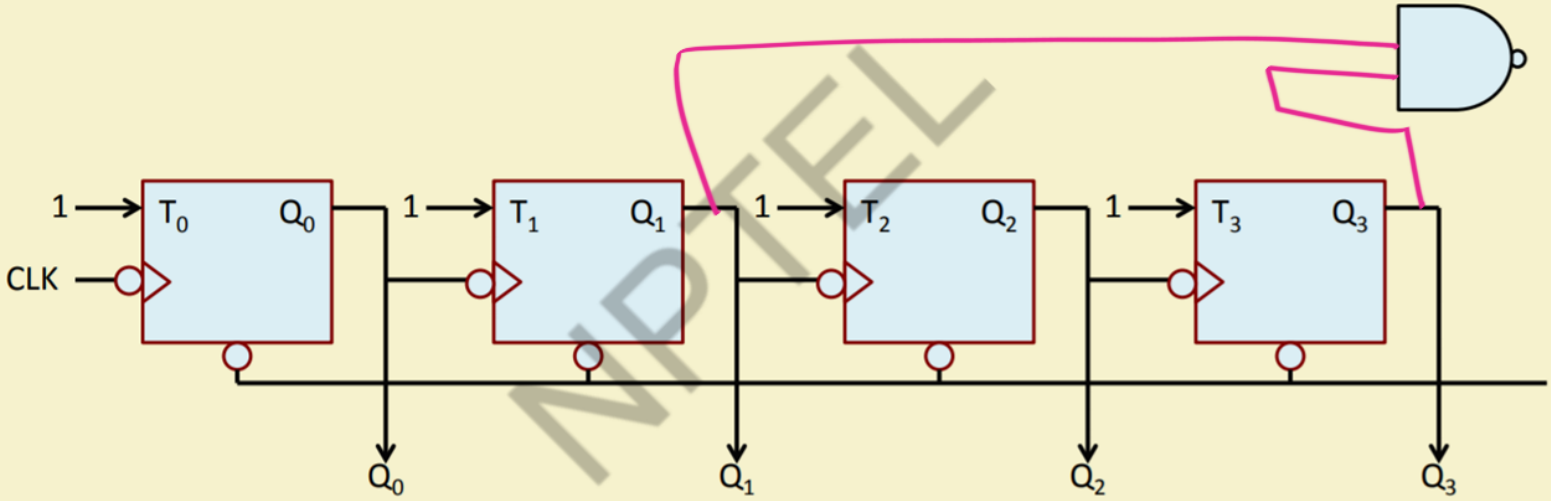
# Example: Design of a decade (modulo-10) counter



