



## CS/ECE/EEE/INSTR F215:Digital Design

### Lecture 19: *Introduction to Sequential Circuits*

*Tue, 12 Oct 2021*





IT IS NEVER TOO LATE  
TO START HEADING IN THE RIGHT DIRECTION

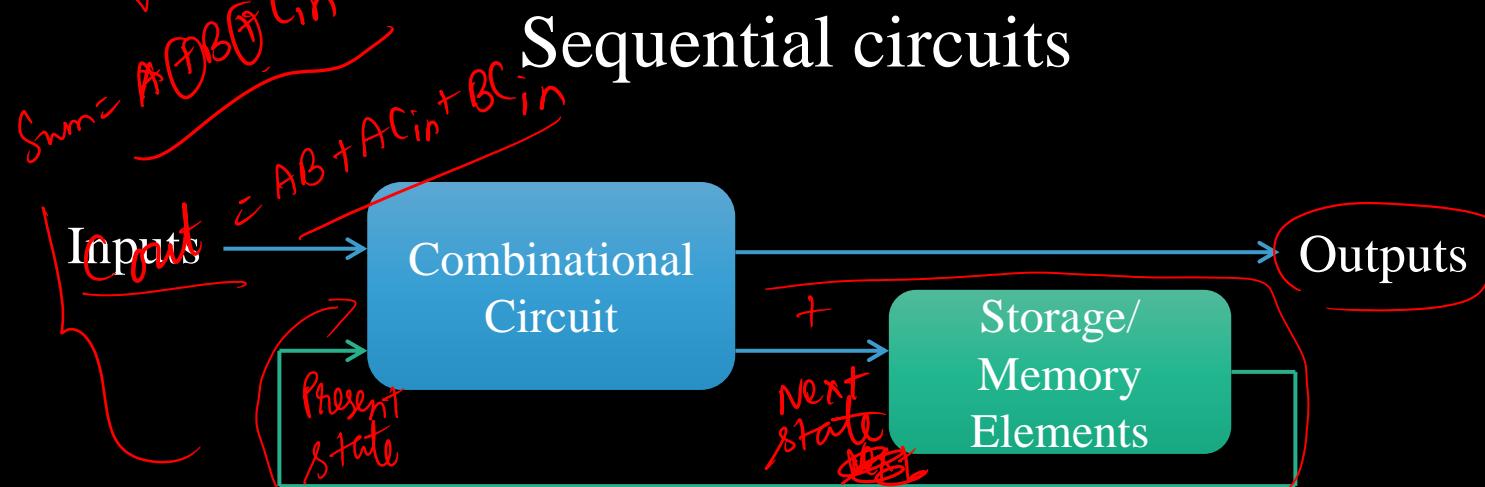
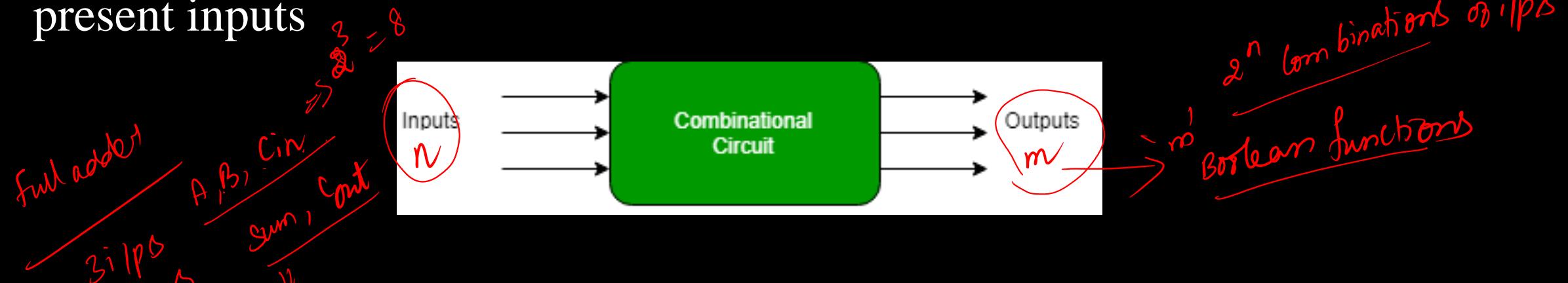
IT IS NEVER TOO LATE

# Logic Circuits

## Combinational

## Sequential

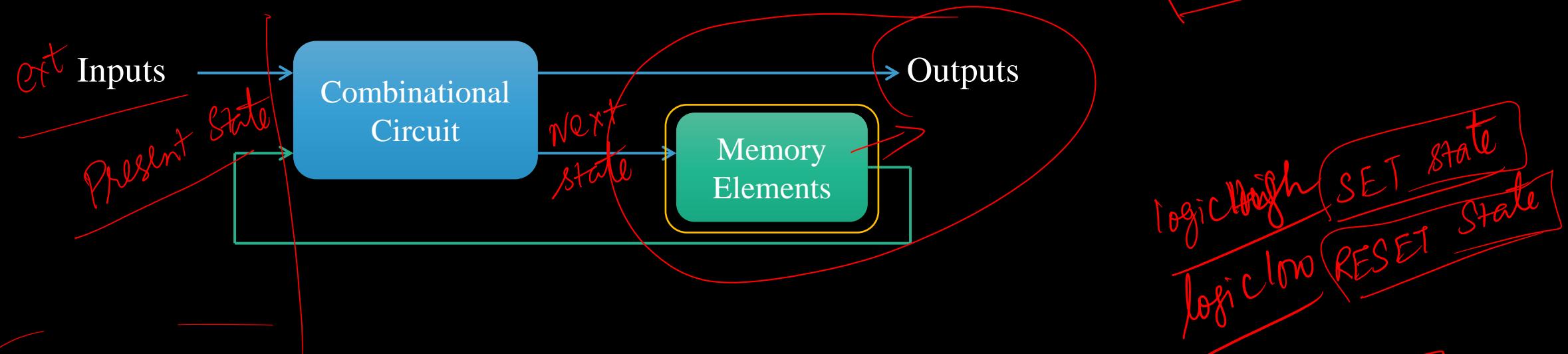
Combinational circuits – The outputs at any time are a function of only the present inputs



# *Sequential logic circuits*

- Consists of combinational logic circuit to which storage elements are connected to form a feedback path
- Outputs are a function of inputs and the state of the storage elements.
- As the state of the storage element is a function of previous inputs to the circuit, outputs of a sequential circuit at any time depend not only on the present values of inputs but also on past inputs.
- Circuit behaviour is specified by a time sequence of inputs and internal states.

# Sequential logic circuits



Storage / Memory elements, capable of storing **binary information**

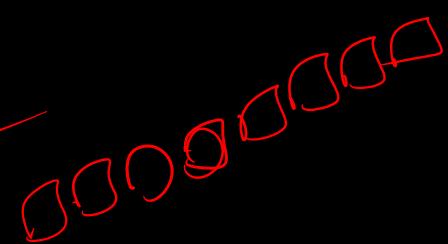
Binary information stored at any given time → state of the sequential circuit at that time

External input + present state → Determine output of sequential circuit

External input + present state → Determine next state of memory element

# Storage / Memory elements

- It should be able to hold a value
- One should be able to read the value stored
- One should be able to change the value.

~~Register~~ 

**Latches** and **flip-flops** are the basic elements for storing information. A latch or flip-flop can store one bit of information.

- Maintain a binary state indefinitely until directed by an input signal to change state.
- Has two useful states.



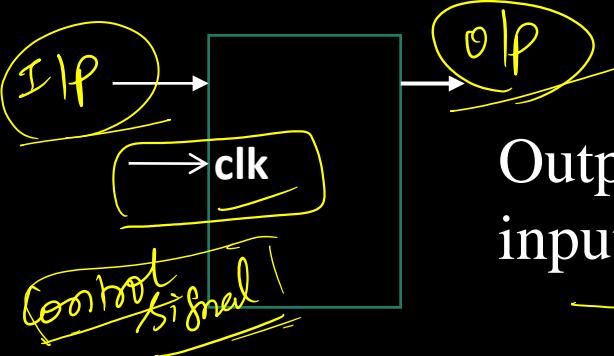
Set & Reset  






# Latches and flip-flops $\rightarrow$ Storage elements

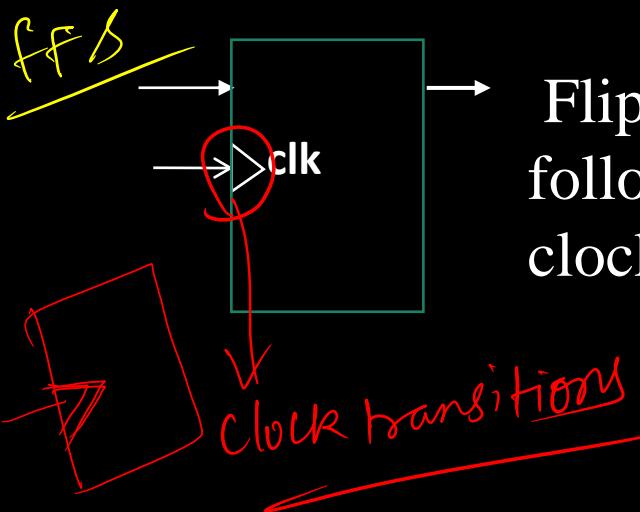
- Storage elements that operate with signal levels (level sensitive) are **Latches**



Output follow the input as long as the clock input is asserted (high/low)



- Storage elements that operate with clock transitions (edge sensitive) are flip-flops

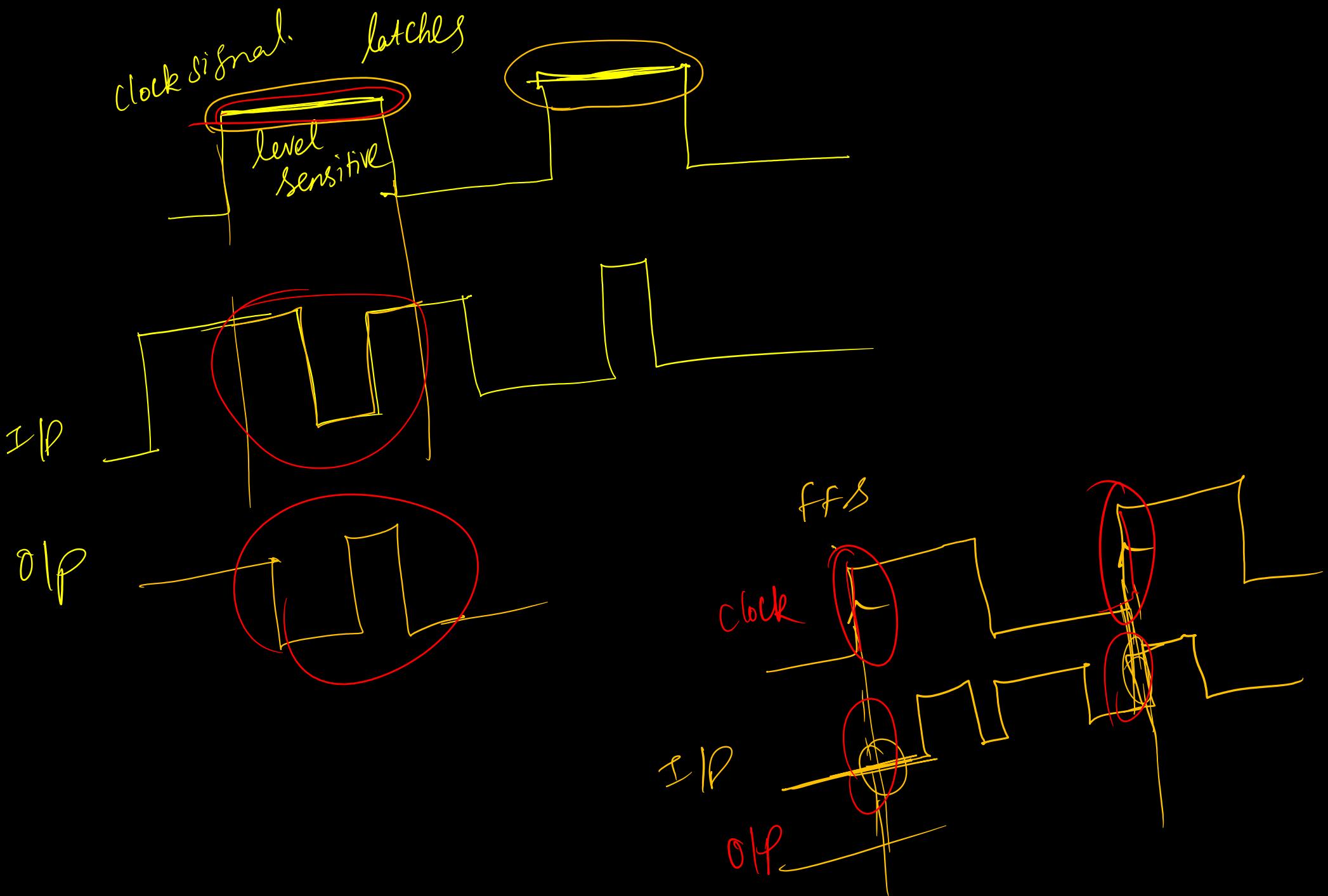


Flip-flops are circuits where the output follows input only during clock transitions (rising edge/falling edge)

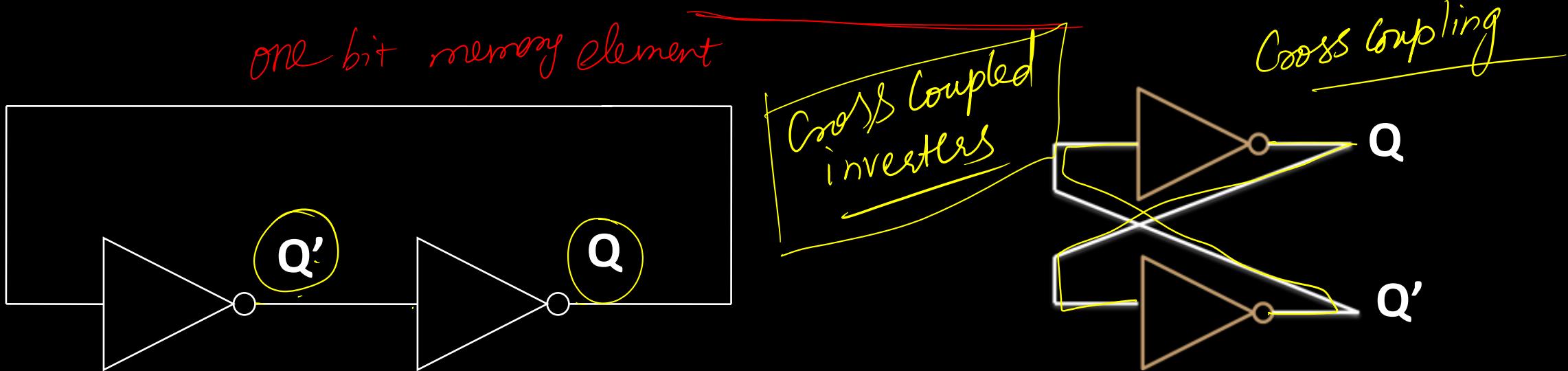


Latches are the basic circuits from which flip-flops are constructed (building blocks of FFs)

FFs are  
controlled  
latches



# Basic Concept of Storage (one bit memory)



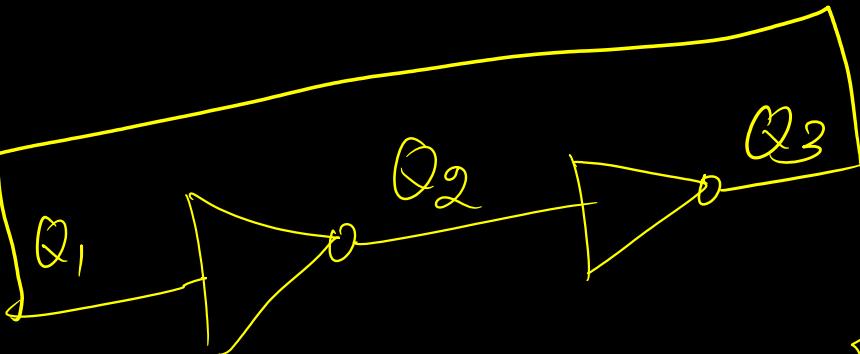
- The circuit remembers  $Q$  (never changes as long as powered on) as well as  $Q'$ .
- We can read  $Q$  by checking the output.

$$\stackrel{\perp}{=} \quad v_{ce}$$

A latch is a simple digital circuit where the output can be set to either logic 1 or logic 0 by an input signal.

The circuit then remains in/“remembers” this state even after the input is removed.

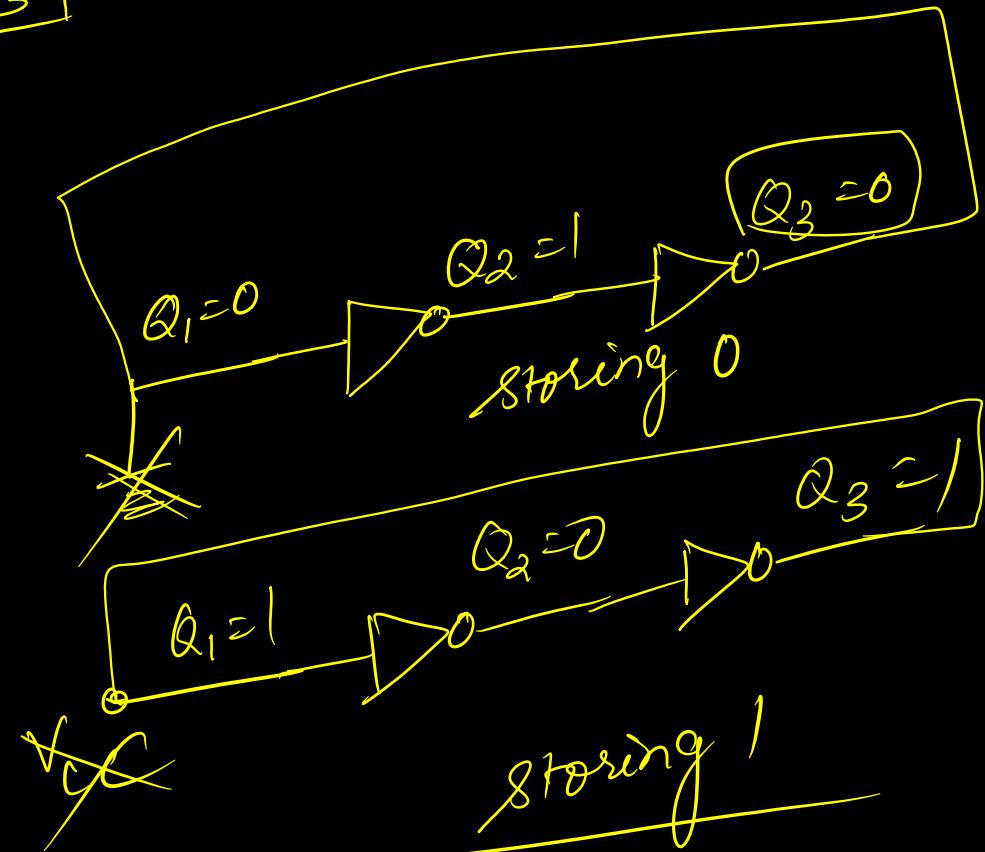
This is a memory circuit that can “remember” a single binary digit (bit). ‘0’ or ‘1’



Remove  $f/b$

Connect  $Q_1$  to ground

Connect  $Q_1$  to  $V_{CC}$





# CS/ECE/EEE/INSTR F215:Digital Design

## Lecture 20: *Latches*

*Thu, 14 Oct 2021*



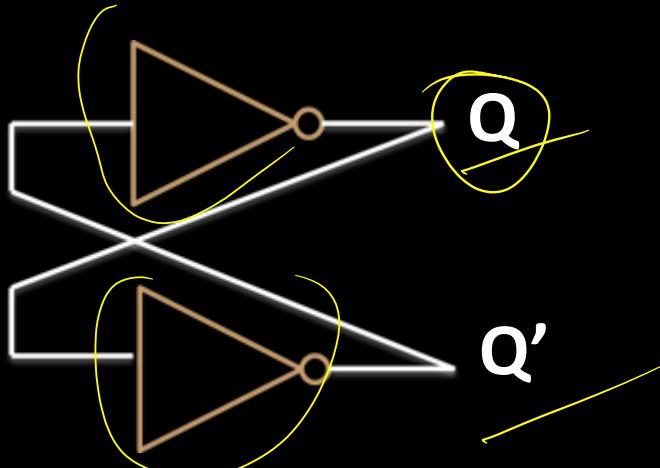
*Challenges are what make  
life interesting.*

*Overcoming them is what make  
life meaningful.*

*- Joshua J. Marine*

# One bit memory element

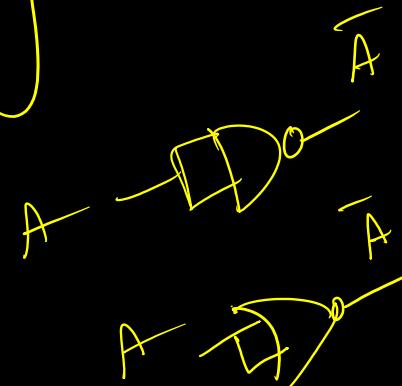
$Q = 1 \rightarrow$  Set State  
 $Q = 0 \rightarrow$  Reset State  
Latch FF  $\Rightarrow$  Unstable element  
2 Stable states



Set state  $Q = 1$   
Reset state  $Q = 0$

Is there any way to control the storage value ?

NAND gate      NOR gate       $\Rightarrow$  Inverter

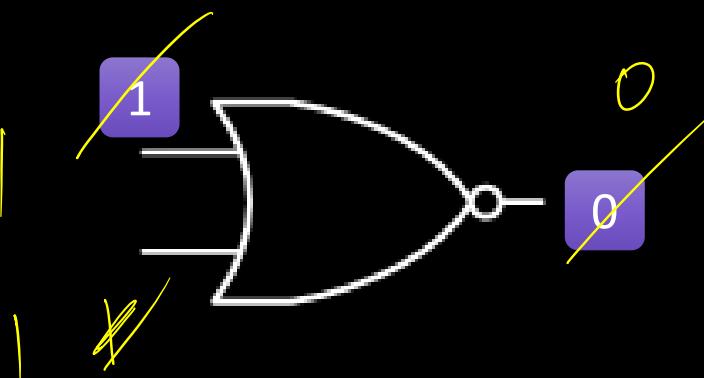


Cross coupling  
lop of one gate  
given as lop to  
another  
& vice versa

## NOR -NAND

Truth table NOR gate

A	B	O/P
0	0	1
0	1	0
1	0	0
1	1	0

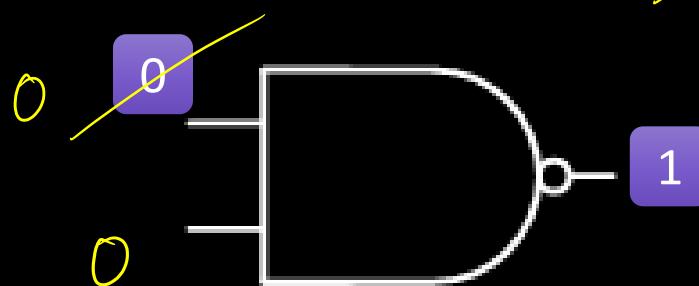


A '1' on any one or both i/p's of NOR gate forces output to '0'

'1'  $\rightarrow$  dominating i/p NOR gate

Truth table NAND gate

A	B	O/P
0	0	1
0	1	1
1	0	1
1	1	0



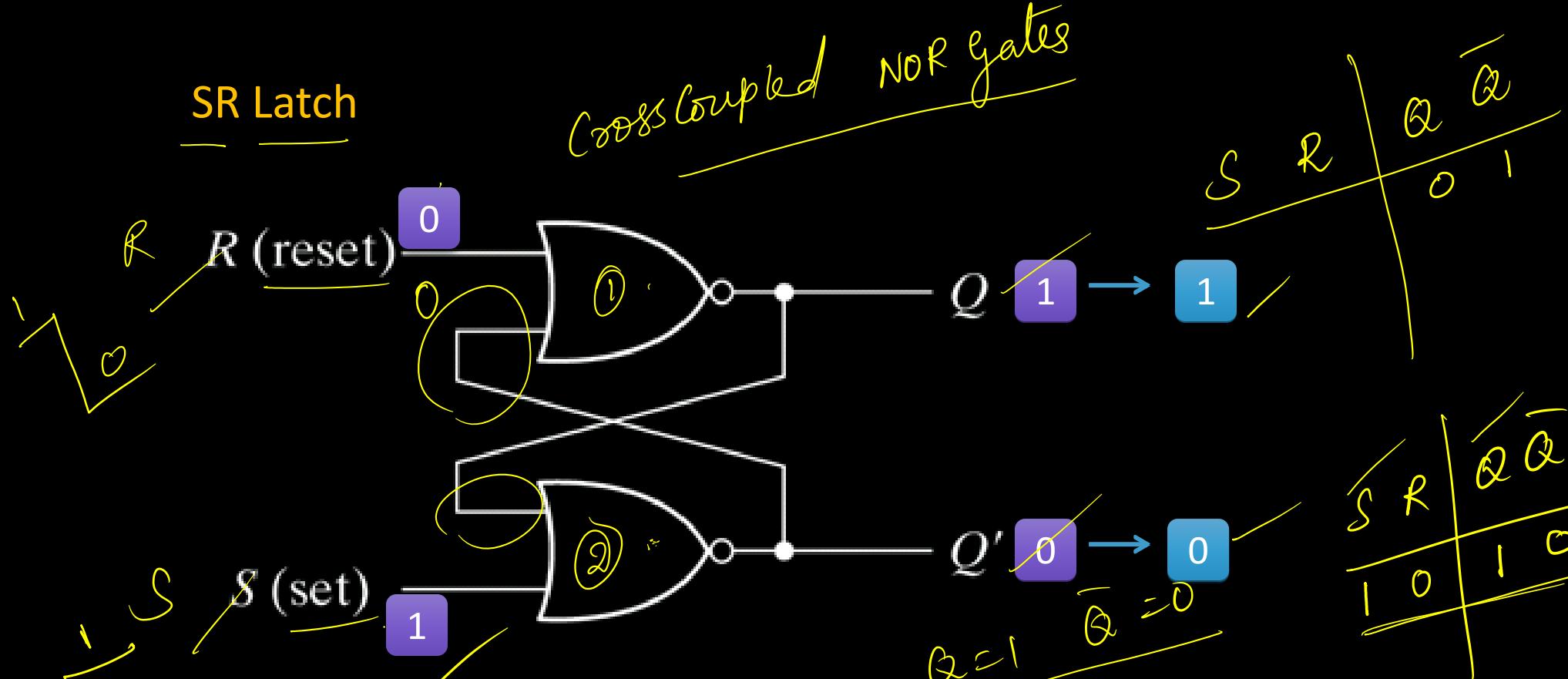
A '0' on any one i/p's of NAND gate or both the forces

output to '1'

'0'  $\rightarrow$  dominating i/p

To Set the latch  $R=0, S=1$

Assume  
 $Q=0 \quad \bar{Q}=1$   
 $S=1 \quad R=0$



latch will be in set state

Assume  
 $Q=1 \quad \bar{Q}=0$   
 $S=1 \quad R=0$   
 $Q=1 \quad \bar{Q}=0$

$S$	$R$	$Q$	$\bar{Q}$
0	0	0	1
1	0	1	0

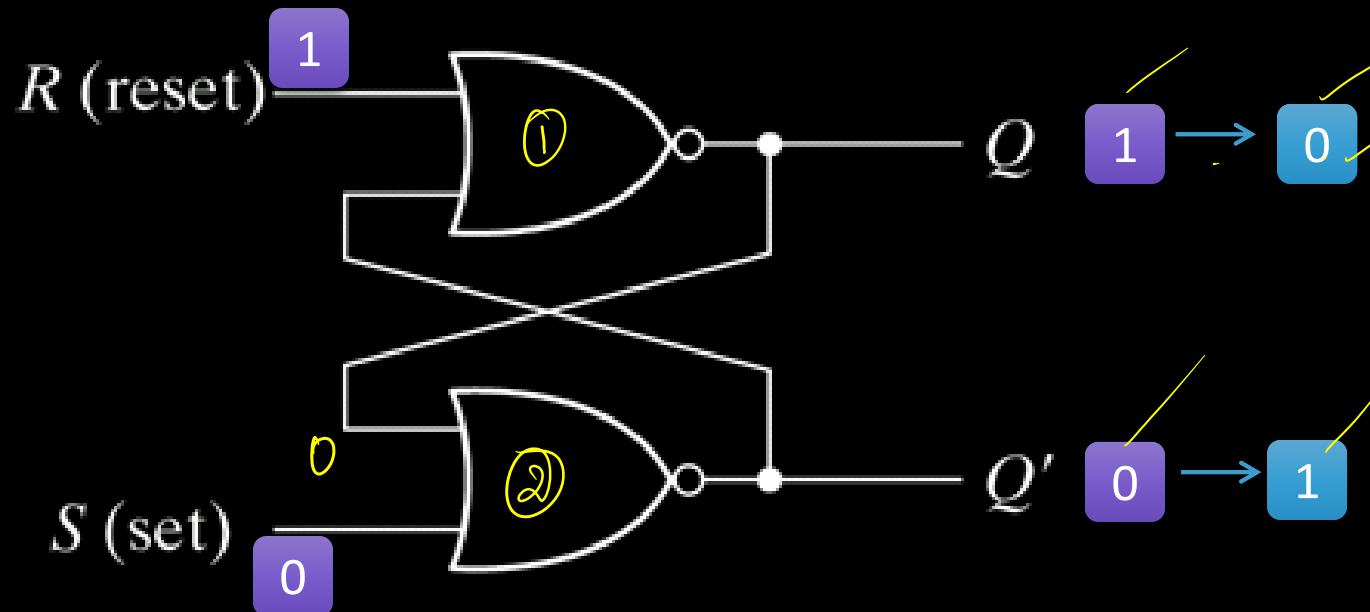
To reset the latch  $R=1$   $S=0$

Assume  
 $Q=1$   
 $\bar{Q}=0$

$Q=1$   
 $S=0$   
 $Q=0$   
 $\bar{Q}=1$

Reset = 1, Set = 0

SR Latch



reset state

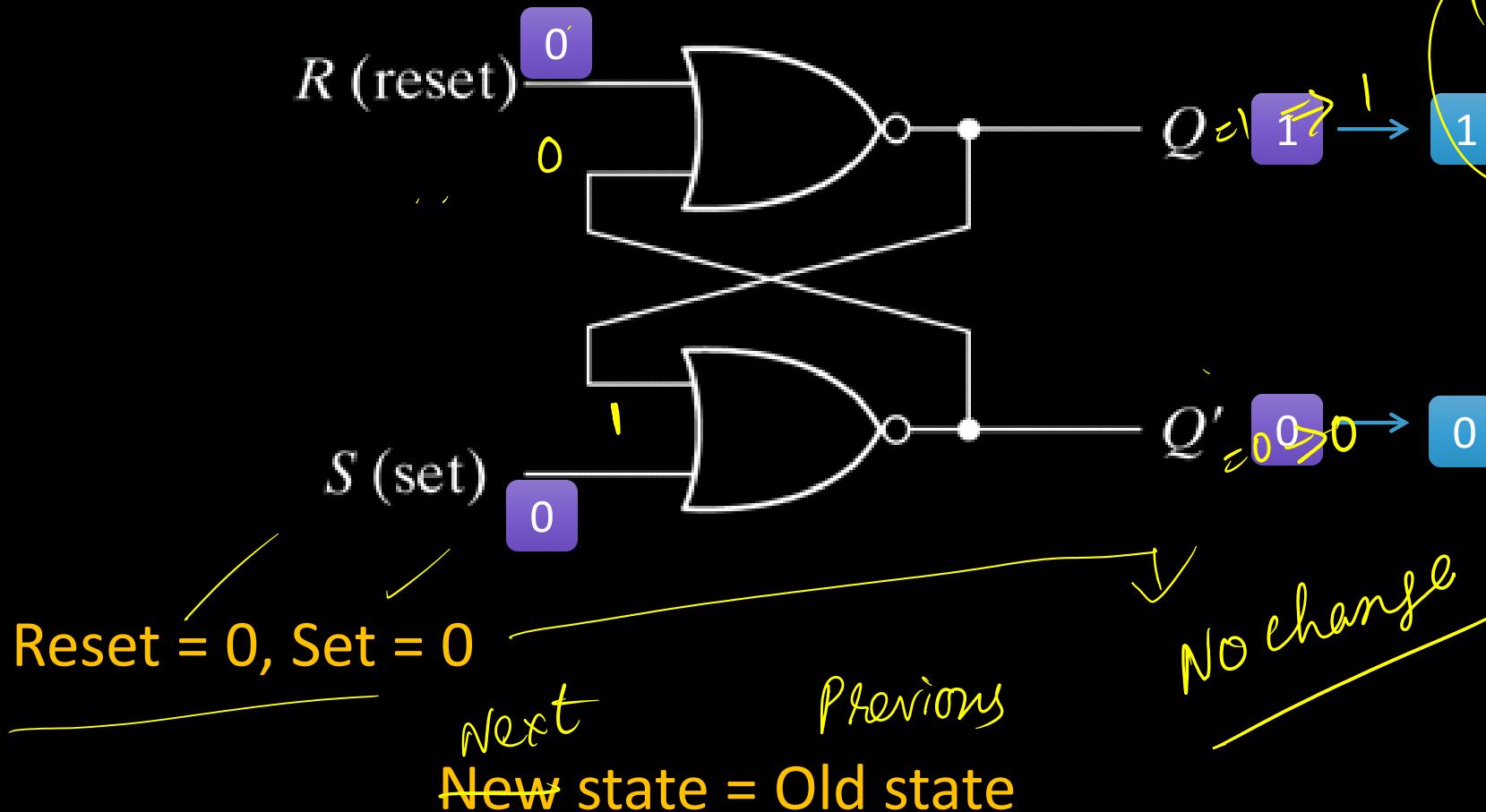
$S_0$   $R_1$   
 $1$   $0$   
 $0$   $0$   
 $1$

Assume  
 $Q=0$   $\bar{Q}=1$   
 $R=1$   $S=0$   
 $Q=0$   $\bar{Q}=1$

CKV

$Q=0$  reset state

## SR Latch

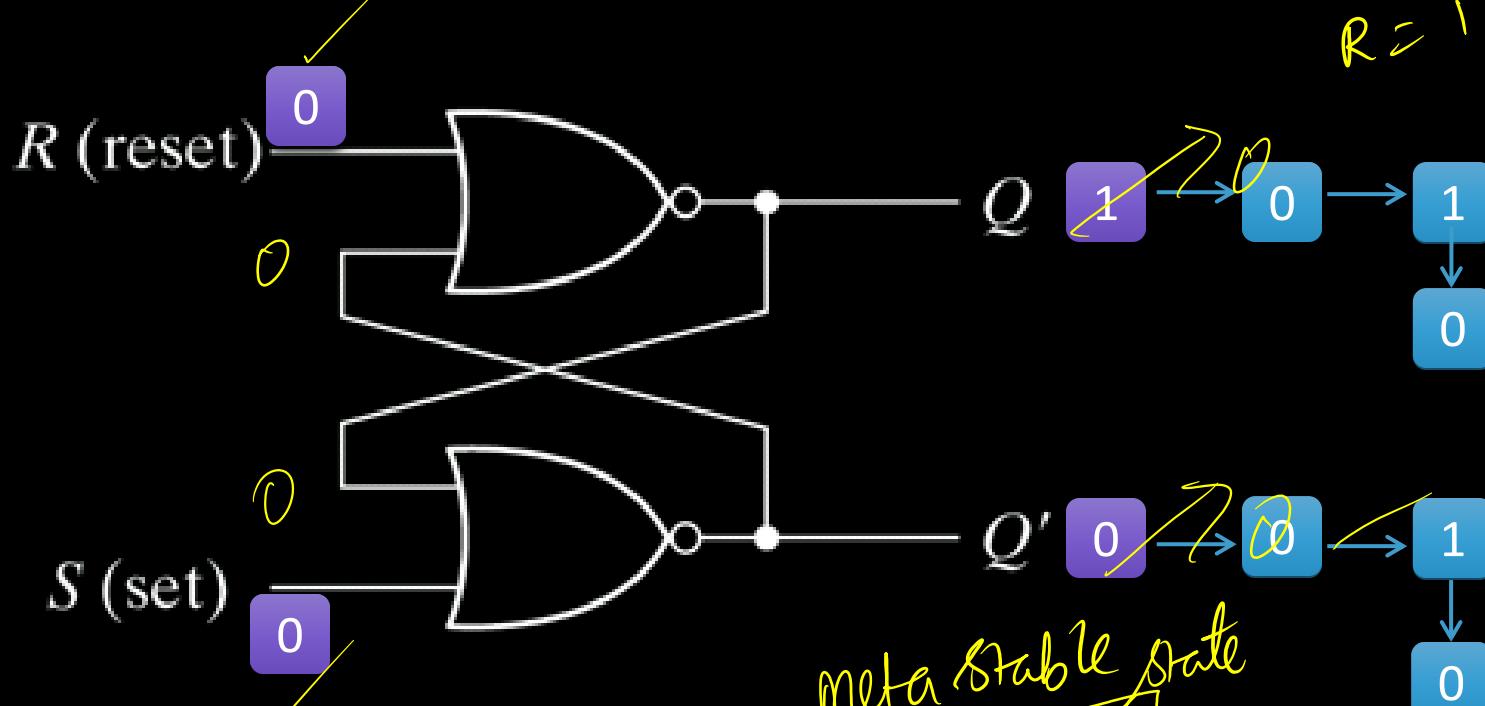


i) Assume  $Q = 0$ ,  $\bar{Q} = 1$   
 $R = 0$ ,  $S = 0$   
 $Q = 0$ ,  $\bar{Q} = 1$

ii) Assume  $Q = 1$ ,  $\bar{Q} = 0$   
 $R = 0$ ,  $S = 0$   
 $Q = 1$ ,  $\bar{Q} = 0$

$\bar{Q} \& Q$  are complementary  
 Assume  
 $\bar{Q} = 0$   
 $Q = 1$   
 $S = 1$   
 $R = 1$

SR Latch



Reset = 1, Set = 1

$Q = 0$  and  $Q' = 0$

metastable state

Not possible

$R = 1$

$S = 1$

Cant predict output  $\rightarrow$  metastable state

avoided

not recommended  
 forbidden  
 not defined

## SR Latch

S	R	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	1	Forbidden	

2 i/p

✓ 20 i/p

No change

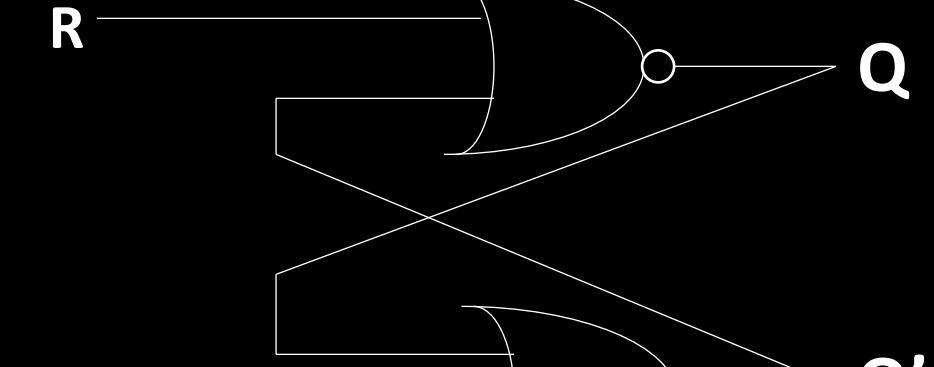
Reset

set

Not recommended

Characteristic table

R → reset i/p  
S → set i/p



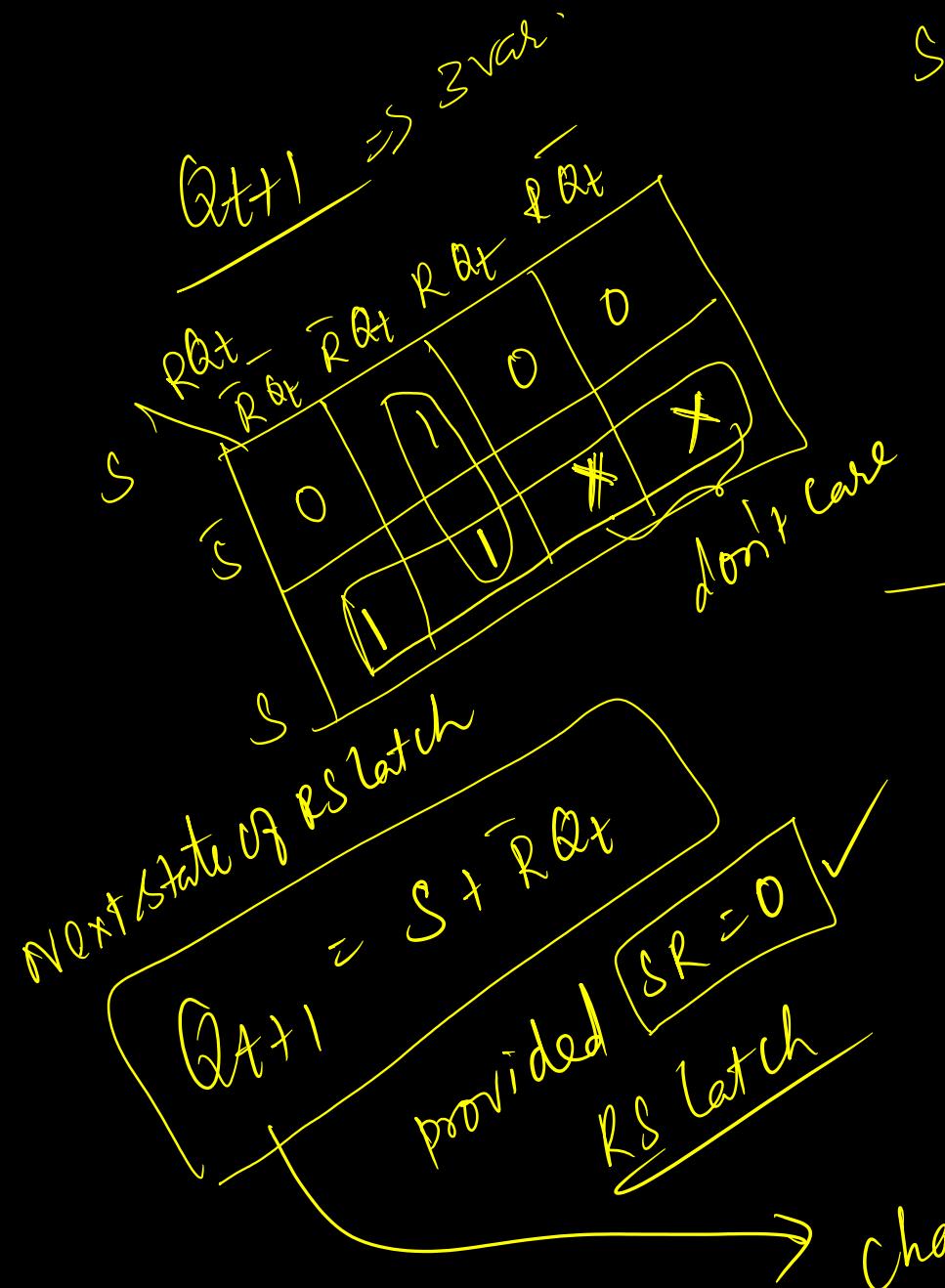
S → Present state  
Qt+1 → Next stable

Q → no change

reset

set

CKV



$S \quad R \quad Q_t \quad Q_{t+1}$

Characteristic table

$S \quad R \quad Q_t \quad Q_{t+1}$

$S$	$R$	$Q_t$	$Q_{t+1}$
0	0	0	0 } No change
0	0	1	1 }
0	1	0	0 } reset
0	1	1	1 } set
1	0	0	x }
1	0	1	x }
1	1	0	x }
1	1	1	Indeterminate undefined

both  $S$  &  $R$  should not be  
'1' at the same time

Characteristic equation



## CS/ECE/EEE/INSTR F215:Digital Design

**Lecture 21: Clocked SR latch, D, JK and T latch**

*Sat, 16 Oct 2021*



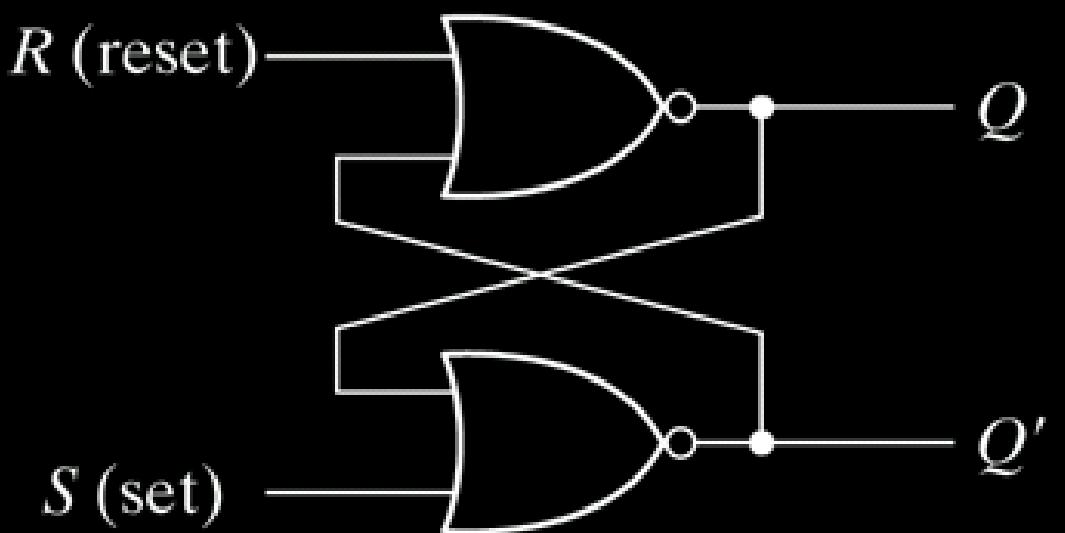
*A seed grows with no sound,  
but a tree falls with huge noise.*

*Destruction has noise,  
but creation is quiet.*

*This is the power of silence.*

*“Grow silently”*

# NOR based RS latch



Characteristic  
equation

$$Q_{t+1} = \overline{S + \overline{R} Q_t}$$

Provided  $SR = 0$

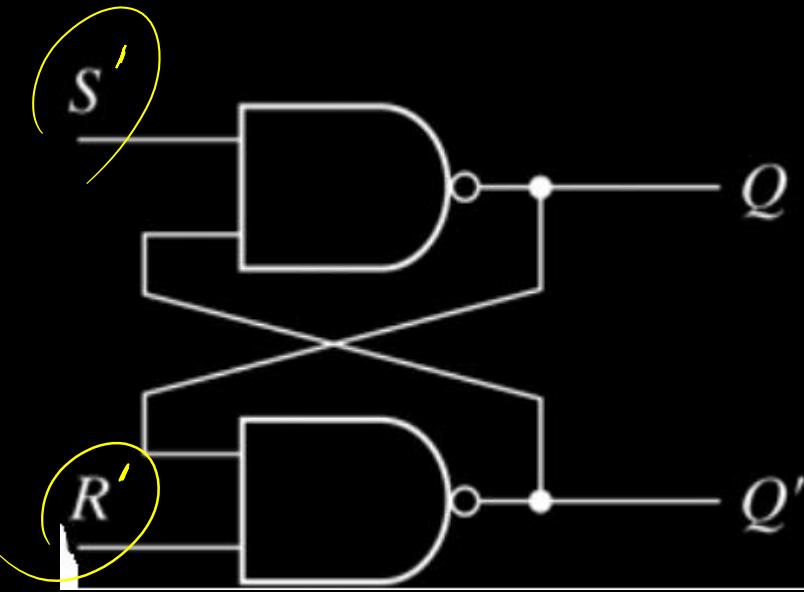
Inputs		Present state	next state of the latch
$S$	$R$	$Q_t$	$Q_{t+1}$
0	0	0	0 } No change
0	1	1	1 } Reset
1	0	0	1 } Set
1	1	1	X } Indet erminate

undefined

# NAND Based Latch

S'R' Latch

Complemented  
version of  
NOR based latch



Two cross-coupled  
NAND gates

$S'$	$R'$	$Q$	$Q'$
0	0	ForbIDDEN	
0	1	1	0
1	0	0	1
1	1	$Q$	$Q'$

not recommended

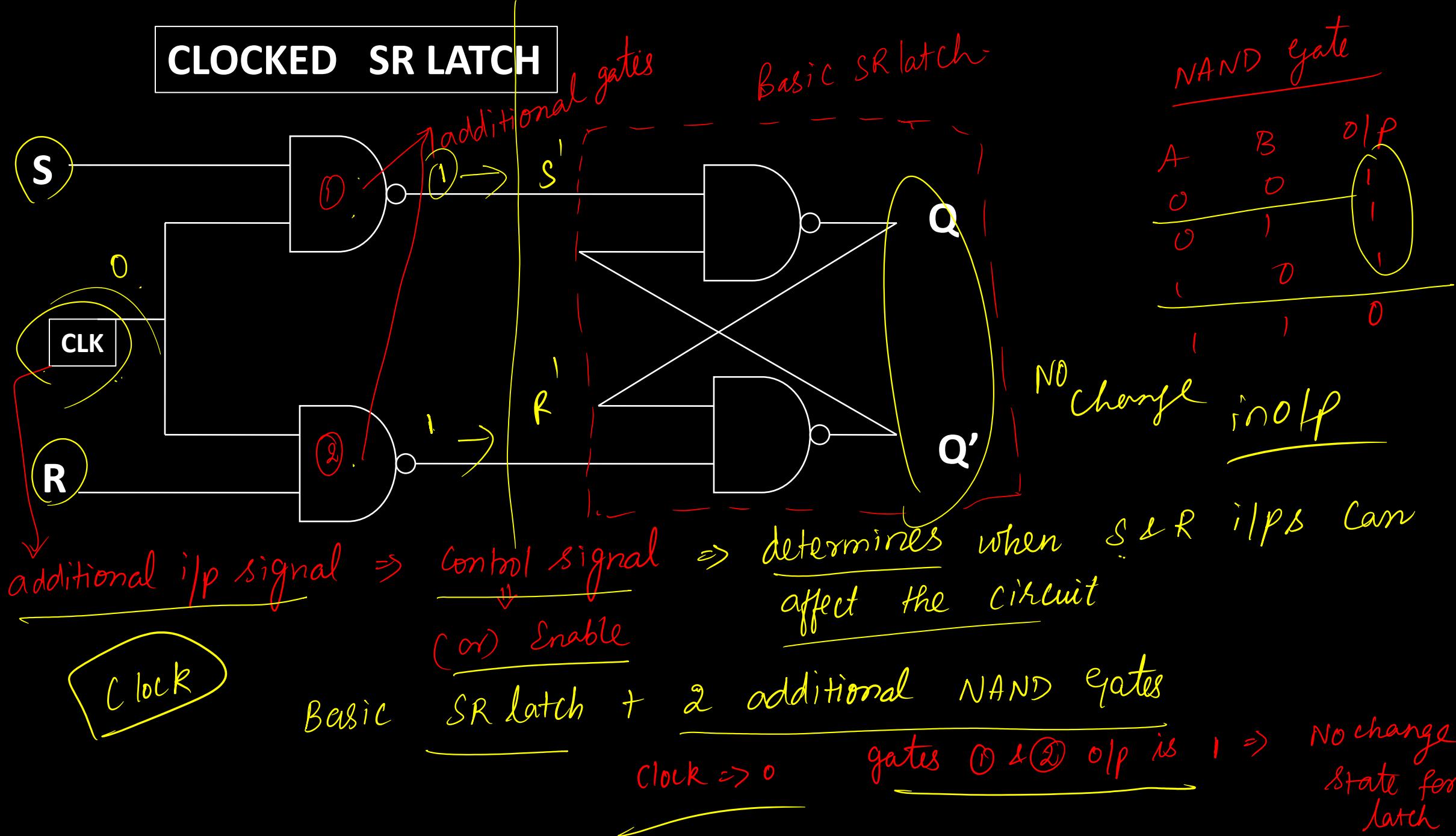
Set

Reset

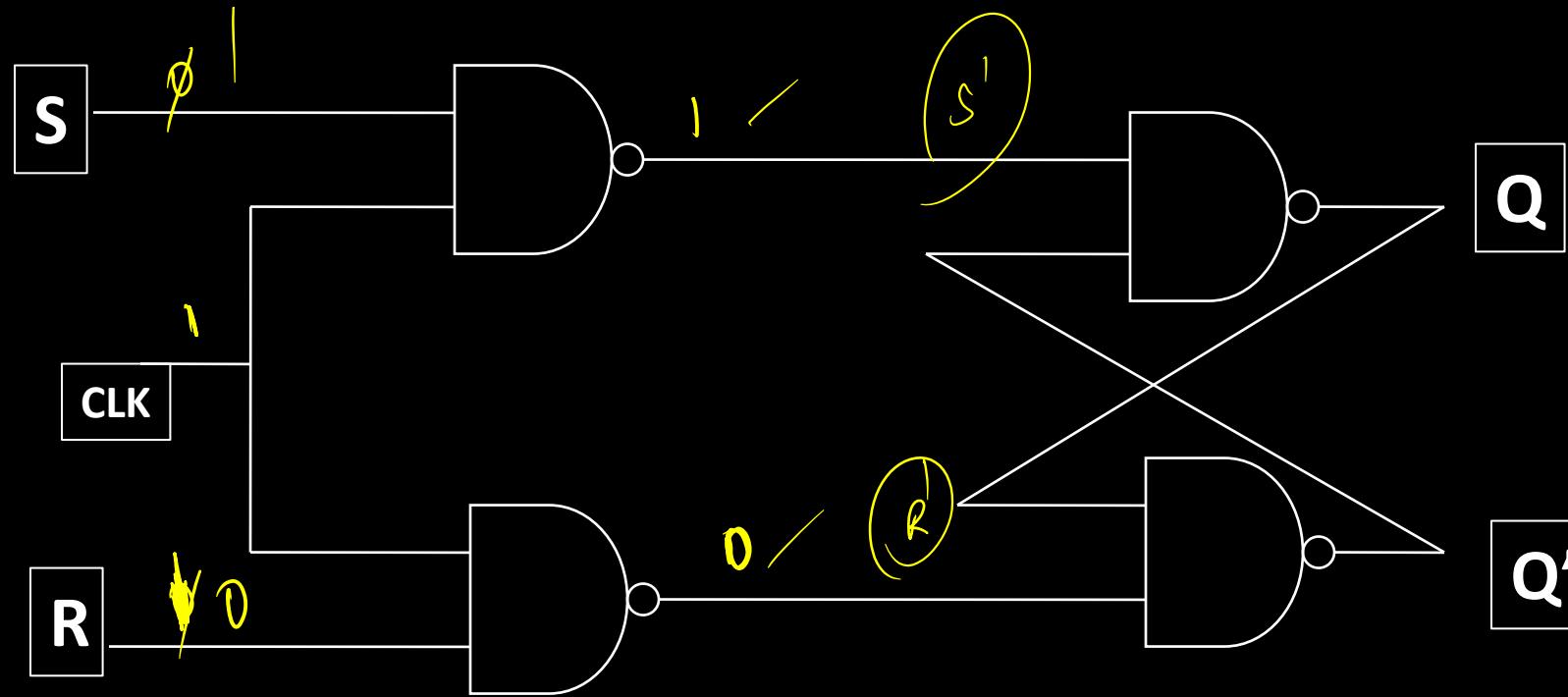
No change

Quiescent state

## CLOCKED SR LATCH

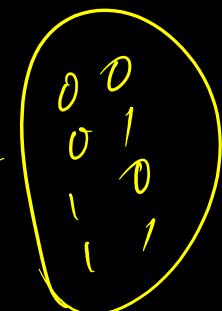


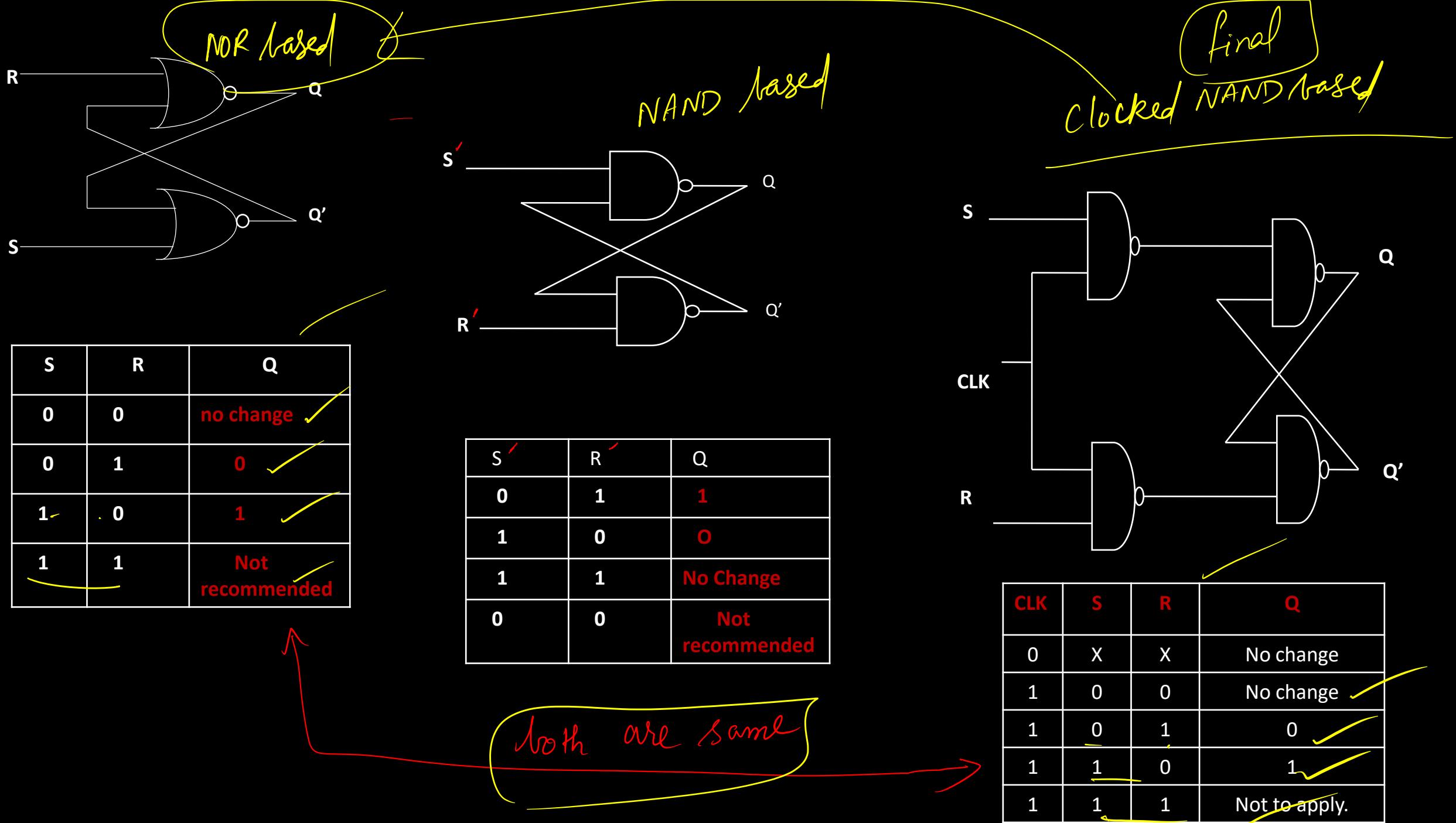
## CLOCKED SR LATCH



A control (or) clock input is added to determine when S & R inputs should affect the circuit

CLK	S	R	Q
0	X	X	No change
1	0	0	No change
1	0	1	0 Reset state
1	1	0	1 Set state
1	1	1	Not to apply / Indeterminate





When an SR NOR latch goes into an indeterminate state?

when both inputs are '1'

When an SR NAND latch goes into an indeterminate state

when both inputs are '0'

Problem in SR latch ?

drawback (or) D-adv

Indeterminate state

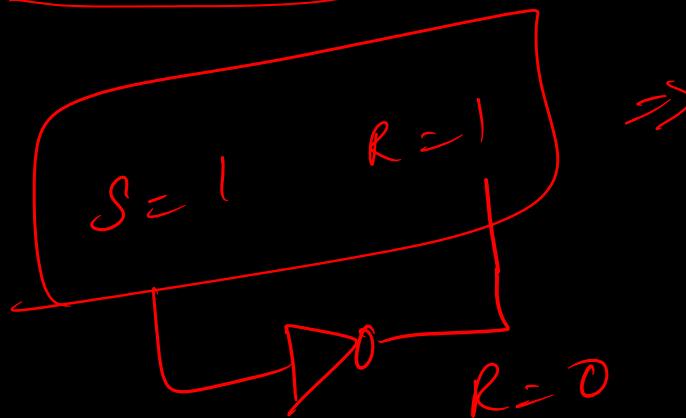
$\Rightarrow$  both inputs S & R should not be '1' at the same time.

SR latch is important circuit as it is useful for other circuits constructed from it.

This makes it difficult to manage

How to avoid this ?

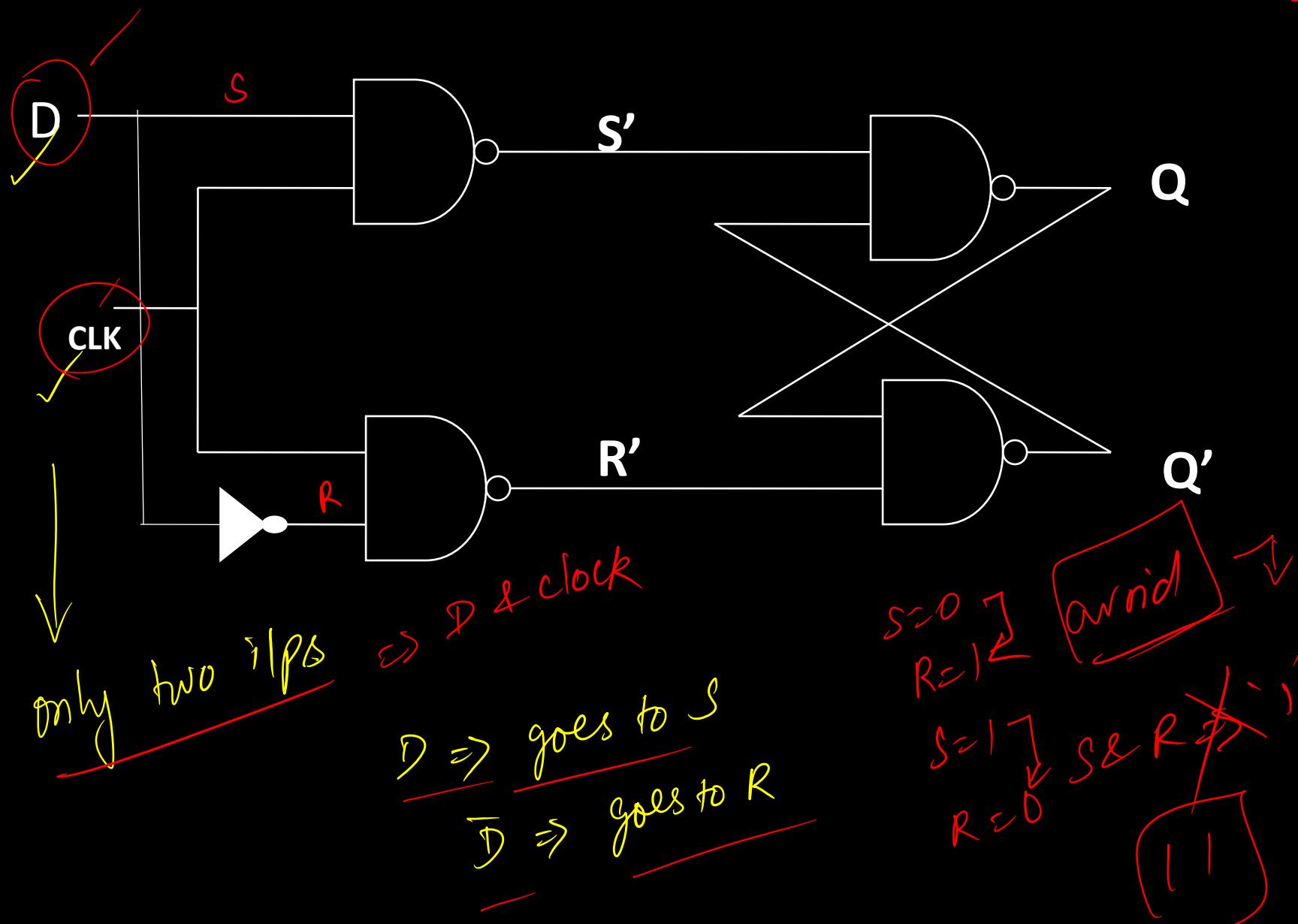
Solution  $\Rightarrow$  ?



O/P keep or change

should not happen

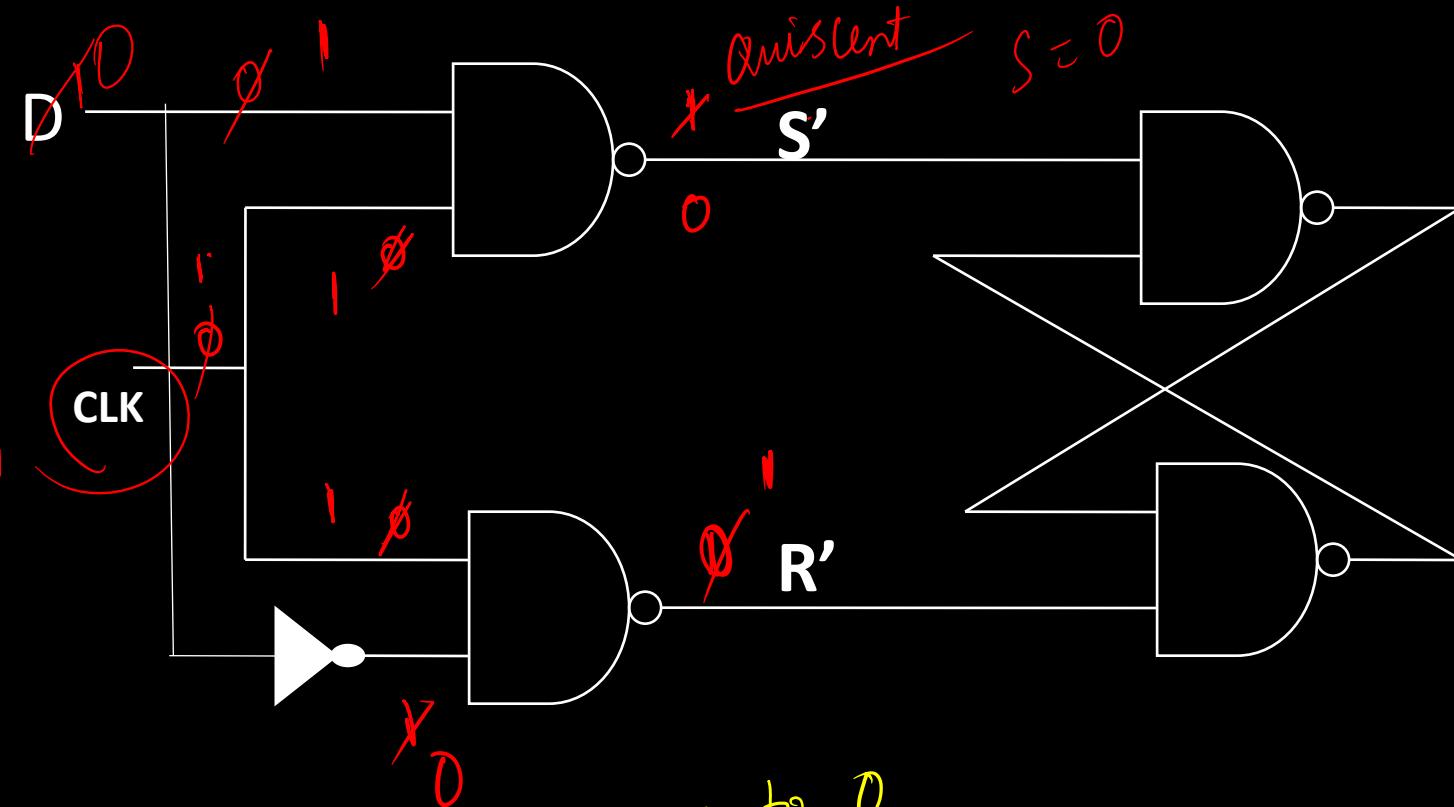
D latch



One way to eliminate  
the undesirable condition  
in SR latch is to  
ensure that both  
 $S'$  &  $R'$  inputs are  
never equal to '1' at  
the same time.

This can be obtained  
by connecting a  
NOT gate b/w them.

↓ Inverted b/w  
S & R inputs



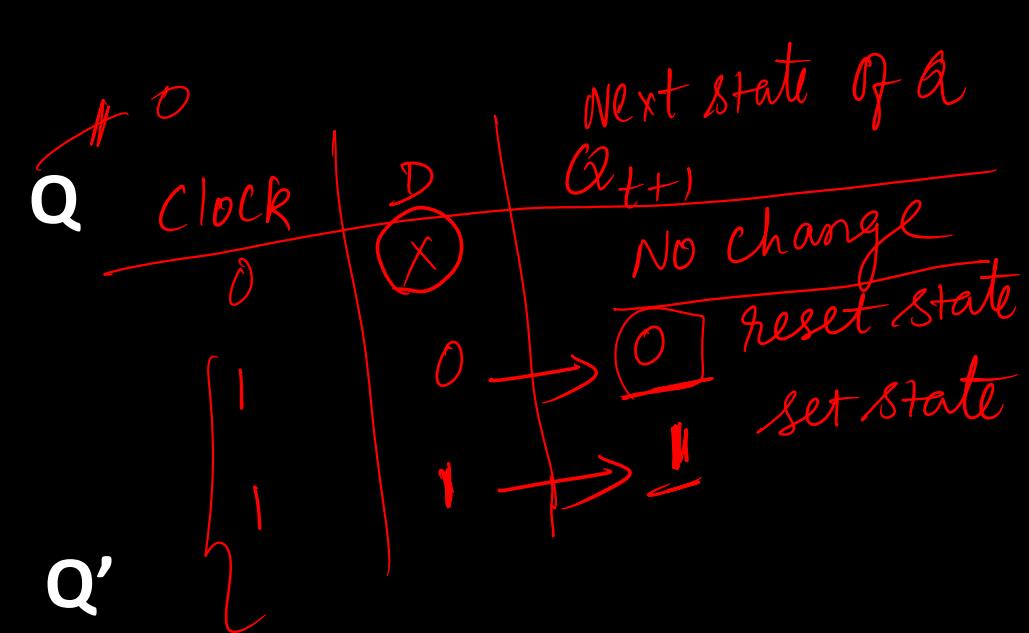
if  $D = 0$

$Q$  goes to 0

if  $D = 1$

$Q$  goes to 1

D latch is transparent latch



Binary information present at the data i/p ( $D$  i/p) of D latch is transmitted to  $Q$  o/p of D latch when clock i/p is asserted. o/p follows changes in i/p when clock is high.



Race around condition

Jk Latch

SR latch is modified  
so that the in determinate

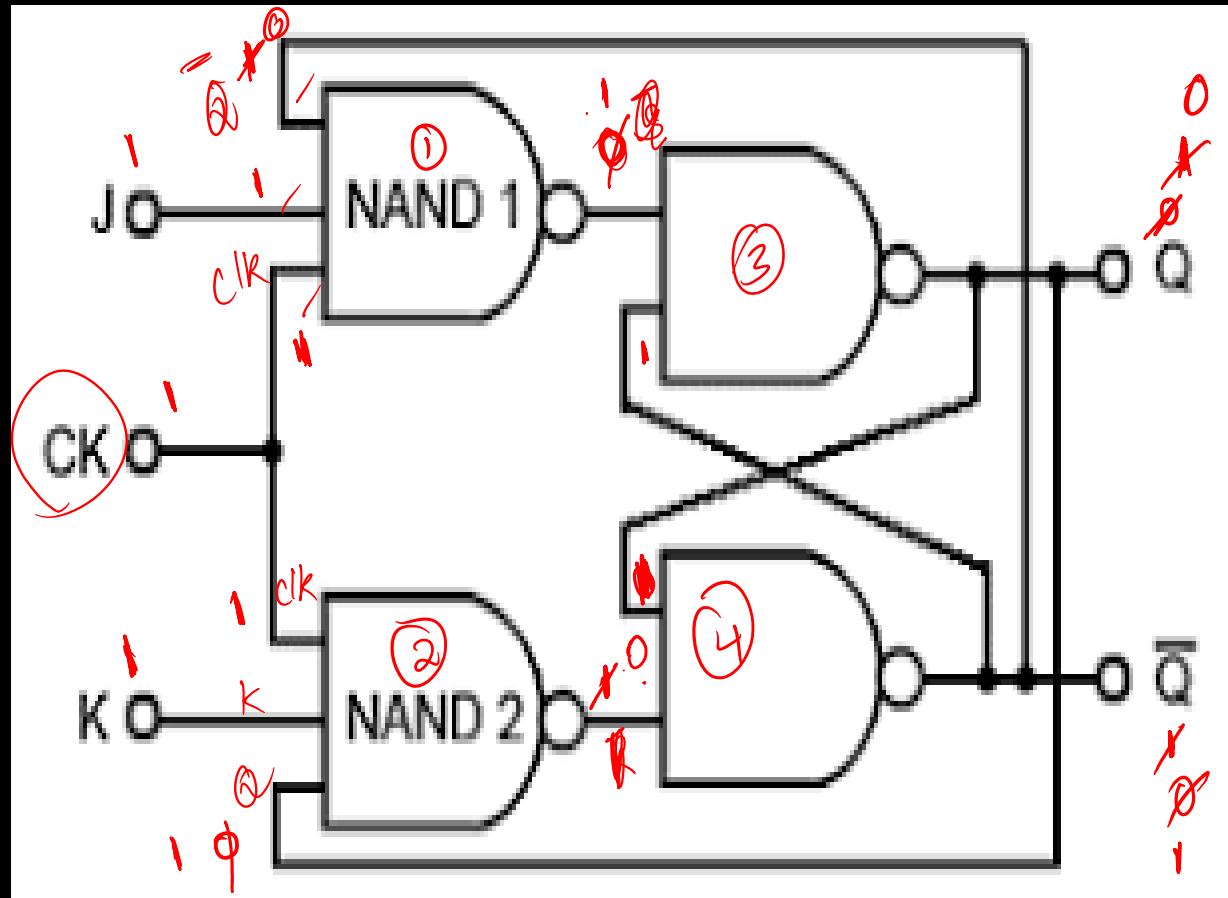
is determined

$\Rightarrow$  Set i/p

$\Rightarrow$  Reset i/p

$$\begin{cases} J = S \\ K = R \end{cases}$$

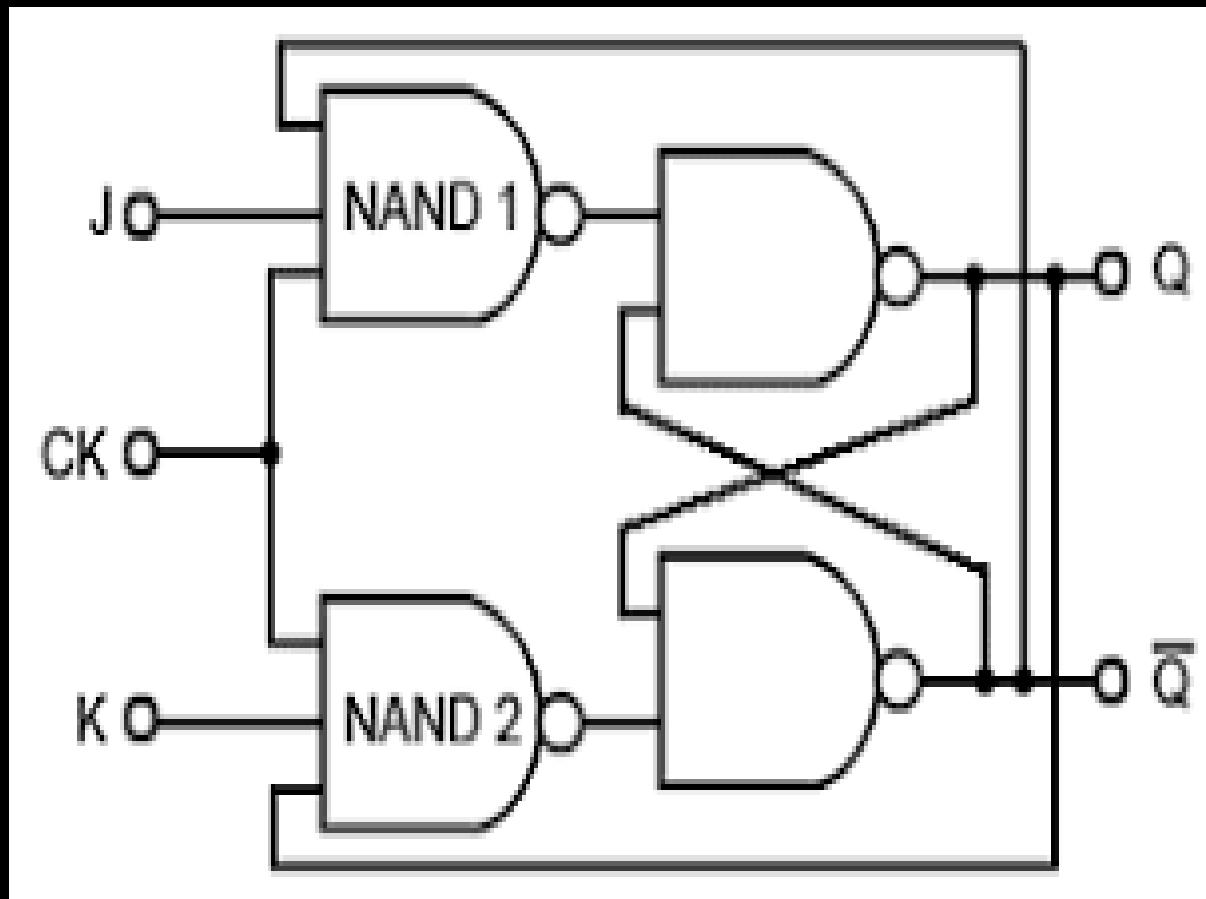
When both J & k are 1  
Jk latch switches to its  
complement state that is



Modified version of SR latch  
where the undefined  
state of both the i/p's  
being '1' is  
defined

$$\text{if } Q_t = 1 \quad Q_{t+1} = 0$$

$$Q_t = 0 \quad Q_{t+1} = 1$$



clock

$J$

$K$

$Q$

$\bar{Q}$

clock

$J$

$K$

toggling

complementing itself

Problem of latch

race around condition

This hand-drawn diagram illustrates the timing and race-around issues in the circuit. It shows two clock signals, one for each flip-flop. The top flip-flop's clock signal is labeled 'clock' and its J and K inputs are labeled 'J' and 'K'. The bottom flip-flop's clock signal is also labeled 'clock' and its J and K inputs are labeled 'J' and 'K'. The output  $Q$  of the top flip-flop and the output  $\bar{Q}$  of the bottom flip-flop are shown. Red annotations explain that the clock signals must be synchronized to avoid a race-around condition where both flip-flops try to change state at the same time, which can lead to a temporary undefined state or latching. The 'complementing itself' note likely refers to the self-complementary nature of the D flip-flop's behavior when the clock edge occurs during a transition.

$Q_{t+1}$ 

$J$	$K$	$\bar{Q}_t$	$\bar{K}\bar{Q}_t$	$K\bar{Q}_t$	$K\bar{Q}_t$
0	0	1	0	0	0
0	0	0	1	0	1
0	1	1	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	0	1	1	0	1
1	0	0	0	0	0
1	1	1	0	1	1
1	1	0	1	0	0

$$Q_{t+1} = J\bar{Q}_t + \bar{K}Q_t$$

Characteristic equation  
of JK latch

$I/Ps$		$P.S$	$N.S$	$Q_{t+1}$
$J$	$K$	$Q_t$	$Q_{t+1}$	
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	1	
1	1	1	0	

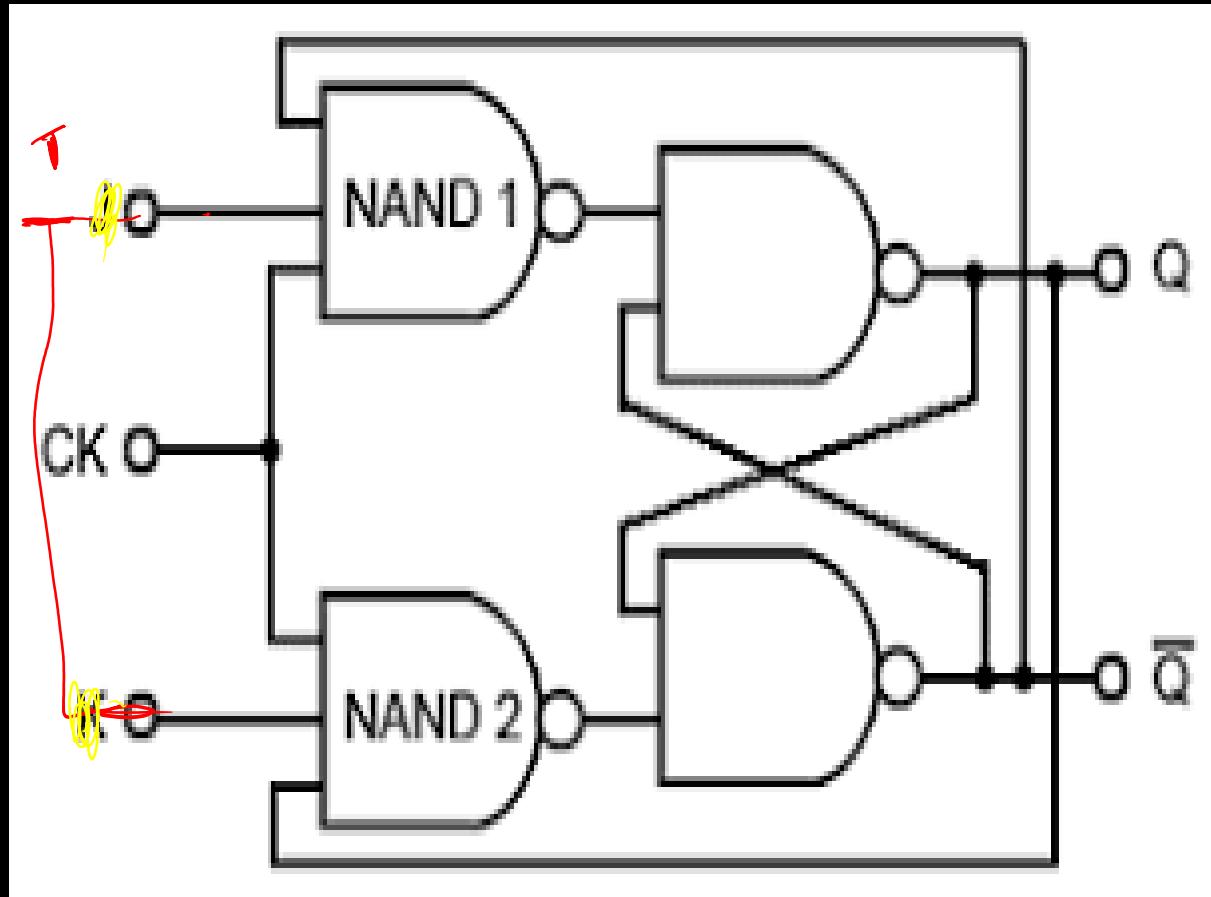
Annotations:

- Row 1:  $Q_t = 0, Q_{t+1} = 0$  } No change
- Row 2:  $Q_t = 1, Q_{t+1} = 1$  } No change
- Row 3:  $Q_t = 0, Q_{t+1} = 0$  } reset state
- Row 4:  $Q_t = 1, Q_{t+1} = 0$  } set state
- Row 5:  $Q_t = 0, Q_{t+1} = 1$  } toggle
- Row 6:  $Q_t = 1, Q_{t+1} = 1$  } toggle

Legend:

- $Q_t \rightarrow \text{No change}$
- 0 reset
- 1 set
- Toggle

T latch



$T$   
 0       $Q_{t+1}$   
 1       $Q_t$       No change  
 Toggle

Char. eqn

$$Q_{t+1} = \bar{T}Q_t + T\bar{Q}_t$$

Both inputs  $J$  &  $K$  are tied together  
& a single input  $T$  is obtained

$$T \geq \underline{J=K}$$

$$\frac{J=0 \ K=0}{J=1 \ K=1}$$

Char. table

$T$	$Q_t$	$Q_{t+1}$	
0	0	0	$\downarrow$ No change
1	1	0	$\downarrow$ Toggle



# CS/ECE/EEE/INSTR F215:Digital Design

## Lecture 22: *Flipflops*

*Tue, 26 Oct 2021*



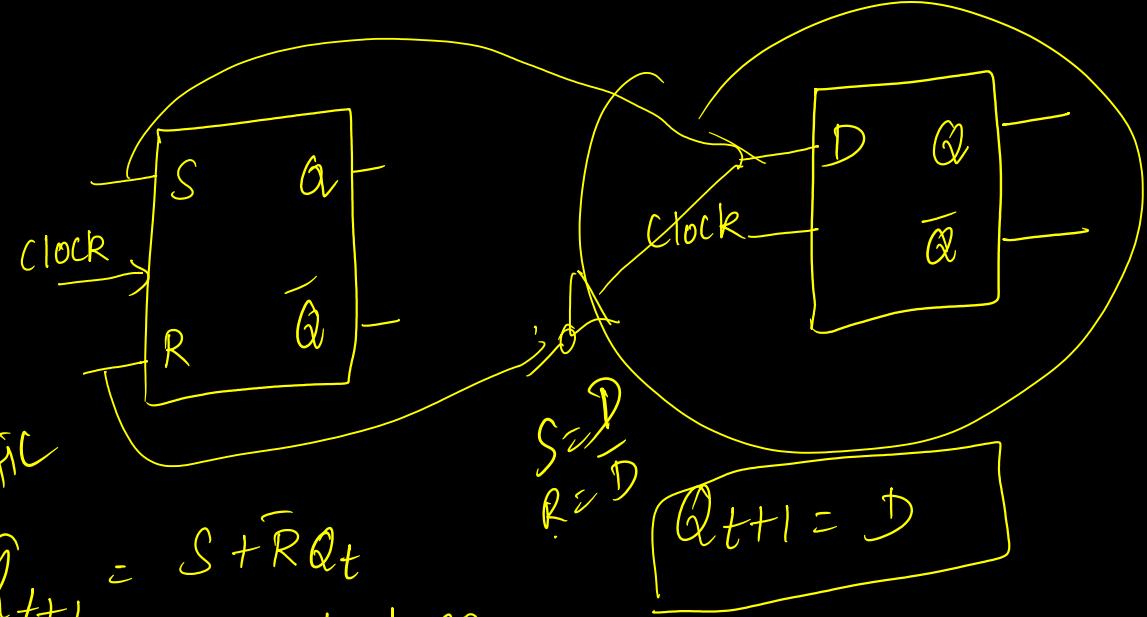
*Our greatest weakness is giving up.*

*The most certain way to succeed*

*is always to try just one more time*

*- Thomas. A. Edison*

Latches - SR, D, JK and T modified

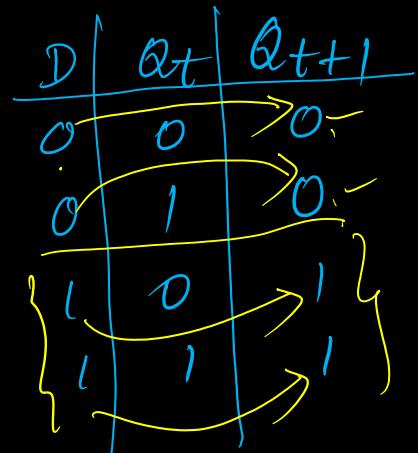


# Symbol

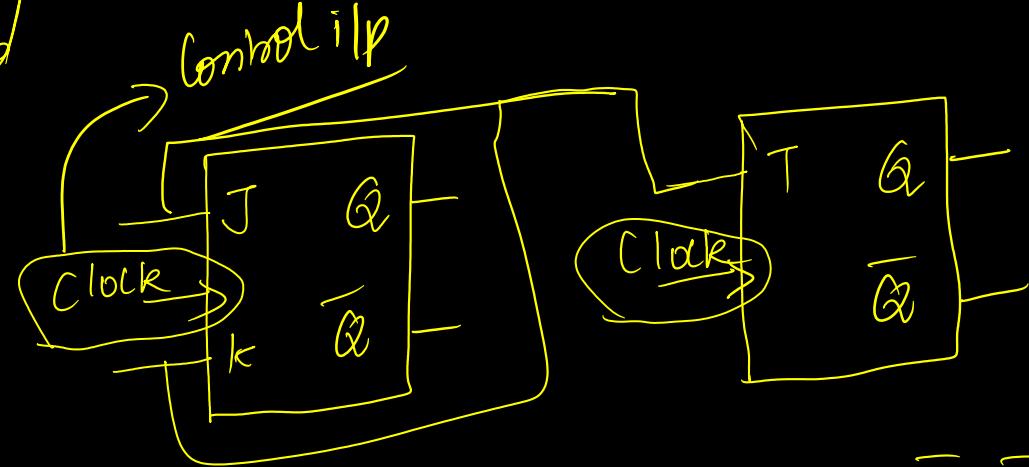
Characteristic  
equation  $Q_{tt}$

+1  
Provided SR = 0

SIPS	PS	NS		
S	R	Q <sub>t</sub>	Q <sub>t+1</sub>	
1	0	0	0	$\rightarrow 0$ } NC
0	0	1	1	$\rightarrow 1$ }
0	1	0	0	$\rightarrow 0$ } Rese
0	1	1	1	$\rightarrow 0$ }
1	0	0	0	$\rightarrow 1$ } set
1	0	1	1	$\rightarrow 1$ }
1	1	0	0	$\times$ } In
1	1	1	1	$\times$ }



terminal  
ate  
overruled



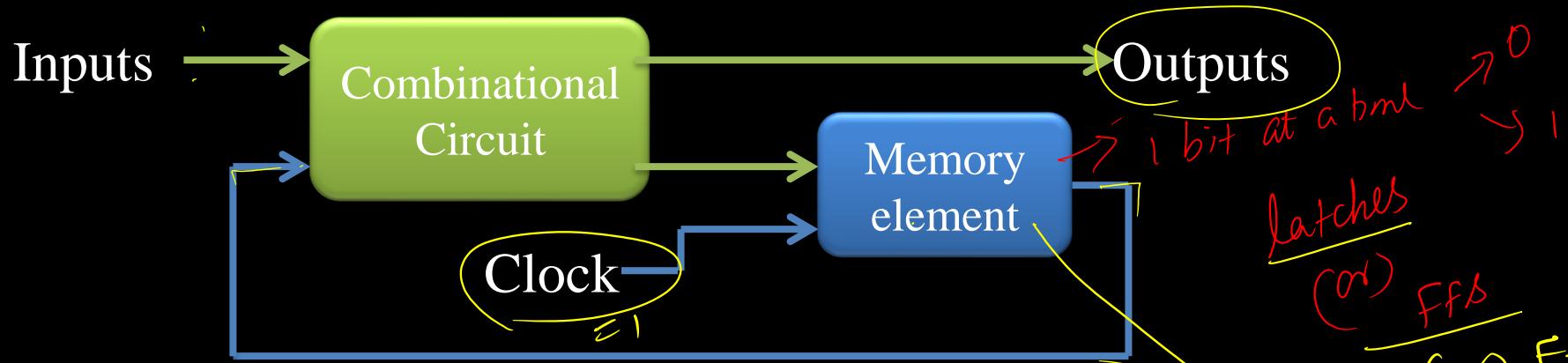
$$Q_{t+1} = J \bar{Q}_t + \bar{\epsilon} Q_t$$

$$\hat{Q}_{t+1} = \bar{T}\hat{Q}_t + \bar{T}\hat{\beta}_t$$

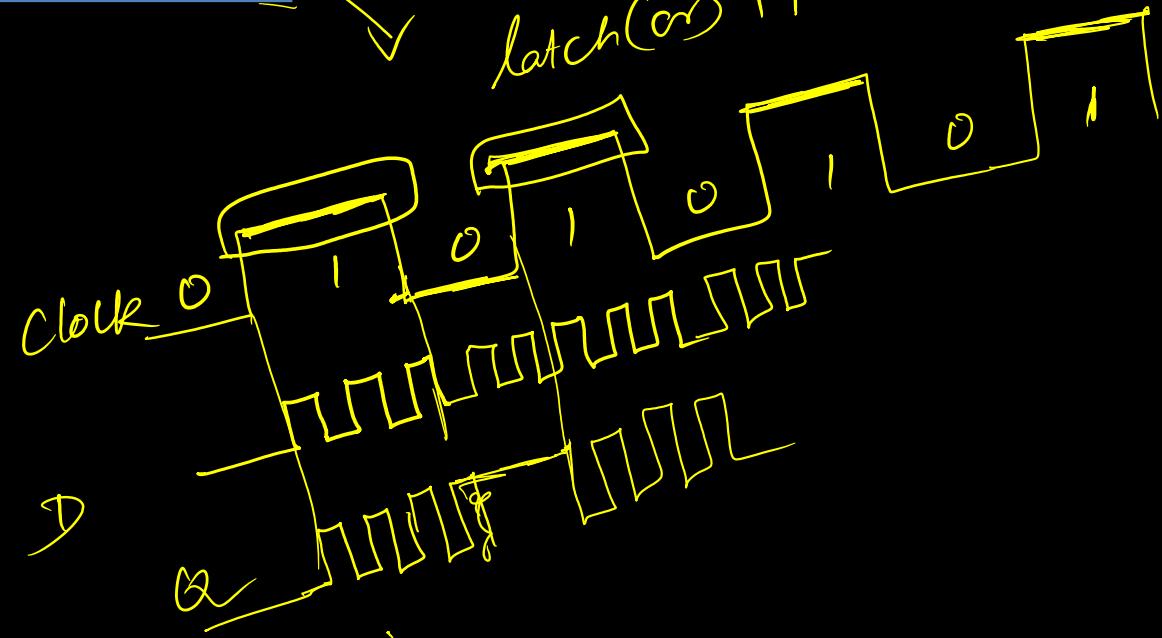
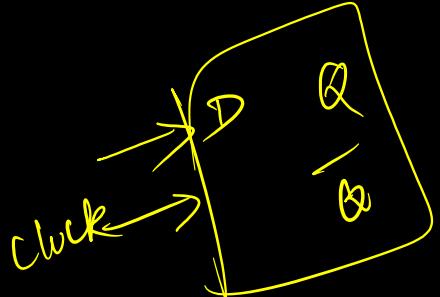
Clock	$J$	$K$	$Q_t$	$Q_{t+1}$
0	0	0	0	0 } NC
0	0	1	1	
0	1	0	0 } reset	
0	1	1	0 } set	
1	0	0	1 } toggle	
1	0	1	1 }	
1	1	0	1 → 0 } toggle	
1	1	1	0 → 1 }	

T	$Q_t$	$Q_{t+1}$
0	0	0
0	1	1
1	0	1
1	1	0

# Sequential Circuit



J  
K  
D



Latches are level sensitive, state transitions of latches start as soon as clock appears

As the output is connected to input of latches through combinational circuit

If input changes while the clock is still '1', the latches respond and give new p/p  
*when clock = 1*

Latches are "transparent" (any change on the inputs is seen at the outputs immediately).

This causes synchronization problems!

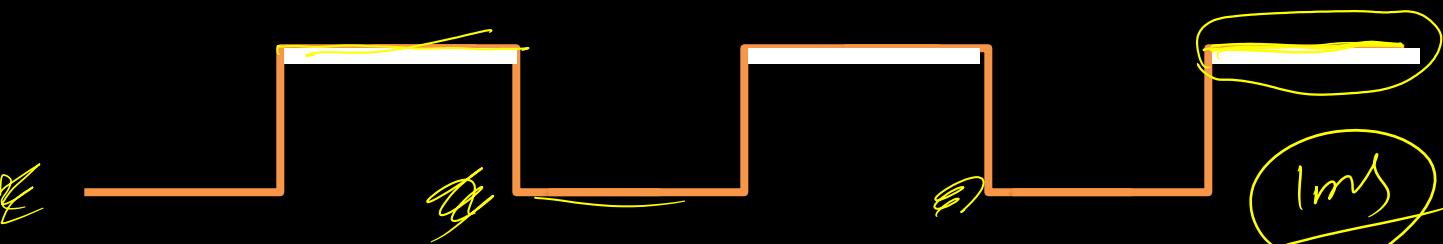
set of latches  
clk = 1  
when it p  
changes  
output of latch change  
connected  
to another as if  
all latches are  
connected through  
common clock

Unpredictable Behaviour -

Solution: Use latches to create flip-flops that can Respond ONLY on  
SPECIFIC times →

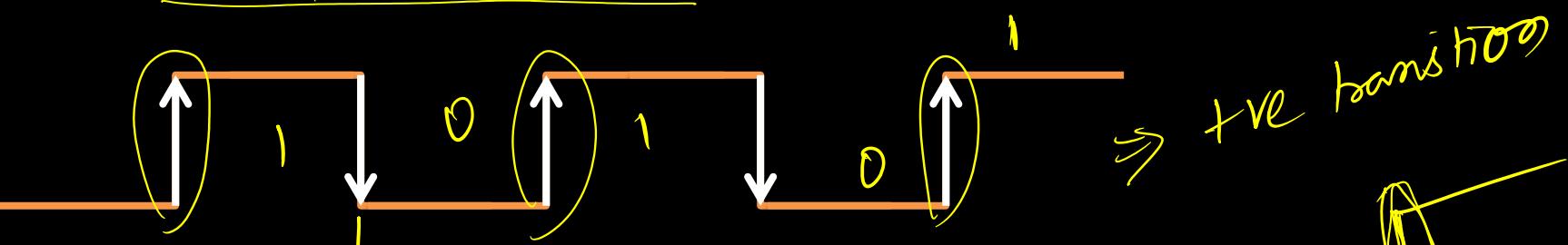
# Sequential Circuits

Latch – Responds to change in level of clock pulse



The key to the proper operation of a flip flop is to trigger it only during signal transition.