# Example: Design of a decade (modulo-10) counter

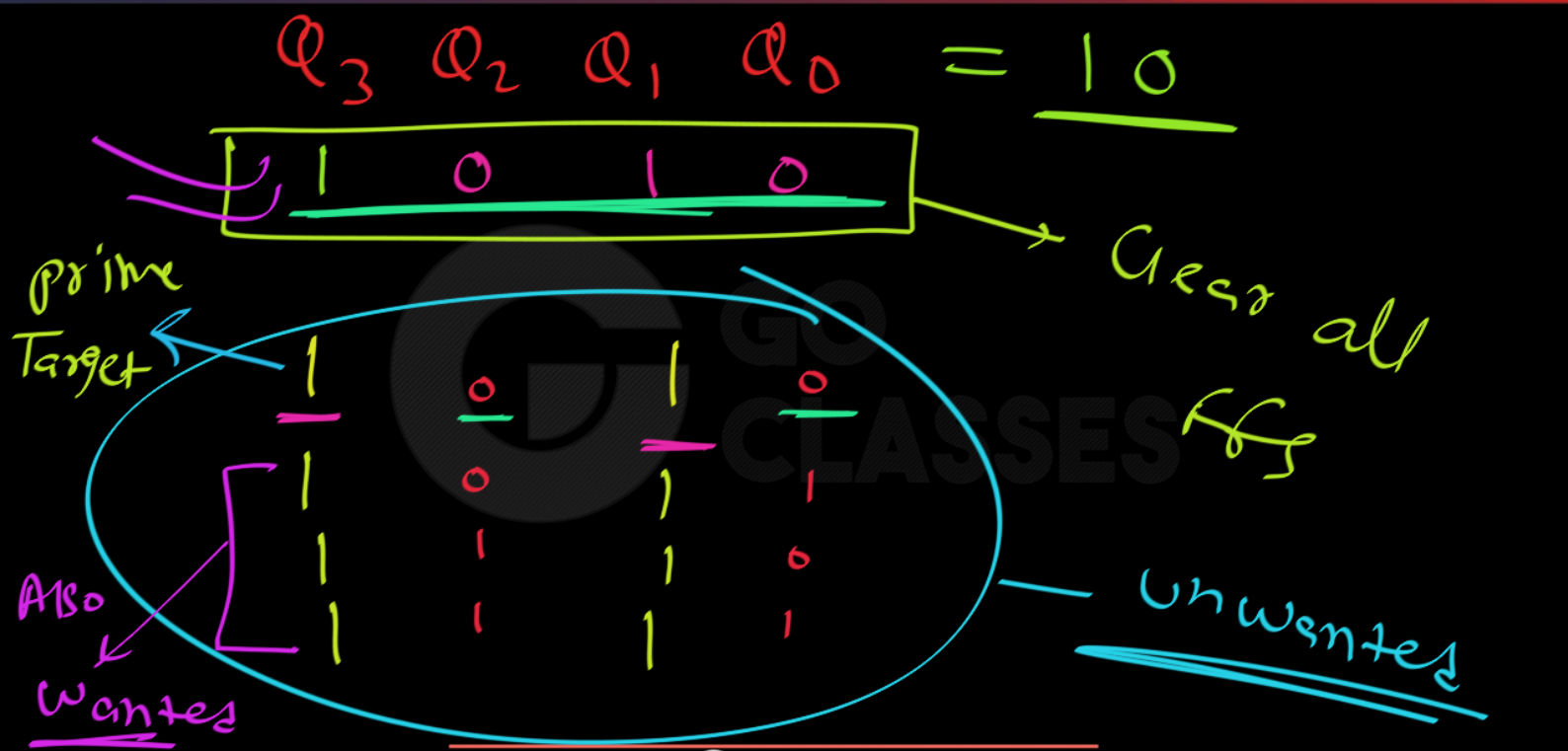


# Example: Design of a decade (modulo-10) counter



$Q_3\ Q_2\ Q_1\ Q_0 = 10$

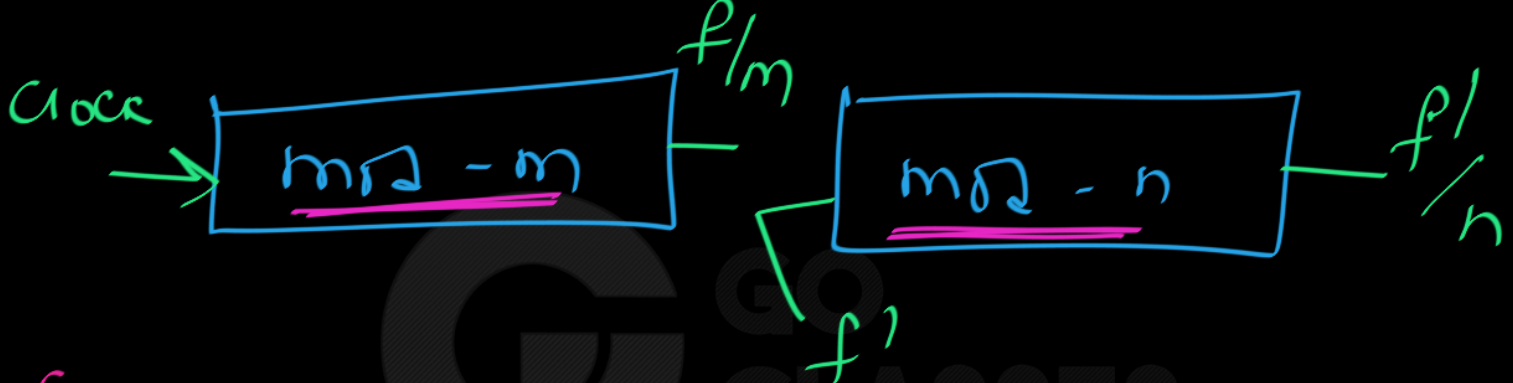

Check all ffs


$Q_3$ $Q_2$ $Q_1$ $Q_0$ = 10
1 0 1 0
Clear all ffs
Prime Target
1 0 1 0
Also
Wantes
Unwantes

# Next Topic:

## Asynchronous Counters

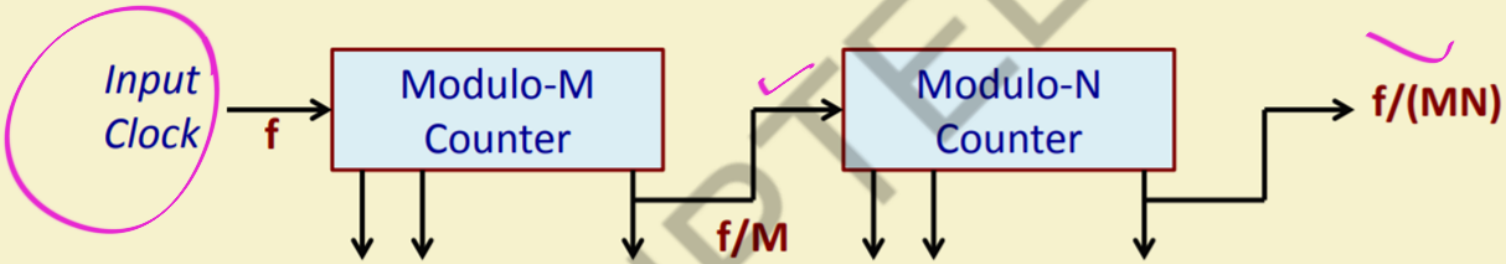
## 3. Cascading counters




Create mod mn :

# Cascading of Ripple Counters

- If a *modulo-M* and a *modulo-N* counter are connected in cascade, the combination will count *modulo-MN*.



**Example: Design a modulo-1000 counter using decade counters**

BCD Counter

# Example: Design a modulo-1000 counter using decade counters

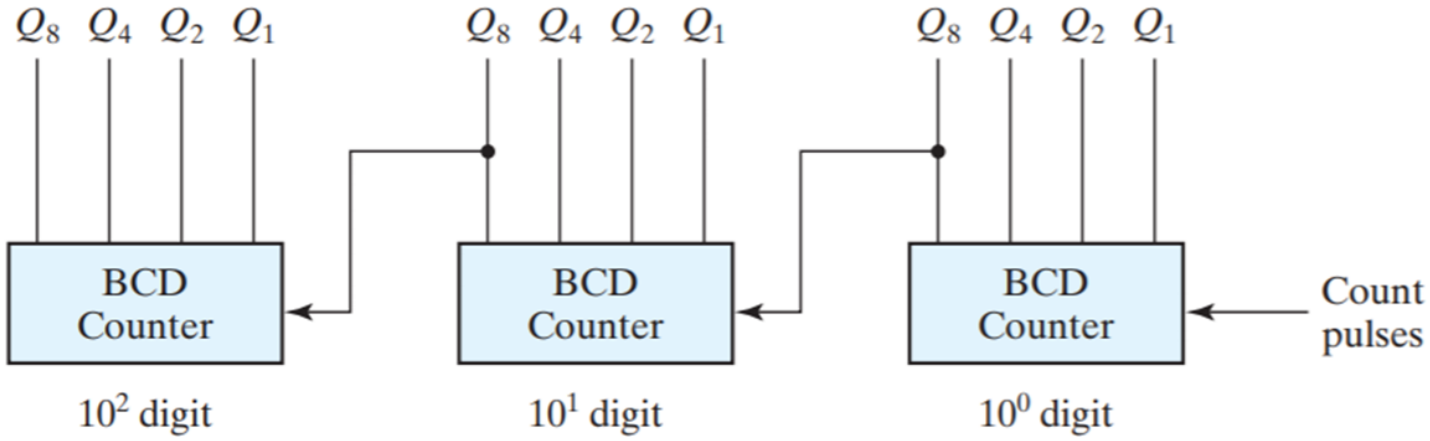



**FIGURE 6.11**
Block diagram of a three-decade decimal BCD counter

# Example: Design a modulo-1000 counter using decade counters

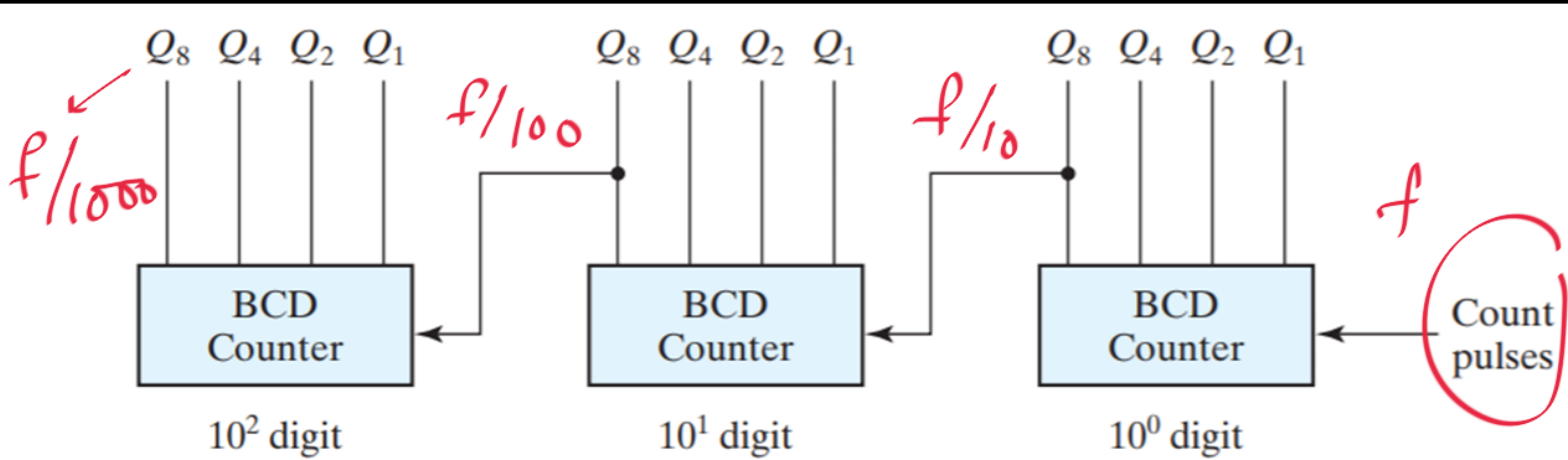
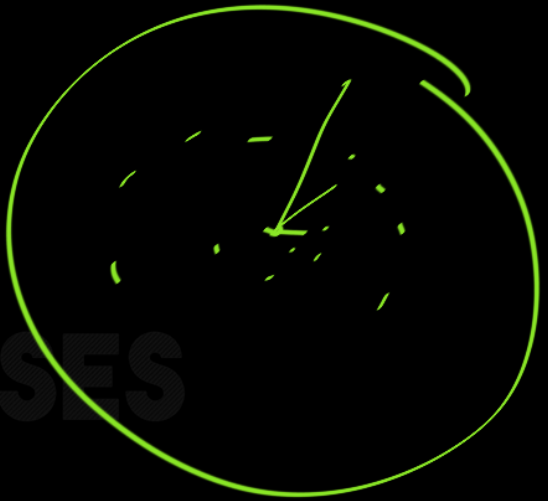