*(Note: The original text 'The key to the proper operation of a flip flop is to trigger it only during signal transition.' is underlined in yellow, and the handwritten note '→ tve edge triggering' is written next to it.)*



*Two ways to build a flip flop*

*Special configuration of two latches to isolate the output of flip-flop  
Gate based design which triggers only during signal transition of clock and is disabled during rest of clock pulse*

*external circuit*

Latch  $\Rightarrow$    $\Rightarrow$

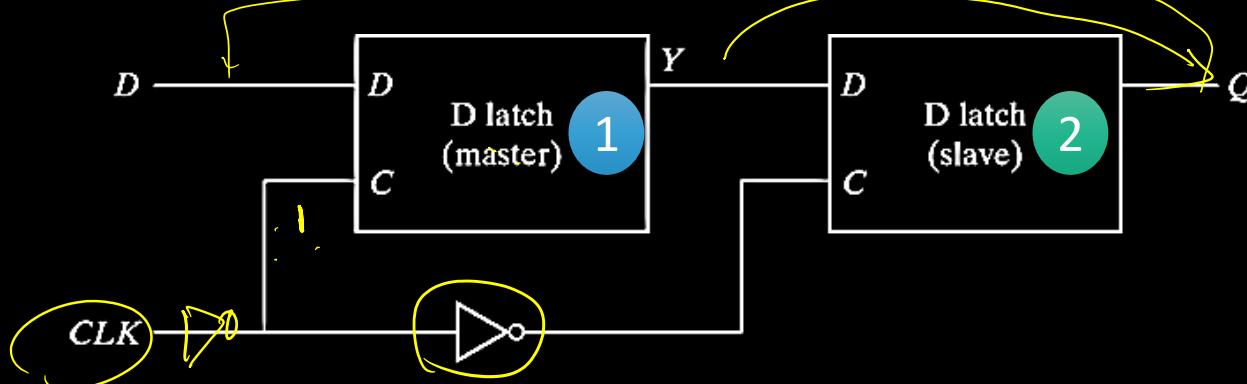
State change will occur  
only during clock transition

how can we modify

2 latches  $\Rightarrow$  o/p of the latch

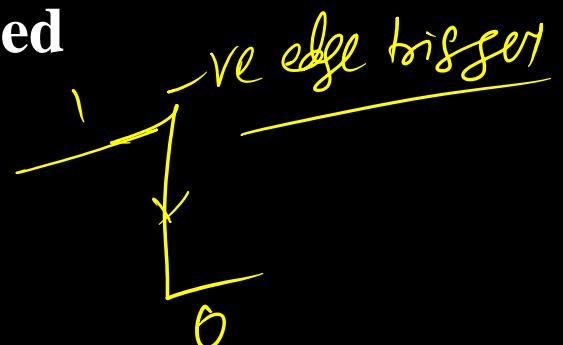
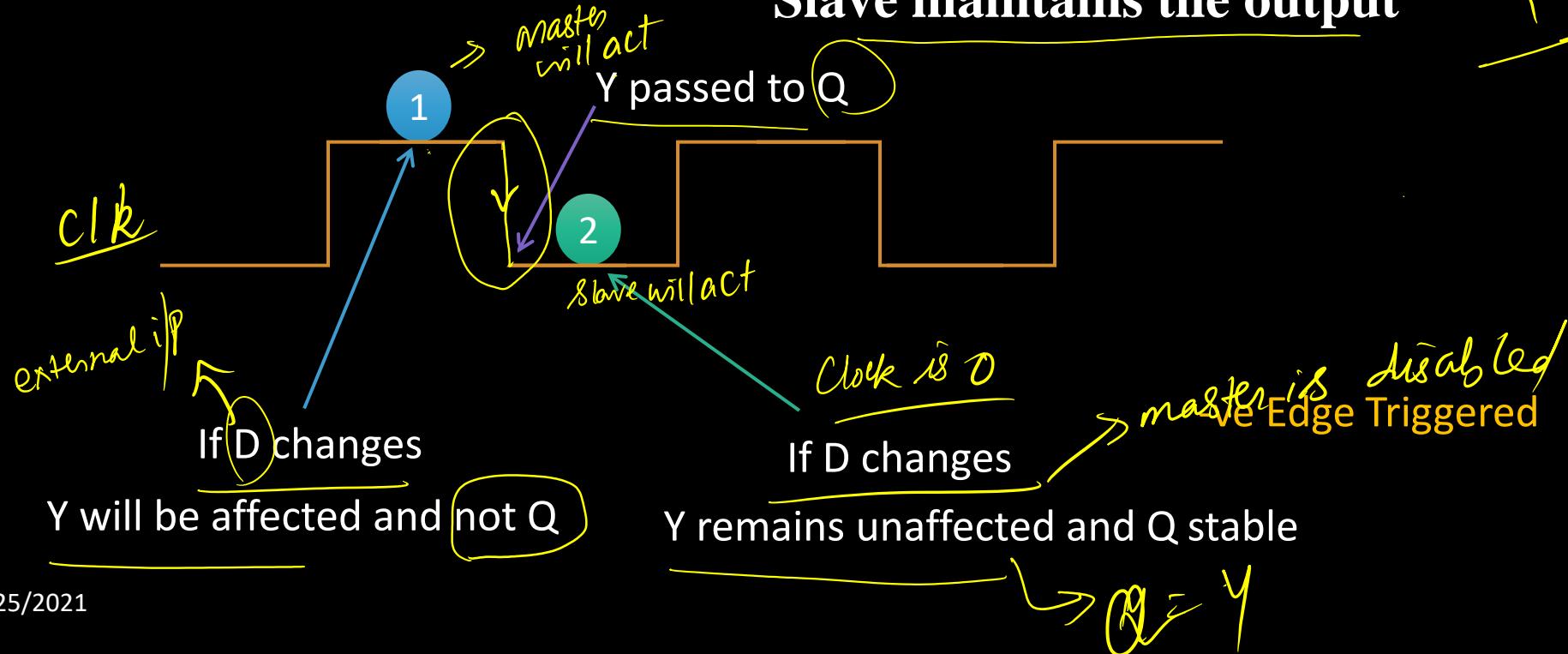
# Master Slave Configuration

⇒ 2 latches + inverter



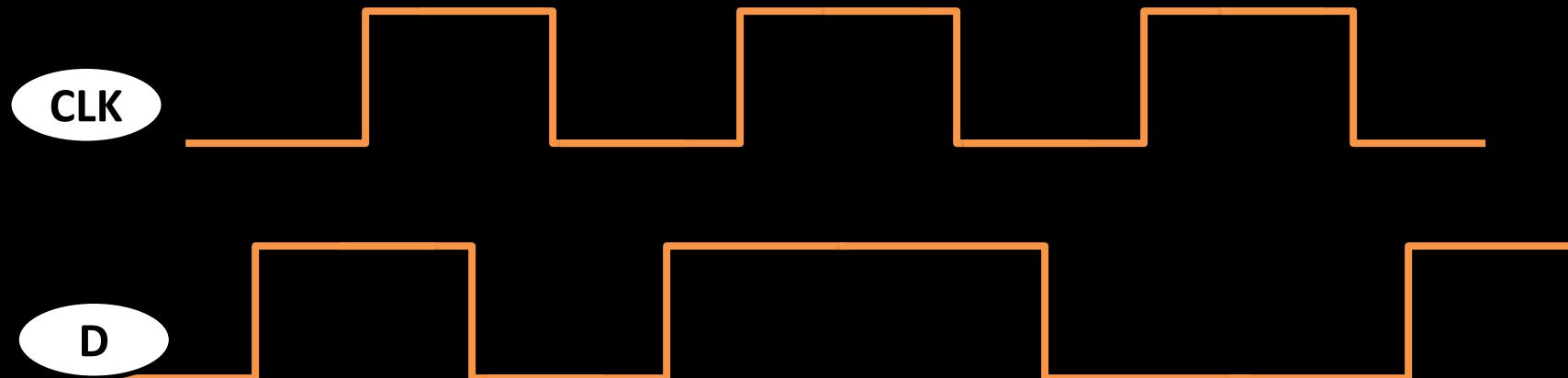
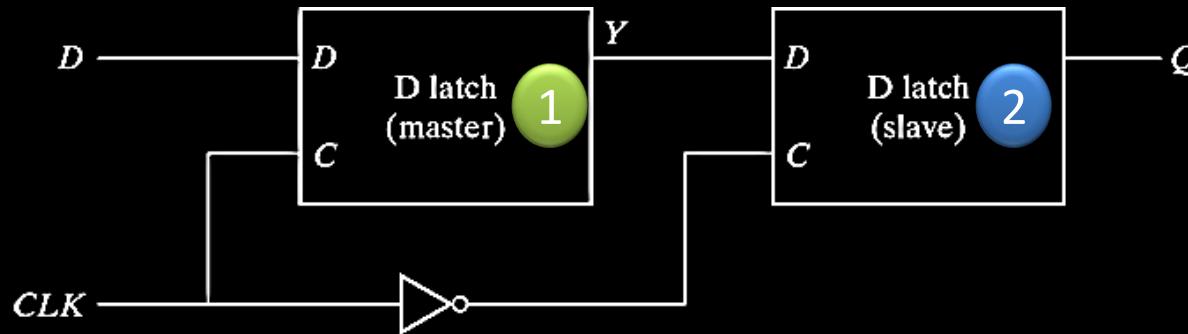
What happens when CLK is logic 1

Master enabled and slave disabled  
Slave maintains the output



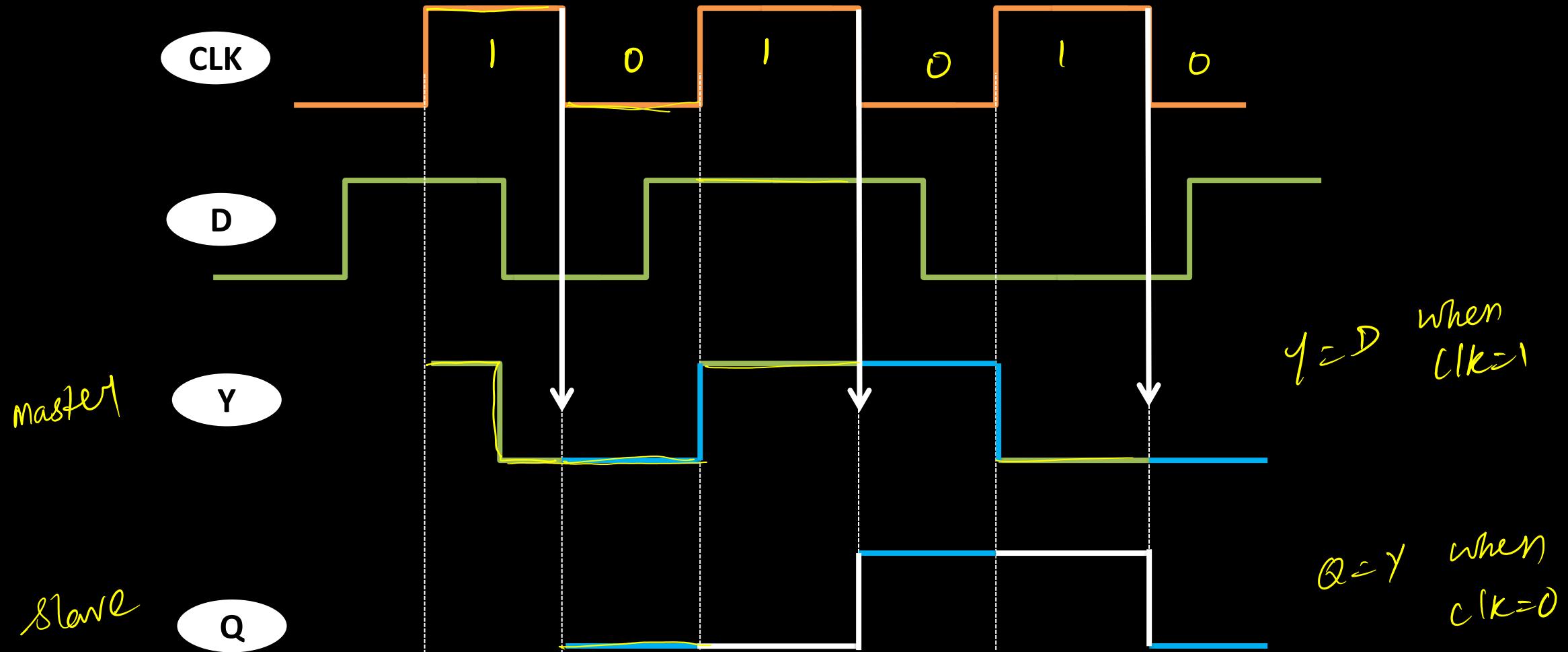
positive edge triggered

# Master Slave Flip-flop



# Edge Triggered D Flip-Flops

## Master Slave Flip-flop





## CS/ECE/EEE/INSTR F215:Digital Design

**Lecture 23: *Flipflops and analysis of clocked sequential circuits***

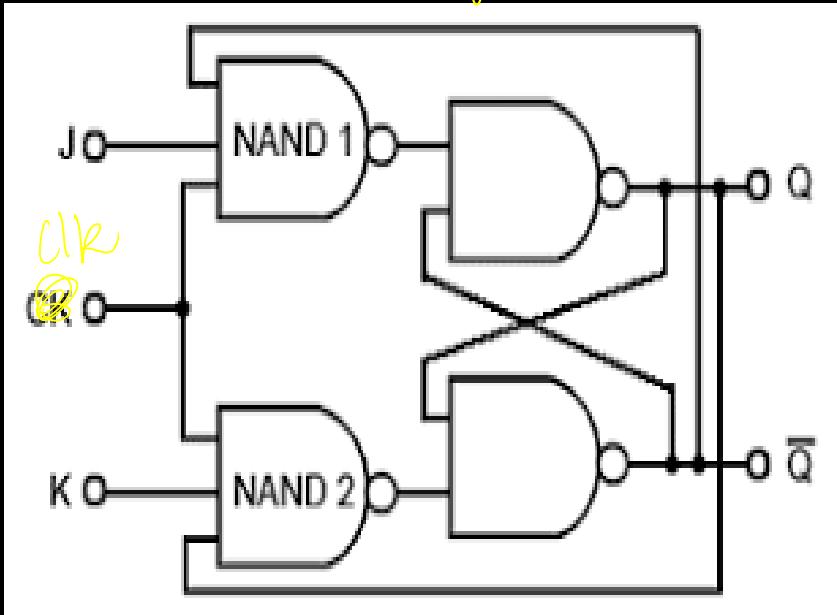
***Thu, 28 Oct 2021***

The strongest factor for success is  
Self-esteem:

Believing you can do it,  
Believing You deserve it and  
Believing you will get it

- Latches are Level Sensitive (level triggered)
  - When the control/clk input is 1, latches change state as and when the input applied to latches changes.
  - Unreliable operation. In JK latch **race around condition** occurs
  - Synchronization is not possible when output of latch is applied directly or through combinational logic circuit to input of the same or another latch and all latches are controlled by same clock
  - Solution: Modify latch to work as flip-flop
  - Flip-flops trigger only during clock transition
  - Positive transition (+ve edge: clock changes from 0 to 1)
  - Negative transition(-ve edge: clock changes from 1 to 0)

# level triggered Jk Latch



Characteristic table

J	K	$Q_{t+1}$
0	0	$Q_t \rightarrow NC$ No change
0	1	$0 \rightarrow$ reset
1	0	$1 \rightarrow$ set
1	1	Toggle



$$T/2 = 5 \text{ ns/8}$$

Race around  
condition  
(or)

Racing

To prevent race around condition

- 1)  $T/2$  < Propagation delay of gate long
- 2) Edge triggering instead of level triggering 20ns
- 3) Master Slave operation

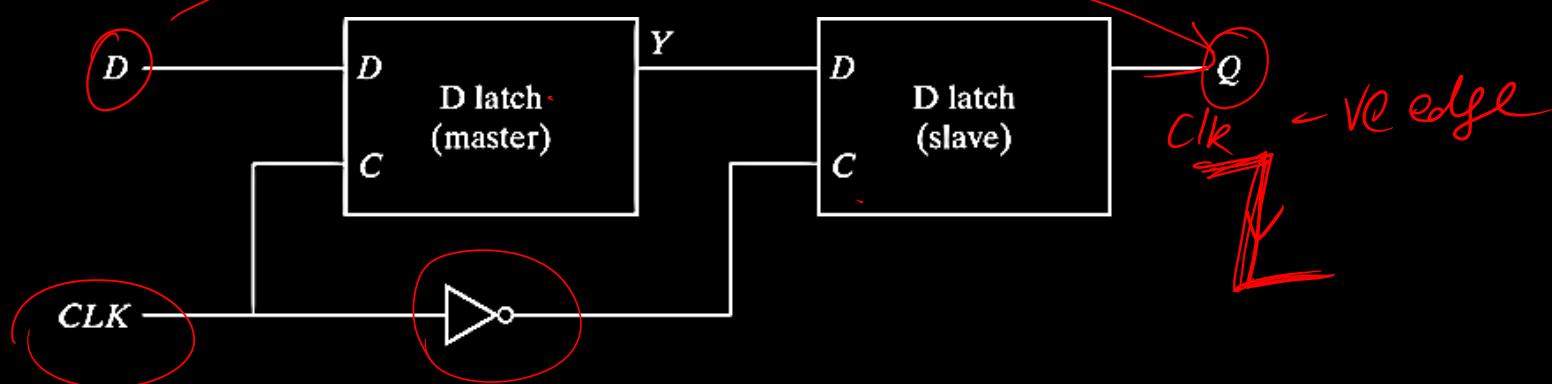




- Modifying Latches to Flipflops

- Two latches connected in a special configuration in such a way that output is isolated and prevented from being affected when the input to the flip flop is changing.

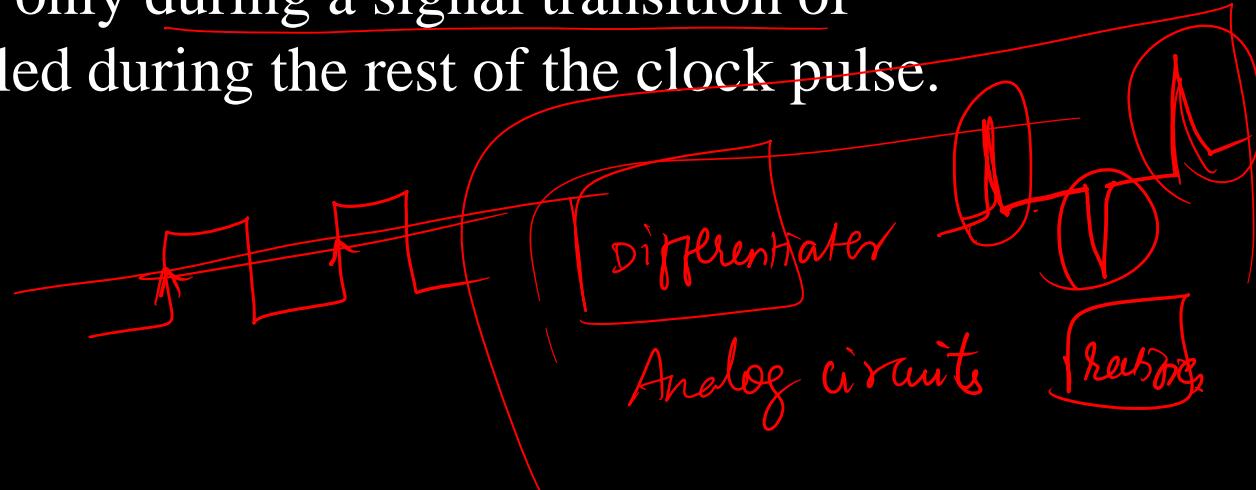
- Master-Slave Configuration



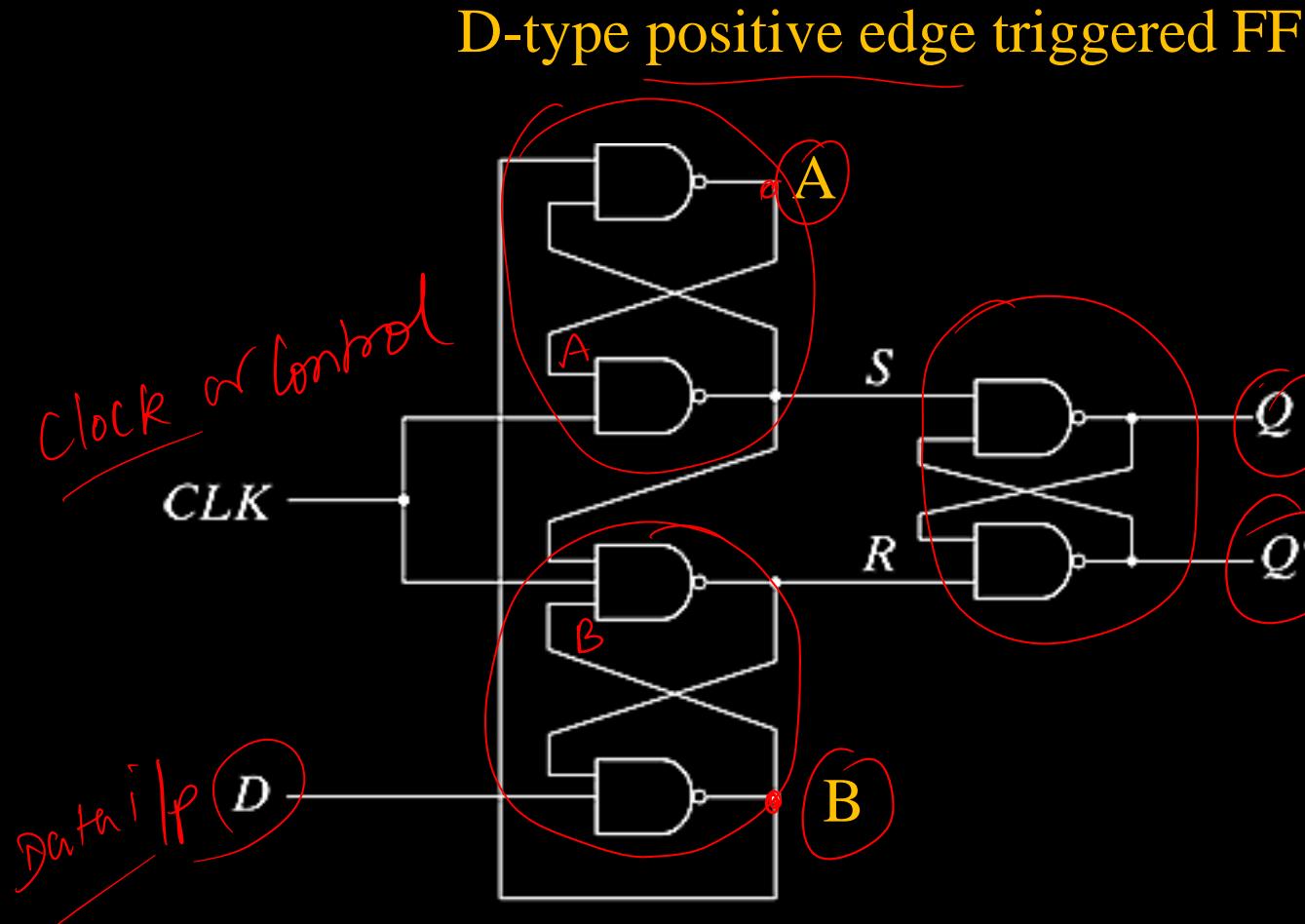
- Gate based design (Latches) which trigger only during a signal transition of synchronizing (Clock) signal and is disabled during the rest of the clock pulse.

*edge triggering*

A handwritten graph shows a square wave signal labeled "x<sup>ve</sup> sig". A red arrow points to the leading edge of the waveform, with the handwritten text "edge triggering" written above it.



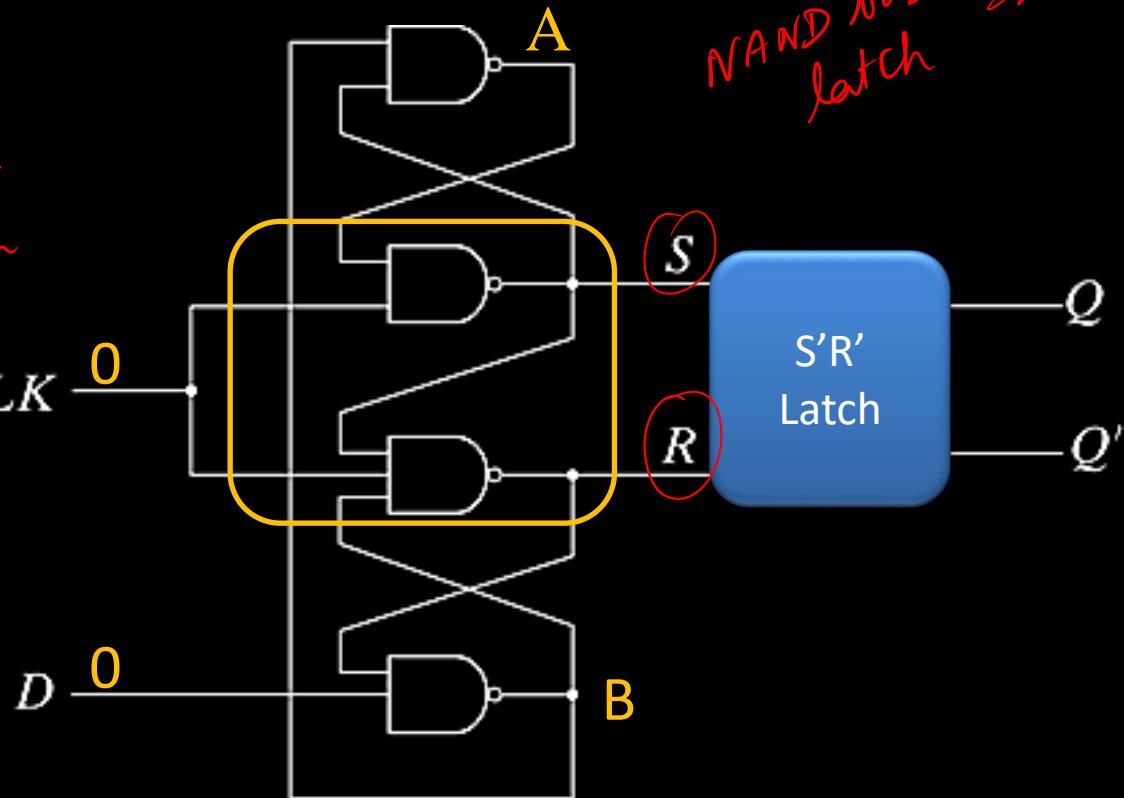
# Edge Triggered D Flip-Flops – Another Construction



Uses 3 latches out of which  
Two latches respond to  
external D (data) input  
and clock (clk) inputs  
Third latch provides the  
output of the flip-flop

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



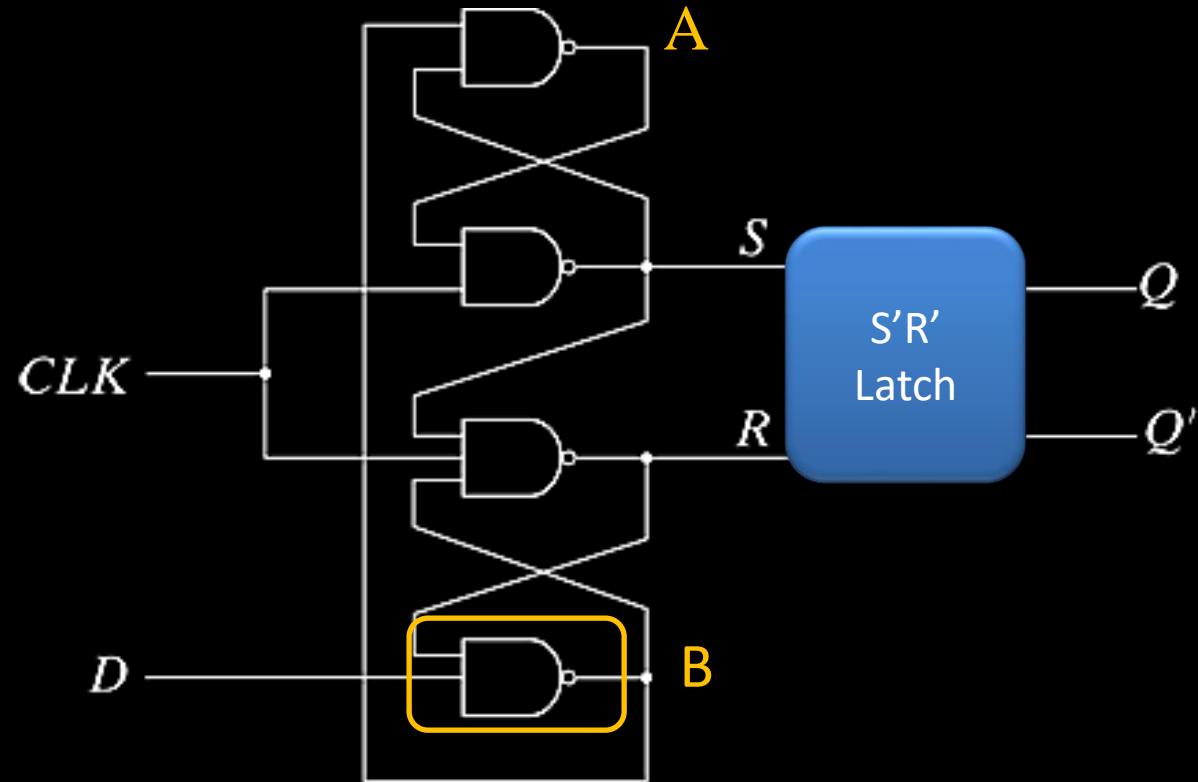
Case I:

D = 0, CLK = 0

CLK = 0 => S=1, R=1

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



Case I:

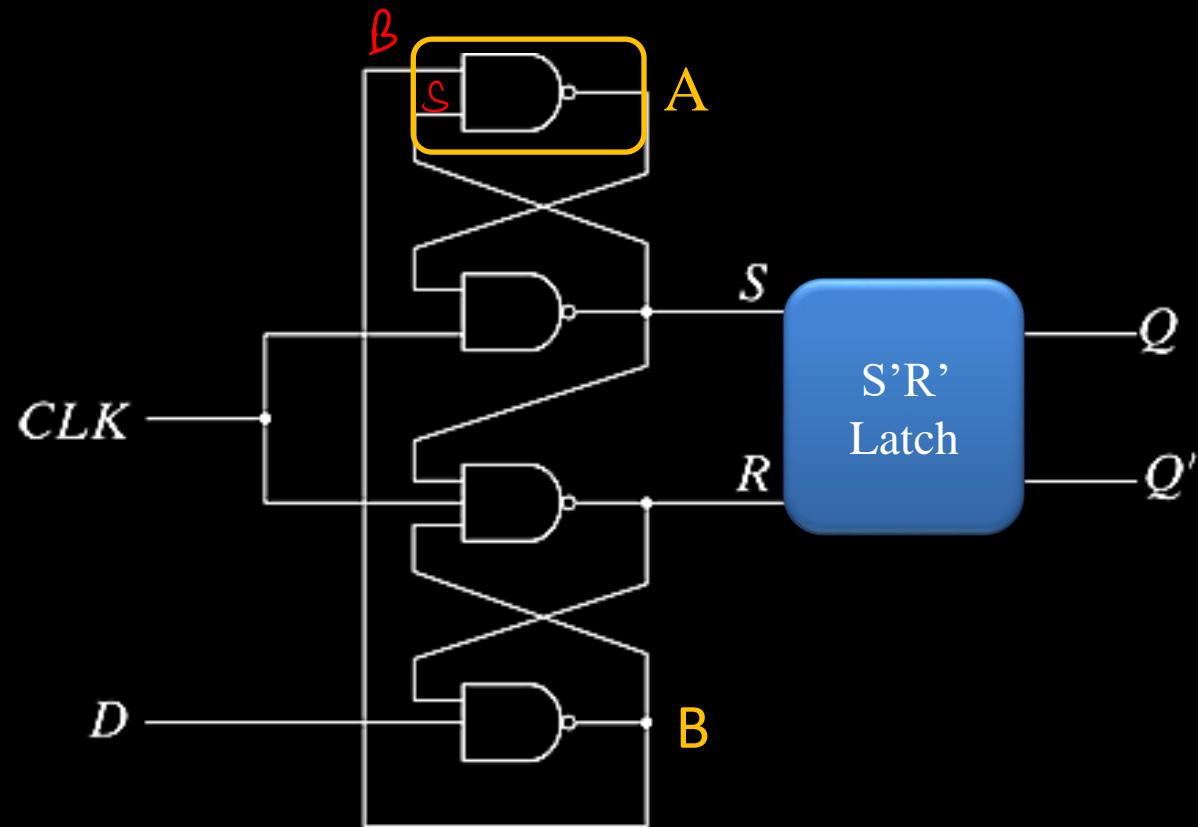
$$D = 0, CLK = 0$$

$$CLK = 0 \Rightarrow S=1, R=1$$

$$D = 0 \Rightarrow B=1$$

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



Case I:

$$D = 0, CLK = 0$$

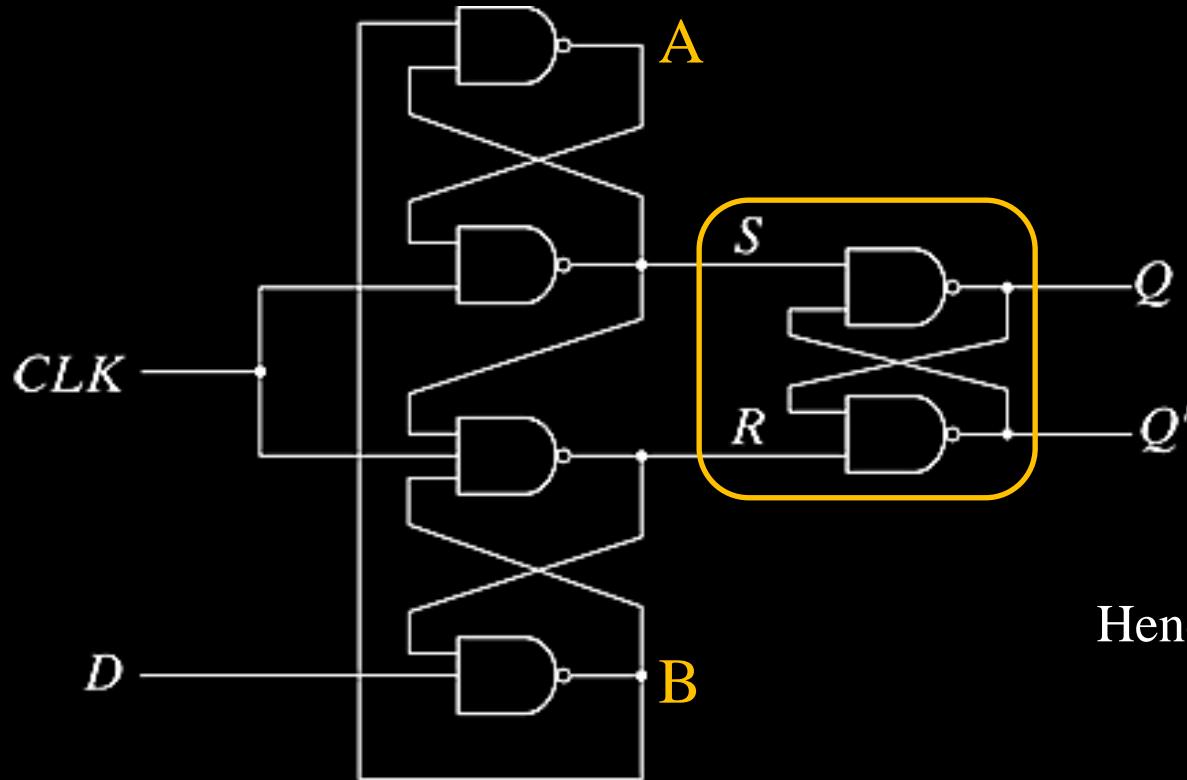
$$CLK = 0 \Rightarrow S=1, R=1$$

$$D = 0 \Rightarrow B=1$$

$$\underline{B=1}, \underline{S=1} \Rightarrow \underline{A=0}$$

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



Case I:

$D = 0, CLK = 0$

$CLK = 0 \Rightarrow S=1, R=1$

$D = 0 \Rightarrow B=1$

$B=1, S=1 \Rightarrow A=0$

$\underline{S = 1}, \underline{R = 1}$  for  $\underline{S'R'}$  latch

No change

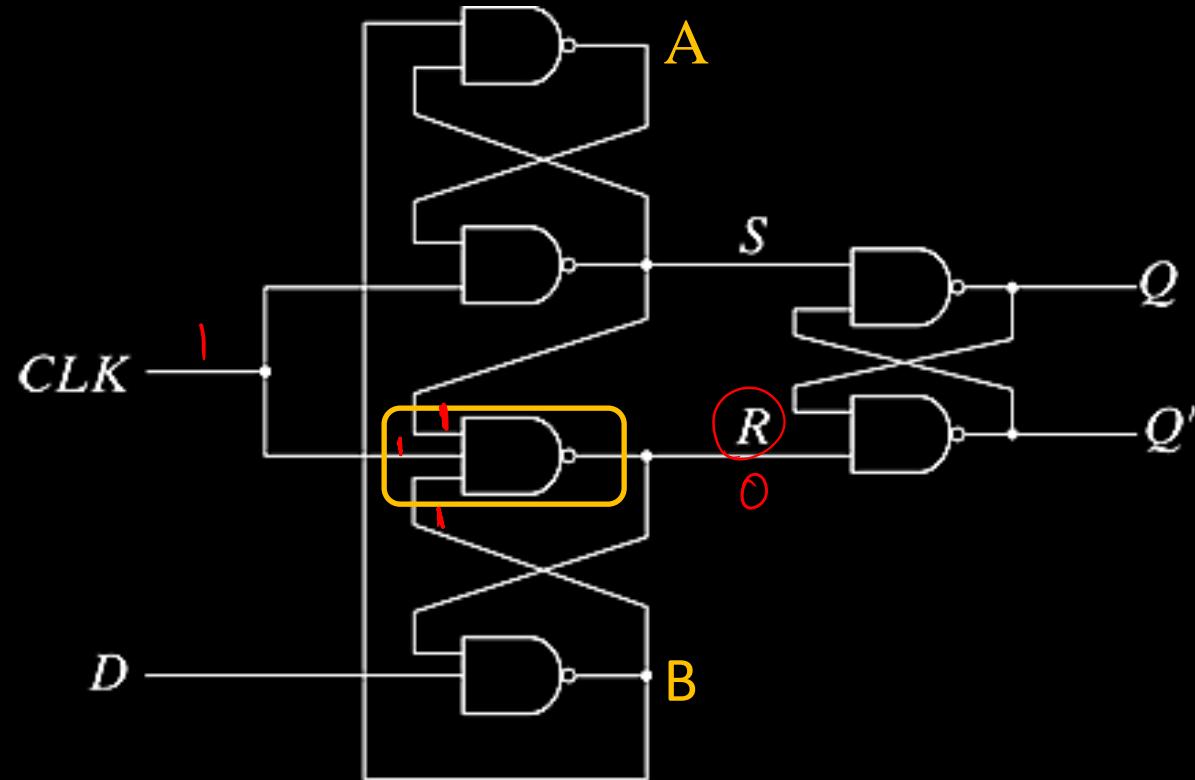
Hence Previous state is maintained

$Q \& \bar{Q}$  ↴

$A = 0, S = 1, R = 1, B = 1$

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



In case I: make CLK = 1

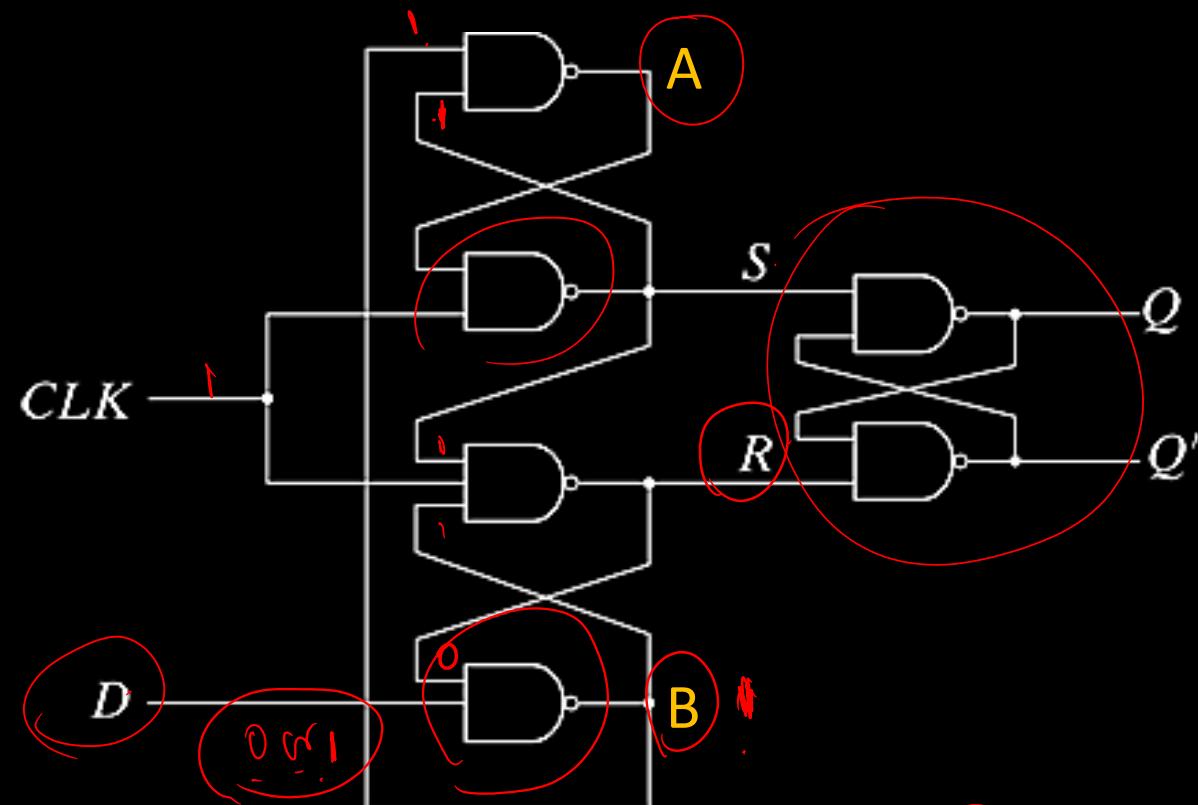
i.e D=0, A = 0, S = 1, R = 1, B = 1

When  $CLK = 1$ ,  $S = 1$ ,  $B = 1 \Rightarrow R = 0$

D=0

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



$A = 0, S = 1, R = 1, B = 1$  (D = 0, CLK = 1)

$CLK = 1, S = 1, B = 1 \Rightarrow R = 0$

$D = 0, R = 0 \Rightarrow (B = 1) \Rightarrow Q = 0$

$B = 1, S = 1 \Rightarrow A = 0$

Only R changes to 0  $\Rightarrow$  R Changes here after one gate delay

$A = 0, S = 1, R = 0, B = 1$

$S = 1, R = 0$  for  $S'R'$  latch

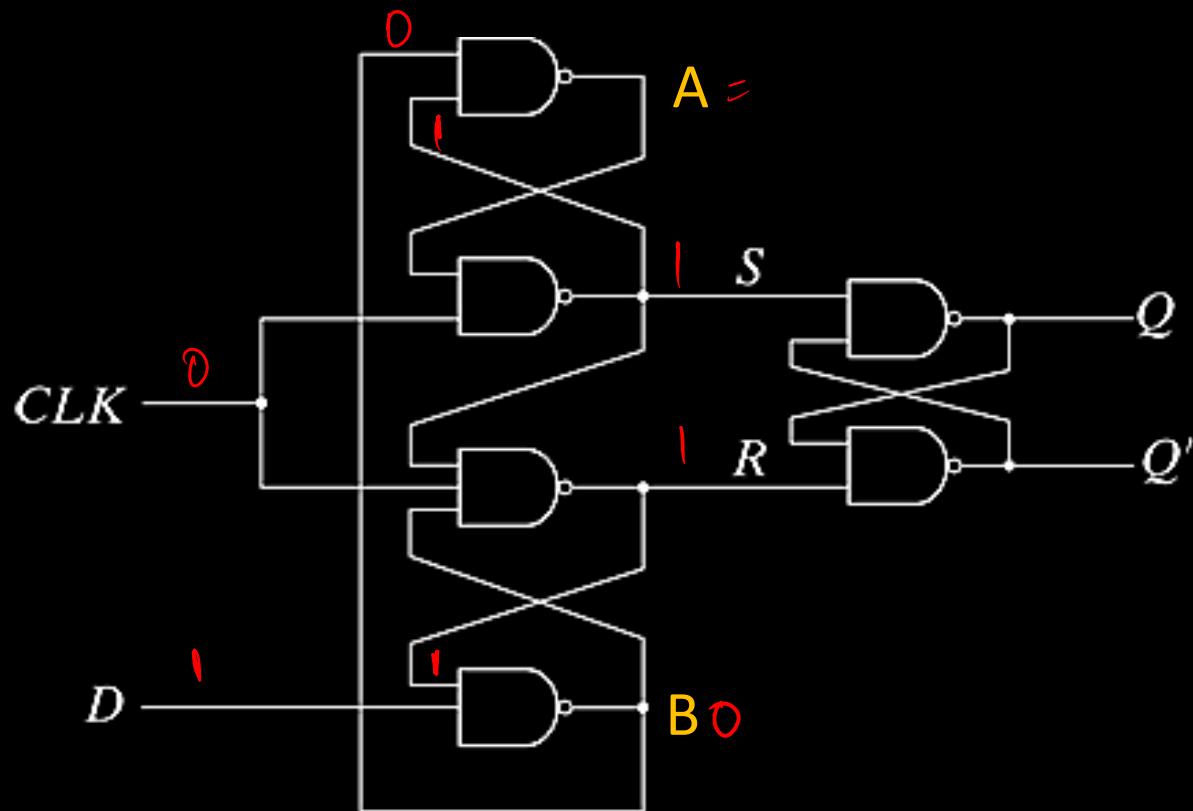
Hence  $Q = 0$  and  $Q' = 1$  Reset state

If D changes after R has stabilized there will be no effect on output.

This is Hold Time.

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF

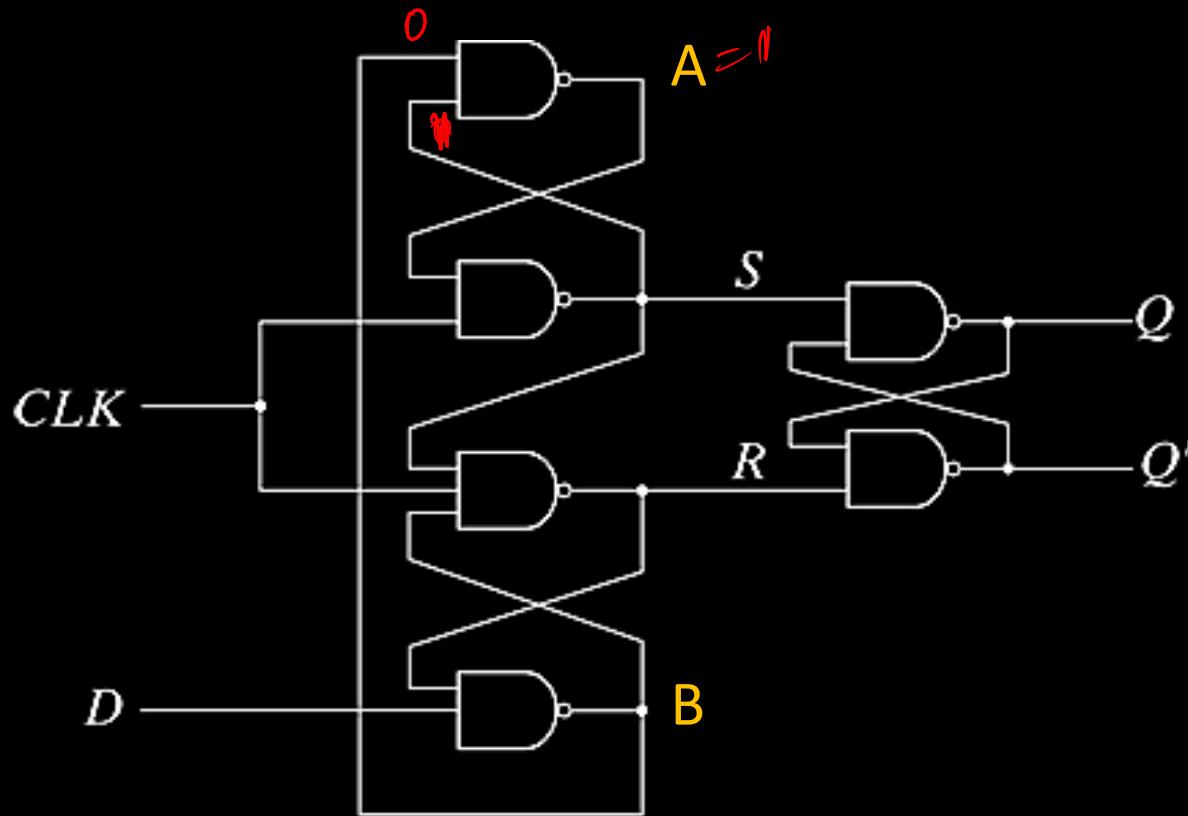


Case II:

- $D = 1$ ,  $CLK = 0$
- $CLK = 0 \Rightarrow R = 1$ ,  $S = 1$

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



Case II:

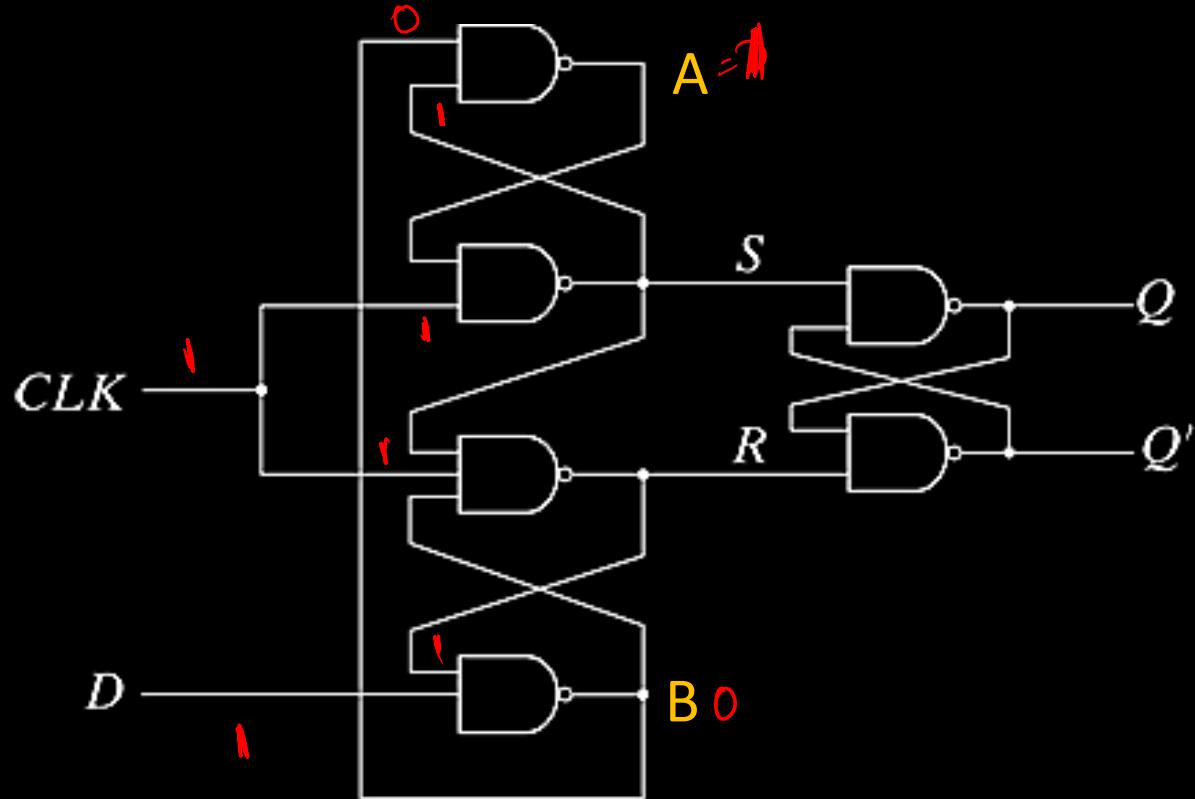
- $D = 1, CLK = 0$
- $CLK = 0 \Rightarrow R = 1, S = 1$
- $D = 1, R = 1 \Rightarrow B = 0$
- $B = 0, S = 1 \Rightarrow A = 1$
- $ASRB = 1110$
- $SR = 11 \Rightarrow \text{No change}$



# Edge Triggered D Flip-Flops

D = 1      Q = 1

D-type positive edge triggered FF



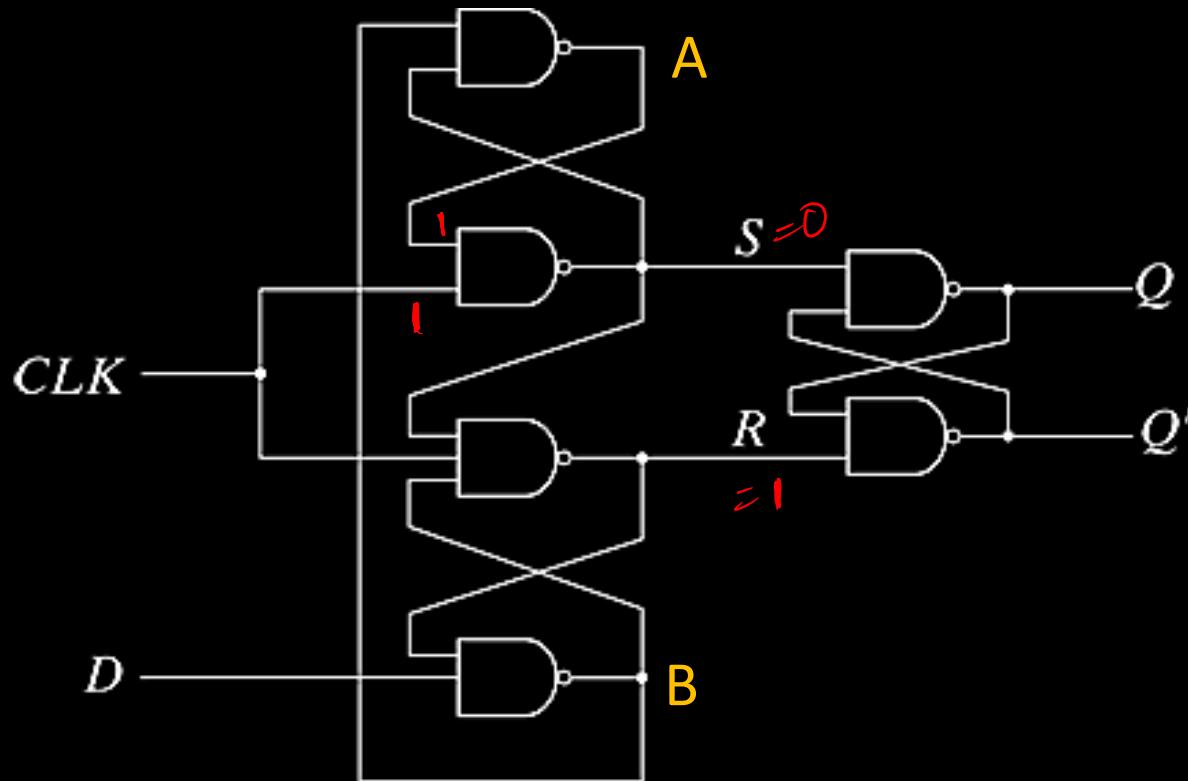
In case II make CLK = 1

- ASRB = 1110
- R=1, D=1 => B=0
- B=0, => A=1

## Edge Triggered D Flip-Flops

$$D=1 \quad Q=1$$

## D-type positive edge triggered FF



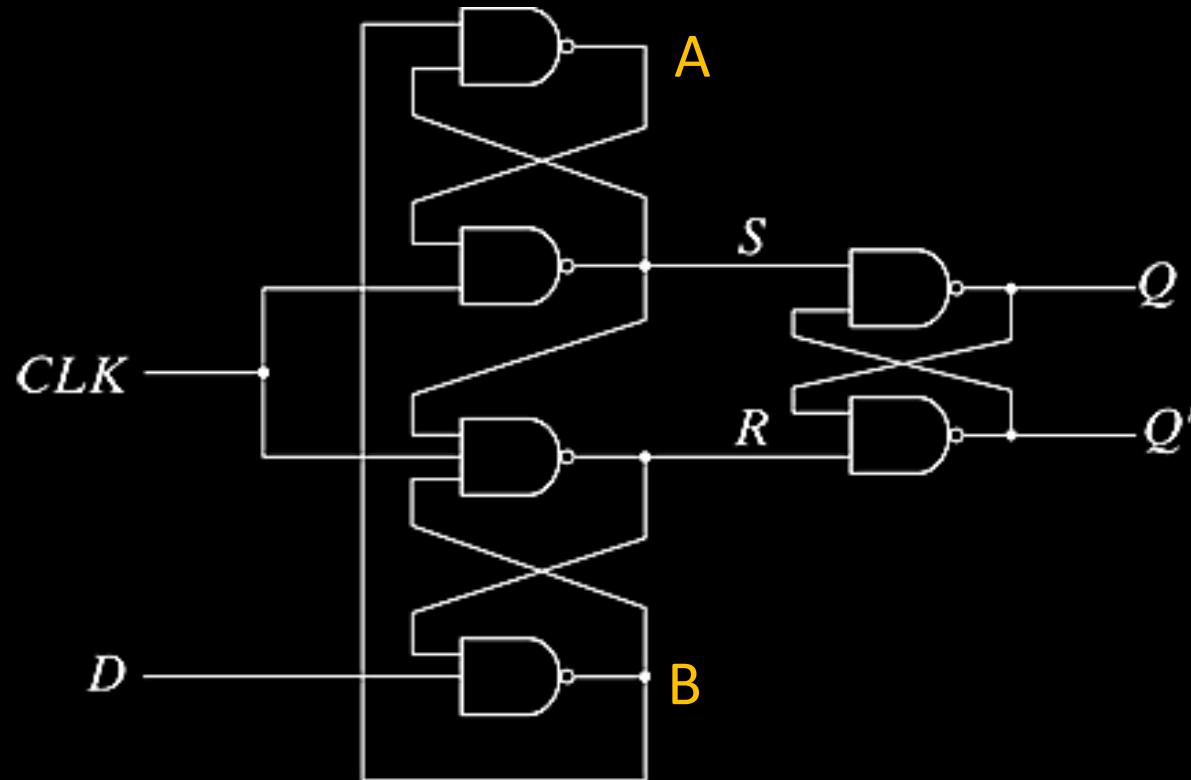
**In case II make CLK = 1**

- **ASRB = 1110**
  - **R=1, D=1 => B=0**
  - **B=0, => A=1**
  - **A=1, CLK=1 => S=0 => R=1**
  - **SR=01 => SET**  $\Rightarrow Q = 1$
  - **ASRB=1010**

# Edge Triggered D Flip-Flops

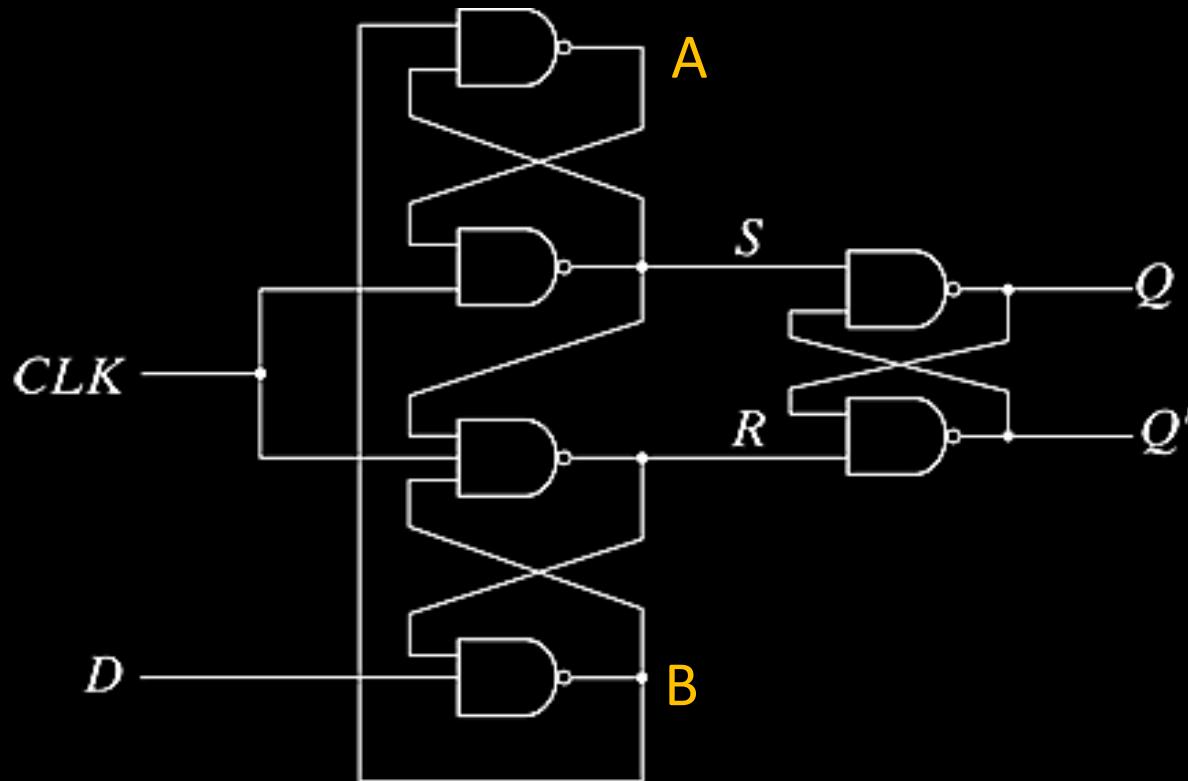
D-type positive edge triggered FF

Going from  $D=1$ ,  $CLK=0$  to  
 $D=0$ ,  $CLK = 1$



# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



Going from  $D=1$ ,  $CLK=0$  to  $D=0$ ,  $CLK = 1$

**ASRB=1110**

- $D=0 \Rightarrow B=1$
- $CLK=1, A=1 \Rightarrow S=0$

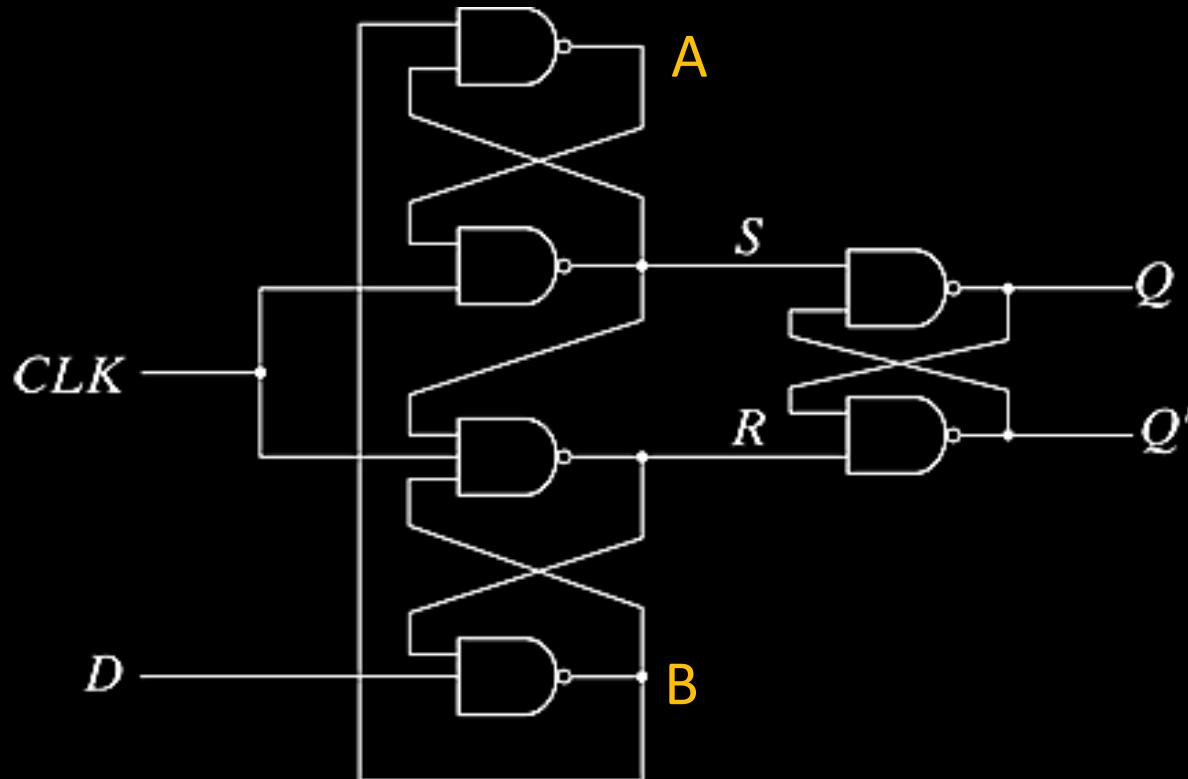
# Edge Triggered D Flip-Flops

Going from  $D=1$ ,  $CLK=0$  to  $D=0$ ,  $CLK = 1$

D-type positive edge triggered FF

ASRB=1110

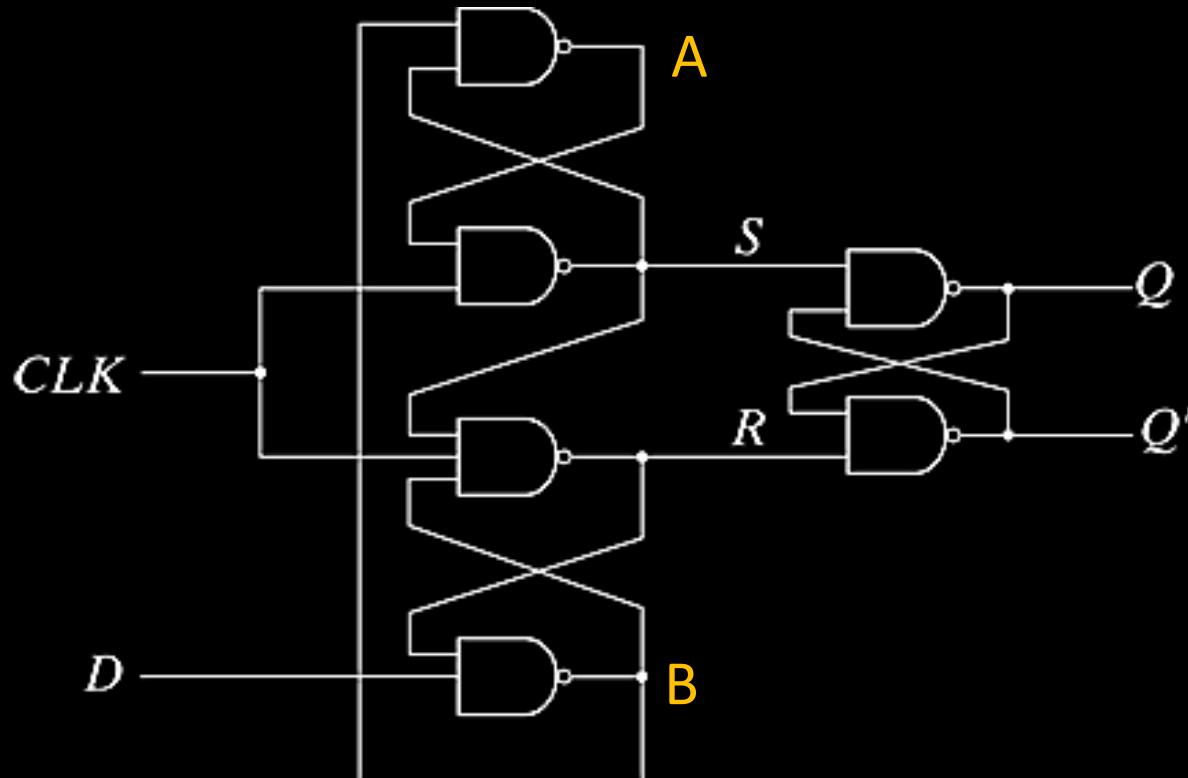
- $D=0 \Rightarrow B=1$
- $CLK=1, A=1 \Rightarrow S=0$
- $S=0, B=1 \Rightarrow A=1$
- $S=0, B=1, CLK=1 \Rightarrow R=1$



# Edge Triggered D Flip-Flops

Going from  $D=1$ ,  $CLK=0$  to  $D=0$ ,  $CLK = 1$

D-type positive edge triggered FF



$ASRB = \underline{\underline{1110}}$

- $D = 0 \Rightarrow B = 1$
- $CLK = 1, A = 1 \Rightarrow S = 0$
- $S = 0, B = 1 \Rightarrow A = 1$
- $S = 0, B = 1, CLK = 1 \Rightarrow R = 1$
- $SR = 01 \Rightarrow \underline{\underline{SET}}$
- $\underline{\underline{ASRB = 1011}}$

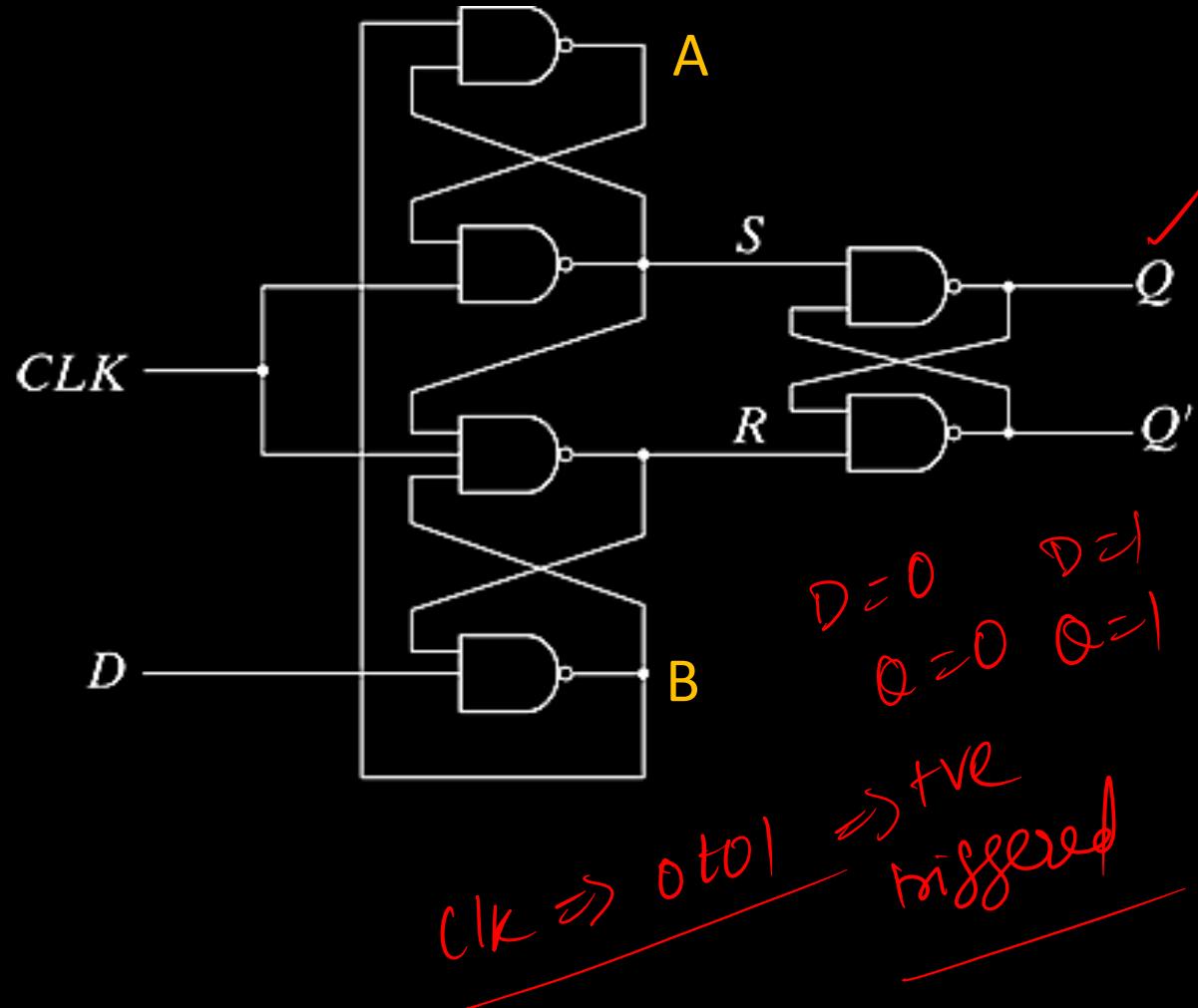
Wrong result

Correct sequence will be:

1. Go from case II to case I, i.e., let ASRB change from 1110 to 0111. Here A takes two gate delays to change.

# Edge Triggered D Flip-Flops

D-type positive edge triggered FF



Going from  $D=1, CLK=0$  to  $D=0, CLK = 1$

ASRB=1110

- $D=0 \Rightarrow B=1$
- $CLK=1, A=1 \Rightarrow S=0$
- $S=0, B=1 \Rightarrow A=1$
- $S=0, B=1, CLK=1 \Rightarrow R=1$
- $SR=01 \Rightarrow SET$
- $ASRB=1011$

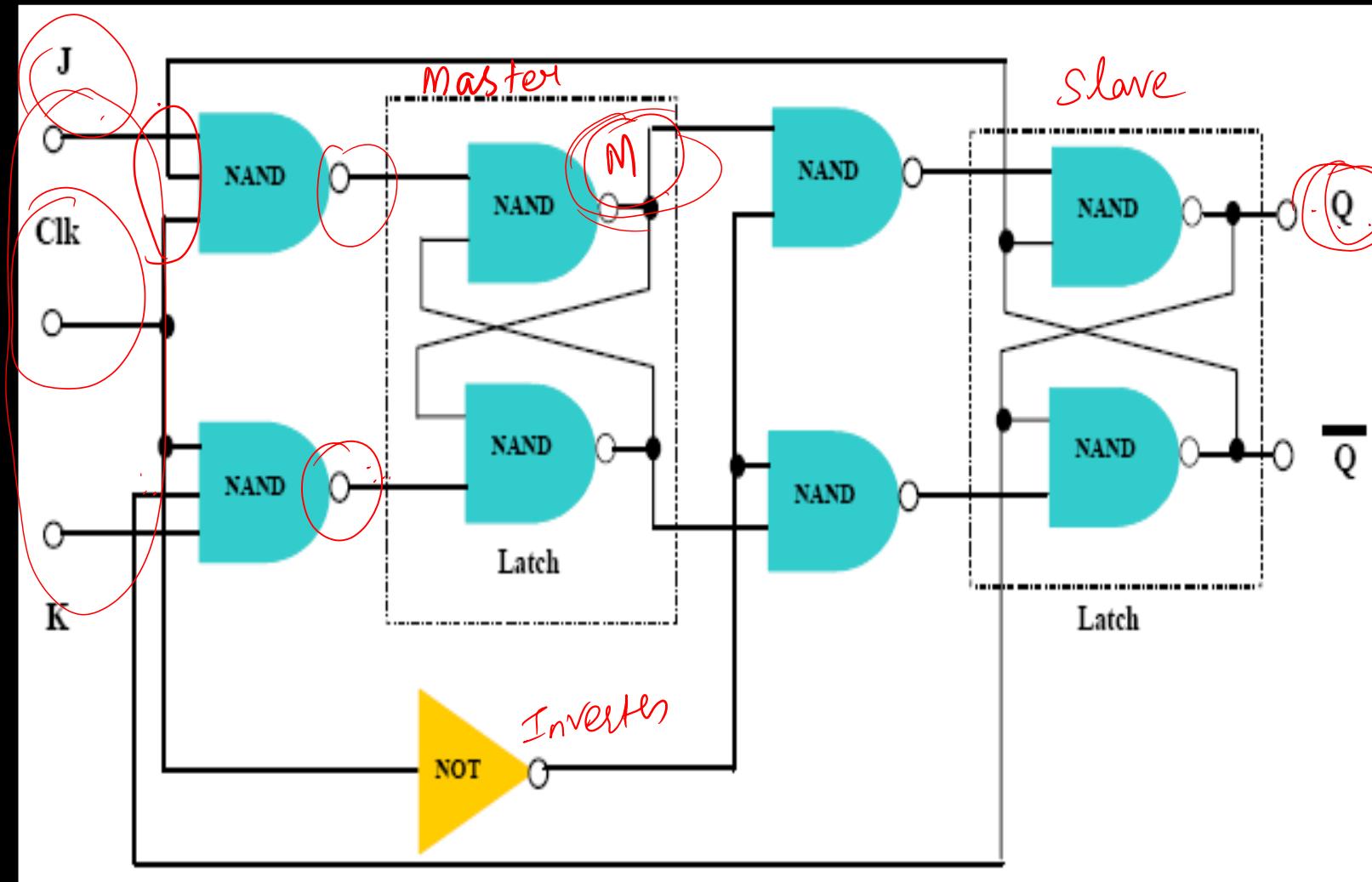
Wrong result

Correct sequence will be:

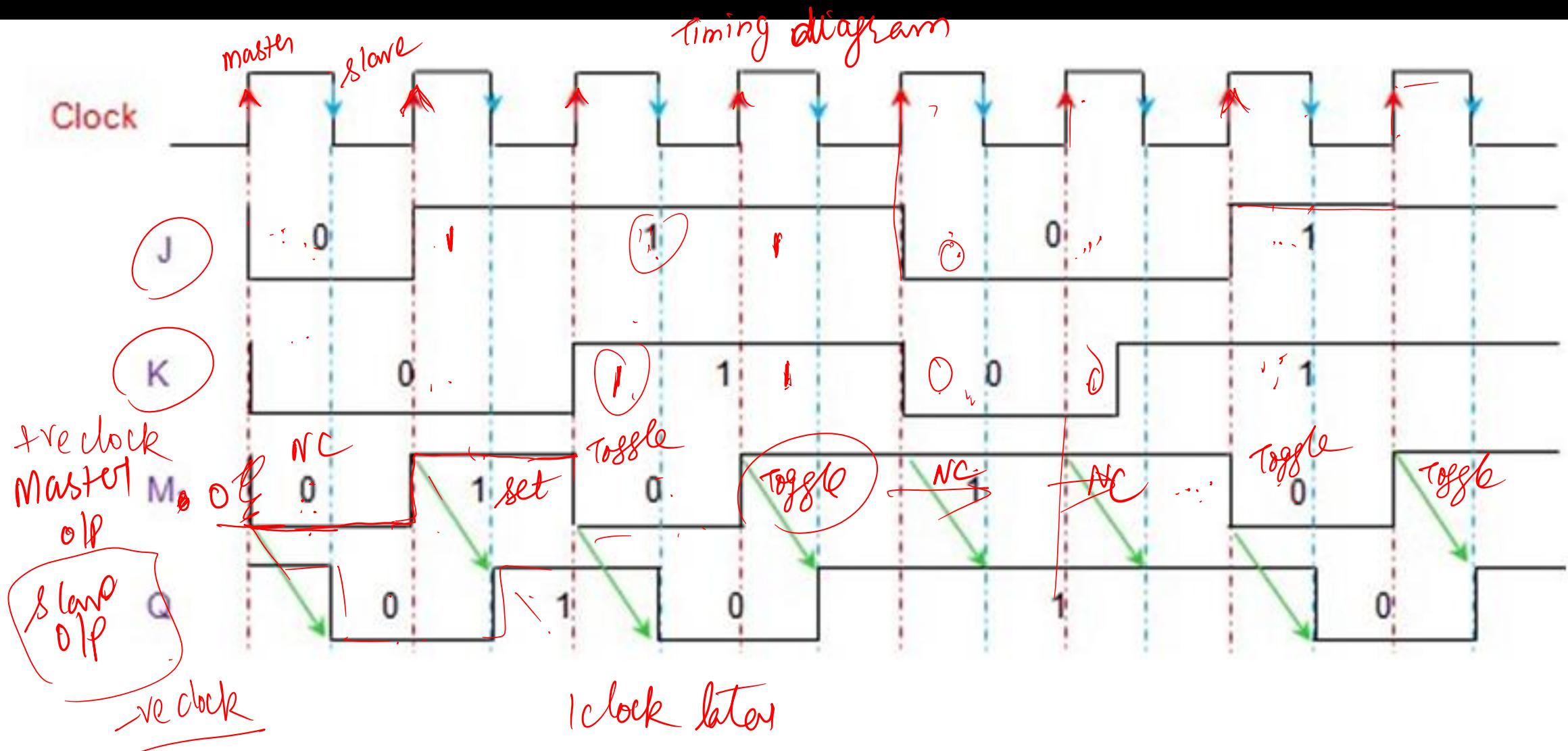
1. Go from case II to case I, i.e., let ASRB change from 1110 to 0111. Here A takes two gate delays to change. This is Setup time.
2. Now go from **case I** to  $D=0, CLK=1$ .

# JK Master - Slave FF

T<sub>WO</sub>



J & K is [PS are]



# Edge Triggered JK Flip-Flops

Symbol +ve edge triggered

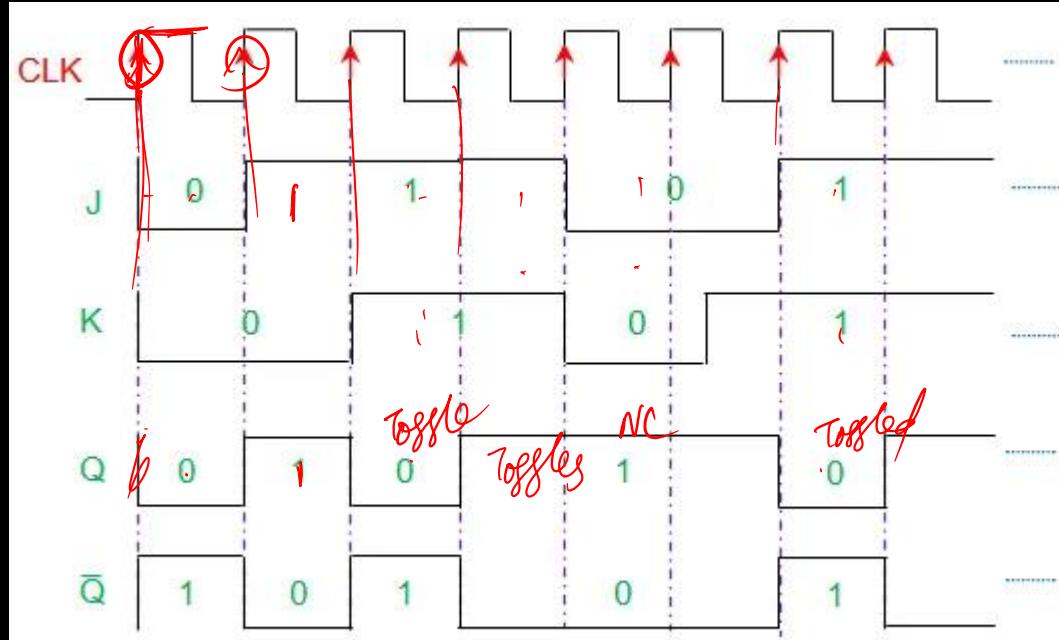
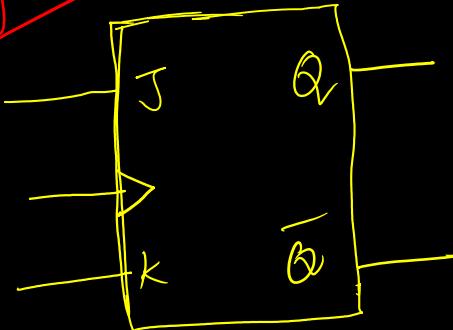
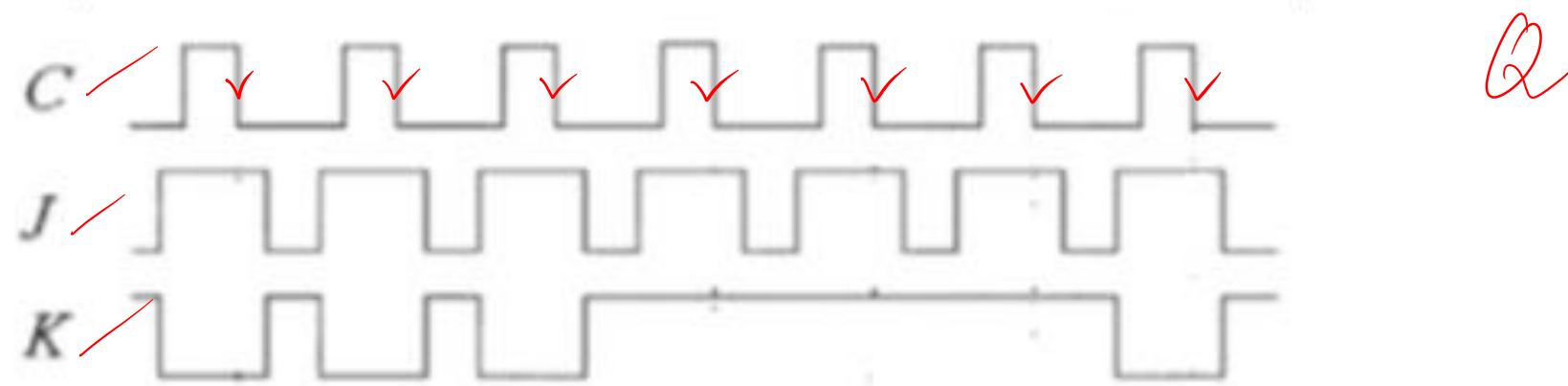


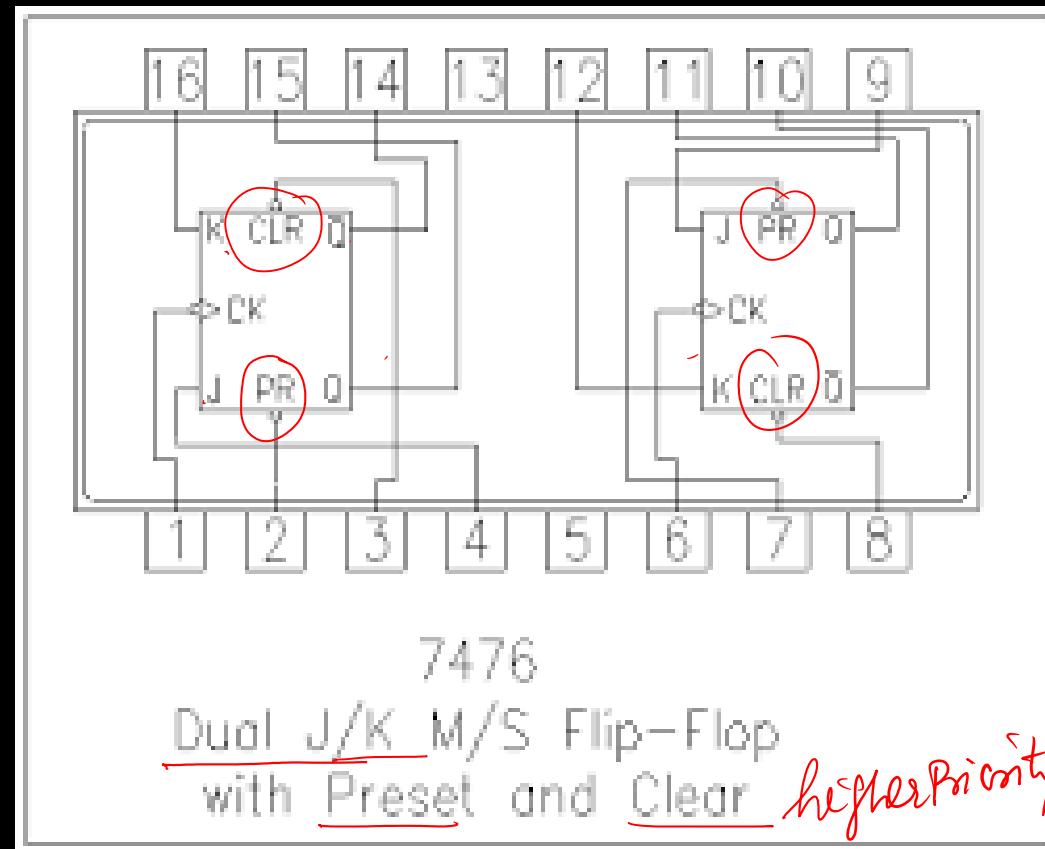
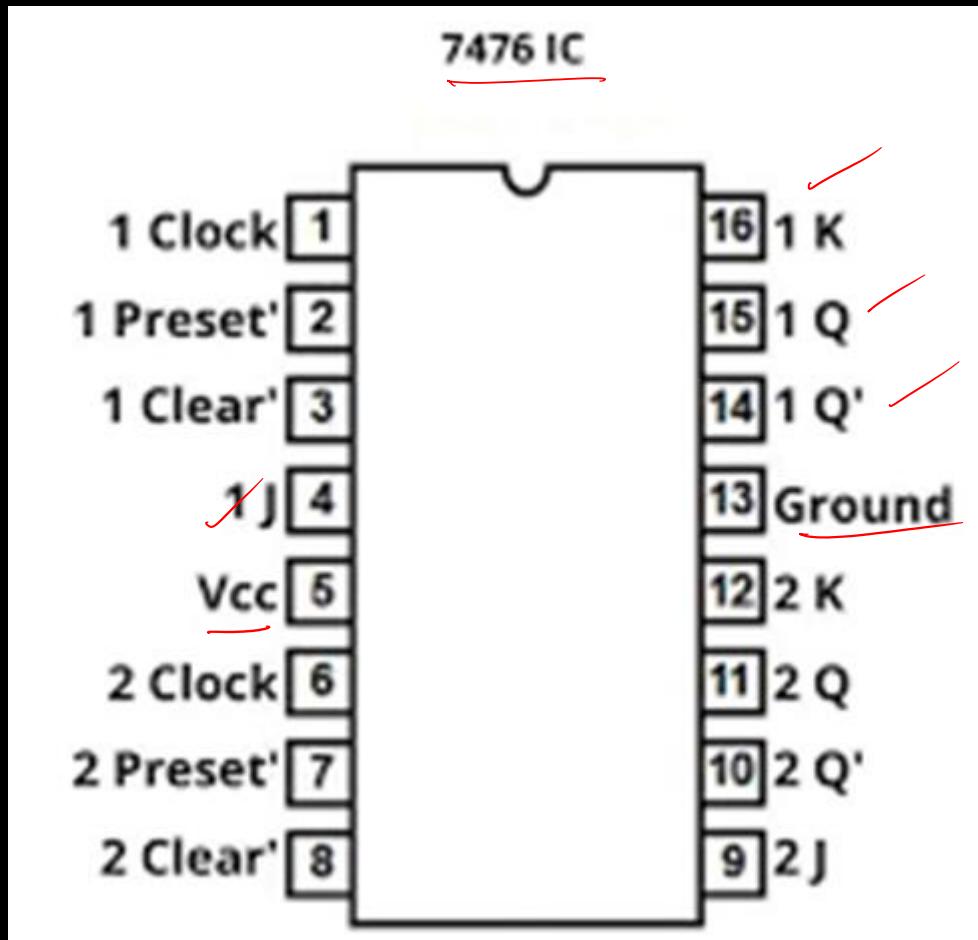
Figure 3 Timing diagram for positive edge-triggered JK flip-flop



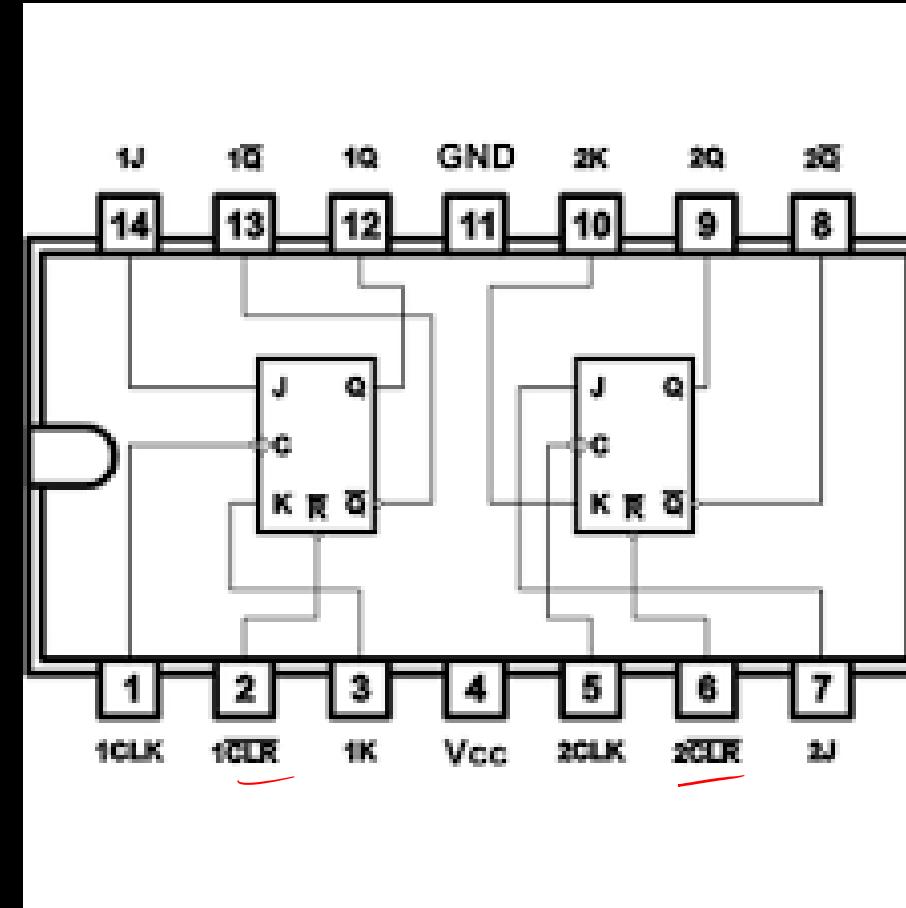
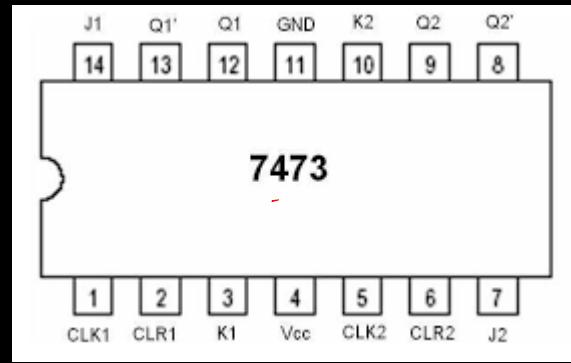
-ve edge triggered

For a negative edge-triggered J-K flip-flop with the inputs shown below, develop the Q output waveform relative to the clock. Assume that Q is initially LOW.



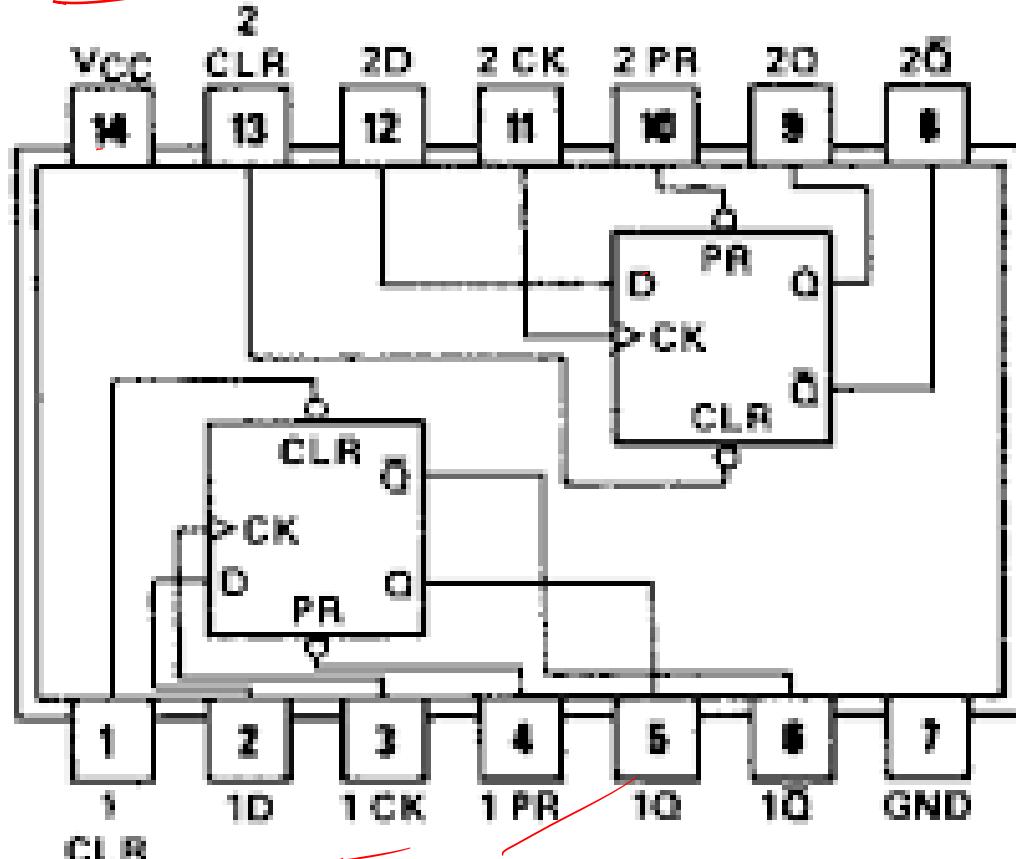


CLR               $Q = 0$   
Preset             $Q = 1$



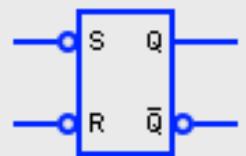
7474

D ff IC

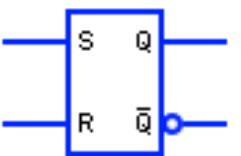


# Flip-Flop Symbols

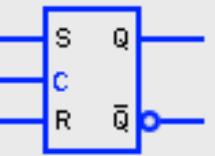
www.electricaltechnology.org



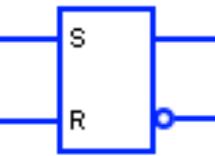
Active Low NAND  
SR Flipflop



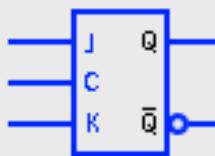
Active High NAND  
SR Flipflop



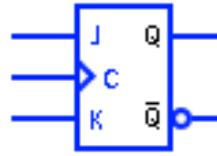
Clocked NAND  
SR Flipflop



SR Flip-Flop  
Set Reset



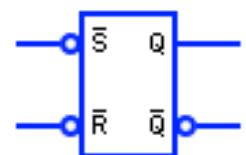
JK Flipflop High  
level Triggered



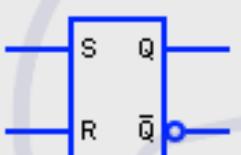
JK Flipflop Rising  
Edge Triggered



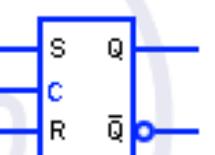
JK Flip-Flop  
Activated by  
the falling edge



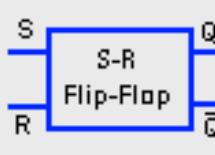
Active Low NOR  
SR Flipflop



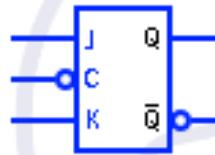
Active High NOR  
SR Flipflop



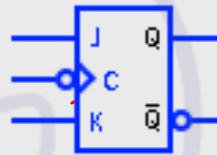
Clocked NOR  
SR Flipflop



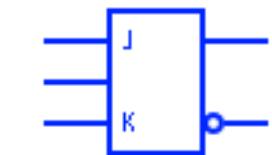
SR Flip-Flop  
Set Reset



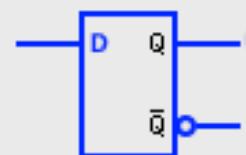
JK Flipflop Low  
level Triggered



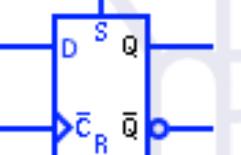
JK Flipflop Falling  
Edge Triggered



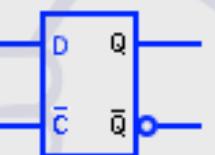
JK Flip-Flop  
Generic symbol



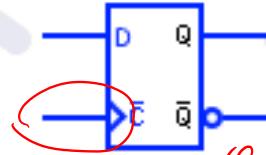
D Flipflop



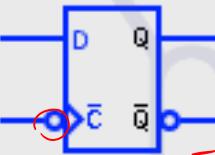
D Flipflop  
With Preset-Clear



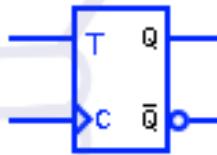
Gated  
D Flipflop



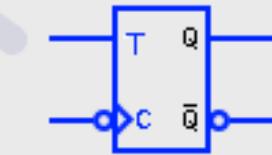
D Flipflop Rising  
Edge Triggered



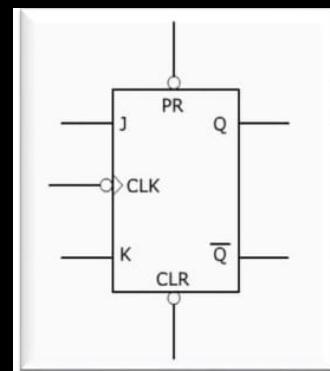
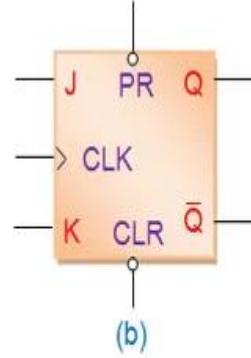
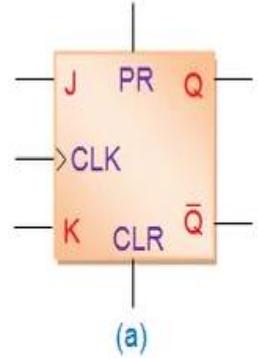
D Flipflop Falling  
Edge Triggered



T Flipflop Rising  
Edge Triggered



T Flipflop Falling  
Edge Triggered



Clear and preset  
inputs have highest  
priority

JK flip-flop with (a) active high preset and clear pins (b) active low preset and clear pins

Has 5 inputs named:  
J(set), K(reset), PR, CLR, and CLK

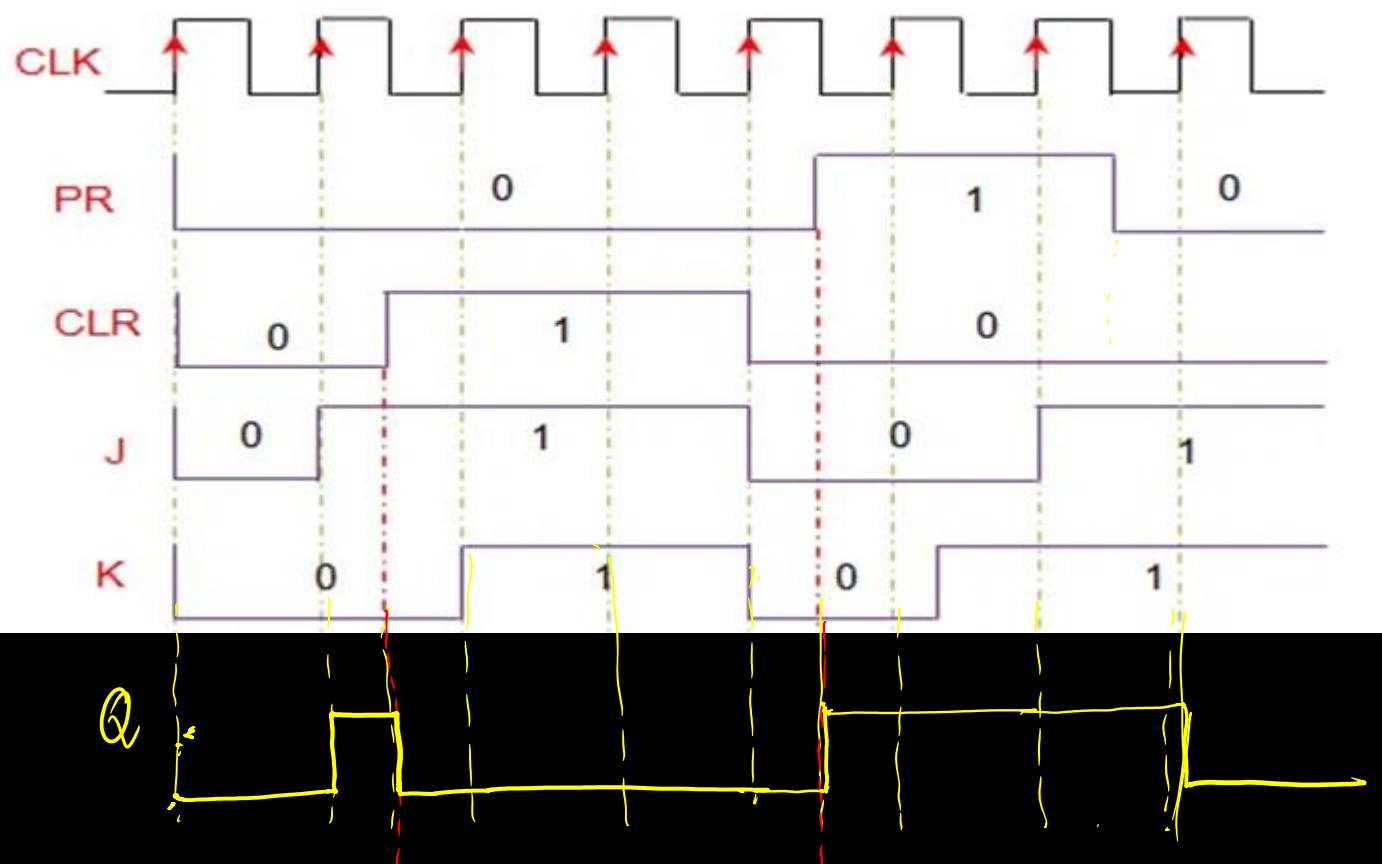
Has 2 outputs: Q and Q'

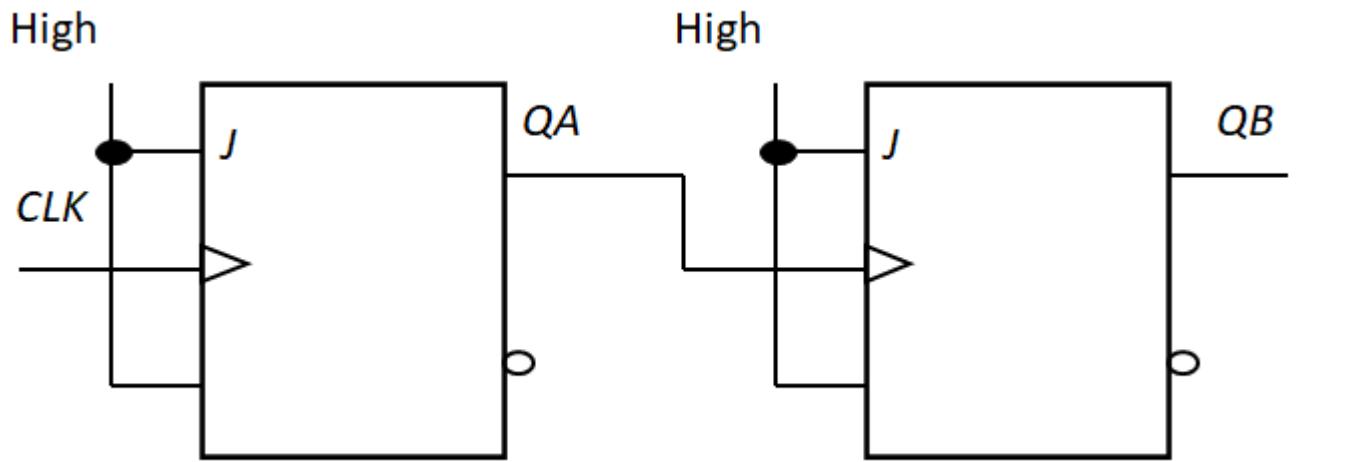
PR = Preset  $\rightarrow Q = 1$   
 CLR = Clear  $\rightarrow Q = 0$   
 CLK = Clock

**Set:** when it stores a binary 1

**Cleared (reset):** when it stores a binary 0

The PR and CLR inputs always  
**override** the J,K inputs.