**FIGURE 6.11**
Block diagram of a three-decade decimal BCD counter

# Frequency Division :

## Real Life Example :

Clock



Clock :

last

second hand — freq = f

slow

minute hand — freq = f/60

hour hand — freq = f/60x60

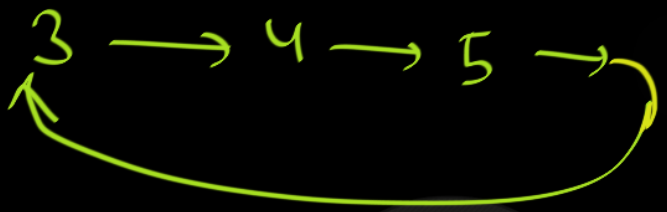
very slow

# Next Topic:

# Asynchronous Counters

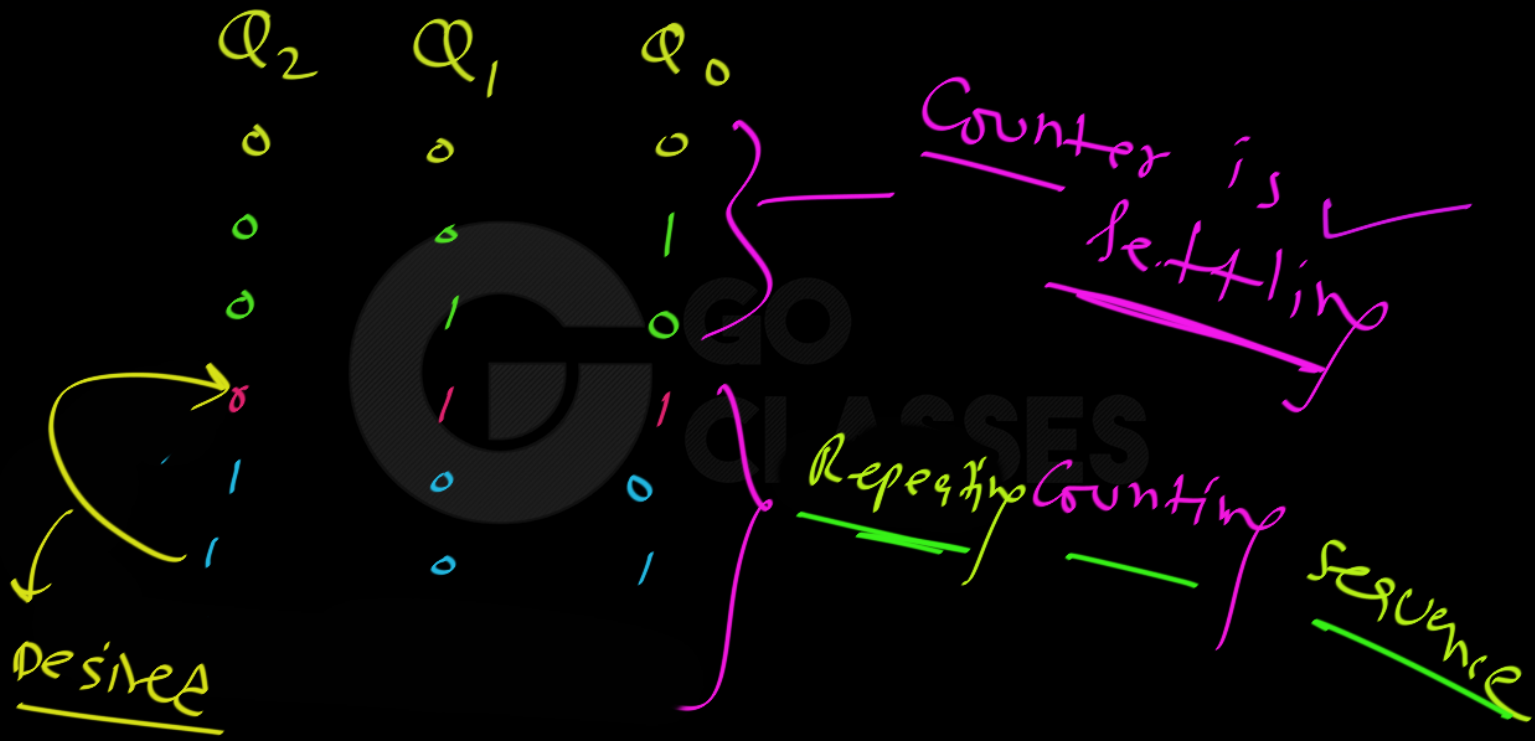
## 4. From Anywhere to Anywhere counting