$Q_A$  &  $Q_B$  = ?  
Inference



CS/ECE/EEE/INSTR F215:Digital Design

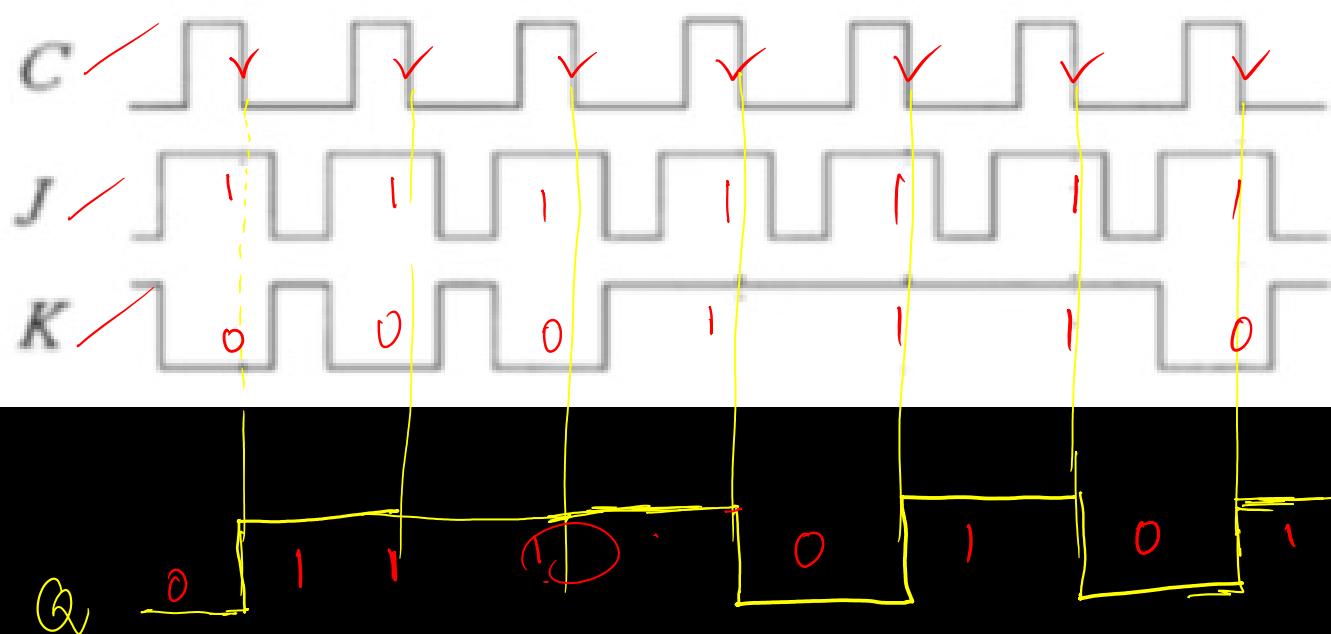
Lecture 24: *Analysis of clocked sequential circuits*  
*Sat, 30 Oct 2021*

Dr. R. N. Ponnalagu, EEE

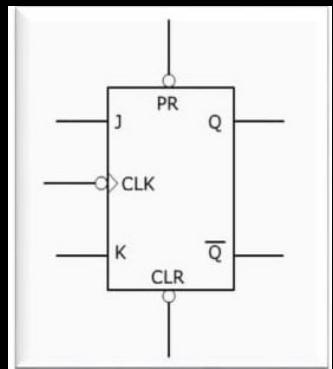
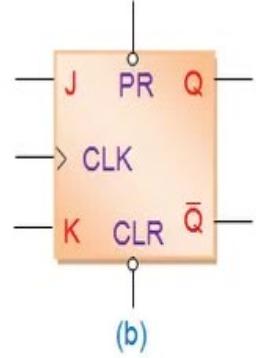
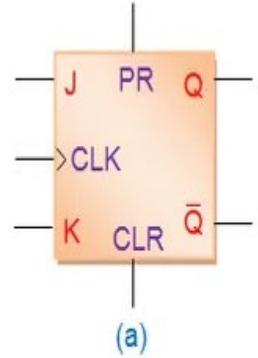
*Strength and Growth  
come only through  
continuous effort  
and struggle*

*- Napoleon Hill*

For a negative edge-triggered J-K flip-flop with the inputs shown below, develop the Q output waveform relative to the clock. Assume that Q is initially LOW.



$J$	$K$	$Q$	$NC$
0	0	0	0
0	1	1	
1	0	1	
1	1	1	
0	1	0	
0	0	0	
0	0	0	Toggle



$Q = 0$  and Preset  $\Rightarrow Q = 1$   
 Clear pins have highest priority

JK flip-flop with (a) active high preset and clear pins (b) active low preset and clear pins

Has 5 inputs named:  
 J(set), K(reset), PR, CLR, and CLK

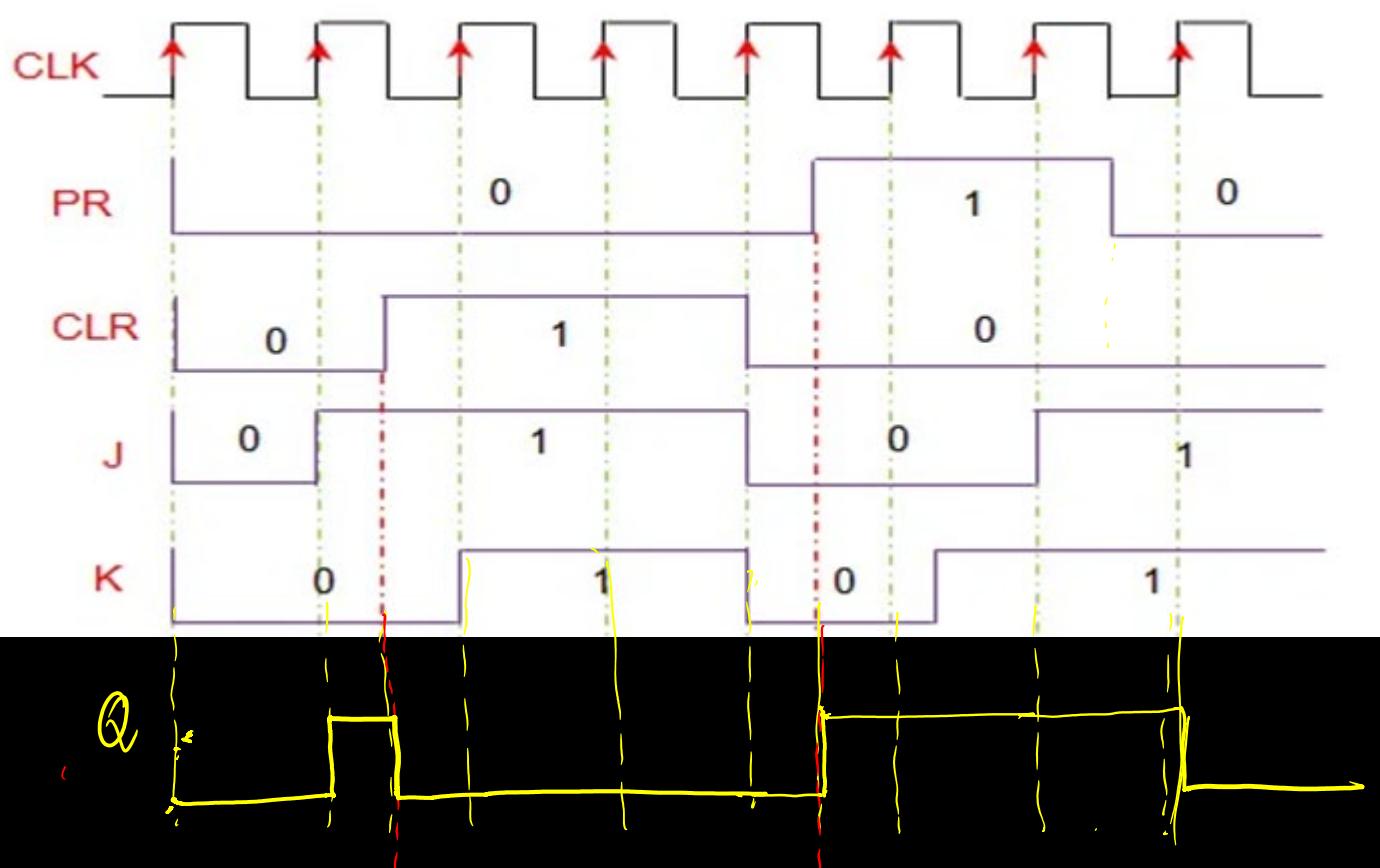
Has 2 outputs: Q and Q'

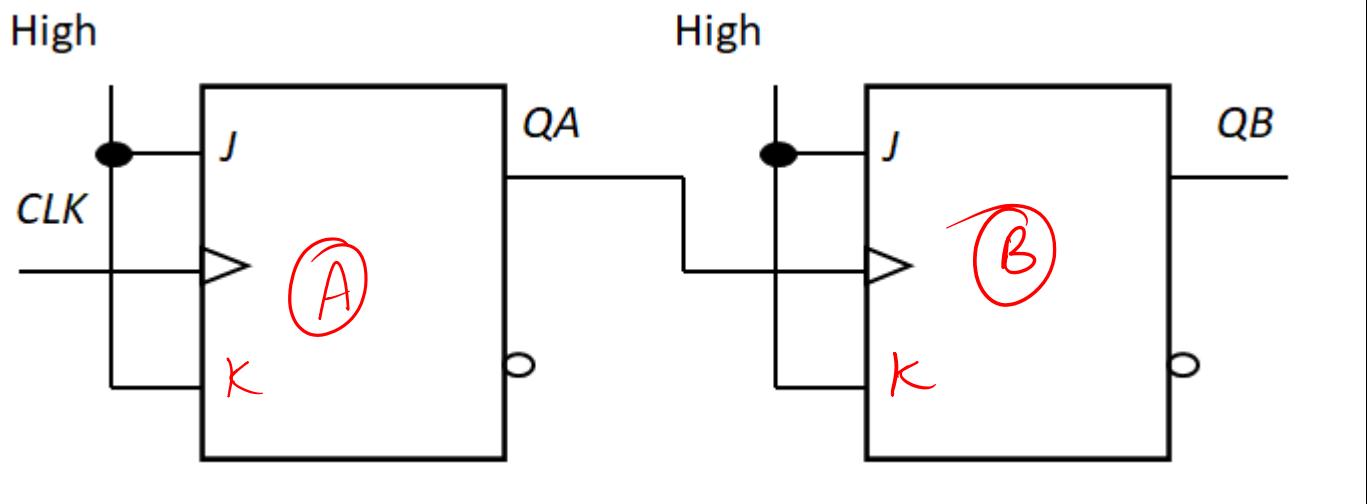
PR = Preset  $\rightarrow Q = 1$   
 CLR = Clear  $\rightarrow Q = 0$   
 CLK = Clock

**Set:** when it stores a binary 1

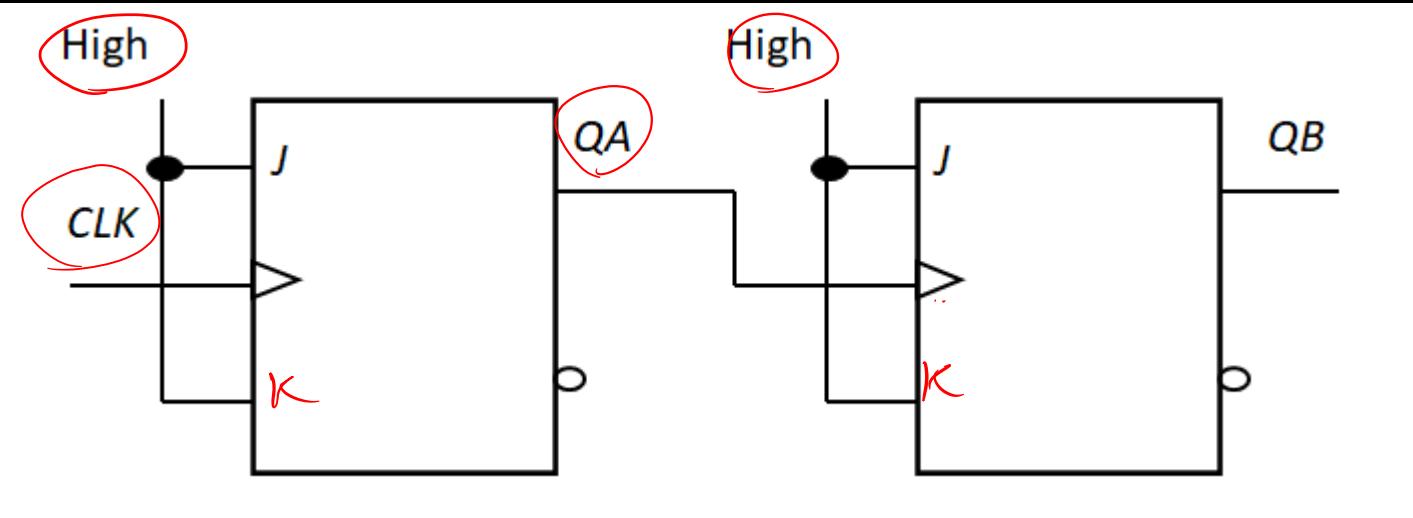
**Cleared (reset):** when it stores a binary 0

The PR and CLR inputs always  
 override the J,K inputs.

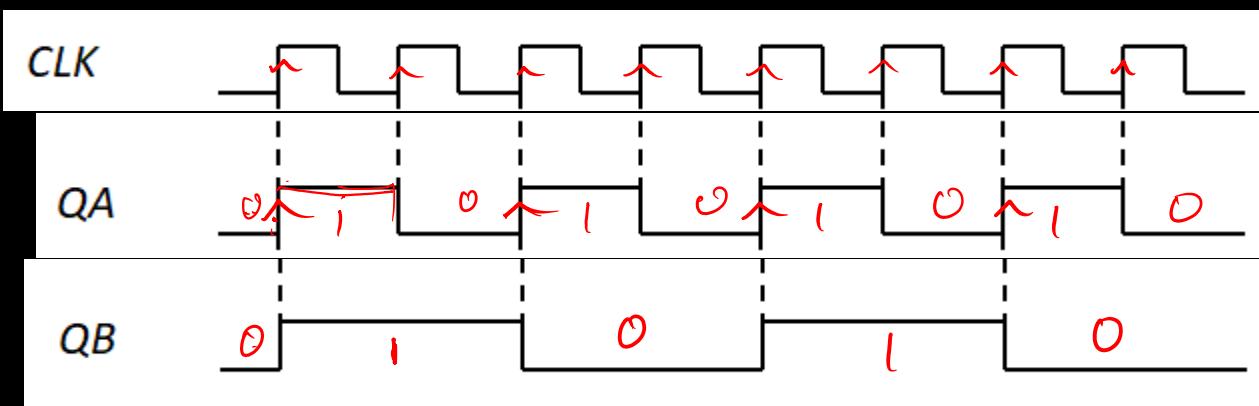


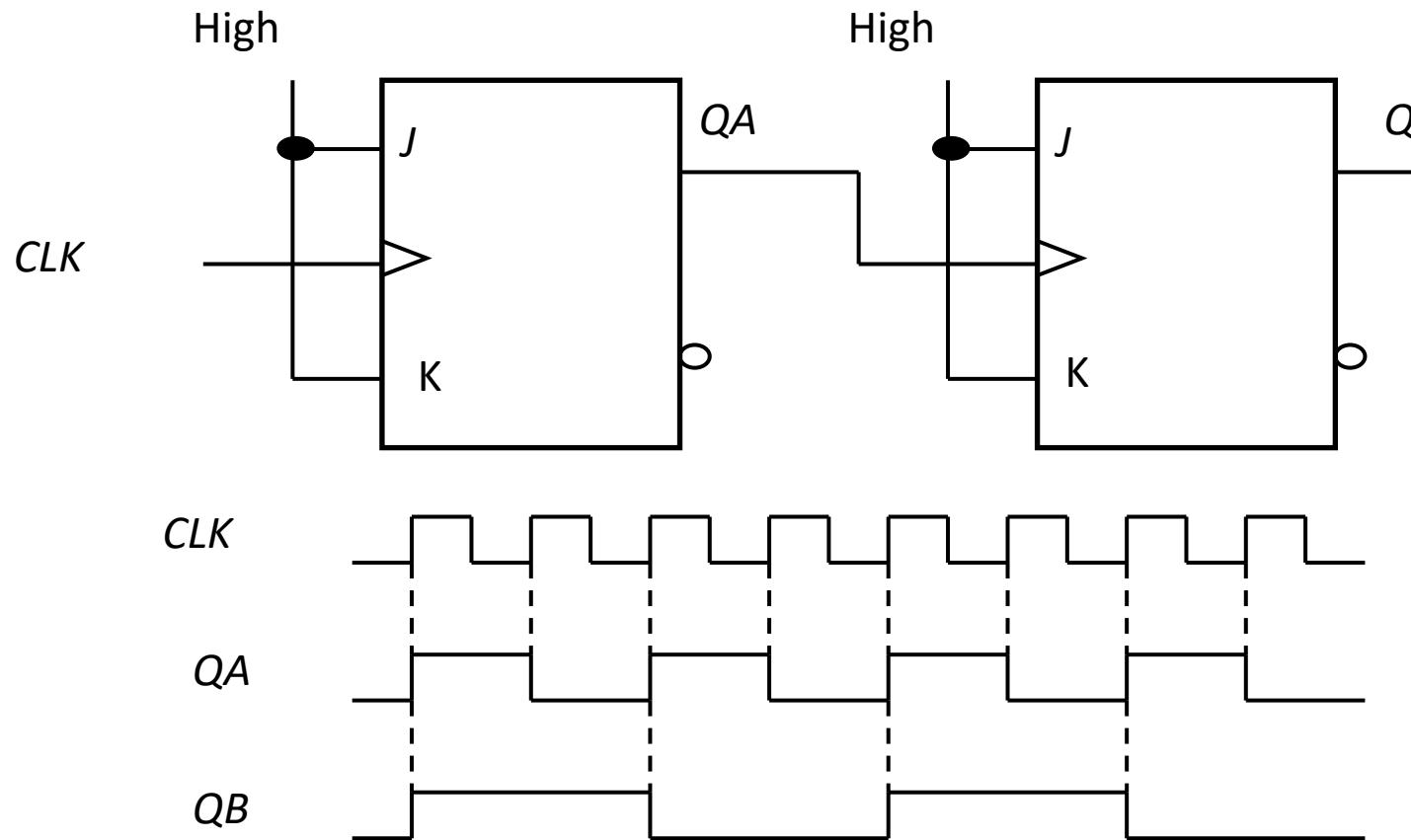


$Q_A$  &  $Q_B = ?$   
Inference



$Q_A$  &  $Q_B$  = ?  
Inference





Applications of  $\frac{f_c}{2}$

Frequency division

$f_c$

$f_c/2$

$f_c/4$

## Characteristic equations

- We can also write **characteristic equations**, where the next state  $Q(t+1)$  is defined in terms of the current state  $Q(t)$  and the flip-flop inputs.

*Char. Table*

D	$Q(t+1)$	Operation
0	0	Reset
1	1	Set

*D flip flop*

J	K	$Q(t+1)$	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

*JK flip flop*

T	$Q(t+1)$	Operation
0	$Q(t)$	No change
1	$Q'(t)$	Complement

*T flip flop*

*Characteristic equations*

$$Q(t+1) = D$$

$$\bar{K}Q + \bar{J}\bar{Q}$$

$$Q(t+1) = K'Q(t) + JQ'(t)$$

$$\bar{T}Q + T\bar{Q} = T \oplus Q$$

$$Q(t+1) = T'Q(t) + TQ'(t)$$

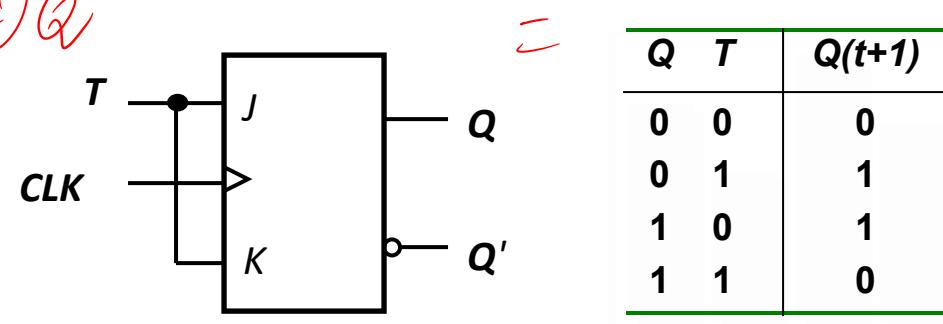
$$= T \oplus Q(t)$$

*For SR Flip flop*

S	R	$Q(t+1)$
0	0	$Q(t)$ $\Rightarrow$ no change
0	1	0 $\Rightarrow$ reset
1	0	1 $\Rightarrow$ set
1	1	undefined

$$Q(t+1) = S + \bar{R}Q(t)$$

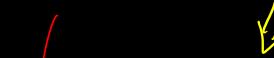
provided  $SR = 0$



---

# ANALYSIS OF SEQUENTIAL CIRCUITS

Clocked sequential circuit :

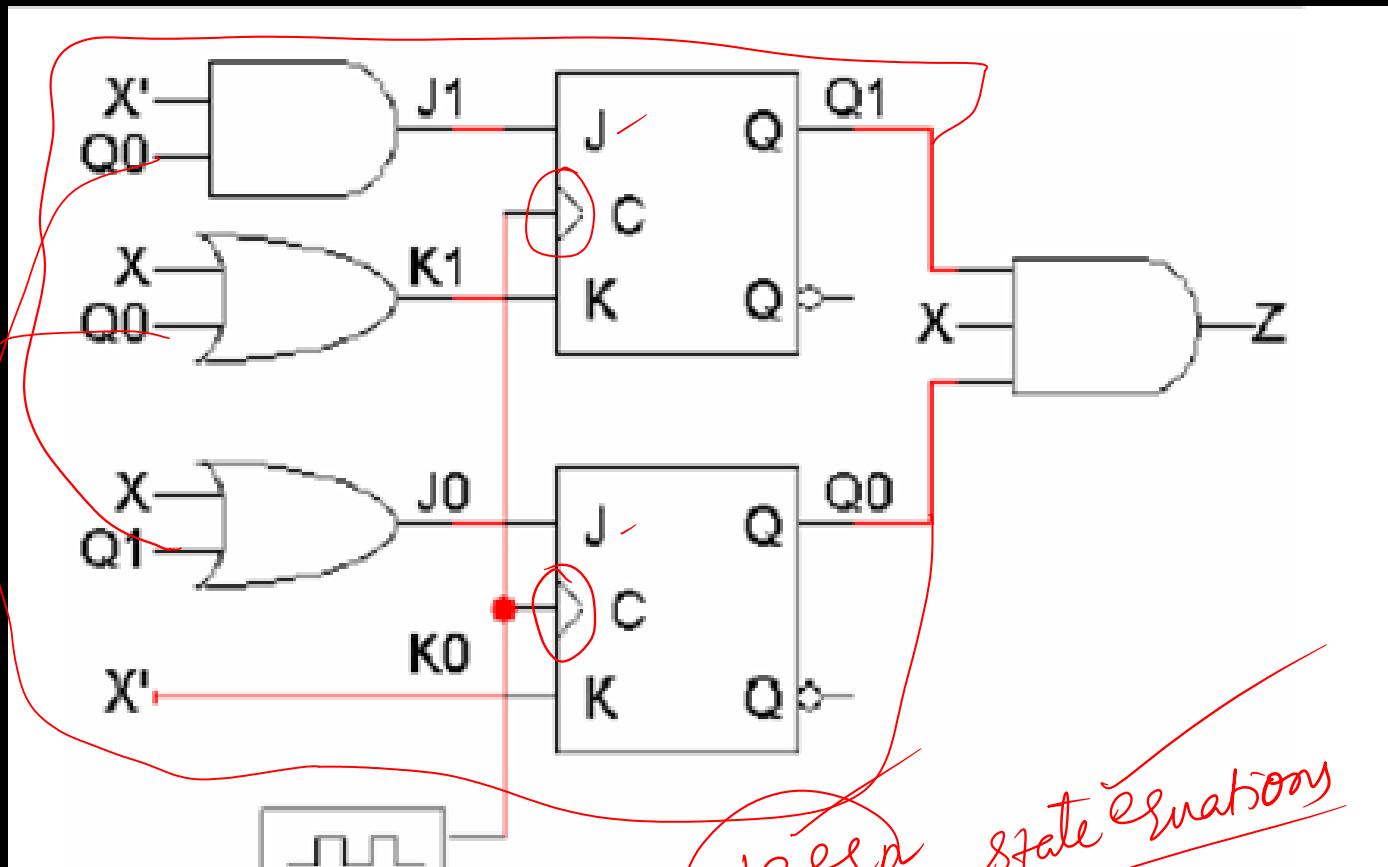


Clocked FFs used as  
memory element

A logic diagram consisting  
of  $FF(s)$  with clock inputs

Analysis: \* obtaining a table or diagram for the time  
sequence of i/p's, o/p's & internal states  
\* o/p's and Next state of FFs are both a function  
of the i/p's & Present state.

logic dgm



➤ Sequential Circuit with  
2 JK FFs; +ve edge triggered  
input X and output Z

➤ Flip flop inputs J1, K1 and  
J0, K0, outputs Q1 and Q0

➤ State table:  
Inputs, Flip flop states  
(current and next), outputs

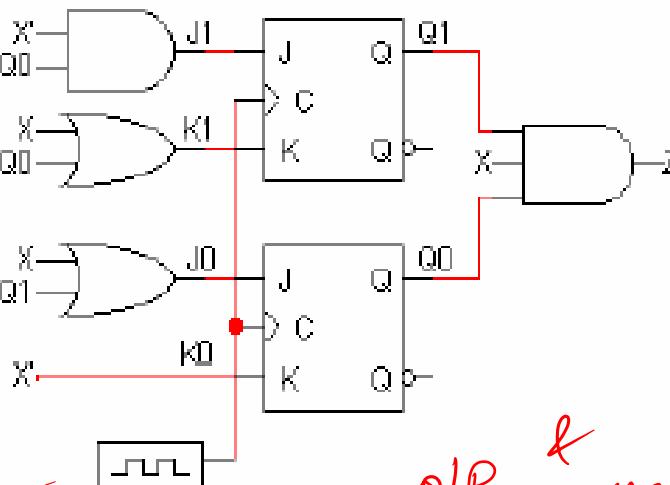
P.S $Q_1\ Q_0$	I.P.S $X$	N.S $Q_1\ Q_0$	O.P.S $Z$
-------------------	--------------	-------------------	--------------

# Making a State Table

# Analyzing our example circuit

- A state table for our example circuit is shown below.
- The present state  $Q_1Q_0$  and the input  $X$  will determine the next state  $Q_1Q_0$  and the output  $Z$ .

$$3 \Rightarrow 2^3 \Rightarrow 8$$



State table

Present State		Input X	Next State		Output Z
$Q_1$	$Q_0$		$Q_1$	$Q_0$	
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

all possible combinations of ZP & P present state

Assume binary values

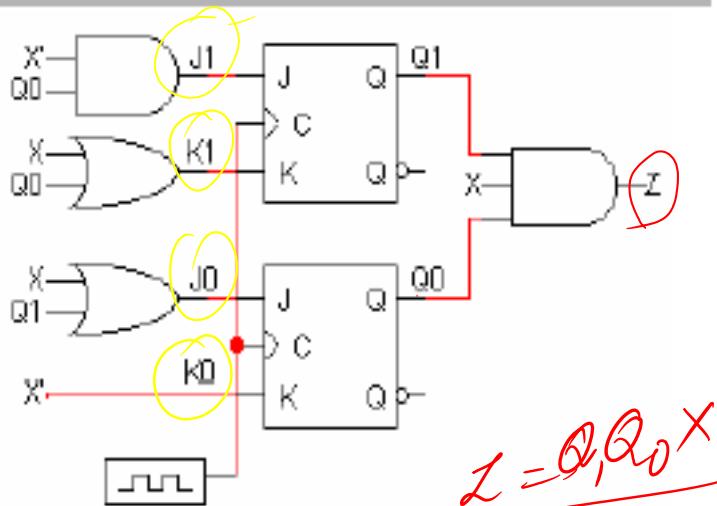
O/P & Next state depends  
ZP & Present state

# The outputs are easy

- From the diagram, you can see that

$$Z = Q_1 Q_0 X \text{ Boolean eqn}$$

- This is an example of a Mealy machine, where the output depends on both the present state ( $Q_1 Q_0$ ) and the input ( $X$ ).



Present State		Inputs	Next State		Outputs
$Q_1$	$Q_0$	$X$	$Q_1$	$Q_0$	$Z$
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

Mealy  $\Rightarrow$  O/P depends on both present state & I/P  
 Moore Present state  
 $Z = Q_1 Q_0 X \rightarrow \text{O/P}$

Moore  $\Rightarrow$ ? circuit

$$\begin{aligned}
 Q(t+1) &= \bar{J}Q_t + \bar{K}Q_t \\
 Q_1(t+1) &= J_1Q_{1t} + \bar{K}_1Q_t
 \end{aligned}$$

Method  
①

## Finding the Next State of the circuit $\Rightarrow$ 2 methods

- Find the Boolean Equations for the flip flop inputs in terms of present state and ext. inputs

$$\overbrace{J_1 \quad K_1 \quad J_0 \quad K_0}^{\text{Inputs}} \quad Q_1 \quad Q_0 \quad \cancel{Q_1 \quad Q_0 \quad \cancel{X}}$$

- Using the equations find the actual values of flip-flop inputs for each possible combinations of present states and inputs

- Use Characteristic tables/equations to find the next state based on flip-flop input values and the present state

$$\overbrace{Q_1 \quad Q_0 \quad \cancel{X}}^{\text{P.S}}$$

$$\left[ \begin{array}{cccc} J_1 & K_1 & J_0 & K_0 \\ \parallel & \parallel & \parallel & \parallel \end{array} \right]$$

$$\overbrace{Q_1 \quad Q_0}^{\text{N.S}} \quad \cancel{Q_1 \quad Q_0}$$

## Method 2

$J_1, K_1, J_0, K_0$

- Find the Boolean Equations for the flip flop inputs in terms of present state and ext. inputs
- Derive state equations from flip flop input equation and characteristic equation of flip flops
- Derive the next state values from state equations by substituting binary values of all combinations of present state and input(s).

$$Q_1(t+1) = J_1 \overline{Q}_1 + \overline{K}_1 Q_1$$

$$Q_0(t+1) = J_0 \overline{Q}_0 + \overline{K}_0 Q_0$$

## Step 1: Flip-flop input equations

- For our example, the **flip-flop input equations** are:

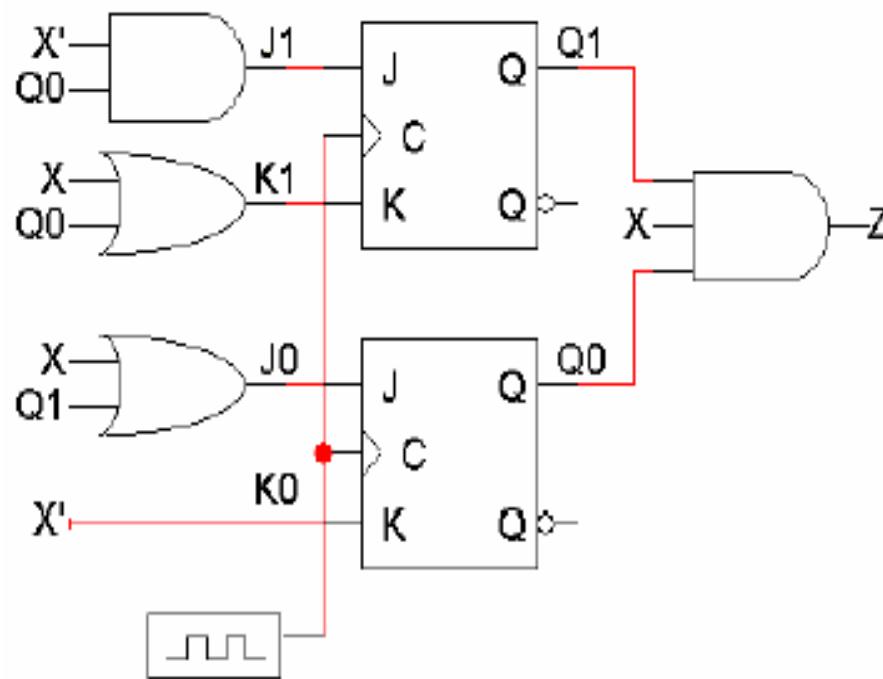
$$J_1 = X' Q_0$$

$$K_1 = X + Q_0$$

$$J_0 = X + Q_1$$

$$K_0 = X'$$

- JK flip-flops each have two inputs, J and K. (D and T flip-flops have one input each.)



## Step 2: Flip-flop input values

- With these equations, we can make a table showing  $J_1$ ,  $K_1$ ,  $J_0$  and  $K_0$  for the different combinations of present state  $Q_1Q_0$  and input  $X$ .

$$\underline{J_1 = X' Q_0}$$

$$K_1 = X + Q_0$$

$$J_0 = X + Q_1$$

$$K_0 = X'$$

Present State		Inputs	Flip-flop Inputs			
$Q_1$	$Q_0$	$X$	$J_1$	$K_1$	$J_0$	$K_0$
0	0	0	0	0	0	1
0	0	1	0	1	1	0
0	1	0	1	1	0	1
0	1	1	0	1	1	0
1	0	0	0	0	1	1
1	0	1	0	1	1	0
1	1	0	1	1	1	1
1	1	1	0	1	1	0

## Step 3: Find the next states

- Finally, use the JK flip-flop characteristic tables or equations to find the next state of *each* flip-flop, based on its present state and inputs.
- The general JK flip-flop characteristic equation was given earlier today.

$$Q(t+1) = K'Q(t) + JQ'(t)$$

- In our example circuit, we have two JK flip-flops, so we have to apply this equation to *each* of them.

$$Q_1(t+1) = K_1'Q_1(t) + J_1Q_1'(t)$$

$$Q_0(t+1) = K_0'Q_0(t) + J_0Q_0'(t)$$

## Step 3 concluded

- Finally, here are the next states for  $Q_1$  and  $Q_0$ , using these equations.

Characteristic  
eqns

$$Q_1(t+1) = K_1'Q_1(t) + J_1Q_1'(t)$$

$$Q_0(t+1) = K_0'Q_0(t) + J_0Q_0'(t)$$

Present State		Inputs $X$	Flip-flop Inputs				Next State	
$Q_1$	$Q_0$		$J_1$	$K_1$	$J_0$	$K_0$	$Q_1$	$Q_0$
0	0	0	0	0	0	1	0	0
0	0	1	0	1	1	0	0	1
0	1	0	1	1	0	1	1	0
0	1	1	0	1	1	0	0	1
1	0	0	0	0	1	1	1	1
1	0	1	0	1	1	0	0	1
1	1	0	1	1	1	1	0	0
1	1	1	0	1	1	0	0	1

# Getting the state table columns straight

- The table starts with Present State and Inputs.
  - Present State and Inputs yield FF Inputs.
  - Present State and FF Inputs yield Next State, based on the flip-flop characteristic tables.
  - Present State and Inputs yield Output.
- We really only care about FF Inputs in order to find Next State.

method ①

Present State		Inputs	Flip-flop Inputs				Next State		Output
$Q_1$	$Q_0$	X	$J_1$	$K_1$	$J_0$	$K_0$	$Q_1$	$Q_0$	Z
0	0	0	0	0	0	1	0	0	0
0	0	1	0	1	1	0	0	1	0
0	1	0	1	1	0	1	1	0	0
0	1	1	0	1	1	0	0	1	0
1	0	0	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0	1	0
1	1	0	1	1	1	1	0	0	0
1	1	1	0	1	1	0	0	1	1

Method 2

$$\text{Characteristic equations } Q_1(t+1) = J_1 \bar{Q}_1 + \bar{K}_1 Q_1$$

$$Q_0(t+1) = J_0 \bar{Q}_0 + \bar{K}_0 Q_0$$

flip flop  $J_1 = \bar{x} Q_0$        $J_0 = x + Q_0$   
 $K_1 = x + Q_0$        $K_0 = \bar{x}$

ff ilp  
equations

Substitute

$$Q_1(t+1) = (\cancel{\bar{x} Q_0} \bar{Q}_1) + (\cancel{(x + Q_0)} \bar{x}) Q_1$$

$$= \cancel{\bar{x} Q_0} \bar{Q}_1 + \cancel{\bar{x} Q_0} Q_1$$

$$= \cancel{x} \bar{Q}_0 + \cancel{x} Q_0$$

$$Q_0(t+1) = (\cancel{x + Q_1}) \bar{Q}_0 + \cancel{x} Q_0$$

$$= \cancel{x} \bar{Q}_0 + Q_1 \bar{Q}_0 + x Q_0$$

State Equations

$$Q_1(t+1) = \cancel{x} \bar{Q}_0 \cancel{Q_1} + \cancel{x} \bar{Q}_0 Q_1$$

$$Q_0(t+1) = \cancel{x} \bar{Q}_0 + Q_1 \bar{Q}_0 + x Q_0$$

Method 2

State  
Equations

$$Q_1(t+1) = \bar{x} Q_0 Q_1 + \bar{x} \bar{Q}_0 Q_1$$

$$Q_0(t+1) = x \bar{Q}_0 + Q_1 \bar{Q}_0 + x Q_0$$

$$\begin{matrix} Q_1 \\ Q_0 \end{matrix} \times \begin{matrix} x \\ 1 \end{matrix}, \quad \begin{matrix} \bar{Q}_0 \\ 0 \end{matrix}$$

$$Q_1(t+1) = 1 \cdot 0 \cdot 1 + 1 \cdot 1 \cdot 1$$

$$= 0 + 1$$

$$= 1$$

$$Q_0(t+1) = 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0$$

$$= 0 + 1 + 0$$

$$= 1$$

P. S	I/P	N.S	O/P
$Q_1$	$Q_0$	$Q_1$ $Q_0$	$Z$
0      0	x	0      0	0
0      0	0	0      1	0
0	1	1      0	0
0	0	0      1	0
0	1	0      0	0
0	0	1      0	0
0	1	0      1	0
0	0	0      0	0
1	0	0      0	0
1	1	0      1	1

P.S

$Q_1, Q_0$

$$\begin{array}{r} 0 \quad 0 \\ \times \quad \underline{Q_1 \quad Q_0} \\ 0 \quad 0 \end{array}$$

$$\begin{array}{r} 0 \quad 1 \\ \times \quad \underline{Q_1 \quad Q_0} \\ 0 \quad 1 \end{array}$$

$$\begin{array}{r} 1 \quad 0 \\ \times \quad \underline{Q_1 \quad Q_0} \\ 1 \quad 0 \end{array}$$

$$\begin{array}{r} 1 \quad 1 \\ \times \quad \underline{Q_1 \quad Q_0} \\ 1 \quad 1 \end{array}$$

$$\begin{array}{r} I/P \quad N.S \\ \times \quad Q_1 \quad Q_0 \\ 0 \quad 0 \end{array}$$

$$\begin{array}{r} 0 \quad 0 \\ \xrightarrow{0 \quad 1} \\ 0 \quad 1 \end{array}$$

$$\begin{array}{r} 0 \quad 1 \\ \xrightarrow{1 \quad 0} \\ 0 \quad 1 \end{array}$$

$$\begin{array}{r} 0 \quad 1 \quad 1 \quad 0 \\ \xrightarrow{1 \quad 0 \quad 1 \quad 0} \\ 0 \quad 0 \quad 0 \quad 0 \end{array}$$

$$\begin{array}{r} 0 \quad 0 \quad 1 \quad 0 \\ \xrightarrow{1 \quad 0 \quad 1 \quad 0} \\ 0 \quad 0 \quad 1 \quad 1 \end{array}$$

$$\begin{array}{r} 0 \quad 0 \\ \xrightarrow{0 \quad 1} \\ 0 \quad 1 \end{array}$$

$$\begin{array}{r} 0 \quad 1 \\ \xrightarrow{1 \quad 0} \\ 1 \quad 0 \end{array}$$

$$\begin{array}{r} 1 \quad 1 \\ \xrightarrow{0 \quad 0} \\ 0 \quad 0 \end{array}$$

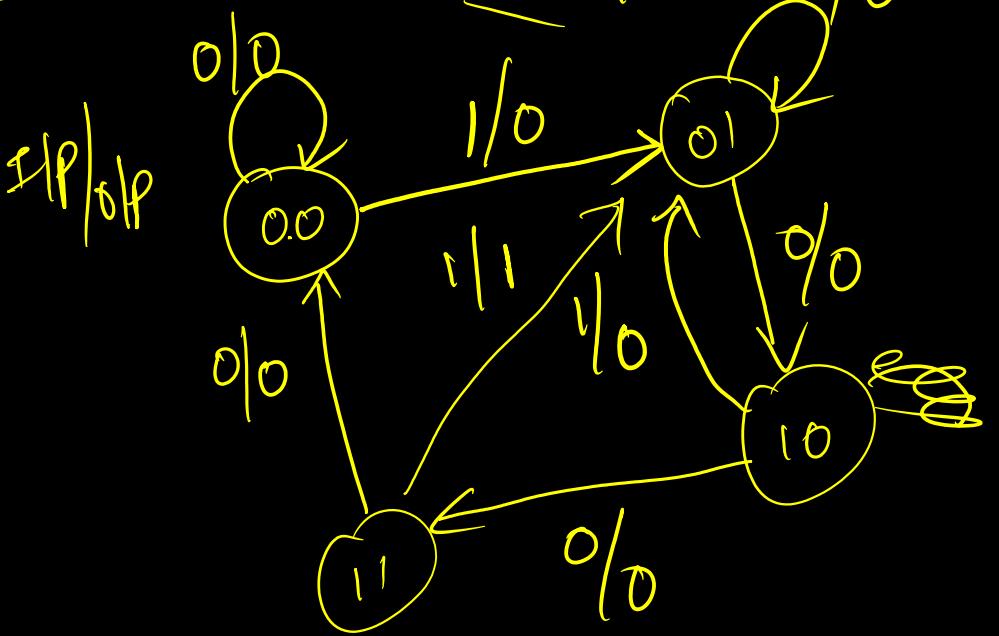
State diagram

states  $\Rightarrow$  circles

2 FFs

4 states

60  
01  
10  
11



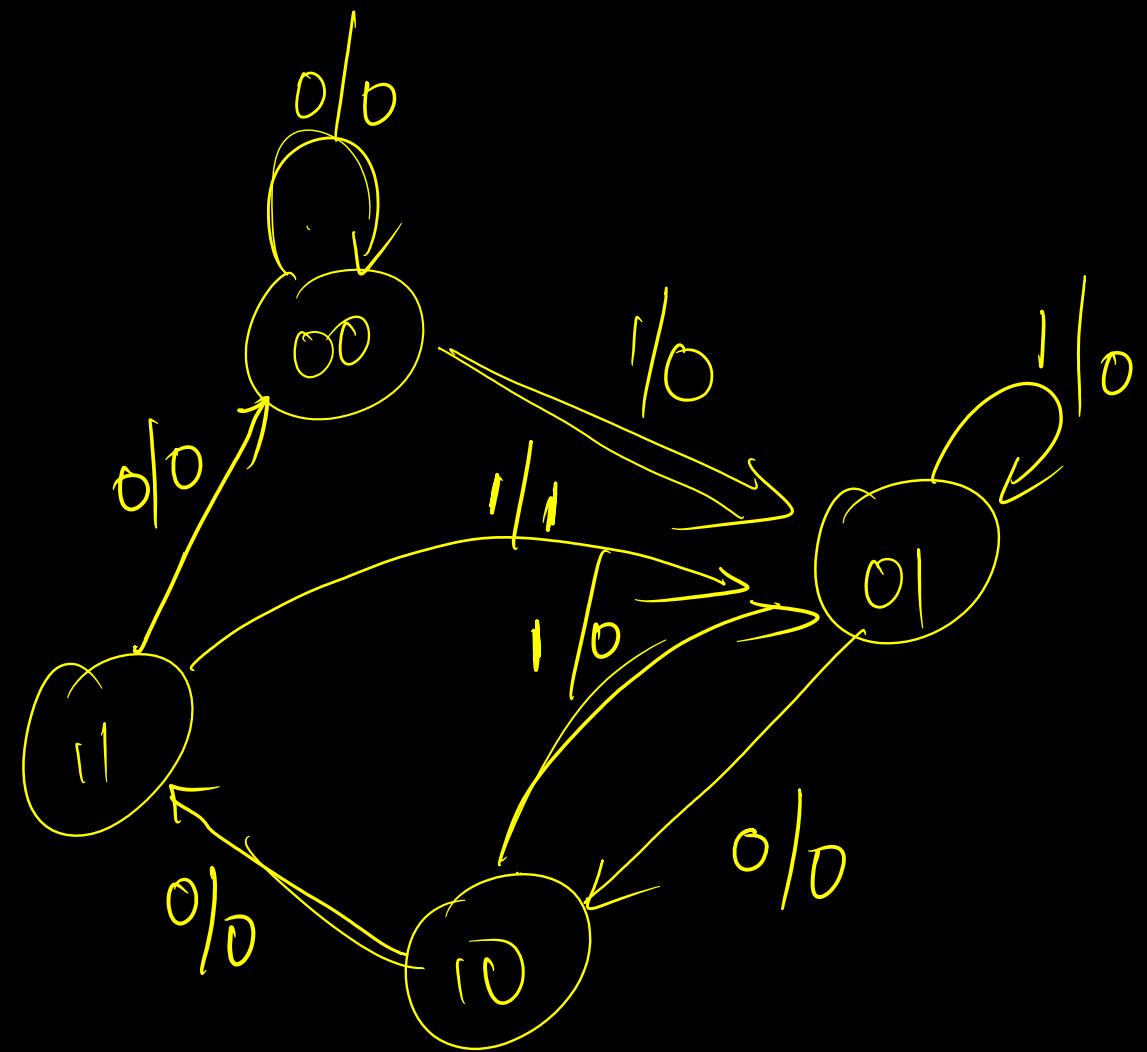
graphical (or) pictorial representation

of state tab.

$n$  FFs  $\Rightarrow 2^n$  states

$00 \xrightarrow{0} 00$
$00 \xrightarrow{1} 01$
$01 \xrightarrow{0} 10$
$01 \xrightarrow{1} 11$
$10 \xrightarrow{0} 01$
$10 \xrightarrow{1} 11$
$11 \xrightarrow{0} 00$
$11 \xrightarrow{1} 01$

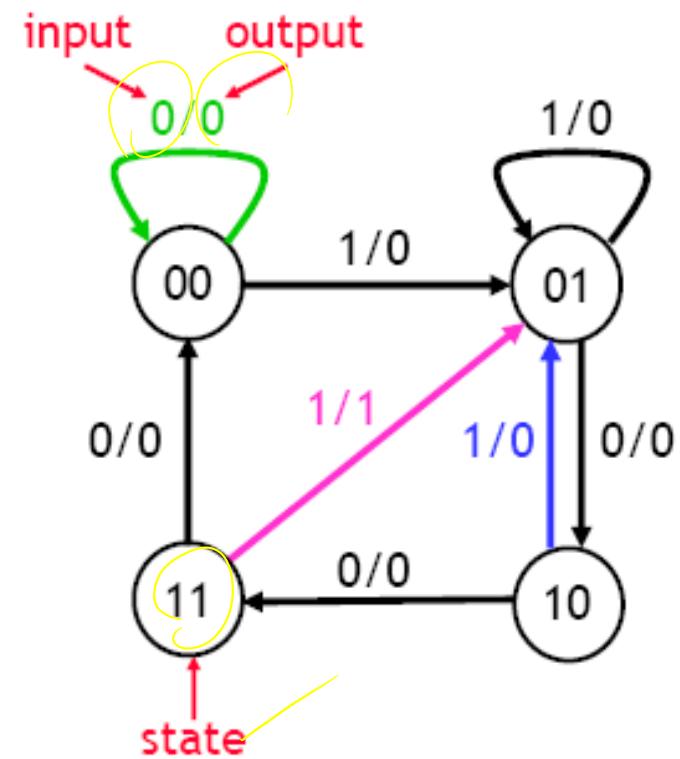
P.S	I/P	N.S	O/P
Q, Q <sub>D</sub>	X	Q <sub>I</sub>	Q <sub>D</sub>
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
0 1	1	0 1	0
1 0	0	1 1	0
1 0	1	0 1	0
1 1	0	0 0	0
1 1	1	0 1	1



# State diagrams

- We can also represent the state table graphically with a **state diagram**.
  - The diagram has one **node** for each possible state.
  - **Arrows** in the diagram connect present states to next states, and are labelled with “input/output.”
- A diagram corresponding to our example state table is shown below.