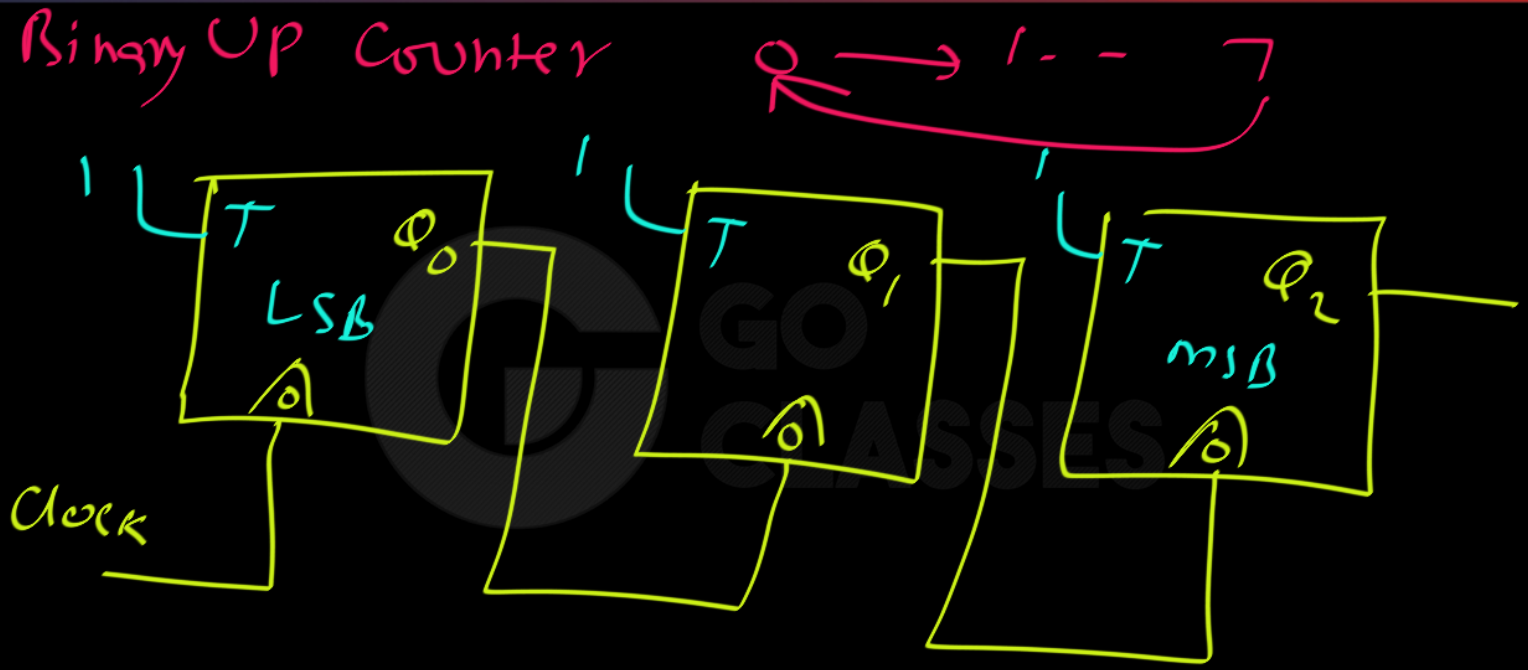
Asynchronous Counter to count from anywhere to anywhere :