Present State	Inputs		Next State	Output	
$Q_1$	$Q_0$	X	$Q_1$	$Q_0$	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	1

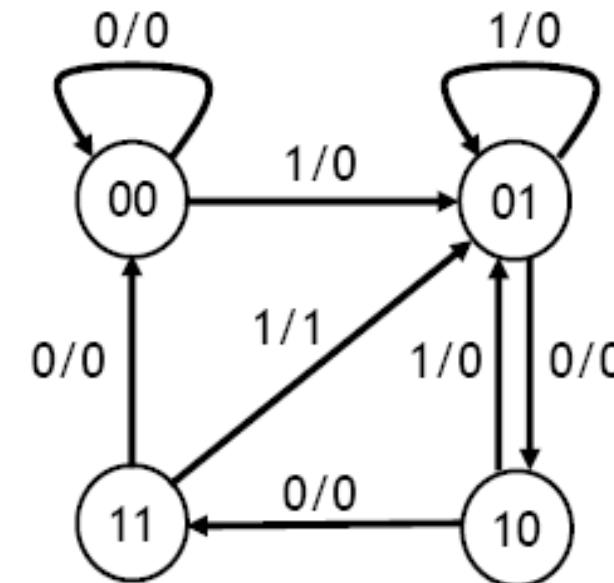


## Size of the state diagram

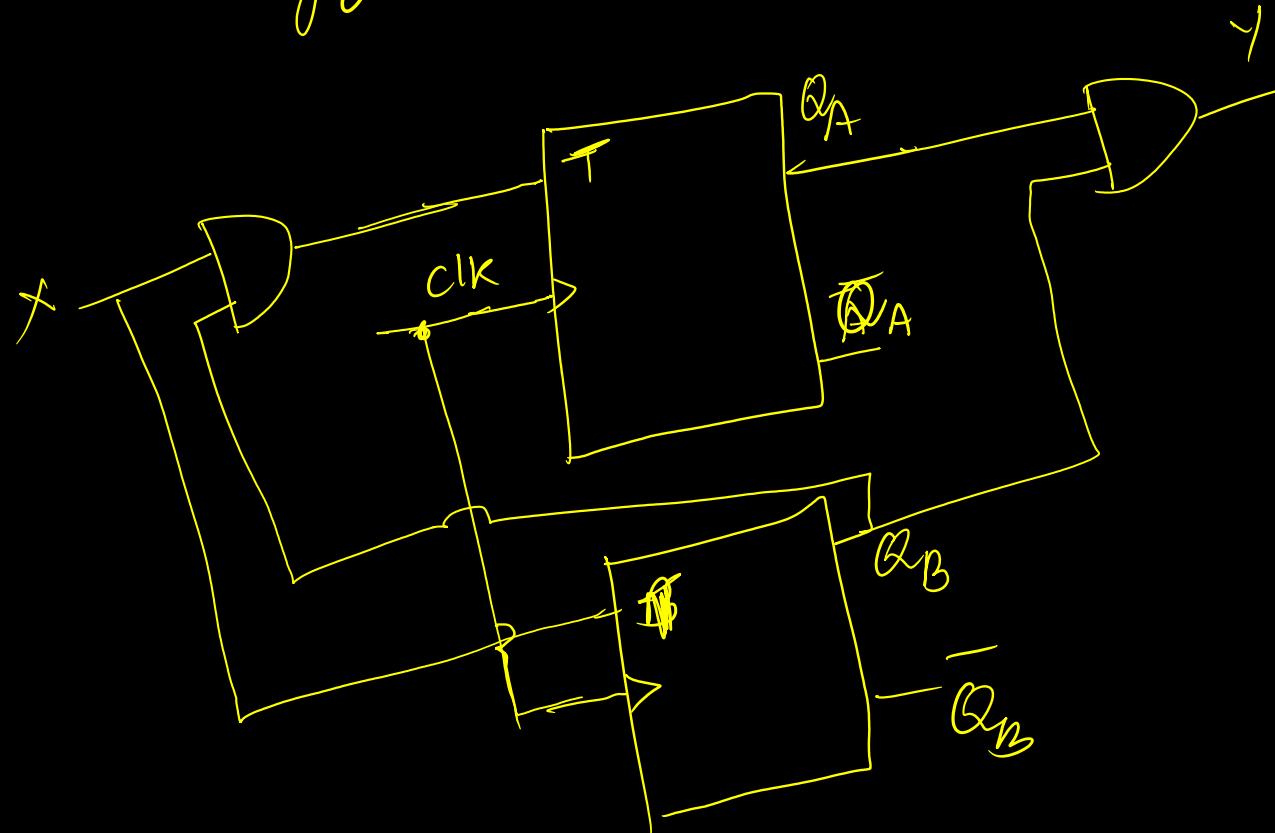
- Always check the size of your state diagrams.
  - If there are  $n$  flip-flops, there should be  $2^n$  nodes in the diagram.
  - If there are  $m$  inputs, then each node will have  $2^m$  outgoing arrows.
- Our example circuit has two flip-flops and one input, so the state diagram should have four nodes, each with two outgoing arrows.

✓ one ilp ✗ <

Present State		Inputs X	Next State		Output Z
$Q_1$	$Q_0$		$Q_1$	$Q_0$	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	1



Analyse





## CS/ECE/EEE/INSTR F215:Digital Design



**Lecture 25: *Design of clocked sequential circuits***

*Sat, 06 Nov 2021*

**BITS** Pilani  
Hyderabad Campus

**Dr. R. N. Ponnalagu, EEE**

*Before You hit You must target*

*If you are taking a single step with confidence  
and strong belief towards your goal,  
definitely majority of the obstacles you face  
will take their way directing you to move  
towards your target to hit it powerfully*



Analyse



$$\begin{aligned} T_A &= Bx \\ T_B &= x \\ Y &= AB \end{aligned} \quad \rightarrow \text{OLP}$$

I/p eqns

Characteristic equation

of T flip flop

$$Q(t+1) = \bar{T}_A \bar{x} + \bar{T}_B x \rightarrow \text{next state}$$

here

$$A(t+1) = T_A \bar{A} + \bar{T}_A A$$

$$B(t+1) = T_B \bar{B} + \bar{T}_B B$$

Substituting  $T_A$  &  $T_B$  values

$$A(t+1) = \bar{A}Bx + A\bar{B} + A\bar{x}$$

$$B(t+1) = \bar{B}x + B\bar{x}$$

State equations

depends on  
Present values  
of  $A$  &  $B$  &  
also on i/p  $x$

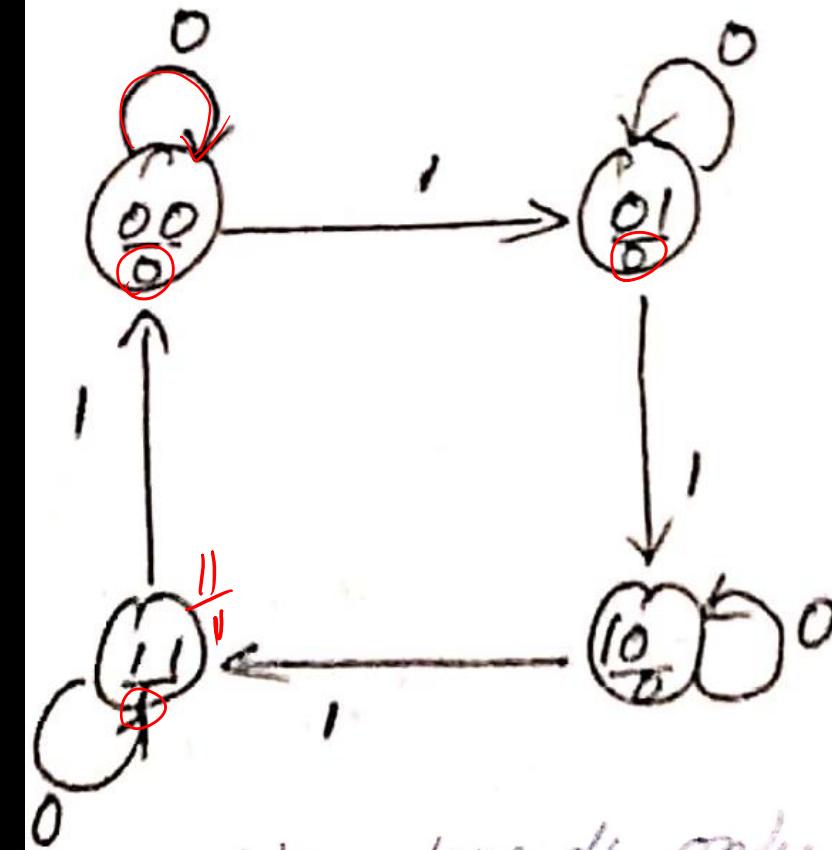
## State table

P.S	I/P	N.S	O/P
A	B	X	X
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

Not depends  
on  $x$

$$Y = \overbrace{A B}^{\text{Present state values}}$$

## State dgm



O/P depends only  
on present state  
values A & B  
This is Moore model

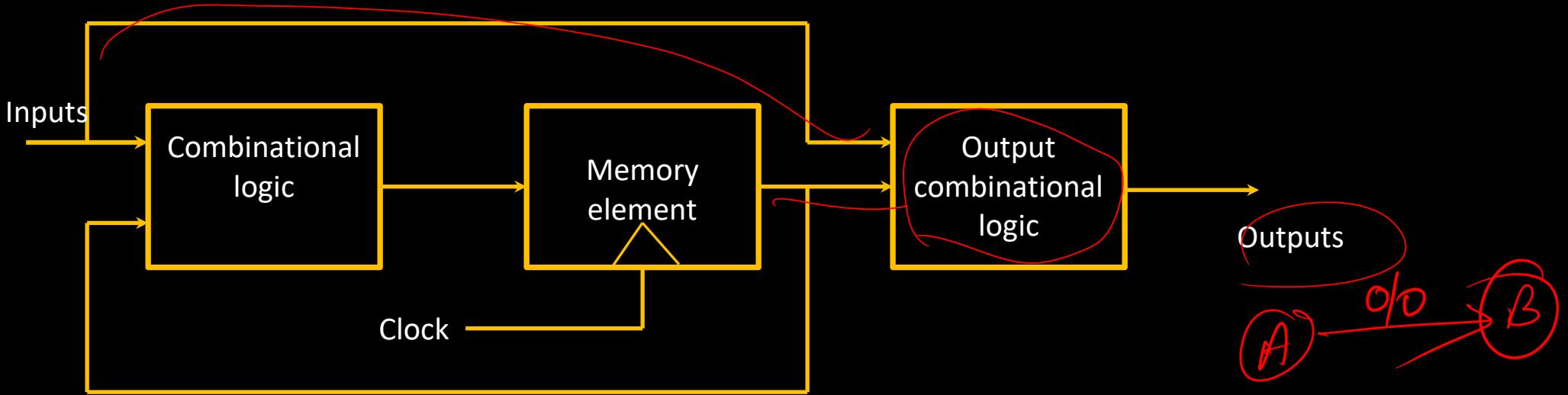
O/Ps are  
indicated  
along  
with states

Moore  
Machine



# Models of sequential circuits

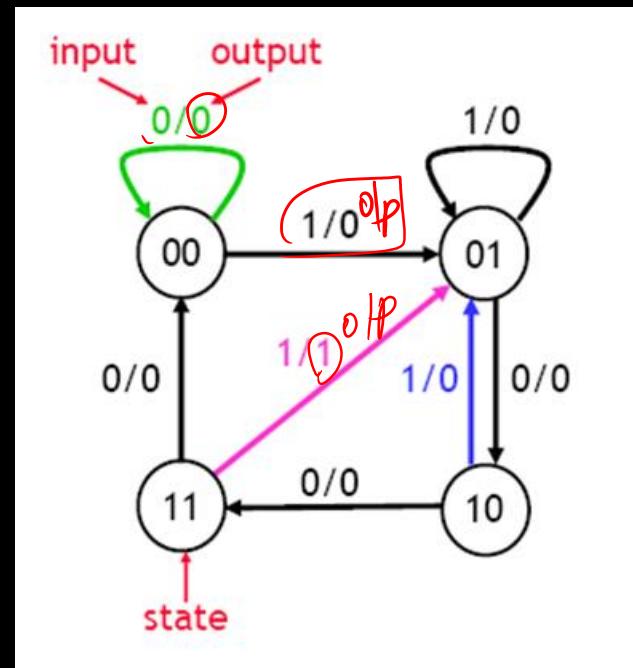
**Mealy Model** – output depends on present state and input



Changes in input may affect the output of the circuit.

To avoid, inputs should be allowed to change only during clock transition.

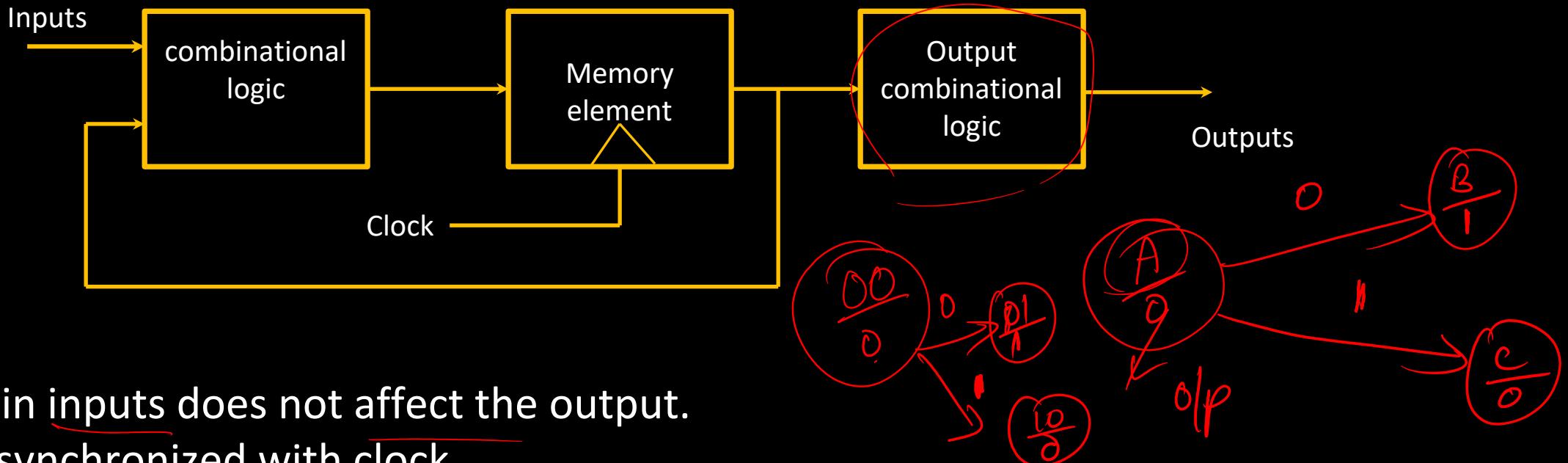
In state diagram representation, outputs are indicated along with inputs in the lines showing transition between the states.





# Models of sequential circuits

Moore Model – output depends only on present state only



Changes in inputs does not affect the output.

Outputs synchronized with clock

In state diagram representation, outputs are indicated within the circle below the present state



# Sequential Circuits

Analysis

Given a circuit

flip flop ip eqns &  
characteristic eqn of ff

Derive state equations & op eqn

State table P.S I/P N.S O/P

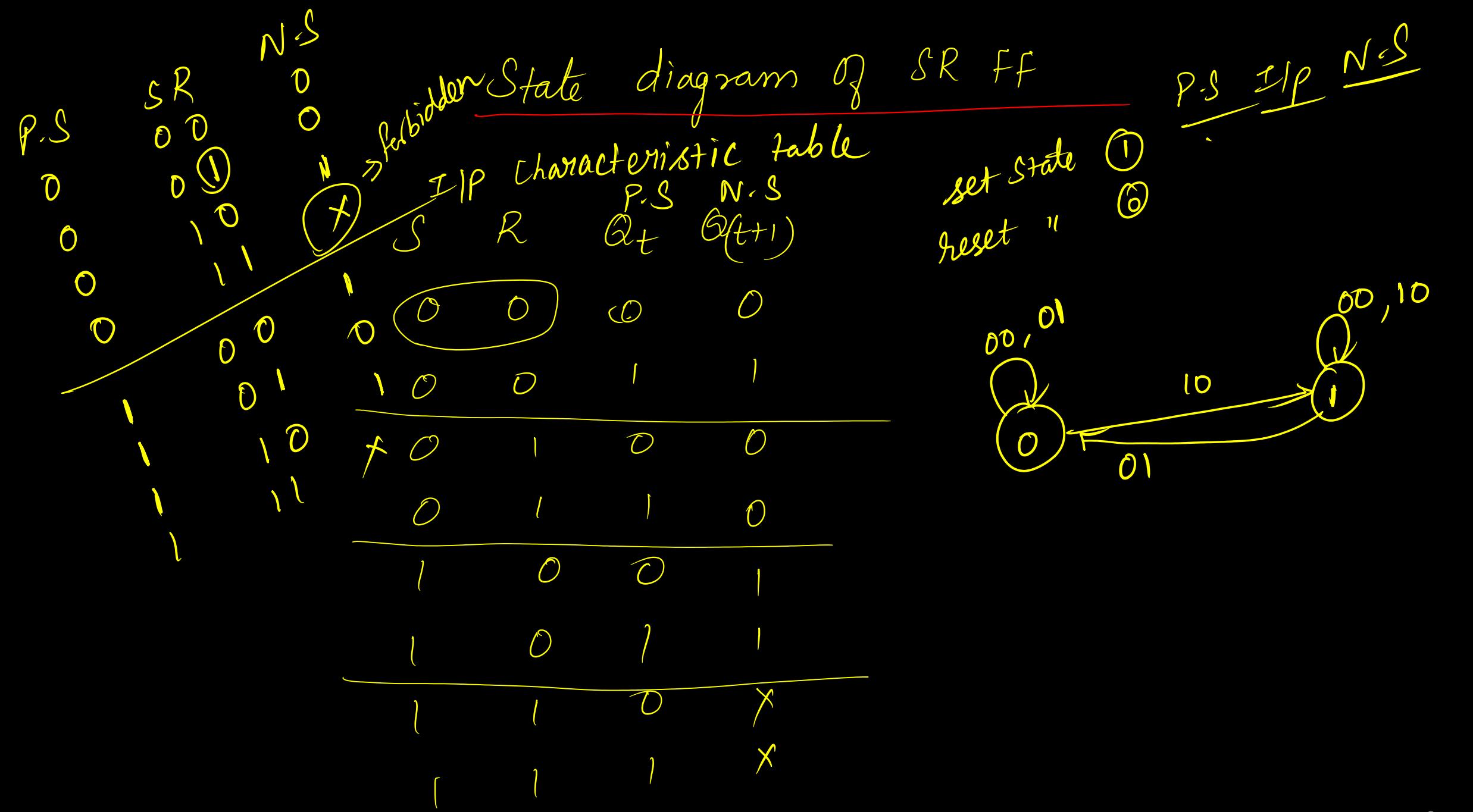
State diagram ✓

sequential  
ckt

Flip Flop State diagram

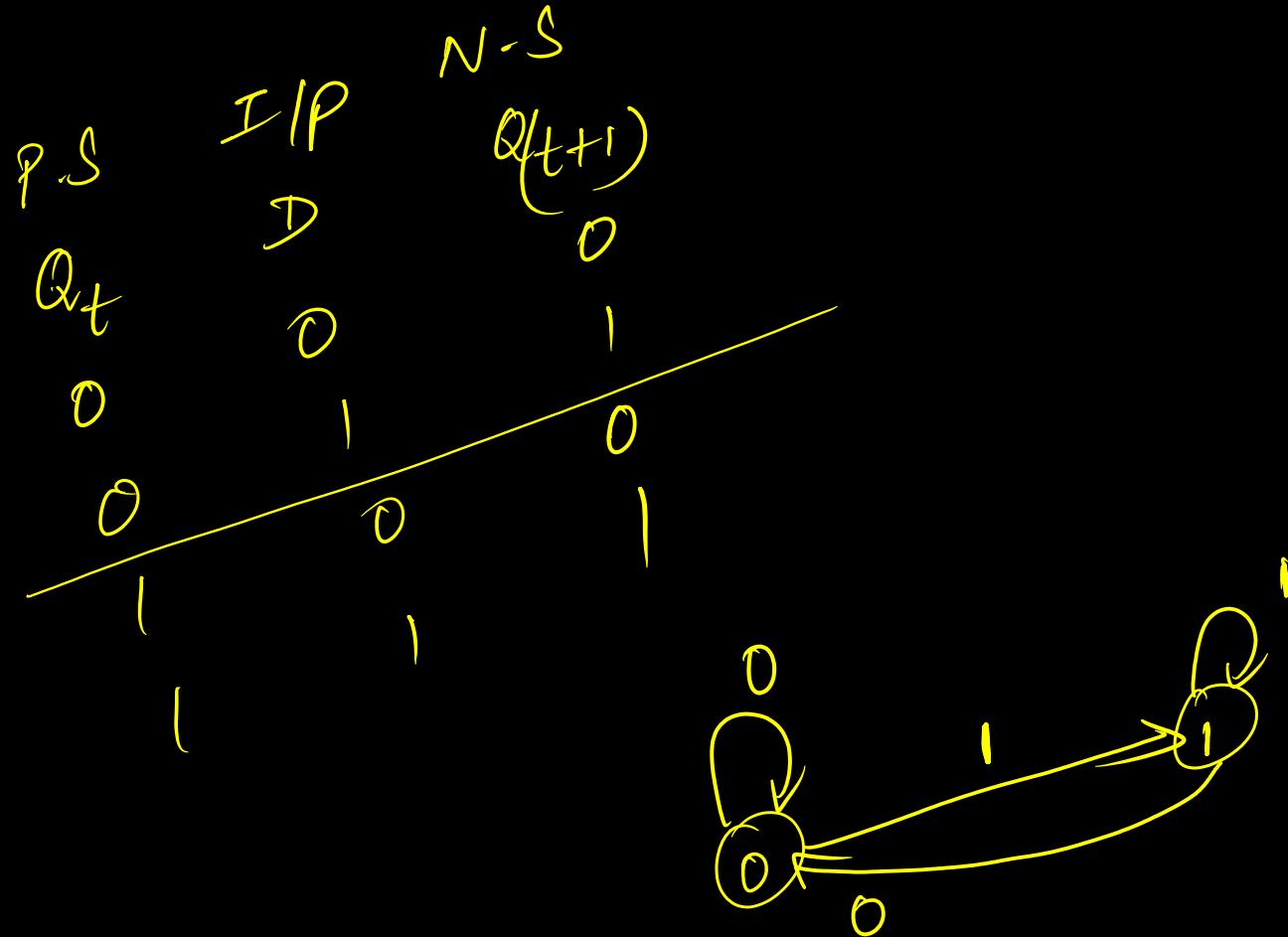
Try for SR, D, JK and T

ff



$Q(t+1) = D$

JK FF  
&  
T FF



# **DESIGN OF CLOCKED SEQUENTIAL CIRCUITS**

# Design of sequential Circuits

Given a Functionality/problem description

① Derive State diagram / state table which ever is easy

State reduction ( if possible)

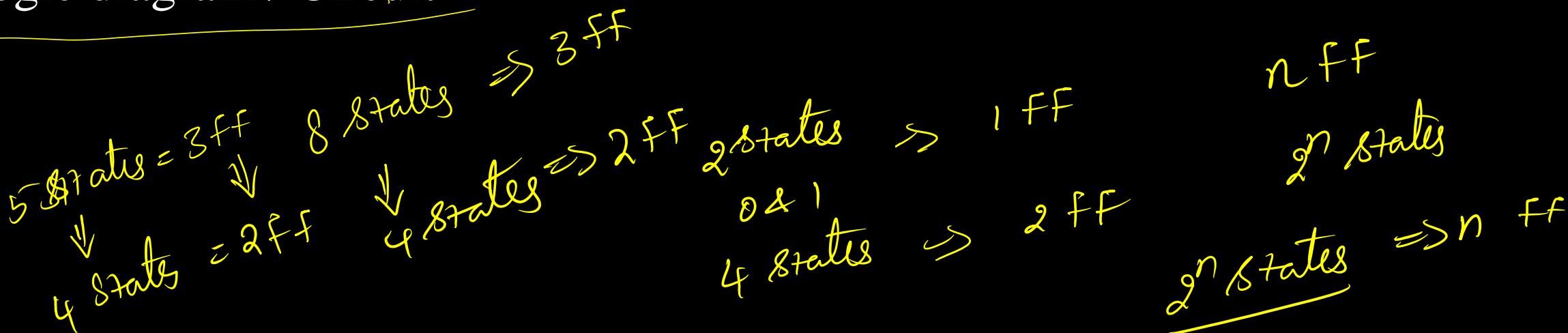
Decide on No. of flipflops (depends on no. of states)

Assign Flipflops

$SR, D, JK \& T$

Write Boolean equations for the inputs and outputs using K-map

Logic diagram / Circuit



# STATE REDUCTION

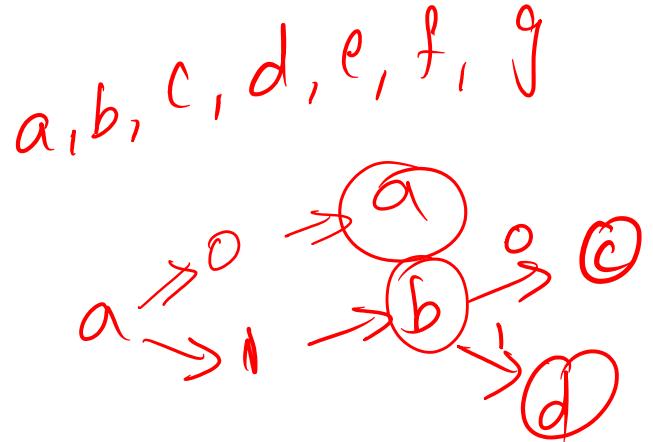
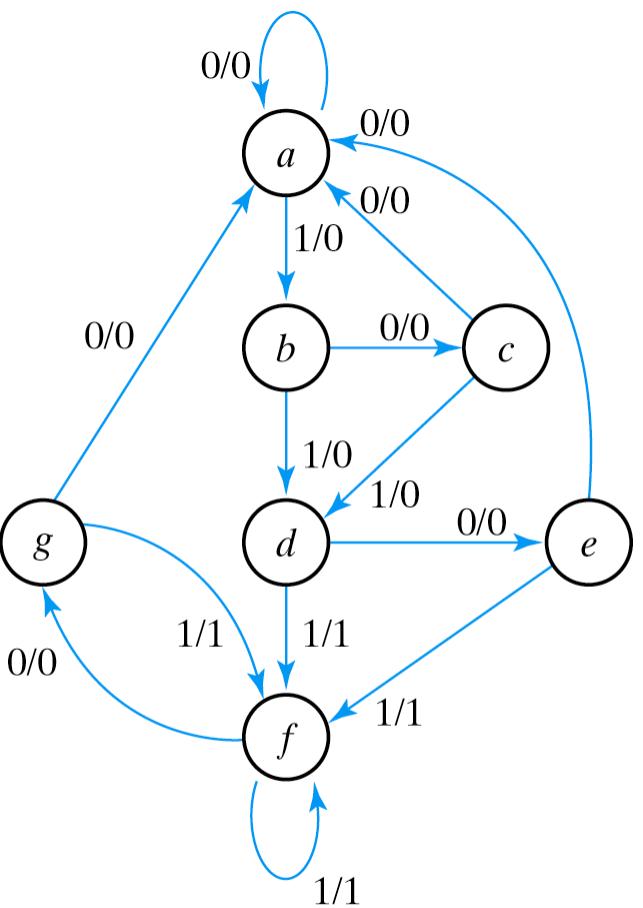
- \* Reduces No. of states
- \* In turn reduces No. of flip flops

$$\hookrightarrow \underbrace{n \text{ flipflops}}_{\text{--}} \Rightarrow \underbrace{2^n \text{ states}}$$

\* Algorithm: Two states are said to be equivalent if every possible set of inputs generate exactly same output & same next state.

When 2 states are equivalent, one of them can be removed.

P.S



State Diagram

## STATE TABLE

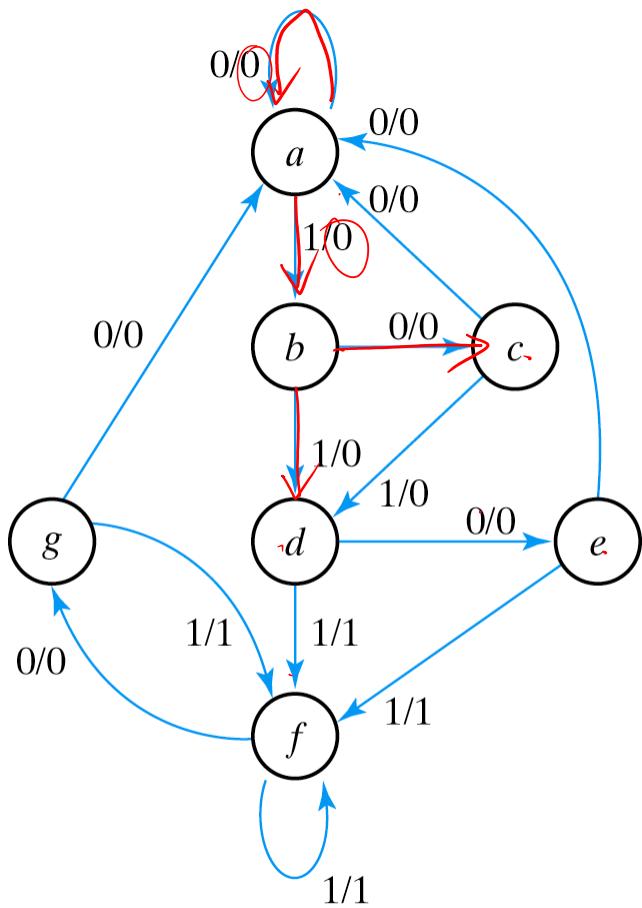


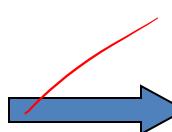
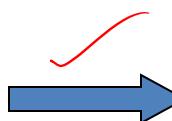
Fig. 5-22 State Diagram

Present state	Next State		Output	
	<del>I/P</del> $X=0$	<del>I/P</del> $X=1$	<del>E/P</del> $X=0$	<del>E/P</del> $X=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

## STATE TABLE

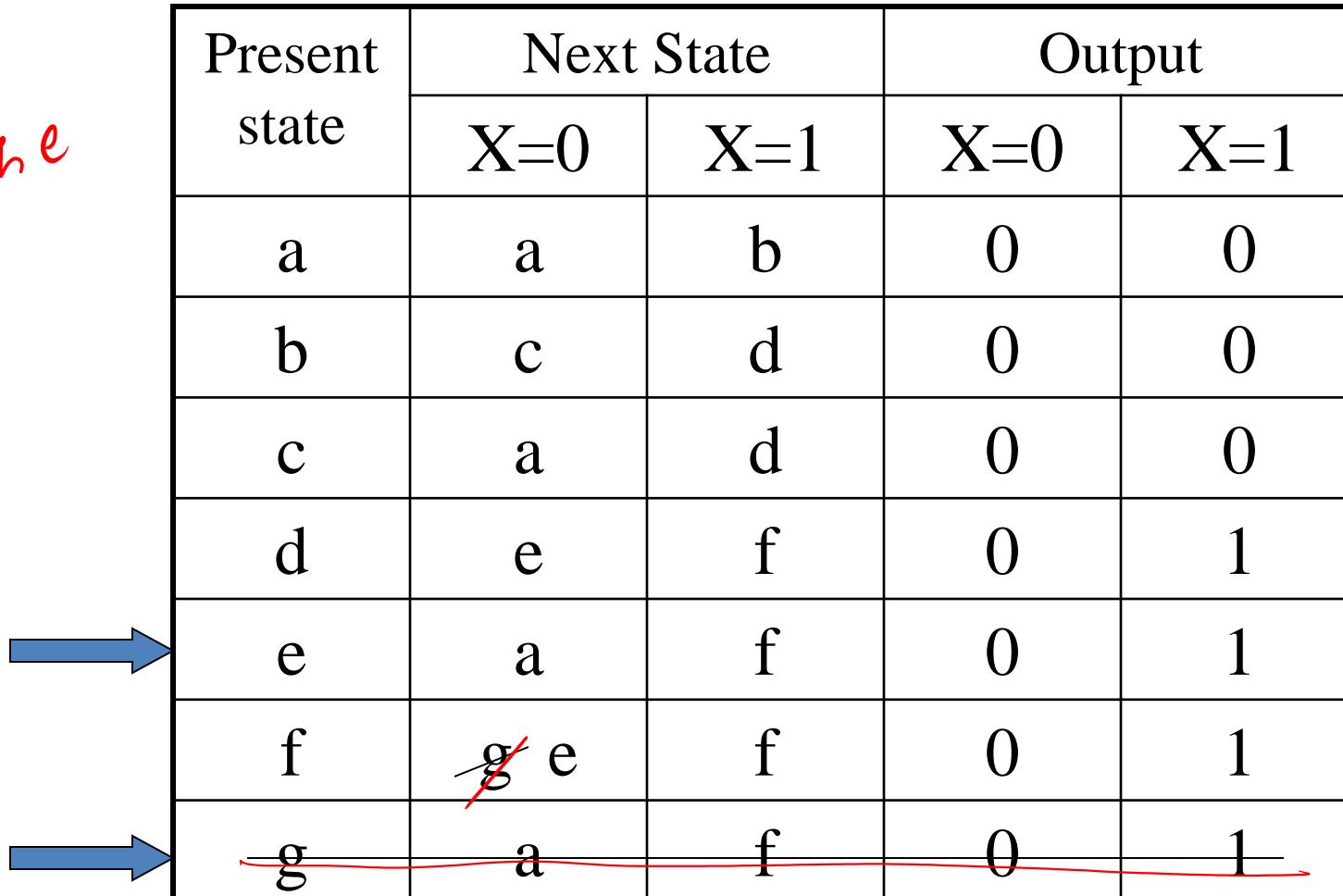
Present state	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

$e \equiv g$



## STATE TABLE

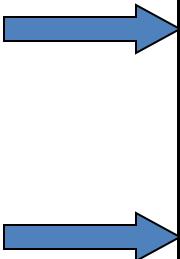
remove g  
replace g with e



Present state	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	<del>g</del> e	f	0	1
g	a	f	0	1

## STATE TABLE

Present state	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1



## STATE TABLE

Present state	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

$d \equiv f$



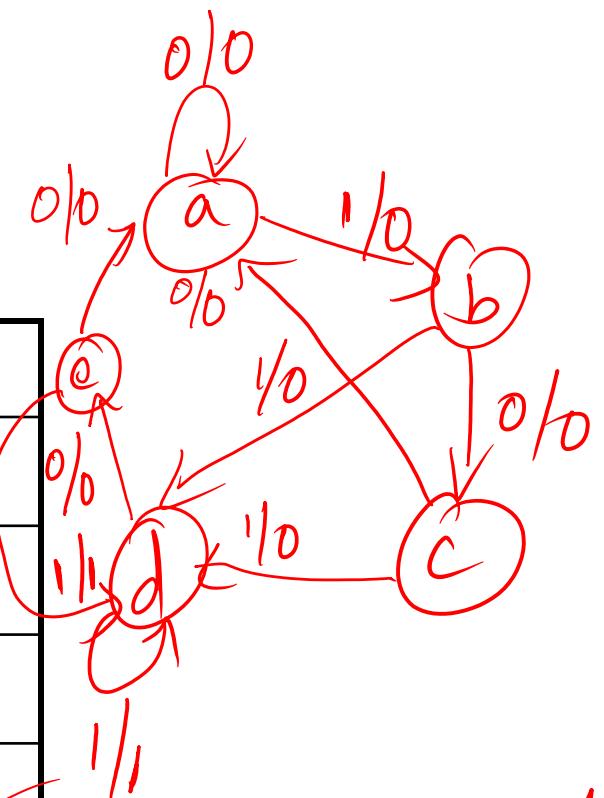
## STATE TABLE

No more reduction possible

Present state	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

3 FFS

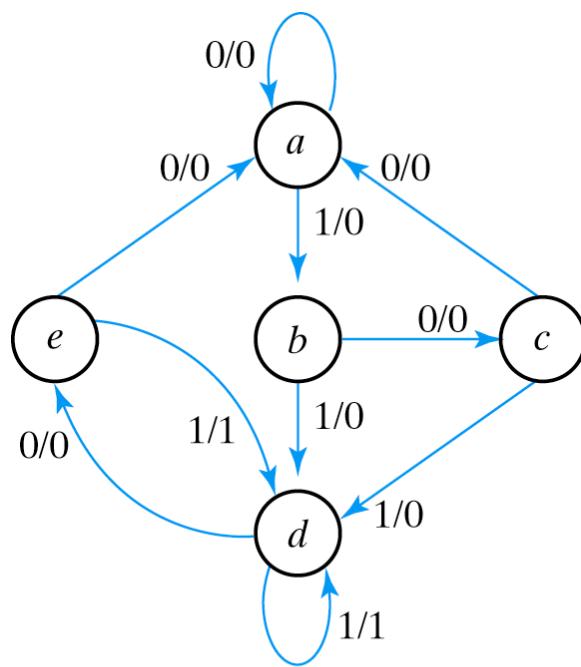
$$\begin{array}{l}
 \begin{array}{l}
 \begin{array}{l}
 a \xrightarrow{000} \\
 b \xrightarrow{001} \\
 c \xrightarrow{010}
 \end{array}
 \} \quad
 \begin{array}{l}
 d \xrightarrow{011} \\
 e \xrightarrow{100}
 \end{array}
 \}
 \end{array}$$



state assignment  
binary assignment

000	/	111
010	/	110
000	/	110

reduced state diagram



# Sequence Detector

- A Sequence Detector ( Recogniser) look for a specific bit pattern in an input string.
- In the example it has an input line called X. One bit of input is applied on every clock and for example it would take 20 clock cycles to enter a 20 bit string.
- It has one output Z which is 1 when a desired pattern is found.
- To detect a pattern 1001 1001  
Ex:   
Input: 11100110100100110  
Output: 00000100000100100
  - 1> Input: 11100110100100110
  - 2> Output: 00000100000100100

One input and one output appear on each clock
- The circuit need to remember bits to recognise a pattern



## CS/ECE/EEE/INSTR F215:Digital Design



### Lecture 26: *Design of clocked sequential circuits*

*Thu, 11 Nov 2021*



*A dream does not become reality  
through magic;  
it takes sweat, determination  
and hard work*

*~ Colin Powell*









# Excitation Tables



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

Characteristic Table

Q(t)	Q(t+1)	J	K	
0	0	0	X	(0 0) (0 1)
0	1	1	X	(1 0) (1 1)
1	0	X	1	(1 1) (0 1)
1	1	X	0	(1 0) (0 0)

Excitation Table



# Design of Clocked sequential Circuits



D	Q(t+1)
0	0
1	1

Characteristic Table

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table

T	Q(t+1)
0	Q(t)
1	$Q'(t)$

Characteristic Table

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

# Sequence Detector

- A Sequence Detector (recogniser) look for a specific bit pattern in an input string.
- In the example it has an input line called X. One bit of input is applied on every clock and for example it would take 20 clock cycles to enter a 20 bit string.
- It has one output Z which is 1 when a desired pattern is found.
- To detect a pattern 1001
  - Ex:      Input:    11100110100100110
  - Output:   00000100000100100

One input and one output appear on each clock
- The circuit need to remember bits to recognise a pattern

## Step 1: Making a state table / state diagram

- The first thing you have to figure out is precisely how the use of state will help you solve the given problem.
  - Make a state table based on the problem statement. The table should show the present states, inputs, next states and outputs.
  - Sometimes it may be easier to come up with a state diagram first and then convert that to a table.
- This is usually the most difficult step. Once you have the state table, the rest of the design procedure is the same for all sequential circuits.
- Sequence recognizers are one of the harder examples we will see in this class, so if you understand this you're in good shape.

# Sequence detector design procedure

States are used to remember meaningful properties of past input sequences that are essential for predicting future output values.

A sequence detector is a sequential circuit that produces a distinct output value whenever a prescribed pattern of input symbols occur in sequence, i.e., detects an input sequence occurrence.

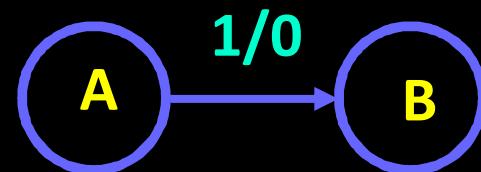
Begin in an **initial state** in which **NONE** of the initial portion of the sequence has occurred (**reset state**)

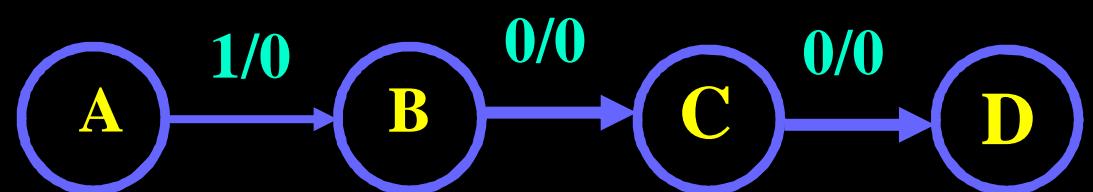
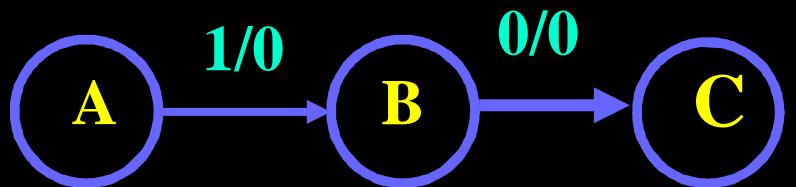
Add a state that detects that first symbol has occurred

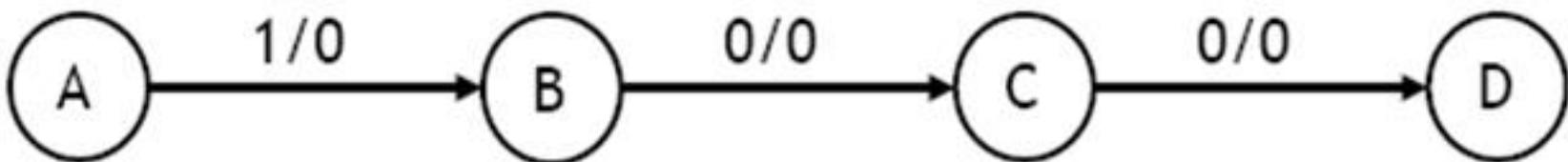
Add states that detect each successive symbol

The final state represents the input sequence occurrence and produces the output

- Start in the initial state
  - State ‘A’ is the initial state
  - Add a state ‘B’ that recognizes the first ‘1’
  - State ‘B’ is the state which represents the fact that the first ‘1’ in the input subsequence has occurred. The output symbol ‘0’ means that the full recognized sequence has not yet occurred



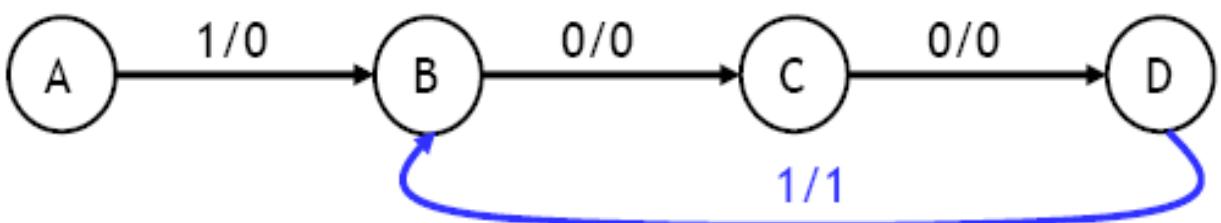




State	Meaning
A	None of the desired pattern (1001) has been entered yet
B	We've already seen the first bit (1) of the desired pattern
C	We've already seen the first two bits (10) of the desired pattern
D	We've already seen the first three bits (100) of the desired pattern

## Overlapping occurrences of the pattern

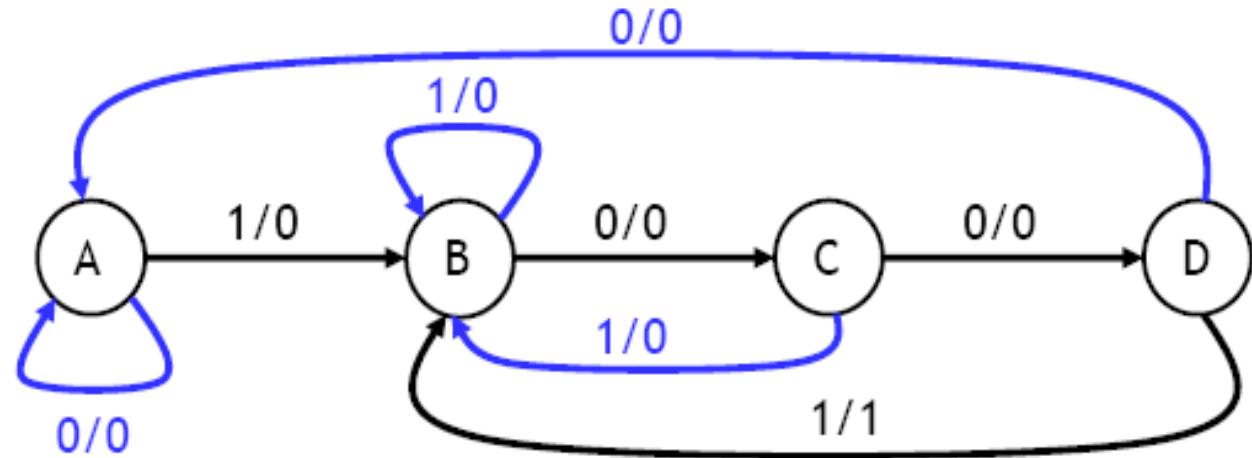
- What happens if we're in state D (the last three inputs were 100), and the current input is 1?
  - The output should be a 1, because we've found the desired pattern.
  - But this last 1 could also be the start of another occurrence of the pattern! For example, 1001001 contains two occurrences of 1001.
  - To properly detect overlapping occurrences of the pattern, the next state should be B.



State	Meaning
A	None of the desired pattern (1001) has been entered yet
B	We've already seen the first bit (1) of the desired pattern
C	We've already seen the first two bits (10) of the desired pattern
D	We've already seen the first three bits (100) of the desired pattern

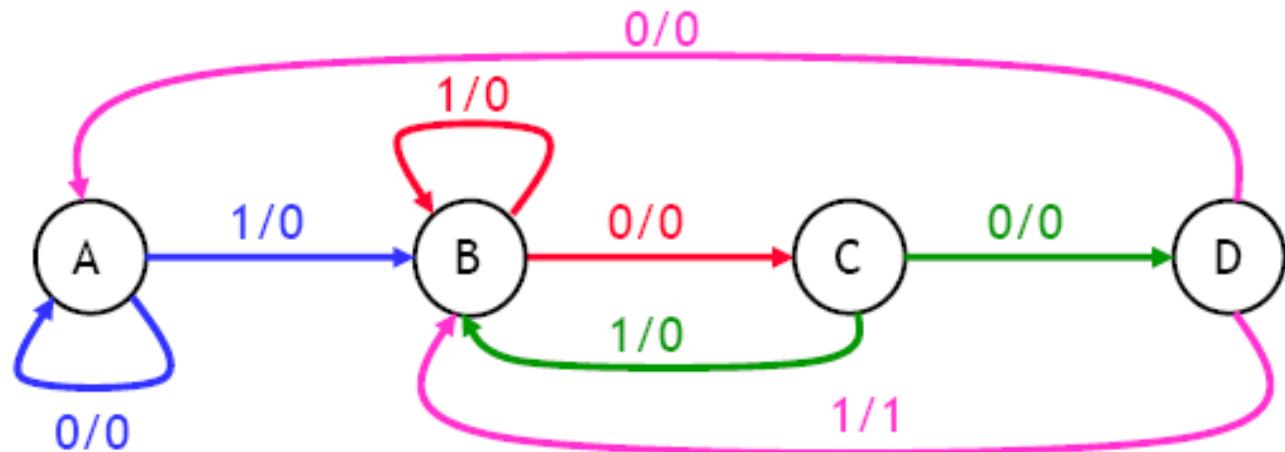
## Filling in the other arrows

- Remember that we need two outgoing arrows for each node, to account for the two input possibilities of  $X = 0$  and  $X = 1$ .
- The remaining arrows we need are shown in blue. They also allow for the correct detection of overlapping occurrences of 1001.

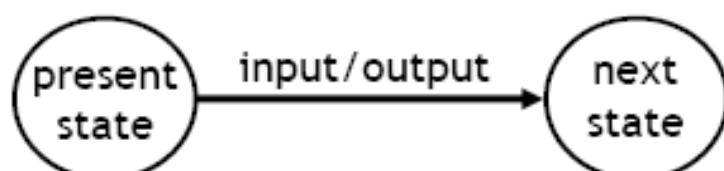


State	Meaning
A	None of the desired pattern (1001) has been entered yet
B	We've already seen the first bit (1) of the desired pattern
C	We've already seen the first two bits (10) of the desired pattern
D	We've already seen the first three bits (100) of the desired pattern

## Finally, making the state table



Remember how the state diagram arrows correspond to rows of the state table.



Present State	Input	Next State	Output
A	0	A	0
	1	B	0
B	0	C	0
	1	B	0
C	0	D	0
	1	B	0
D	0	A	0
	1	B	1

We have four states ABCD, so we need at least two flip-flops  $Q_1 Q_0$ .

With 2 bits, we can have 4 codes: 00, 01, 10, and 11

How many assignments of 2-bit codes for the 4 states?

Answer:  $4 \times 3 \times 2 \times 1 = 24$  possible assignments

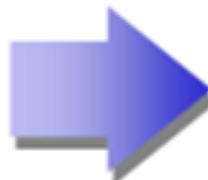
The easiest thing is to represent state A with  $Q_1 Q_0 = 00$ , B with 01, C with 10, and D with 11. (You could have used these codes in Step 1 too, rather than using temporary state labels like ABCD.)

Does code assignment make a difference in cost?

Answer: yes, it affects the cost of the combinational logic

The state assignment can have a big impact on circuit complexity,

Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1



Present State $Q_1\ Q_0$	Input X	Next State $Q_1\ Q_0$		Output Z
		Q <sub>1</sub>	Q <sub>0</sub>	
0 0	0	0	0	0
0 0	1	0	1	0
0 1	0	1	0	0
0 1	1	0	1	0
1 0	0	1	1	0
1 0	1	0	1	0
1 1	0	0	0	0
1 1	1	0	1	1

## Step 3: Finding flip-flop input values

- Next we have to figure out how to actually make the flip-flops change from their present state into the desired next state.
- This depends on what kind of flip-flops you use! We'll use two JKs here.
- For each flip-flop  $Q_i$ , look at its present and next states, and determine what the inputs  $J_i$  and  $K_i$  should be in order to make that state change.

Present State $Q_1$ $Q_0$		Input X	Next State $Q_1$ $Q_0$		Flip-flop Inputs $J_1$ $K_1$ $J_0$ $K_0$				Output Z
0	0	0	0	0					0
0	0	1	0	1					0
0	1	0	1	0					0
0	1	1	0	1					0
1	0	0	1	1					0
1	0	1	0	1					0
1	1	0	0	0					0
1	1	1	0	1					1

# JK excitation table

- An **excitation table** shows what flip-flop inputs are required in order to make a desired state change.

$Q(t)$	$Q(t+1)$	J	K	Operation
0	0	0	x	No change/Reset
0	1	1	x	Set/Complement
1	0	x	1	Reset/Complement
1	1	x	0	No change/Set

- This is the same information that's given in the characteristic table, but presented "backwards."

J	K	$Q(t+1)$	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

## Back to the example

- We can now use the JK excitation table on the right to find the correct values for each flip-flop's inputs, based on its present and next states.