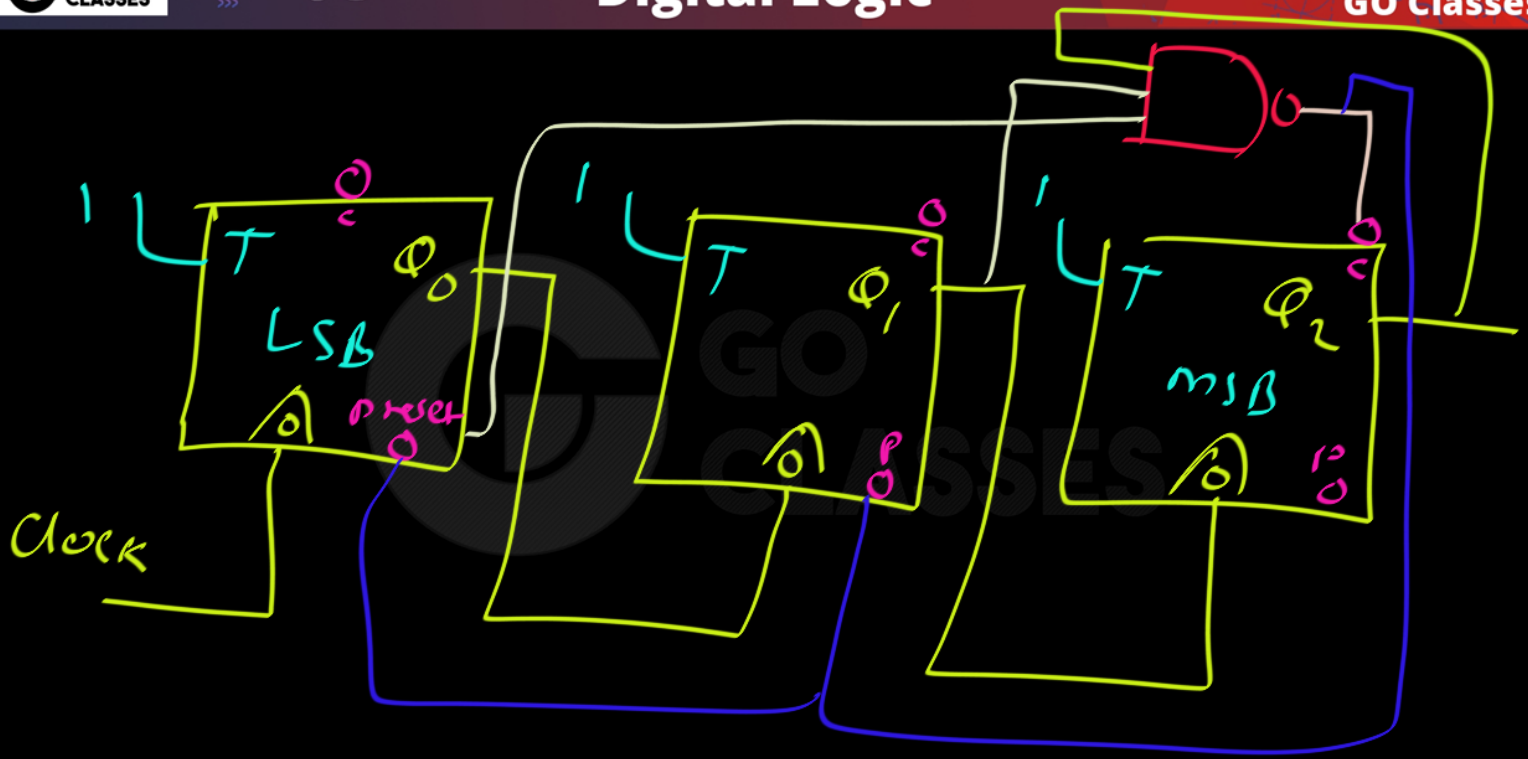


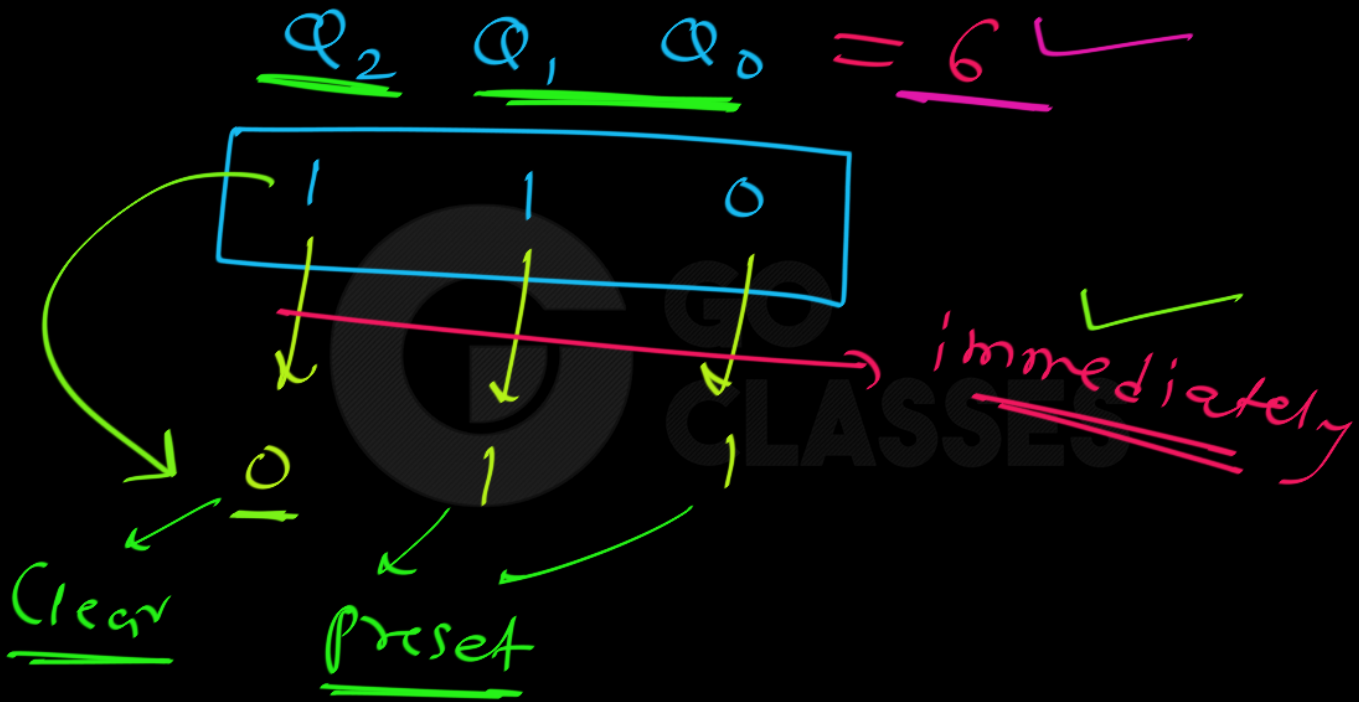

Ripple Counter

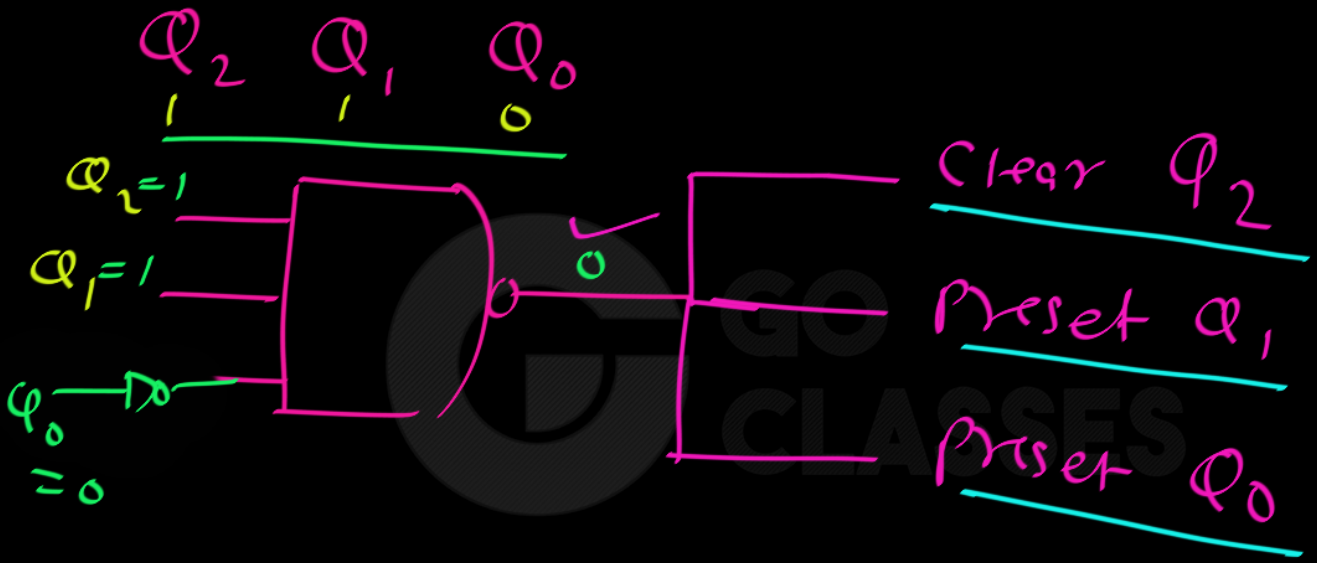
3 FFs


$Q_2$ $Q_1$ $Q_0$
0 0 0
0 0 1
0 1 0
1 0 1
1 1 0
0 0 1
Counter is resetting
Repeating Counting sequence
Desired

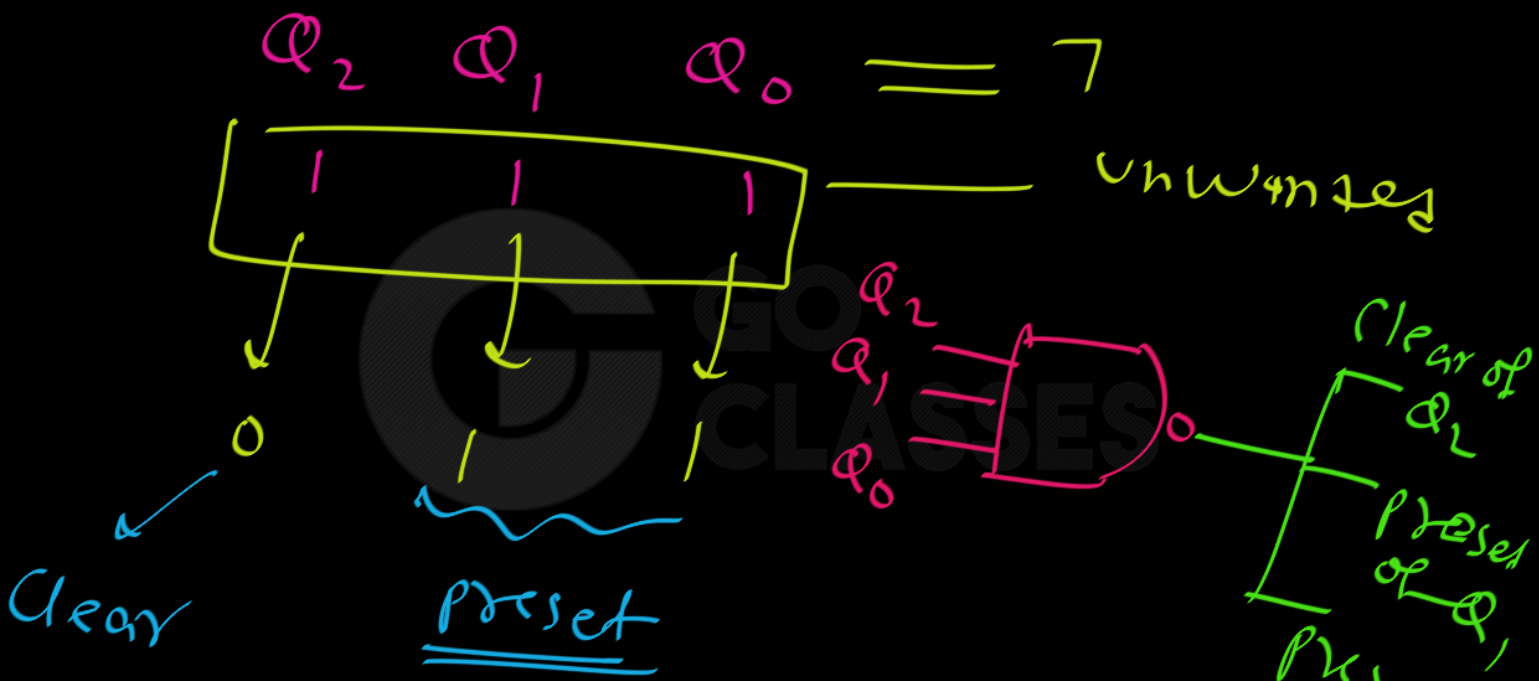

Binary UP Counter
0 → 1 - - 7
1
T
$Q_0$
LSB
1
T
$Q_1$
1
T
$Q_2$
MSB
Clock


1
T
C
$Q_0$
LSB
Preset
Clock
1
T
C
$Q_1$
1
T
C
$Q_2$
MSB


$Q_2$ $Q_1$ $Q_0$ = 6
1 1 0
immediately
0 1 1
Clear
Preset


Q2 Q1 Q0
1 1 0
Q2=1
Q1=1
Q0
=0
0
Clear Q2
Preset Q1
Preset Q0

Q:


$Q_2$ $Q_1$ $Q_0$ = 7
1 1 1
Unwanted
0 1 1
Clear
Preset
$Q_2$
$Q_1$
$Q_0$
Clear of $Q_2$
Preset of $Q_1$
Preset of $Q_0$

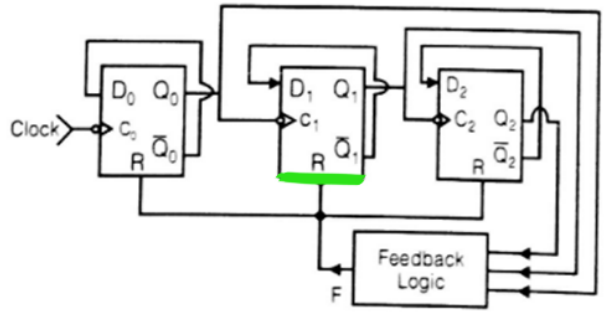
# Next Topic:

## GATE Questions

## GATE ECE 1987 Ripple Counter Question

# 1987

1. A ripple counter using negative edge triggered D flip-flops is shown below. The flip-flops are cleared to '0' at the R input. The feedback logic is to be designed to obtain the count sequence shown in the same figure. The correct feedback logic is :




Count sequence in the order of $Q_2Q_1Q_0$ :

000 → 001 → 010 → 011 → 100 → 101

(a) $F = \overline{Q_2 Q_1 \bar{Q}_0}$

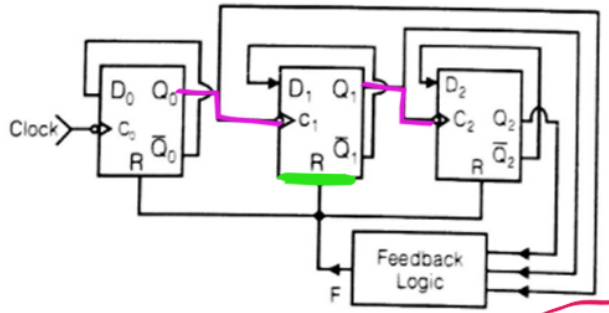
(b) $F = Q_2 \bar{Q}_1 \bar{Q}_0$

(c) $F = \bar{Q}_2 \bar{Q}_1 Q_0$

(d) $F = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0$

# 1987

1. A ripple counter using negative edge triggered D flip-flops is shown below. The flip-flops are cleared to '0' at the R input. The feedback logic is to be designed to obtain the count sequence shown in the same figure. The correct feedback logic is :




Count sequence in the order of $Q_2Q_1Q_0$ :