$Q(t)$	$Q(t+1)$	$J$	$K$
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Present State $Q_1$ $Q_0$	Input X	Next State $Q_1$ $Q_0$	Flip-flop Inputs				Output Z
			$J_1$	$K_1$	$J_0$	$K_0$	
0   0	0	0   0	0	x	0	x	0
0   0	1	0   1	0	x	1	x	0
0   1	0	1   0	1	x	x	1	0
0   1	1	0   1	0	x	x	0	0
1   0	0	1   1	x	0	1	x	0
1   0	1	0   1	x	1	1	x	0
1   1	0	0   0	x	1	x	1	0
1   1	1	0   1	x	1	x	0	1

$\mathcal{J}_1$		$\mathcal{Q}_0 X$			
		$\mathcal{Q}_0 X$	$\mathcal{Q}_0 X$	$\mathcal{Q}_0 X$	$\mathcal{Q}_0 X$
$\mathcal{Q}_1$	$\mathcal{Q}_1$	0	0	0	1
	$\mathcal{Q}_1$	x	x	x	x

$\mathcal{K}_1$		$\mathcal{Q}_0 X$			
		$\mathcal{Q}_0 X$	$\mathcal{Q}_0 X$	$\mathcal{Q}_0 X$	$\mathcal{Q}_0 X$
$\mathcal{Q}_1$	$\mathcal{Q}_1$	x	x	x	x
	$\mathcal{Q}_1$	0	1	1	1

$$k_1 = x + \mathcal{Q}_0$$

$$\mathcal{J}_1 = \mathcal{Q}_0 \bar{x}$$

$$\mathcal{J}_0 = x + \mathcal{Q}_1$$

$$k_0 = \bar{x}$$

$$1 = \mathcal{Q}_1 \mathcal{Q}_0 x$$

## Step 4: Find equations for the FF inputs and output

- Now you can make K-maps and find equations for each of the four flip-flop inputs, as well as for the output Z.
- These equations are in terms of the present state and the inputs.

Present State		Input X	Next State		Flip-flop Inputs				Output Z
$Q_1$	$Q_0$		$Q_1$	$Q_0$	$J_1$	$K_1$	$J_0$	$K_0$	
0	0	0	0	0	0	x	0	x	0
0	0	1	0	1	0	x	1	x	0
0	1	0	1	0	1	x	x	1	0
0	1	1	0	1	0	x	x	0	0
1	0	0	1	1	x	0	1	x	0
1	0	1	0	1	x	1	1	x	0
1	1	0	0	0	x	1	x	1	0
1	1	1	0	1	x	1	x	0	1

## Step 4: Find equations for the FF inputs and output

Use k-Maps to find out flip flop inputs and output Z

- The advantage of using JK flip-flops is that there are many don't care conditions, which can result in simpler MSP equations.

Present State		Input X	Next State		Flip-flop Inputs				Output Z
Q <sub>1</sub>	Q <sub>0</sub>		Q <sub>1</sub>	Q <sub>0</sub>	J <sub>1</sub>	K <sub>1</sub>	J <sub>0</sub>	K <sub>0</sub>	
0	0	0	0	0	0	x	0	x	0
0	0	1	0	1	0	x	1	x	0
0	1	0	1	0	1	x	x	1	0
0	1	1	0	1	0	x	x	0	0
1	0	0	1	1	x	0	1	x	0
1	0	1	0	1	x	1	1	x	0
1	1	0	0	0	x	1	x	1	0
1	1	1	0	1	x	1	x	0	1

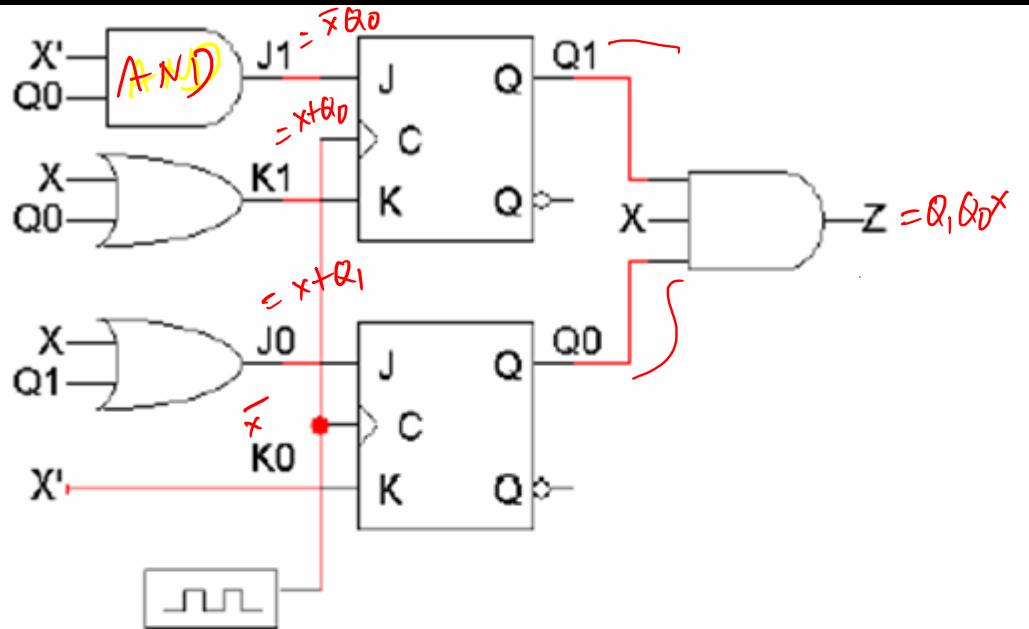
$$J_1 = X'Q_0$$

$$K_1 = X + Q_0$$

$$J_0 = X + Q_1$$

$$K_0 = X'$$

$$Z = Q_1 Q_0 X$$



$$J_1 = X'Q_0$$

$$K_1 = X + Q_0$$

$$J_0 = X + Q_1$$

$$K_0 = X'$$

$$Z = Q_1 Q_0 X$$

This is the same circuit which we have seen in analysis

circuit detects occurrences of the pattern 1001 in a serial input stream  $X$ .



## CS/ECE/EEE/INSTR F215:Digital Design



### Lecture 27: *Design of clocked sequential circuits\_2*

Sat, 13 Nov 2021

**BITS** Pilani  
Hyderabad Campus

Dr. R. N. Ponnalagu, EEE

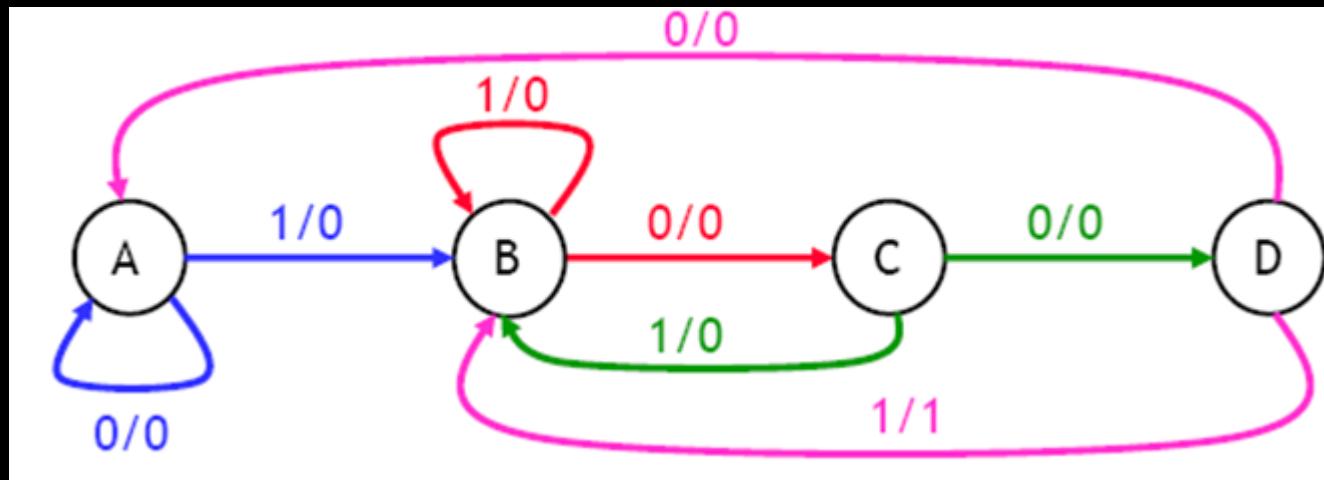
If you have choice

choose the best,

If you have No choice

do the best

► To detect a pattern 1001



Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

Present State		Input			Next State
$Q_1$	$Q_0$	X	$Q_1$	$Q_0$	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	1

Present State $Q_1$ $Q_0$	Input X	Next State $Q_1$ $Q_0$	Flip-flop Inputs				Output Z
			J <sub>1</sub>	K <sub>1</sub>	J <sub>0</sub>	K <sub>0</sub>	
0   0	0	0   0	0	x	0	x	0
0   0	1	0   1	0	x	1	x	0
0   1	0	1   0	1	x	x	1	0
0   1	1	0   1	0	x	x	0	0
1   0	0	1   1	x	0	1	x	0
1   0	1	0   1	x	1	1	x	0
1   1	0	0   0	x	1	x	1	0
1   1	1	0   1	x	1	x	0	1

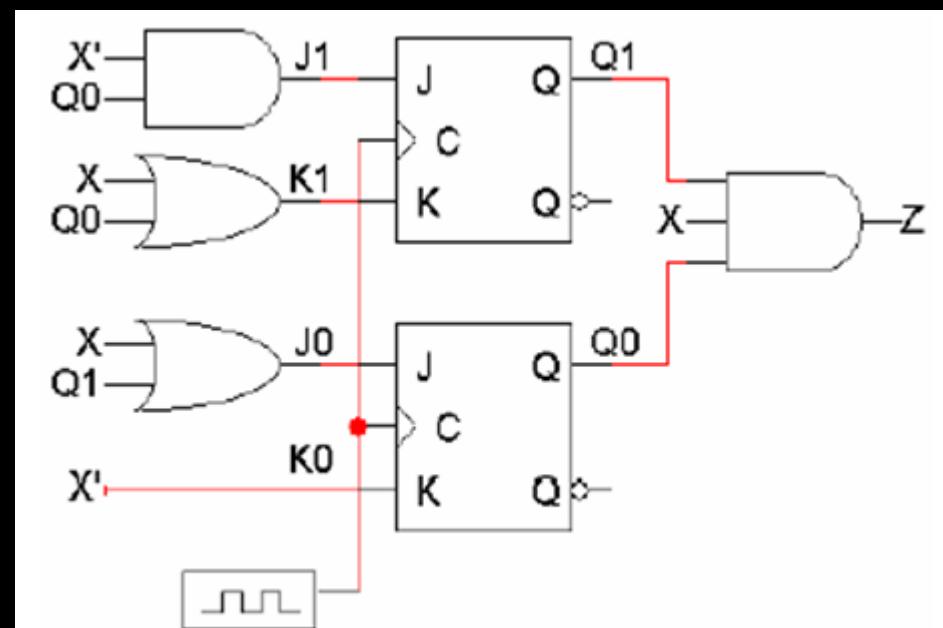
$$J_1 = X'Q_0$$

$$K_1 = X + Q_0$$

$$J_0 = X + Q_1$$

$$K_0 = X'$$

$$Z = Q_1Q_0X$$



## Building the same circuit with D flip-flops

- What if you want to build the circuit using D flip-flops instead?
- We already have the state table and state assignments, so we can just start from Step 3, finding the flip-flop input values.
- D flip-flops have only one input, so our table only needs two columns for  $D_1$  and  $D_0$ .

Present State $Q_1$ $Q_0$	Input X	Next State $Q_1$ $Q_0$	Flip-flop Inputs $D_1$ $D_0$	Output Z
0   0	0	0   0	0   0	0
0   0	1	0   1	0   1	0
0   1	0	1   0	1   0	0
0   1	1	0   1	0   1	0
1   0	0	1   1	1   1	0
1   0	1	0   1	0   1	0
1   1	0	0   0	0   0	0
1   1	1	0   1	0   1	1

## Finding equations (Step 4)

- If you use K-maps again, you should find the following equations.

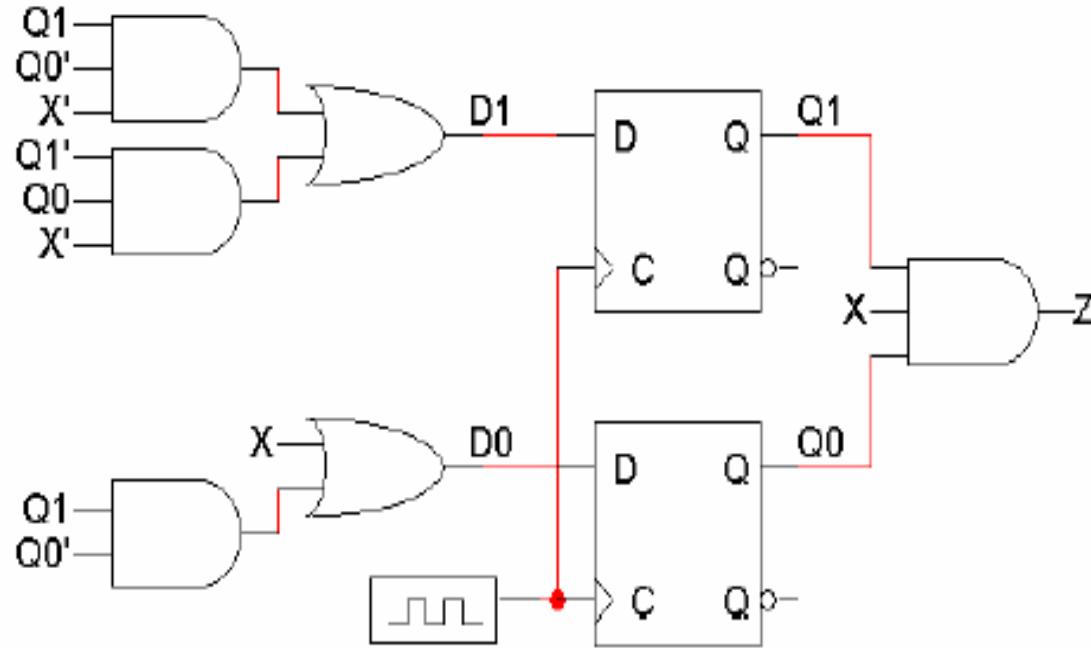
$$\underline{D_1 = Q_1 Q_0' X' + Q_1' Q_0 X'}$$

$$\underline{D_0 = X + Q_1 Q_0'}$$

$$Z = Q_1 Q_0 X$$

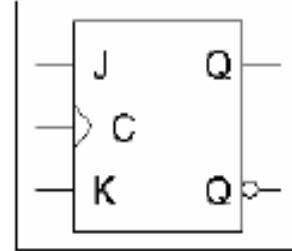
Present State $Q_1 \quad Q_0$		Input X	Next State $Q_1 \quad Q_0$		Flip-flop Inputs $D_1 \quad D_0$		Output Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	0	1	0	1	1

## Building the circuit (Step 5)

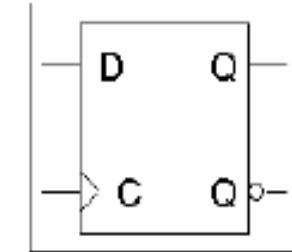


## Flip-flop comparison

- JK flip-flops are good because there are many don't care values in the flip-flop inputs, which can lead to a simpler circuit.



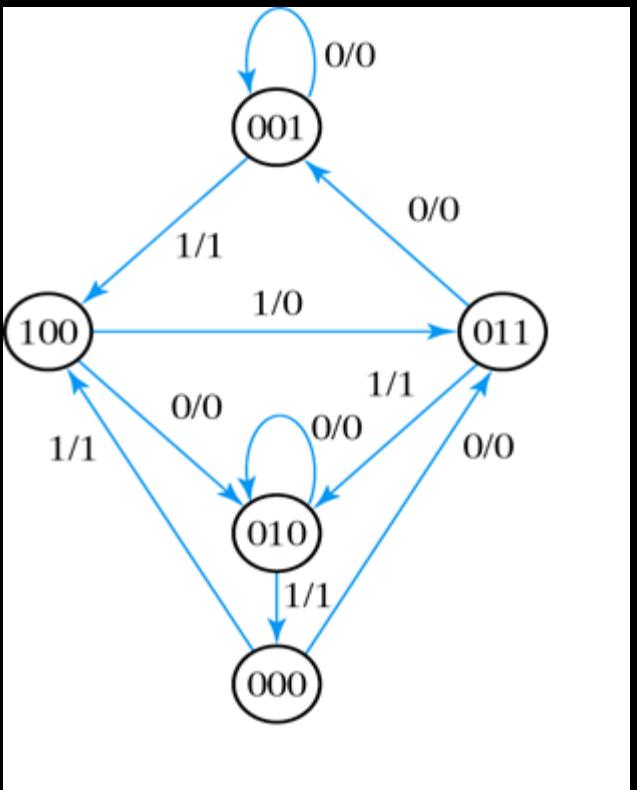
- D flip-flops have the advantage that you don't have to set up flip-flop inputs at all, since  $Q(t+1) = D$ . However, the D input equations are usually more complex than JK input equations.



- In practice, D flip-flops are used more often.
  - There is only one input for each flip-flop, not two.
  - There are no excitation tables to worry about.
  - D flip-flops themselves are simpler to implement than JK flip-flops.

# DESIGN WITH UNUSED STATES

- Design the circuit using ~~D~~ Flip Flops



ff	i		p	b	Tc
TA	TB				
0	0				0
1	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1
1	0				0
0	0				0
0	0				0
0	1				1</

PS			X	NS			$\overline{D_a}$	$\overline{D_b}$	$\overline{D_c}$	Y
A	B	C		A	B	C				
0	0	0	0	0	1	1				0
0	0	0	1	1	0	0				1
0	0	1	0	0	0	1				0
0	0	1	1	1	0	0				1
0	1	0	0	0	1	0				0
0	1	0	1	0	0	0				1
0	1	1	0	0	0	1				0
0	1	1	1	0	1	0				1
1	0	0	0	0	1	0				0
1	0	0	1	0	1	1				0

PS			X	NS			$D_A$	$D_B$	$D_C$	Y
A	B	C		A	B	C				
0	0	0	0	0	1	1	0	1	1	0
0	0	0	1	1	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	1	0	0	1	0	0	1
0	1	0	0	0	1	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	1
0	1	1	0	0	0	1	0	0	1	0
0	1	1	1	0	1	0	0	1	0	1
1	0	0	0	0	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1	1	0

$$T_A = A + \overline{B}X$$

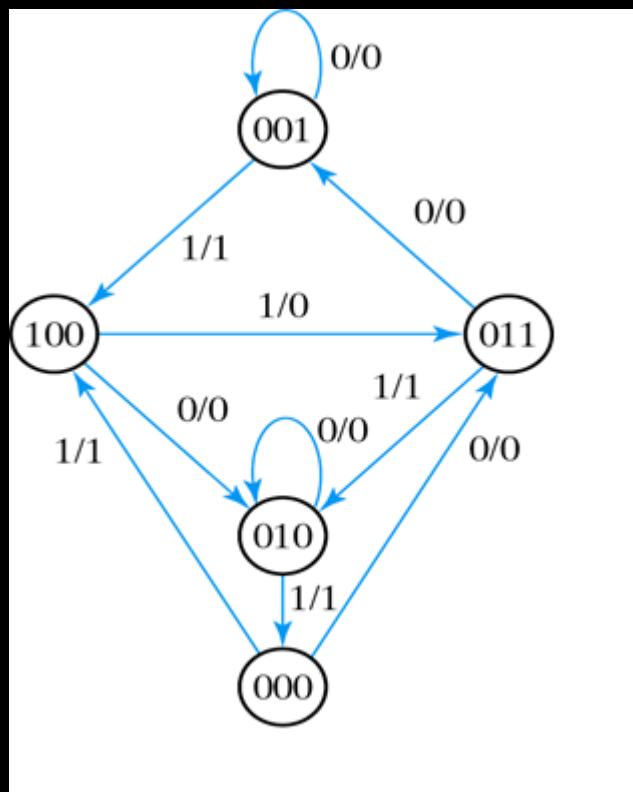
$$T_B = A + \overline{B}C\overline{X}$$

		AB	CX			
		00	01	11	10	
		00	0	1	1	0
		01	0	0	0	0
		11	X	X	X	X
		10	0	0	X	X

		AB	CX			
		00	01	11	10	
		00	1			
		01	1		1	
		11	X	X	X	X
		10	1	1	X	X

		AB	CX			
		00	01	11	10	
		00	1			1
		01				1
		11	X	X	X	X
		10		1	X	X

		AB	CX			
		00	01	11	10	
		00		1	1	
		01		1	1	
		11	X	X	X	X
		10			X	X



**Answer:**

$$D_a = A' B' x$$

$$D_b = C' x' + A + BCx$$

$$D_c = A' B' x' + Cx' + Ax$$

$$Y = A' x$$

Answer:

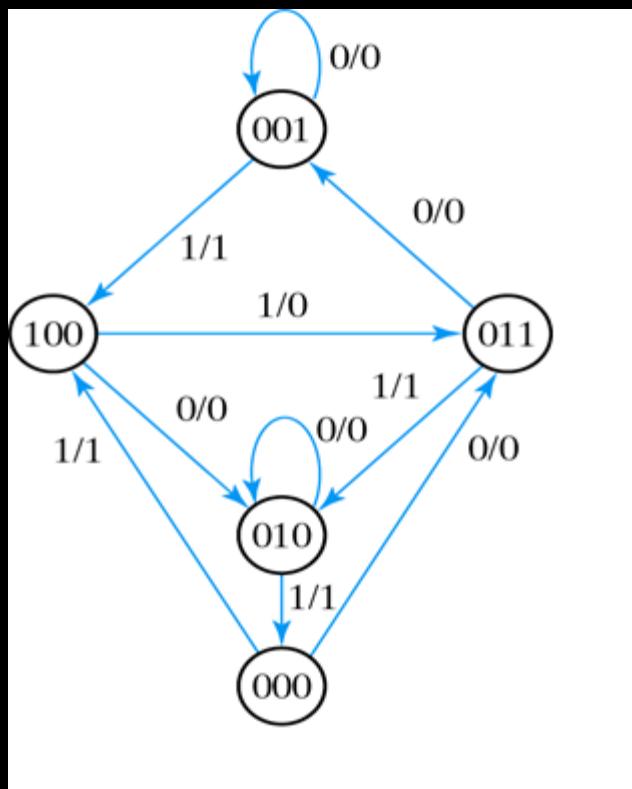
$$D_a = A'B'x$$

$$D_b = C'x' + A + BCx$$

$$D_c = A'B'x' + Cx' + Ax$$

$$Y = A'x$$

The circuit is self correcting



PS			X	NS			T <sub>a</sub>	T <sub>b</sub>	T <sub>c</sub>	Y
A	B	C		A	B	C				
0	0	0	0	0	1	1				0
0	0	0	1	1	0	0				1
0	0	1	0	0	0	1				0
0	0	1	1	1	0	0				1
0	1	0	0	0	1	0				0
0	1	0	1	0	0	0				1
0	1	1	0	0	0	1				0
0	1	1	1	0	1	0				1
1	0	0	0	0	1	0				0
1	0	0	1	0	1	1				0

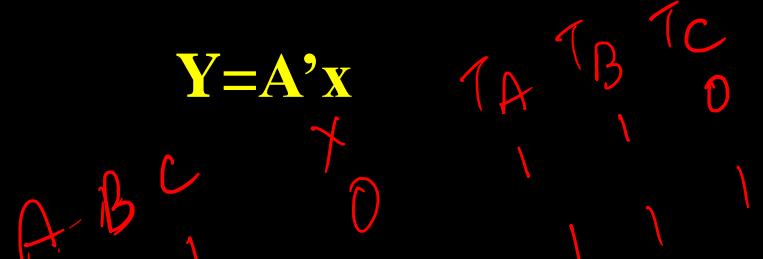
Answer:

$$T_a = A + B'x$$

$$T_b = B'C'x' + A + BCx' + BC'X$$

$$T_c = A'B'C'x' + Cx + Ax$$

$$Y = A'x$$



0+  
0+0  
0+0  
0+1  
0+1

$T_A, T_B, T_C$   
N.S  
A B C  
0 1 2  
0 1 0  
0 1 0  
0 1 1

000  
001  
010  
011  
100

used States

unused States

Self Correcting

# Flip flop Conversions

SR  
D  
JK  
T

*Char-table*      *SR to JK*

	J	K	$Q_t$	$Q_{(t+1)}$	S	R
0	0	0	0	0	x	0
0	0	1	1	0	x	0
0	1	0	0	1	0	x
0	1	1	1	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	0	0	1

*Excitation's table*

	$\bar{J}Q_t$	$\bar{K}Q_t$	$K\bar{Q}_t$	$\bar{K}\bar{Q}_t$
$\bar{J}$	0	x	0	0
$J$	1	x	0	1
$\bar{K}$	1	x	0	1
$K$	0	0	1	0

$S = J Q_t$   
 $R = K \bar{Q}_t$



# Registers

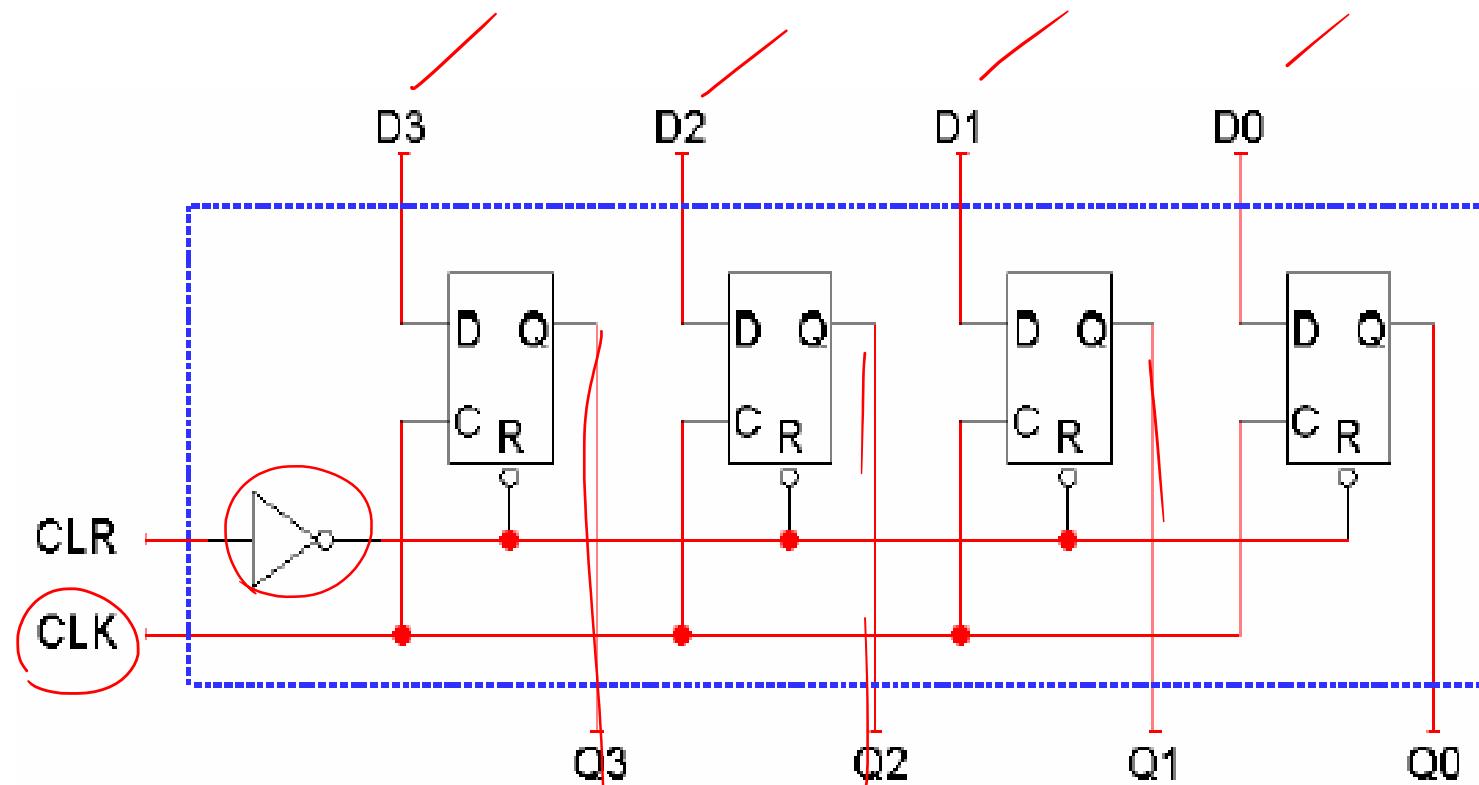
- Register is an extension of a Flip-flop that can store multiple bits. Register consists of group of flip-flops and some combinational gates. Each FF is capable of storing one bit of information. N-bit register has group of n-FFs capable of storing n-bits of binary information. The logic gates determine how the information is transferred into the register.
- Ex: Used as a Temporary storage in Microprocessors

# A basic register

lead

An example: Register constructed using D FFs.

D0-D3 Data inputs; Q0-Q3 data output.





# CS/ECE/EEE/INSTR F215:Digital Design



## Lecture 28: Registers

*Tue, 16 Nov 2021*



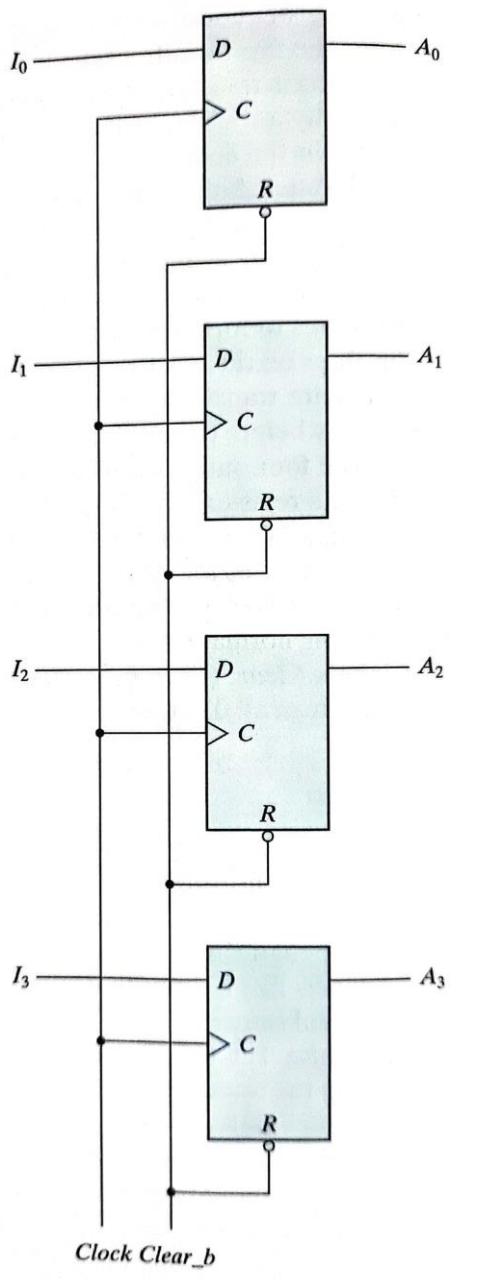
The Capacity to learn is a gift;  
The Ability to learn is a skill;  
The willingness to learn is a choice !

- Brian Herbert

# Registers

---

- Register is an extension of a Flip-flop that can store multiple bits. Register consists of group of flip-flops and some combinational gates. Each FF is capable of storing one bit of information. N-bit register has group of n-FFs capable of storing n-bits of binary information. The logic gates determine how the information is transferred into the register.
- Ex: Used as a Temporary storage element in Microprocessors



Register constructed using four D-FFs

Clock pulse triggers all the FFs

Binary values available at  $I_0I_1I_2I_3$  transferred to register  
and available as  $A_0A_1A_2A_3$

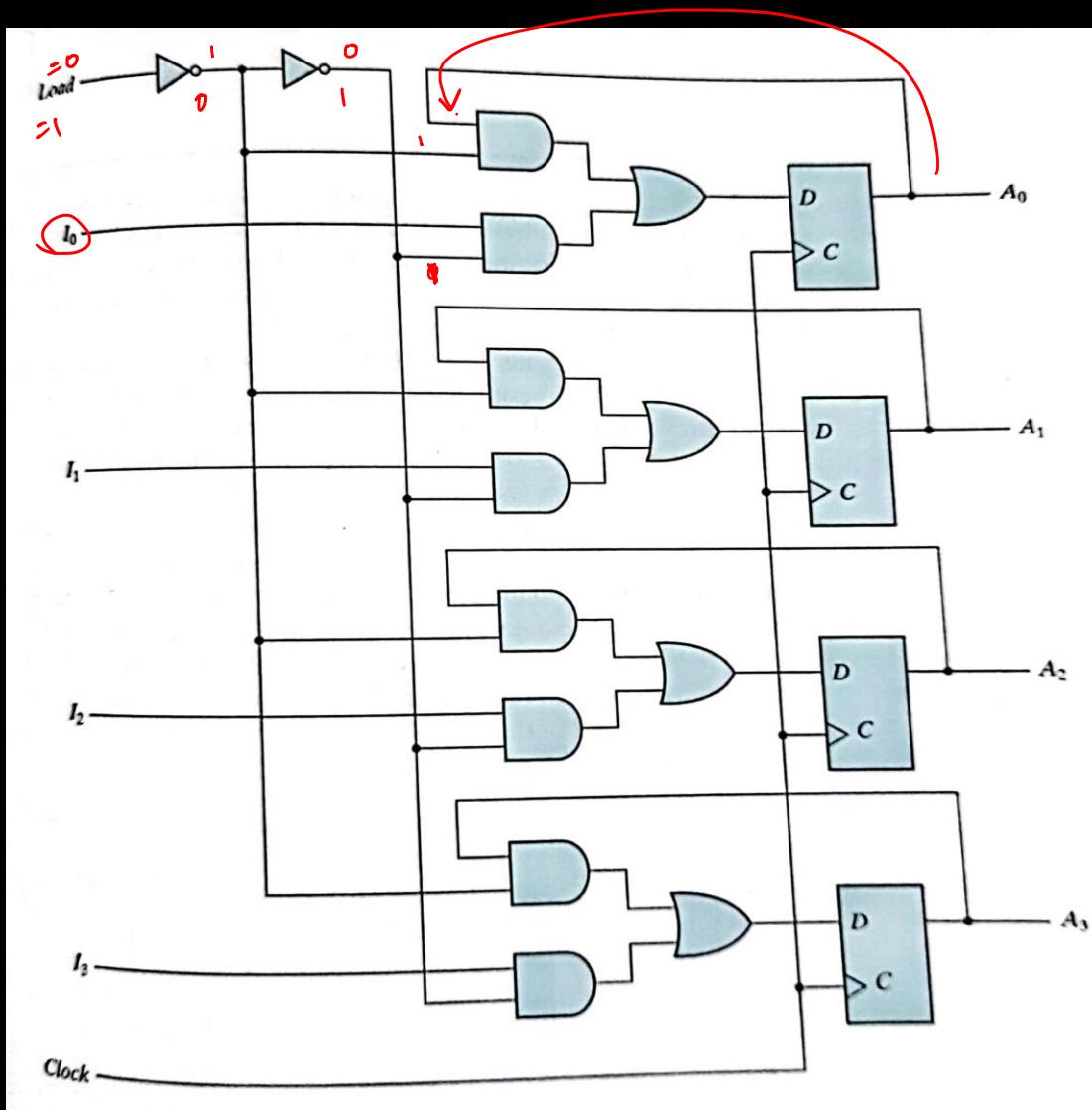
Four outputs can be sampled at any time to obtain the binary information stored in the register.

Clear input clears all the FFs

How can we store the contents of the register for more than one cycle?

Either inputs must be held constant or clock must be inhibited from the circuit by connecting external gates ( this will involve uneven propagation delays and result in clock skew affect synchronization)

Controlling the operation of register by controlling the inputs is better than controlling the clock



**Load = 1** means data at the four ext. i/p transferred to register with the next positive edge of the clock.

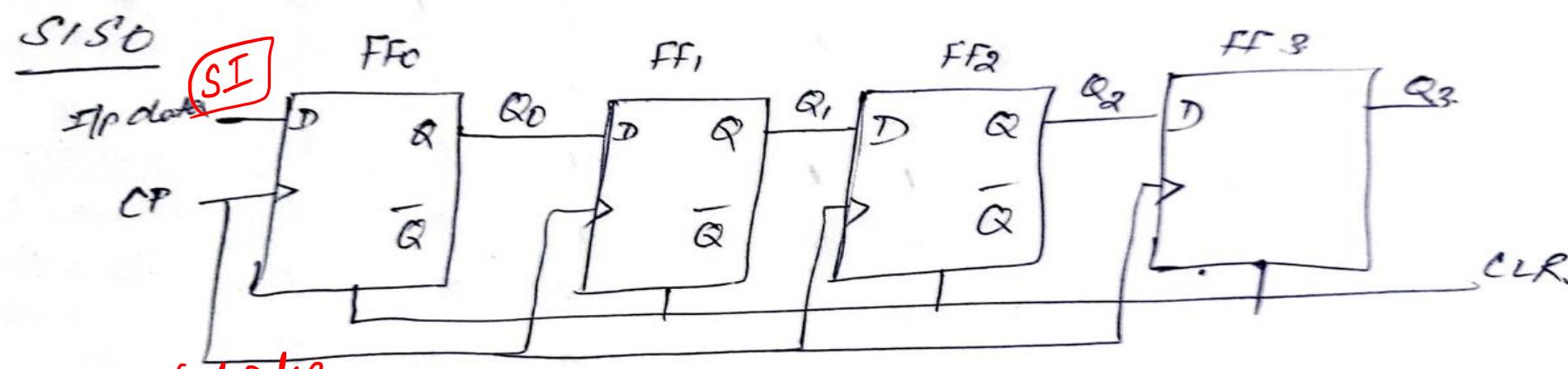
**Load = 0** means outputs of the FFs are connected to their respective inputs.

This implies no change condition leaving the output unchanged.

**Shift register:** A register capable of shifting its binary information in one or both directions is called shift registers. It consists of chain of FFs in cascade with output of one connected to i/p of next. All FFs receive common CP which activate shift from one stage to next.

**Serial Mode-** information altered or manipulated one bit at a time

A **shift register** "shifts" its output once every clock cycle. **SI** is an input that supplies a new bit to shift "into" the register.



101

$$\begin{aligned} Q_0(t+1) &= S \\ Q_1(t+1) &= \underline{Q_0(t)} \\ Q_2(t+1) &= Q_1(t) \\ Q_3(t+1) &= Q_2(t) \end{aligned}$$

Here is one example transition.

Present State	Input	Next State
Q0-Q3	SI	Q0-Q3
0110	1	1011

The current Q3 (0 in this example) will be lost on the next cycle.

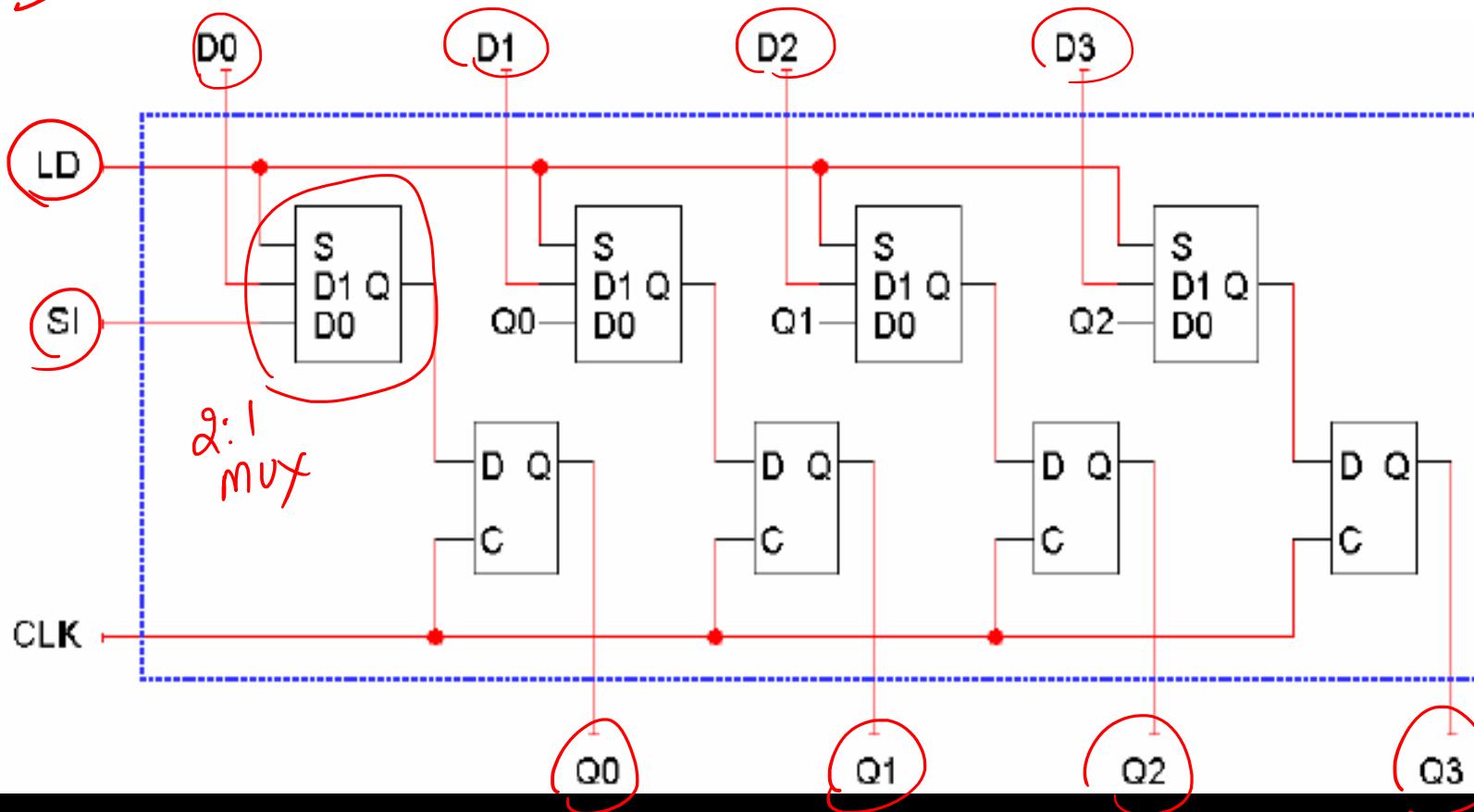
# Applications of shift register transfer – serial data (e.g. USB, UART)

# Shift registers with parallel load

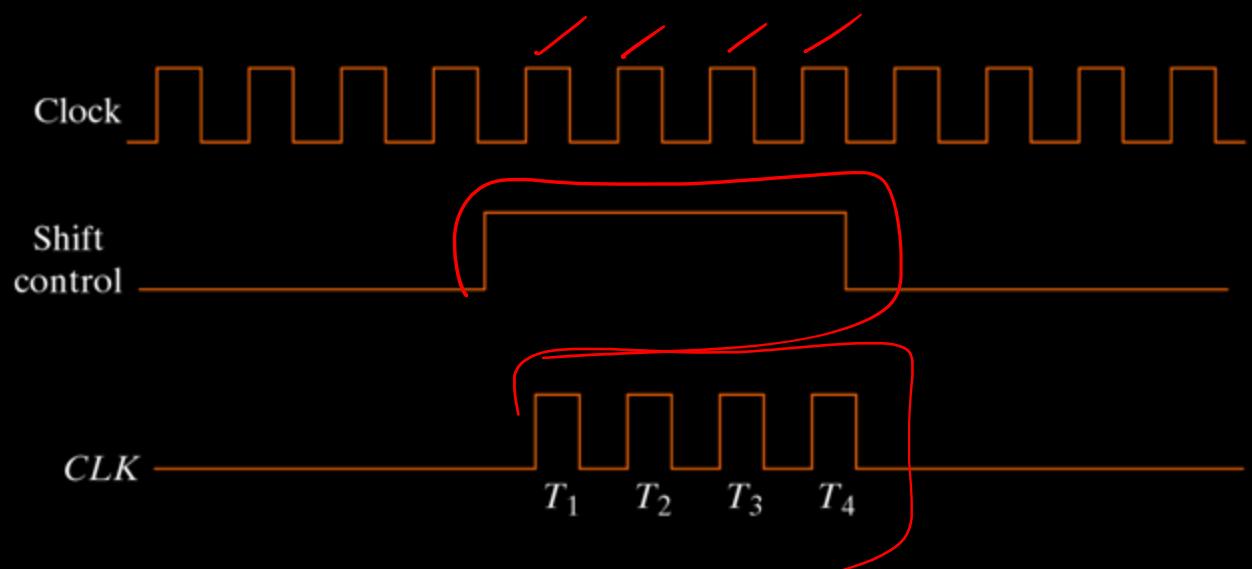
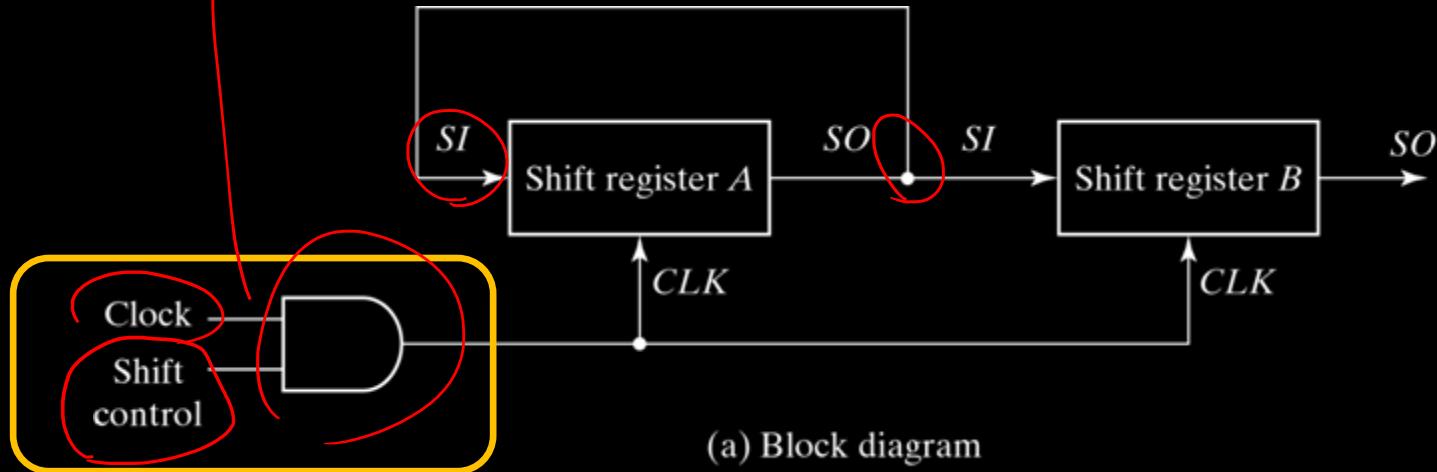
- We can add a parallel load operation, just as we did for regular registers.

*serial* When LD = 0 the flip-flop inputs will be SIQ0Q1Q2, so the register will shift on the next positive clock edge.

*Parallel* When LD = 1, the flip-flop inputs are D0-D3, and a new value is loaded into the register on the next positive clock edge.