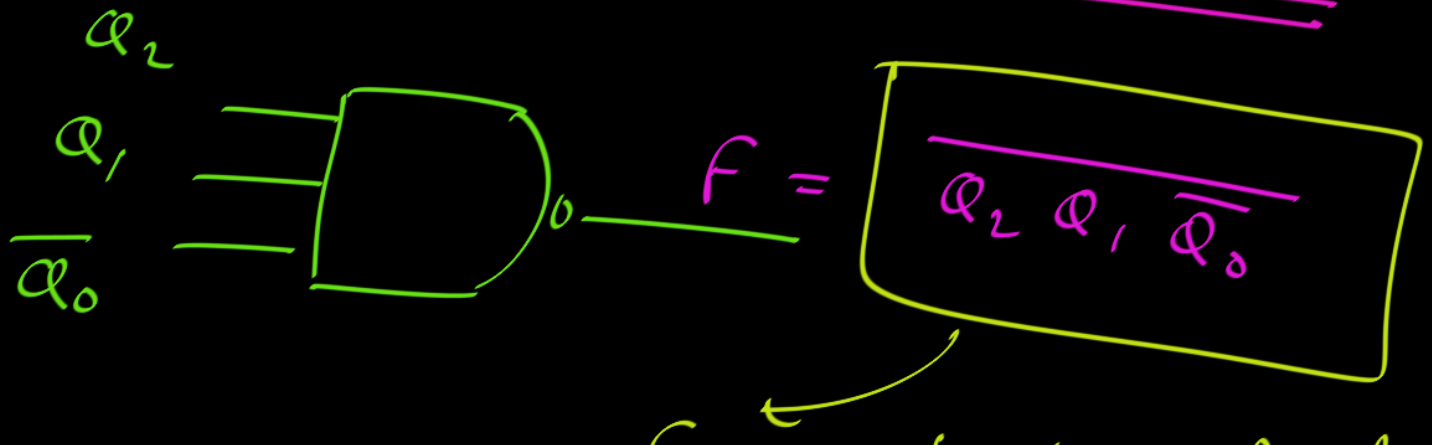
000 → 001 → 010 → 011 → 100 → 101 →

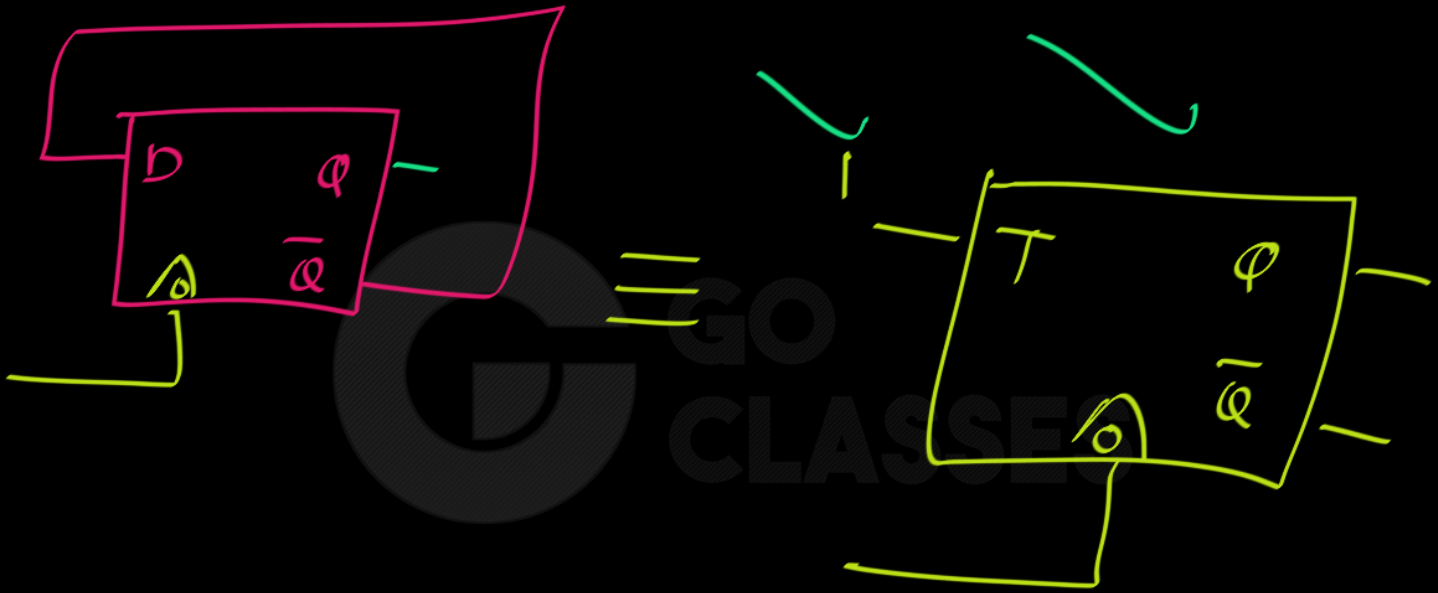
(a) $F = \overline{Q_2 Q_1 \bar{Q}_0}$ (b) $F = Q_2 \bar{Q}_1 \bar{Q}_0$

(c) $F = \bar{Q}_2 \bar{Q}_1 Q_0$ (d) $F = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0$

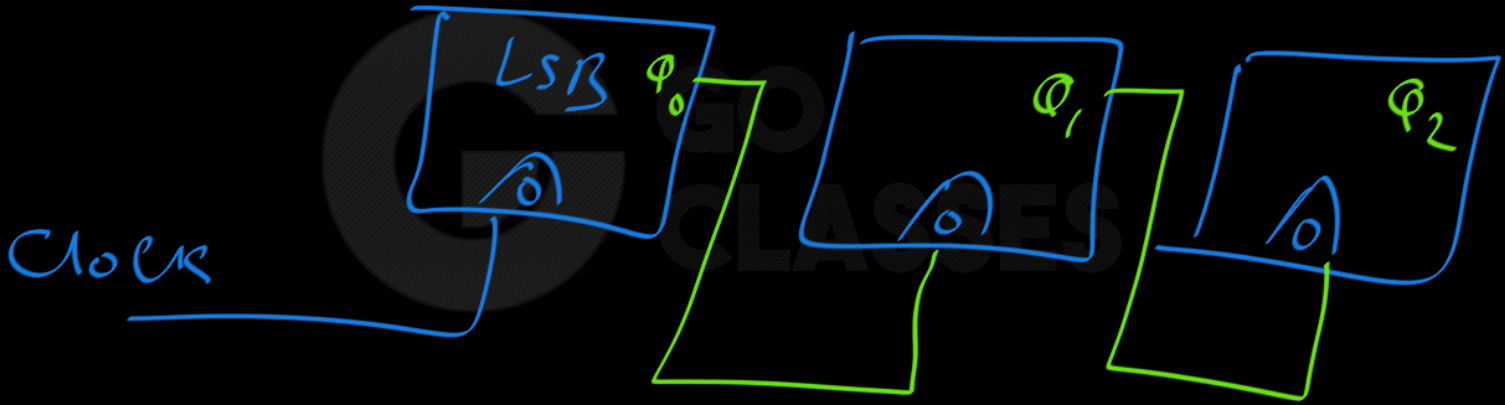
$Q_2\ Q_1\ Q_0 = 6$

$1\quad 1\quad 0 \longrightarrow$ unwanted

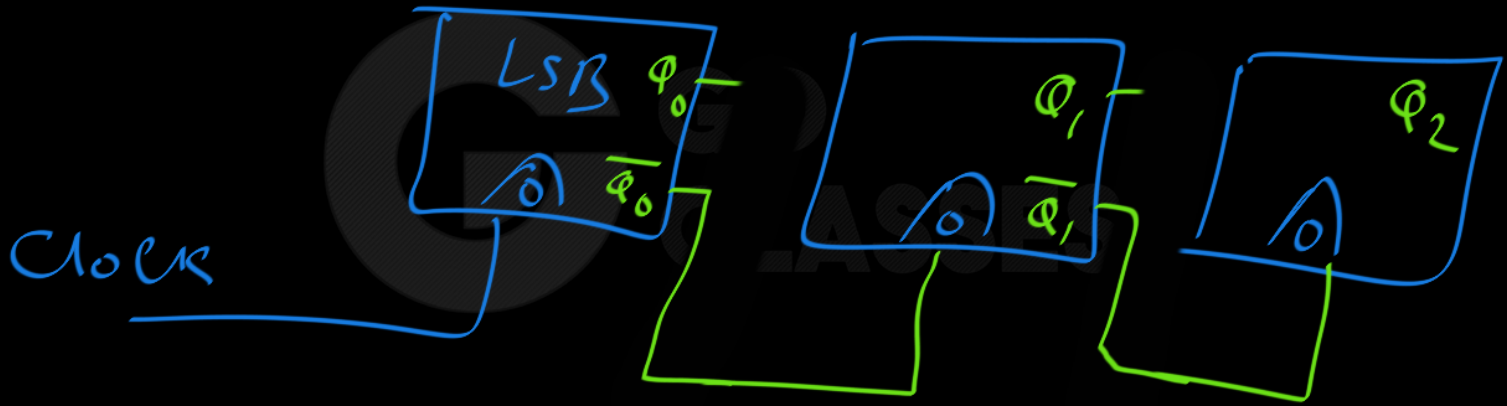



D
Q
Q̄
≡
T
Q
Q̄

Up Counting



Down Counting :



# Next Topic:

## GATE Questions

## GATE ECE 1990 Ripple Counter Question

## 1990

1. A 4 bit modulo-16 ripple counter uses JK flip-flops. If the propagation delay of each flip-flop is 50 nsec, the maximum clock frequency that can be used is equal to ……

a. 20 MHz
b. 10 MHz
c. 5 MHz
d. 4 MHz


**1990**

1. A 4 bit modulo-16 ripple counter uses JK flip-flops. If the propagation delay of each flip-flop is 50 nsec, the maximum clock frequency that can be used is equal to ……

a. 20 MHz
b. 10 MHz
c. 5 MHz
d. 4 MHz

$f \leq \frac{1}{nt} = \frac{1}{4 \times 50\ nsec}$

$f \leq \frac{10^9}{200}\ Hz$

$f \leq 5\ MHz$

$\frac{1}{sec} = Hz$


1
T
Q
1
K
$\bar{Q}$
≡
1
T
Q
$\bar{Q}$

# Ripple Counter:

## Proper functioning:

$$\text{Clock } f_{req} \leq \frac{1}{nt}$$

- $n$ → #ffs
- $t$ → prop Delay of ff

## 1990

1. A 4 bit modulo-16 ripple counter uses JK flip-flops. If the propagation delay of each flip-flop is 50 nsec, the maximum clock frequency that can be used is equal to ……

a. 20 MHz
b. 10 MHz
c. 5 MHz
d. 4 MHz

**Answer:** C

# Next Topic:

## GATE Questions

## GATE ECE 1993 Ripple Counter Question

## 1993

1. A pulse train with a frequency of 1 MHz is counted using a modulo 1024 ripple counter built with JK flip-flops. For proper operation of the counter, the maximum permissible propagation delay per flip flop stage is _____ nsec.

#ffs = 10

**1993**

1. A pulse train with a frequency of 1 MHz is counted using a modulo 1024 ripple counter built with JK flip-flops. For proper operation of the counter, the maximum permissible propagation delay per flip flop stage is ____ nsec.

100

$f \leq \frac{1}{nt}$

$1\,MHz \leq \frac{1}{10t}$

$t \leq \frac{1}{10\,MHz}$

$\frac{1000\,nsec}{10}$

$= 0.1\,\mu sec$

$\frac{1}{Hz} = sec$

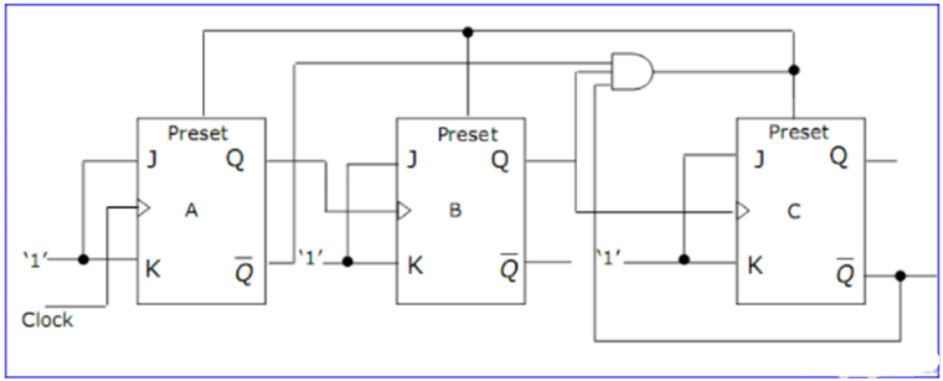
# Next Topic:

# GATE Questions

## GATE ECE 1999 Ripple Counter Question

## 1999

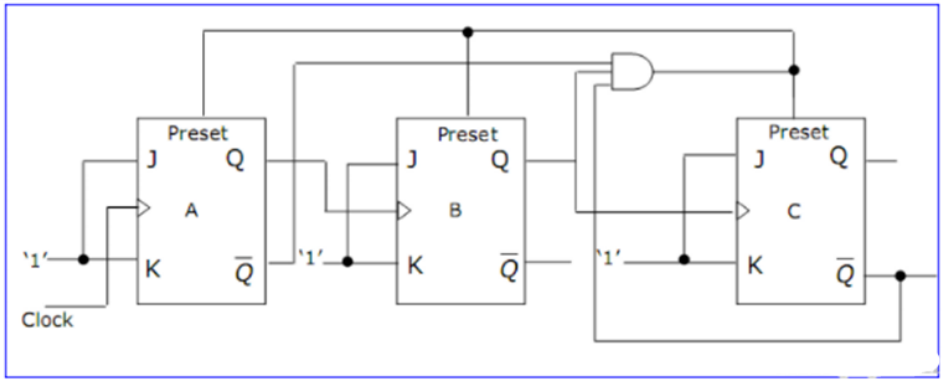
1. The ripple counter shown in the given figure is works as a




a. Mod – 3 up counter
b. Mod – 5 up counter
c. Mod – 3 down counter
d. Mod – 5 down counter

## 1999

1. The ripple counter shown in the given figure is works as a



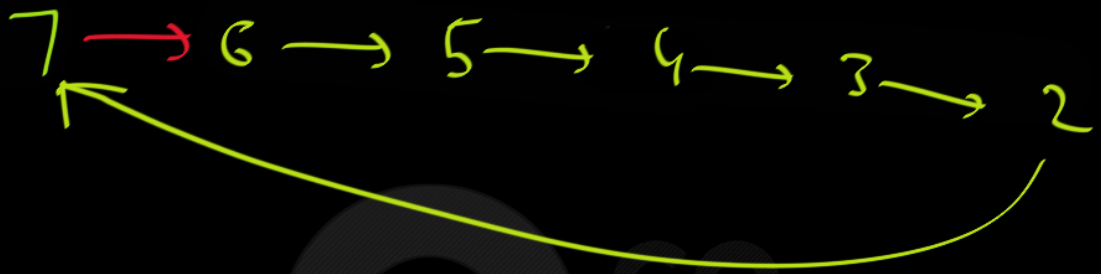

a. Mod – 3 up counter
b. Mod – 5 up counter
c. Mod – 3 down counter
d. Mod – 5 down counter

Down Counter

$\overline{Q_2}\,Q_1\,\overline{Q_0}$ ⇒ Unwanted = 010


1111
1110
0111


7
6
3




1 = Unwanted

Unwanted

$Q_2$ $Q_1$ $Q_0$

0 0 1 = 1

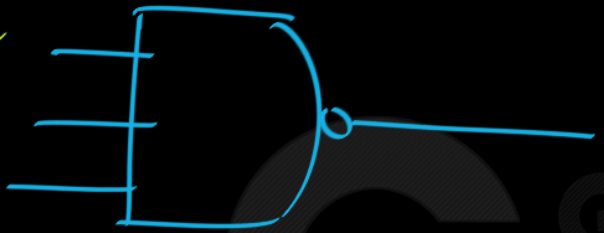
immediately

1 1 1 Preset

$\overline{Q_2}$

$\overline{Q_1}$

$Q_0$

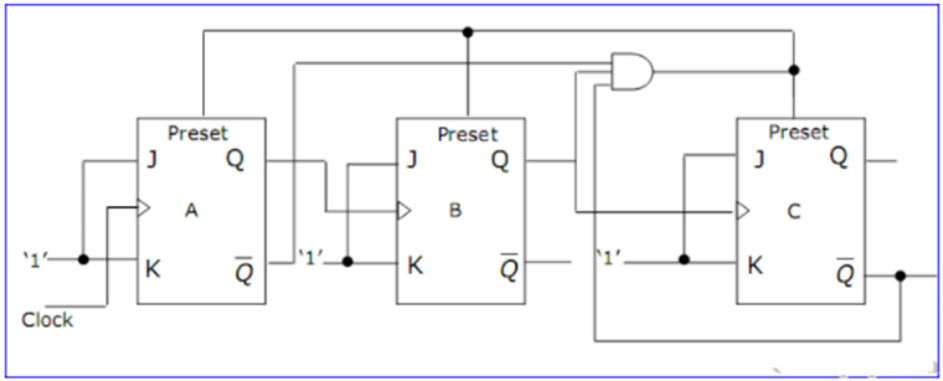


Active low

Preset of all FFs

## 1999

1. The ripple counter shown in the given figure is works as a




a. Mod – 3 up counter
b. Mod – 5 up counter
c. Mod – 3 down counter
d. Mod – 5 down counter

**Answer:** D