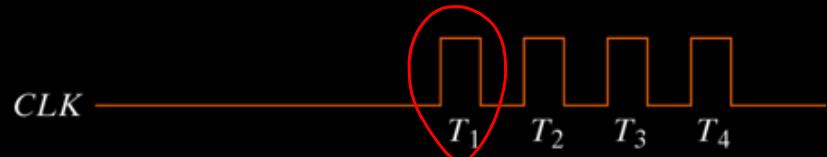
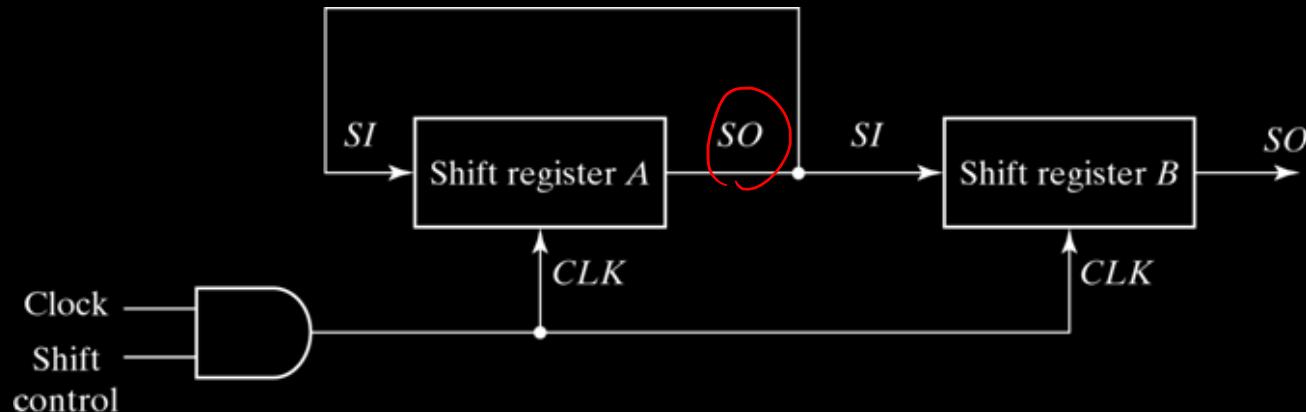


# Serial Transfer

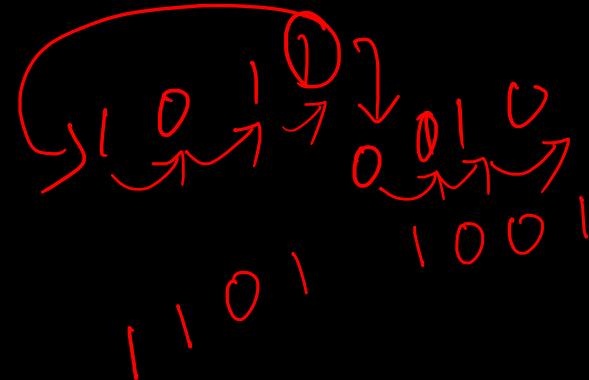


# Serial Transfer

Operation assuming 4-bit shift register



Timing pulse	Shift Register A	Shift Register B
Initial value	1 0 1 1	0 0 1 0
After T <sub>1</sub>	1 1 0 1	1 0 0 1
After T <sub>2</sub>	1 1 1 0	1 1 0 0
After T <sub>3</sub>	0 1 1 1	0 1 1 0
After T <sub>4</sub>	1 0 1 1	1 0 1 1



# Universal Shift register

## Capabilities of Universal shift register

Clear signal to clear the register to 0

Clock input to synchronize

{ Shift-right }  
Shift-left }

Bidirectional shift Register

Parallel load

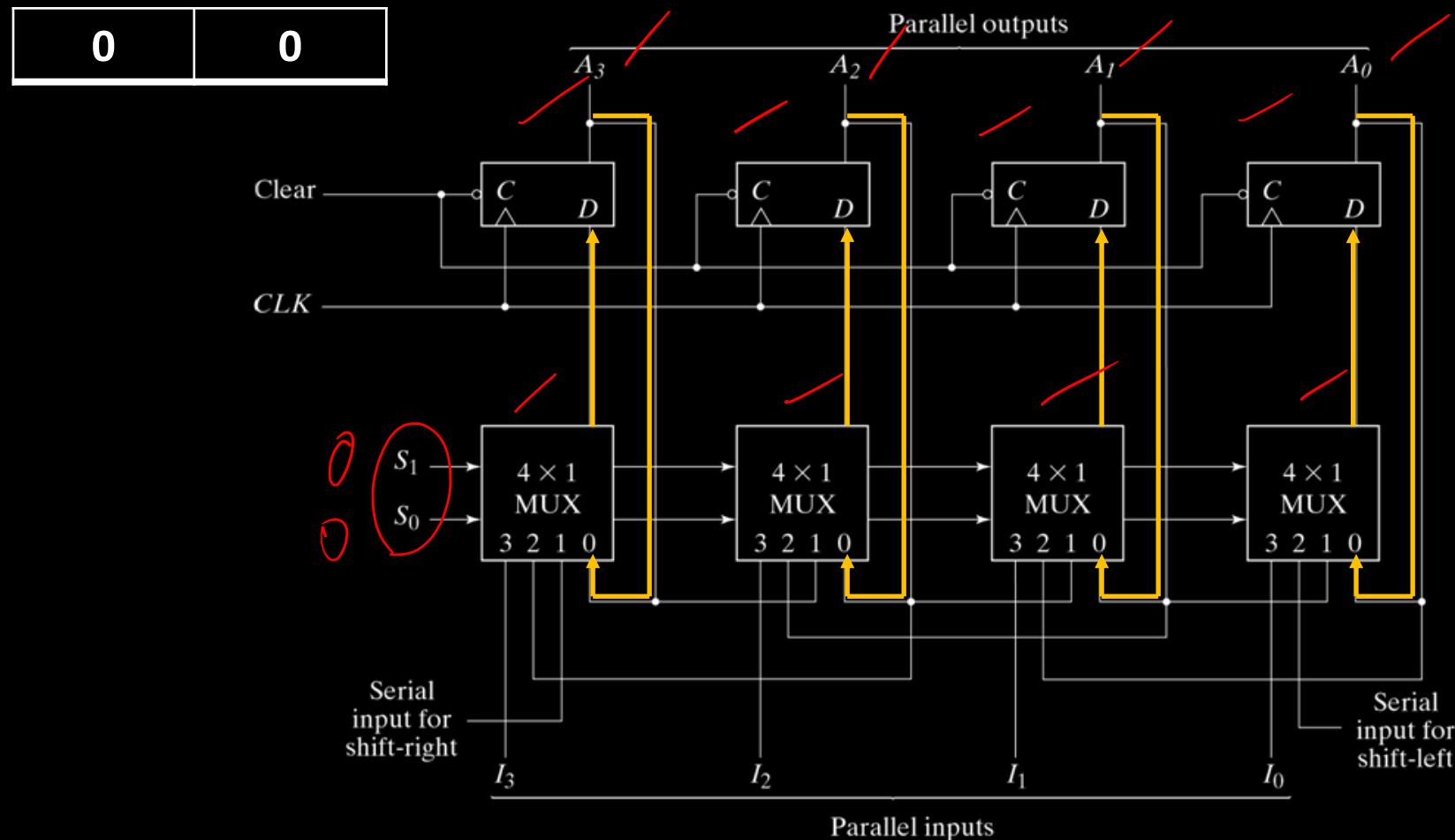
Parallel out

Control state to leave information in register unchanged

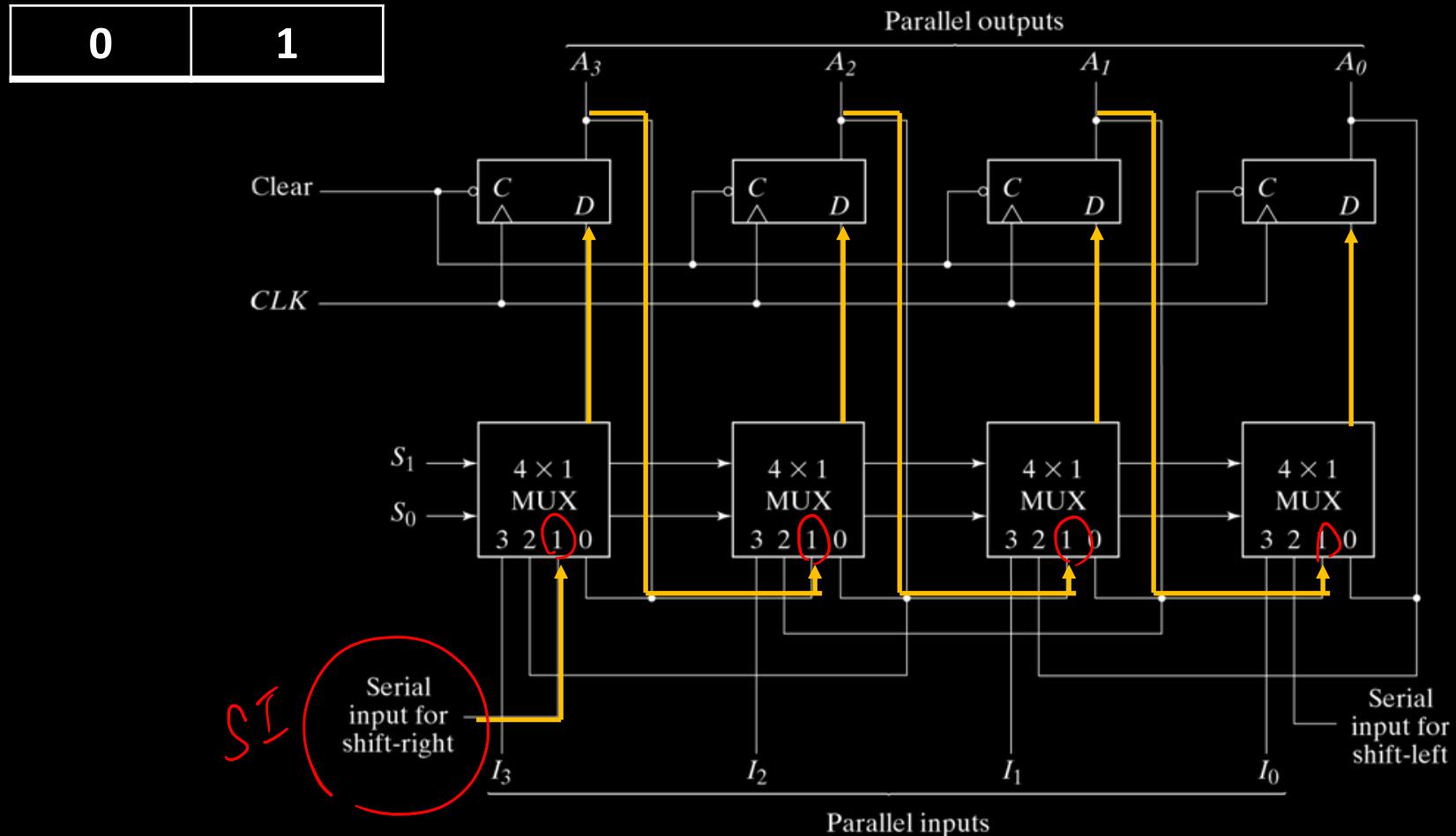
# Universal Shift register

Mode control		Register operation
$S_1$	$S_0$	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

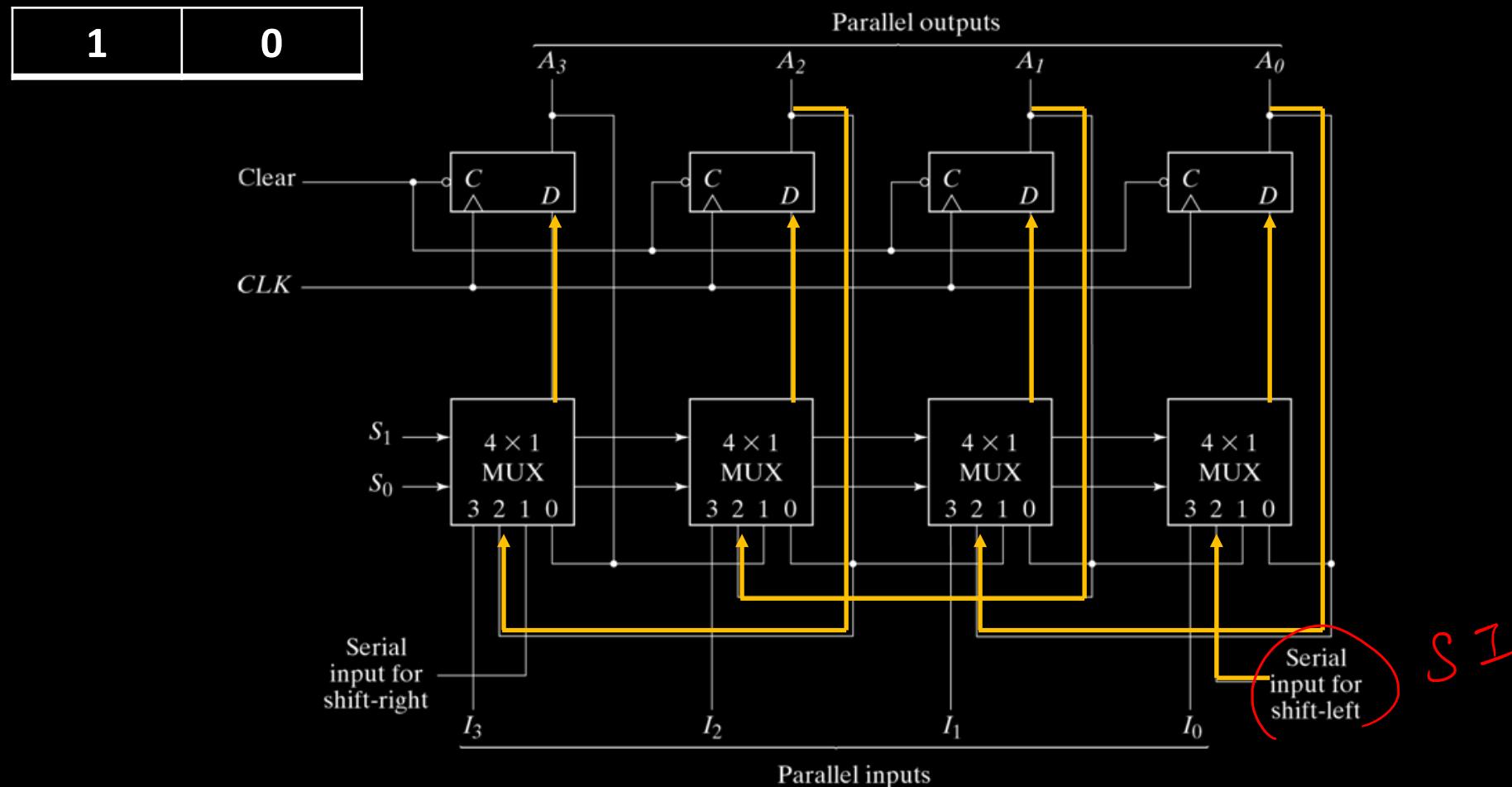
# Universal Shift register



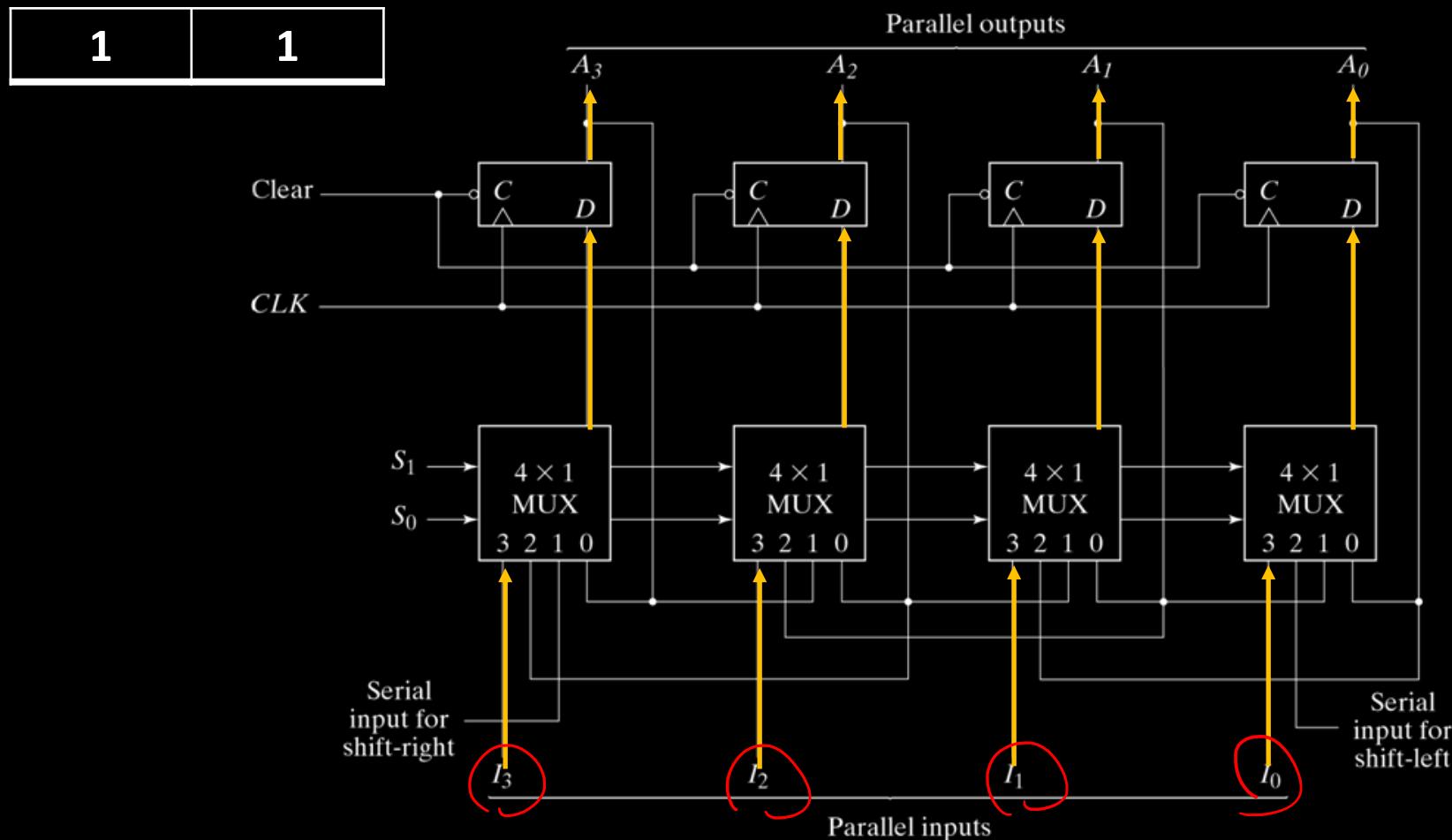
# Universal Shift register



# Universal Shift register

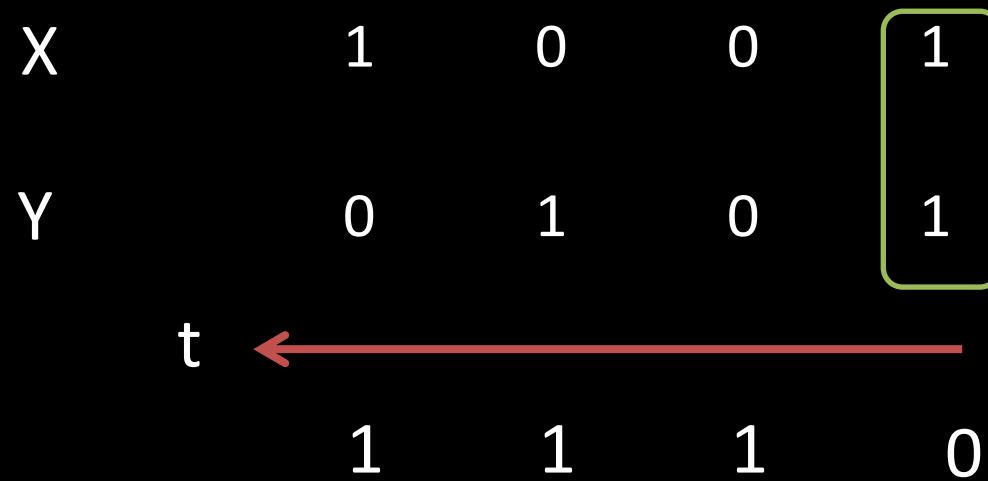


# Universal Shift register

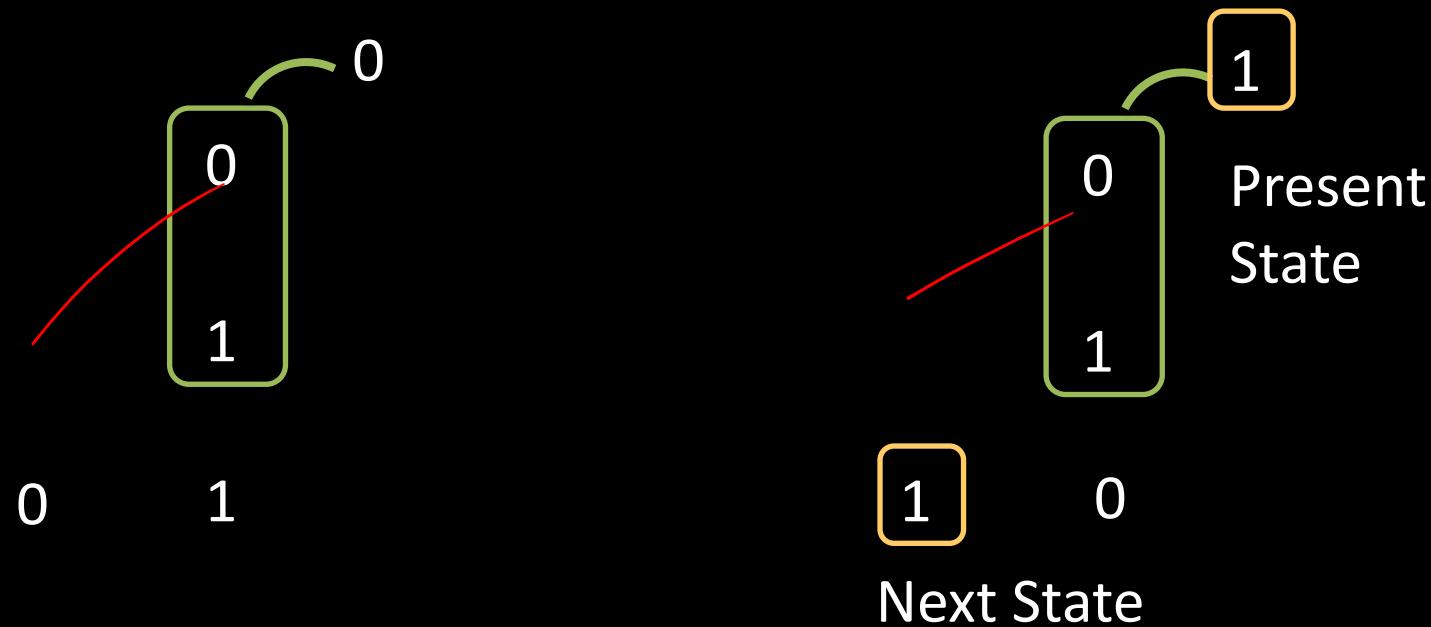


## Serial Adder

bit by bit



# Serial Adder



Define two states S0 and S1

S0 indicates carry = 0

S1 indicates carry = 1

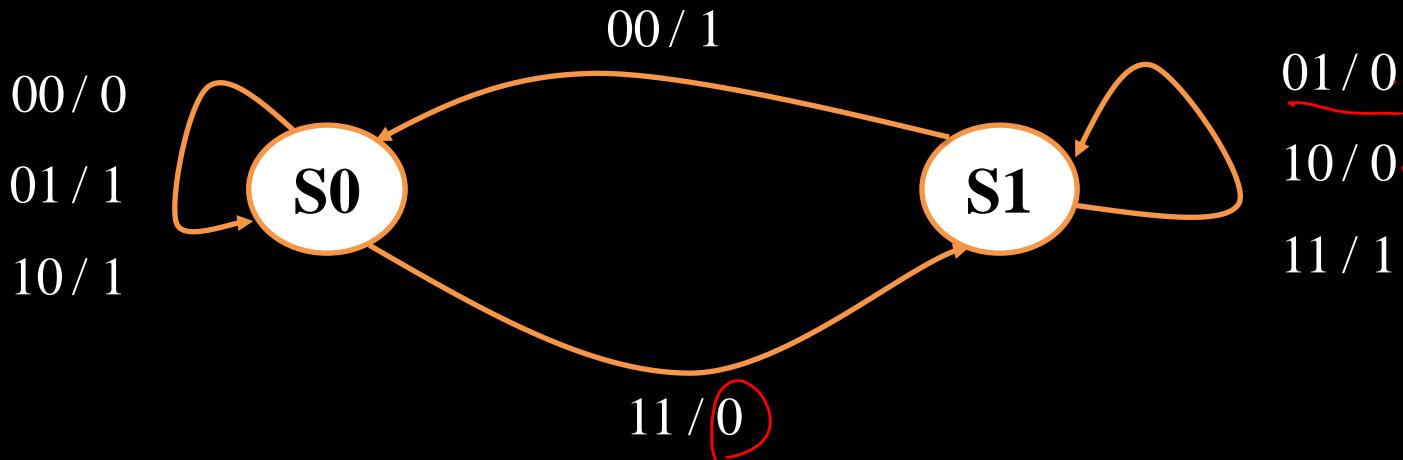
# Serial Adder

Define two states  $S_0$  and  $S_1$

$S_0$  indicates carry = 0

$S_1$  indicates carry = 1

X and Y indicates inputs, Sum is output



$S_0 \rightarrow 0$   
 $S_1 \rightarrow 1$

## Serial Adder

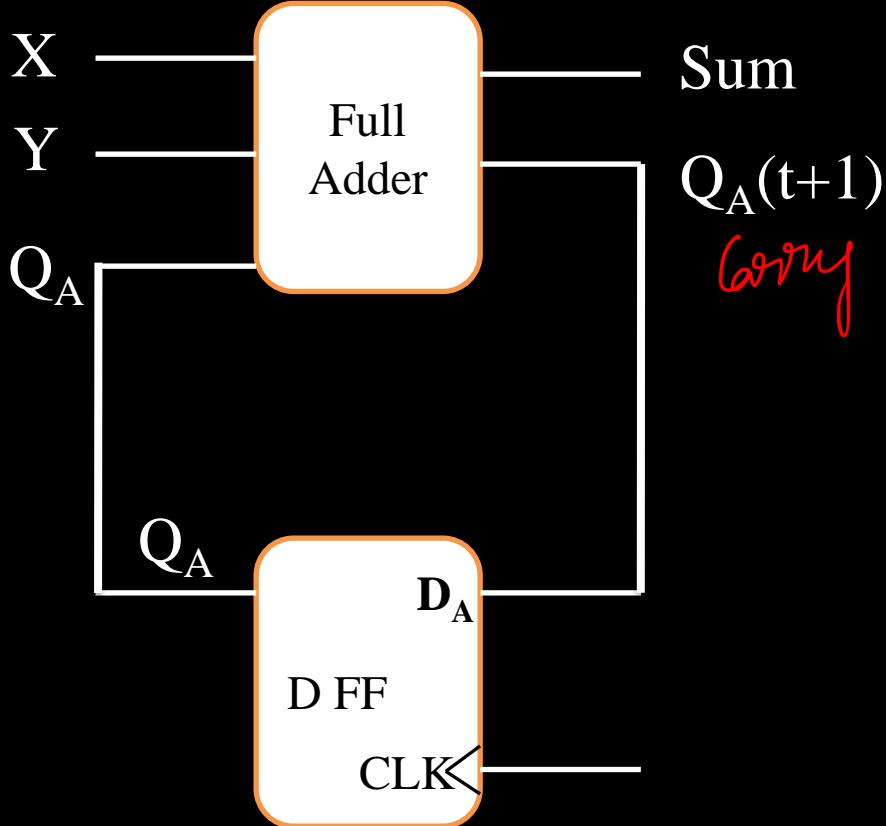
Present State				Next State		Out
	$Q_A$	X	Y		$Q_{A(t+1)}$	Sum
$S_0 \rightarrow 0$	0	0	0	$S_0$	0	0
$S_0$	0	0	1	$S_0$	0	1
$S_0$	0	1	0	$S_0$	0	1
$S_0$	0	1	1	$S_1$	1	0
$S_1 \rightarrow 1$	0	0	0	$S_0$	0	1
$S_1$	1	0	1	$S_1$	1	0
$S_1$	1	1	0	$S_1$	1	0
$S_1$	1	1	1	$S_1$	1	1

*carry expression of full adder*

$$Q_A(t+1) = Q_A X + XY + Q_A Y$$

$$\text{Sum} = \underline{Q_A \oplus X \oplus Y}$$

# Serial Adder

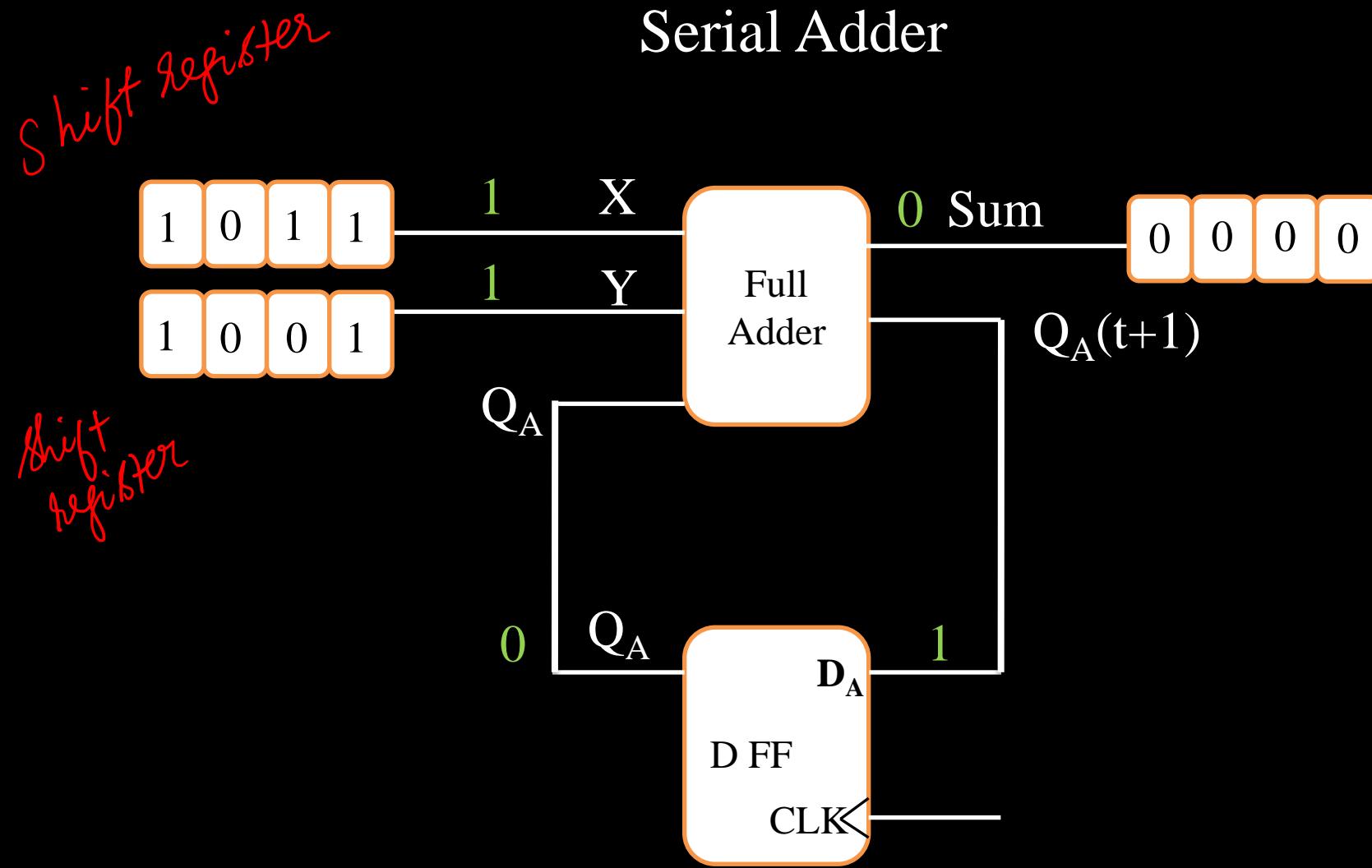


$$Q_A(t+1) = Q_A X + XY + Q_A Y$$

$$\text{Sum} = Q_A \oplus X \oplus Y$$

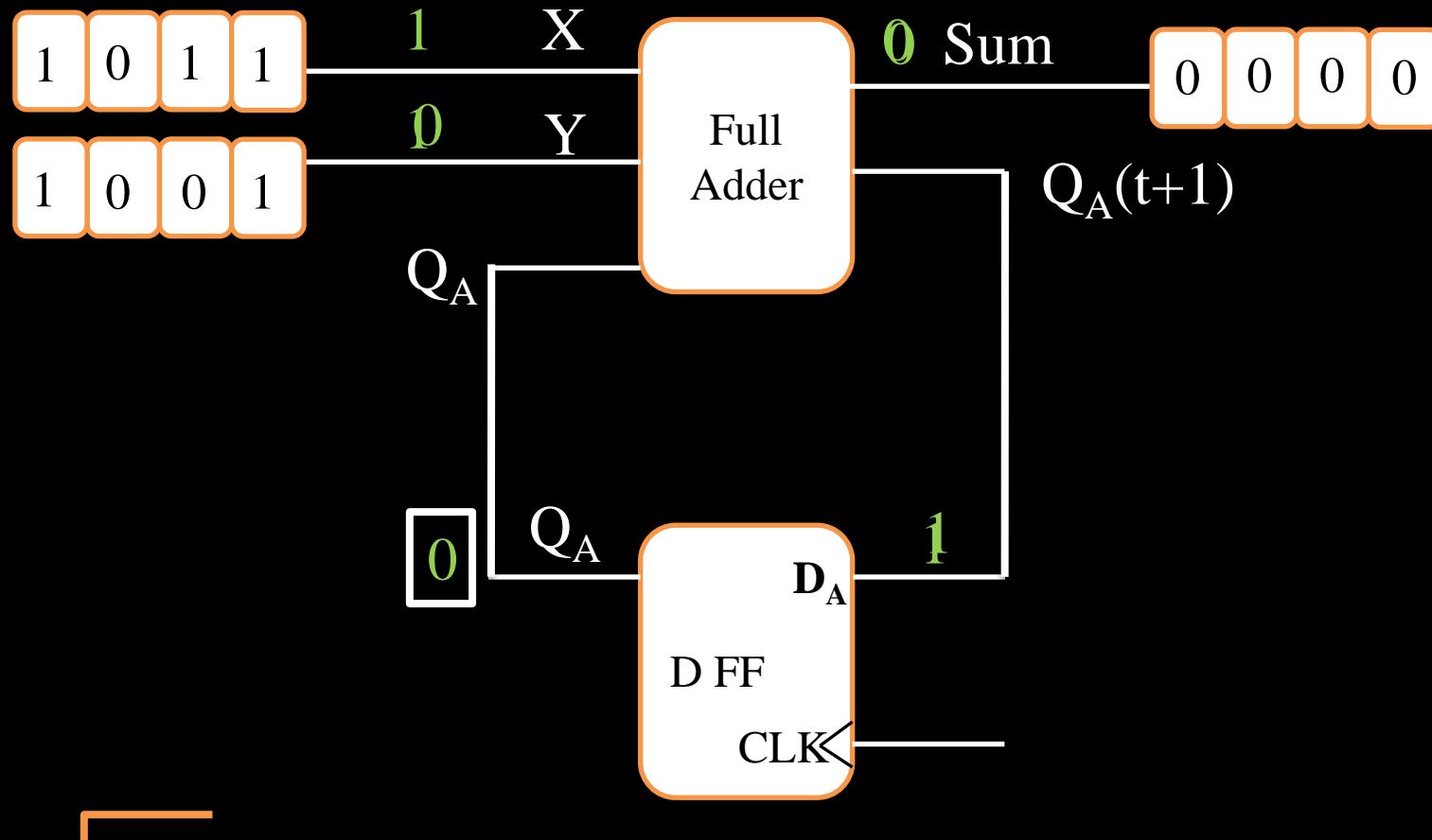
How to control X and Y ?

# Serial Adder

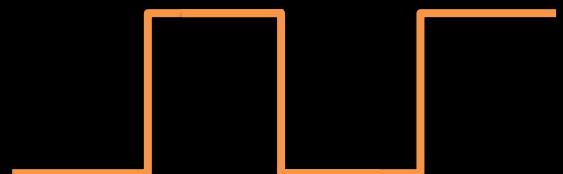
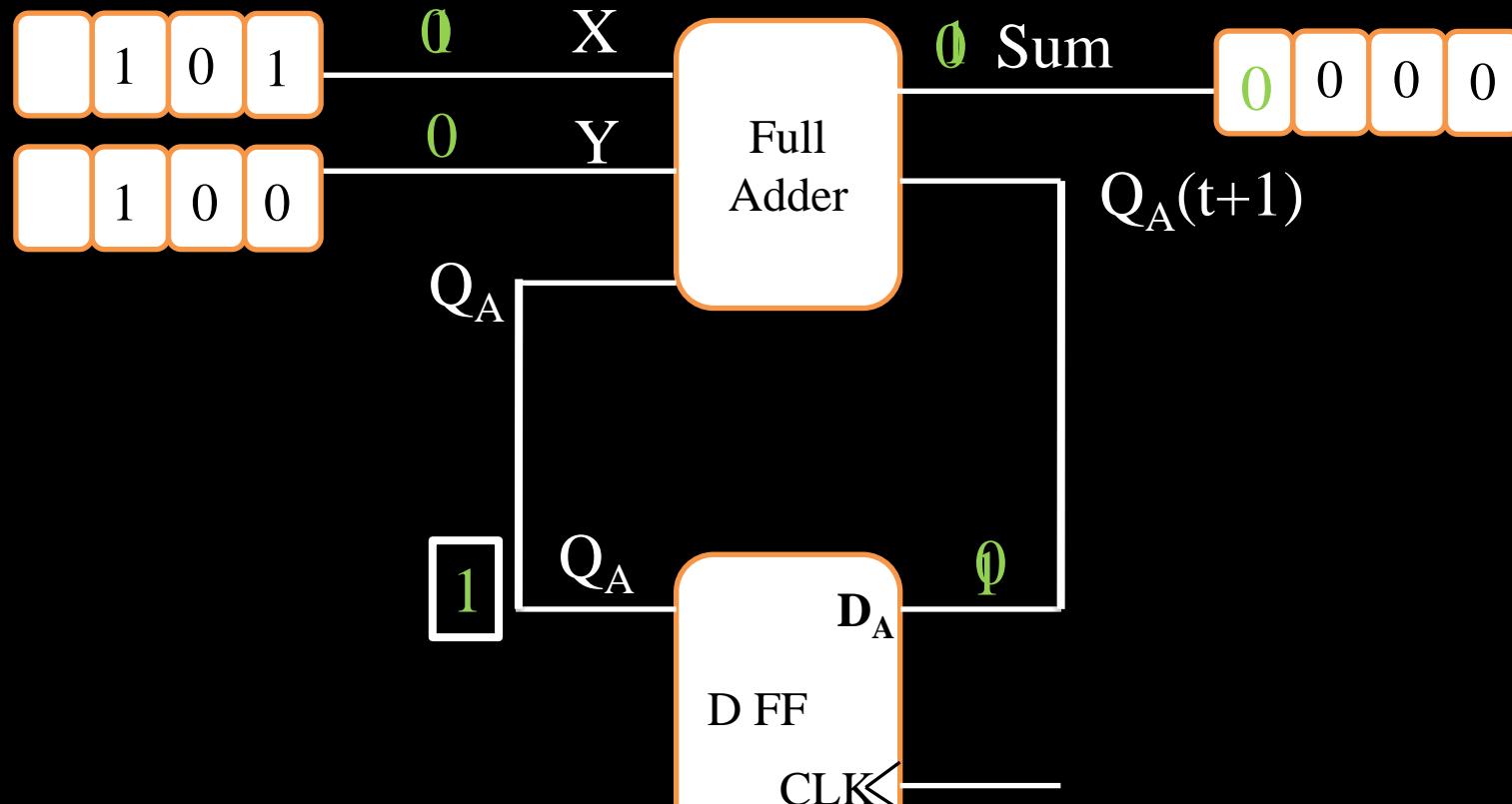


Before any clock pulses

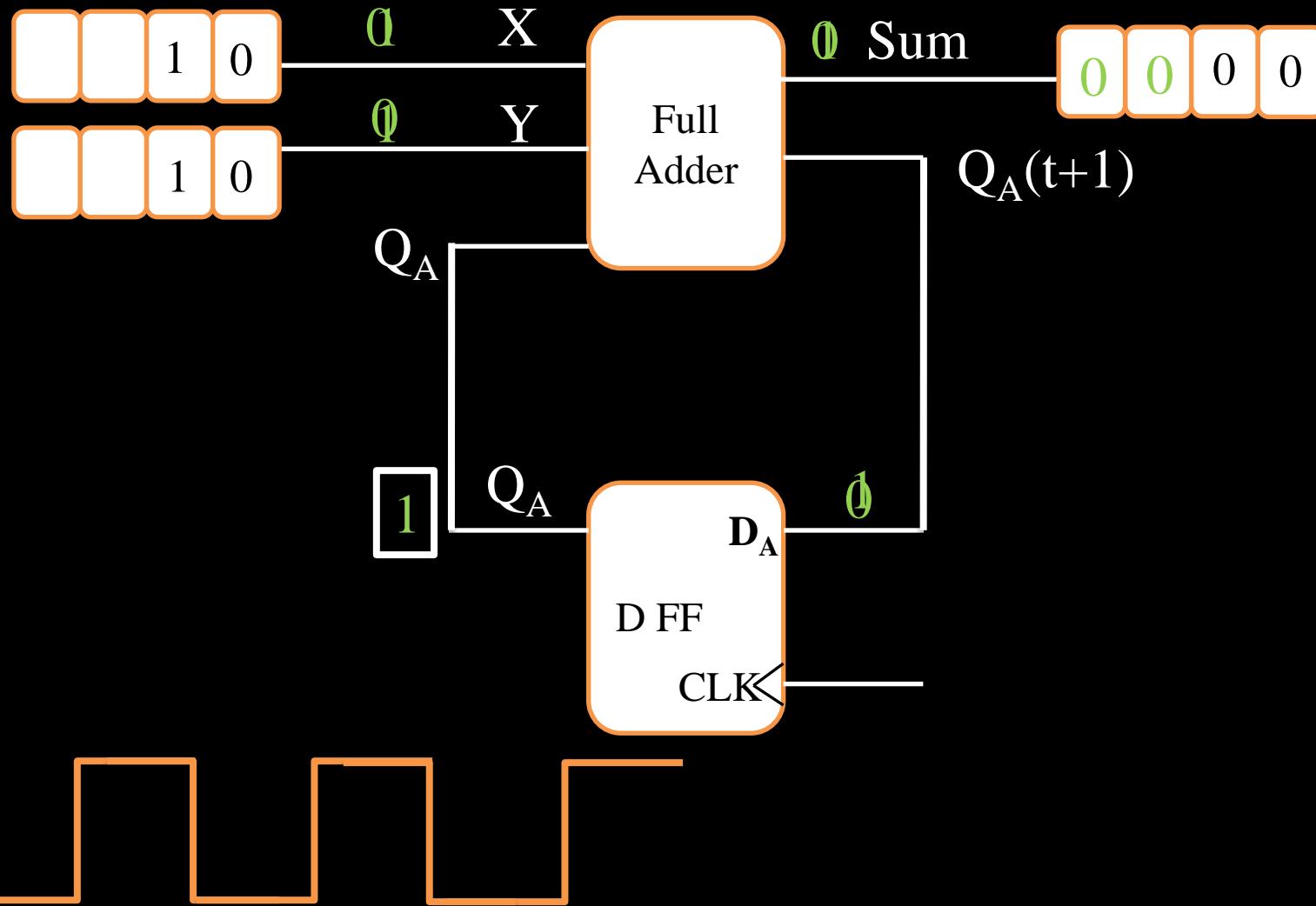
# Serial Adder



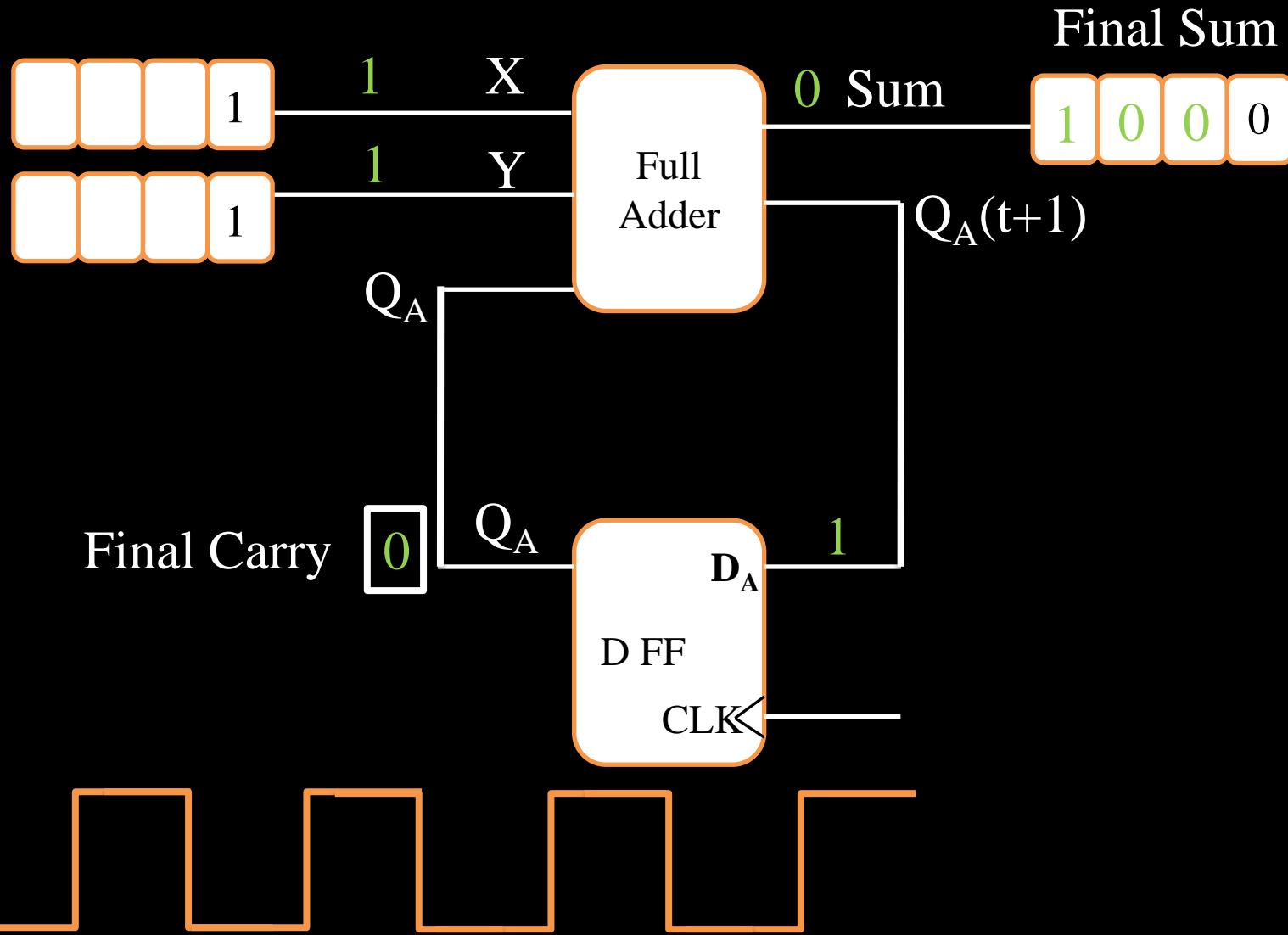
# Serial Adder



# Serial Adder



# Serial Adder



# Serial Adder

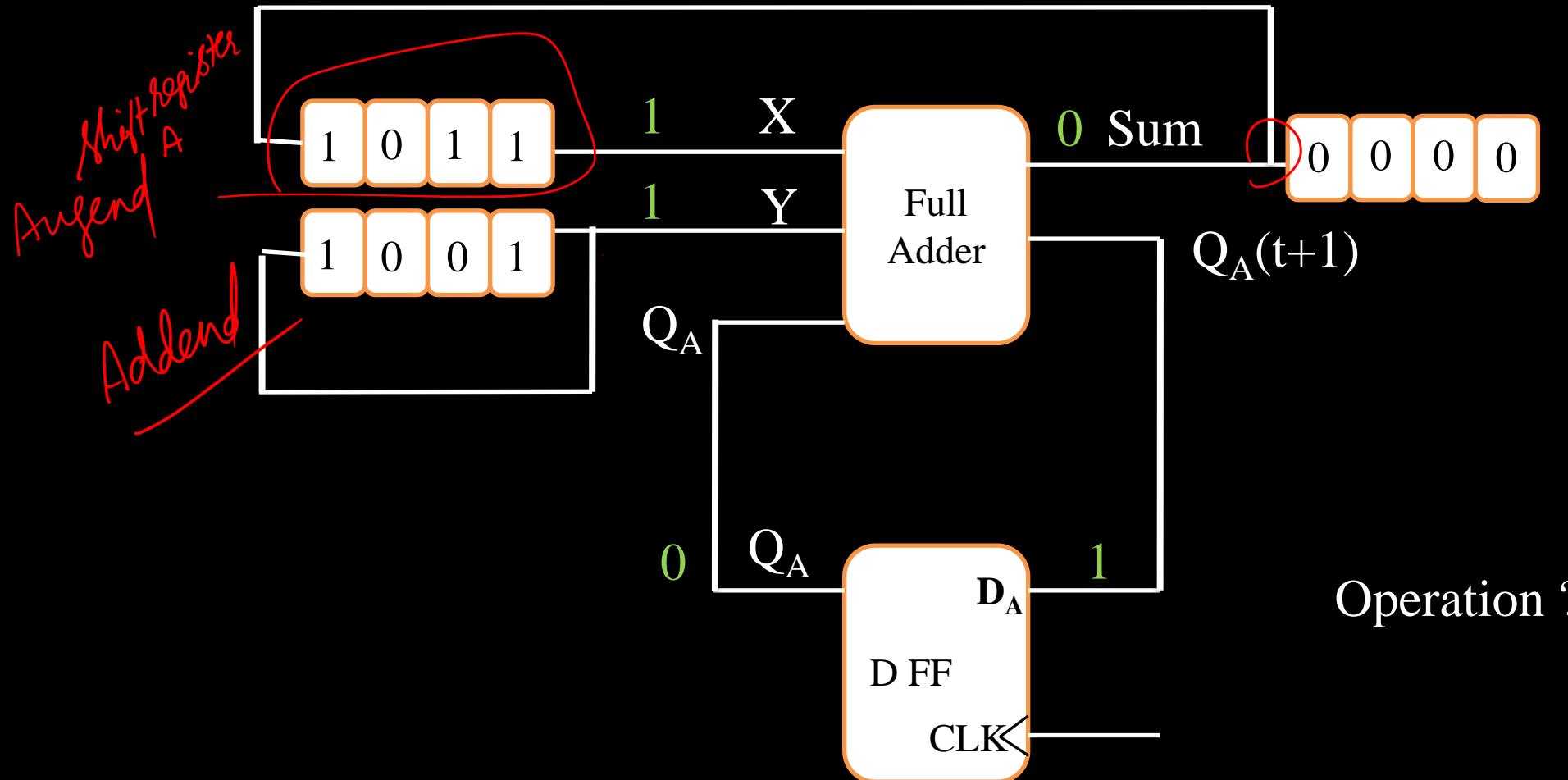
A 1-bit full adder is sufficient for n-bit addition

N-clock cycles for N-bit addition

Complex control circuitry

More memory elements required

# Serial Adder



Shift registers will have additional control signals like load, shift control



# CS/ECE/EEE/INSTR F215:Digital Design



**BITS Pilani**  
Hyderabad Campus



## Lecture 29: *Counters*

*Thu, 18 Nov 2021*

**Dr. R. N. Ponnalagu, EEE**

*Desire for success > Fear for Failure  
is the formula for success*

# Counters

- A sequential circuit that goes through **prescribed sequence of states** upon application of input pulses.
- Input pulses may be clock pulses.
- Sequence of states may follow binary sequence or any other sequence of states.
- A counter that follows binary sequence – **Binary Counter**
- An n-bit binary counter has n-FFs and can count in binary from 0 through  $2^n - 1$ .
- Types of counters: Ripple or Asynchronous counters and Synchronous counters

Ring Counter, Johnson Counter – **Other counters**

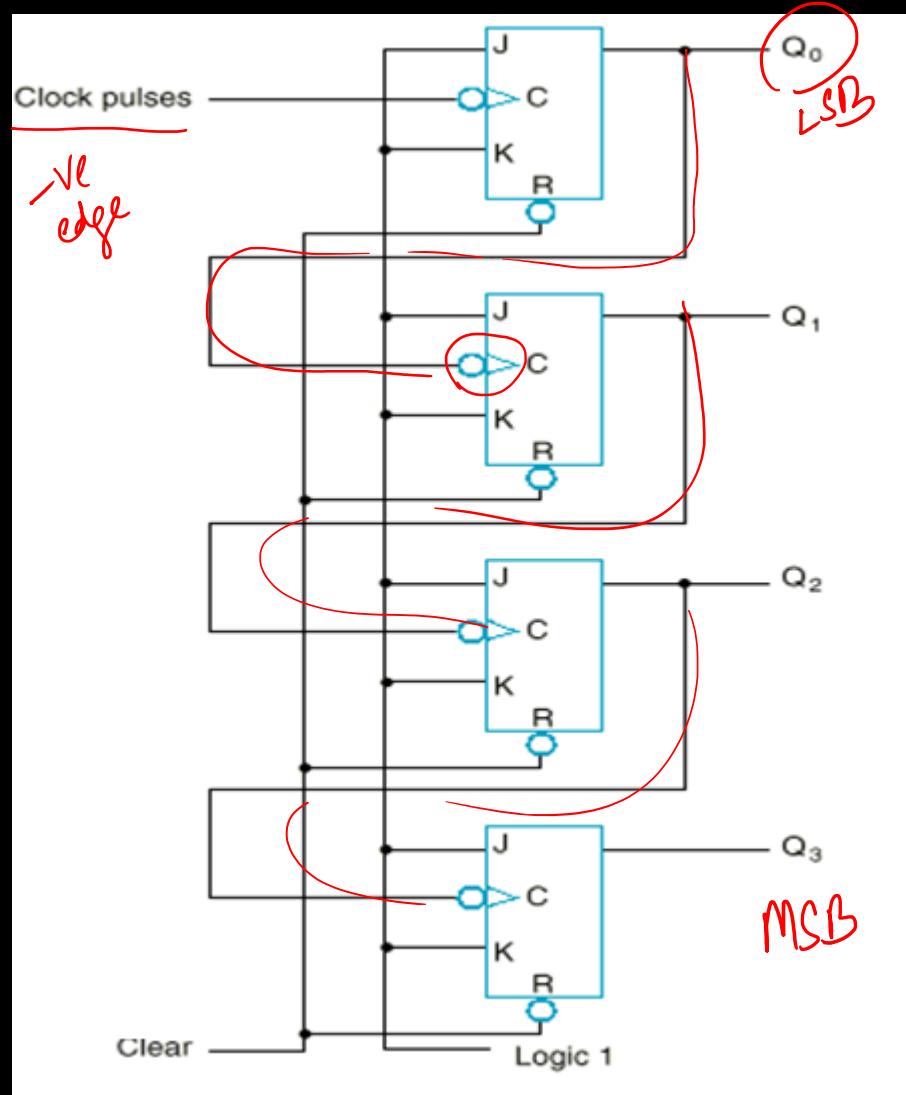
Ripple Counter – Asynchronous Counters

The FF output transition serves as a source for triggering other FFs. No common clock.

**Synchronous Counters**

All FFs receive the common clock pulse, and the change of state is determined from the present state.

# Ripple Counter – Asynchronous Counters



In Asynchronous counter, ext. clock pulse is applied only to the flip flop (LSB).

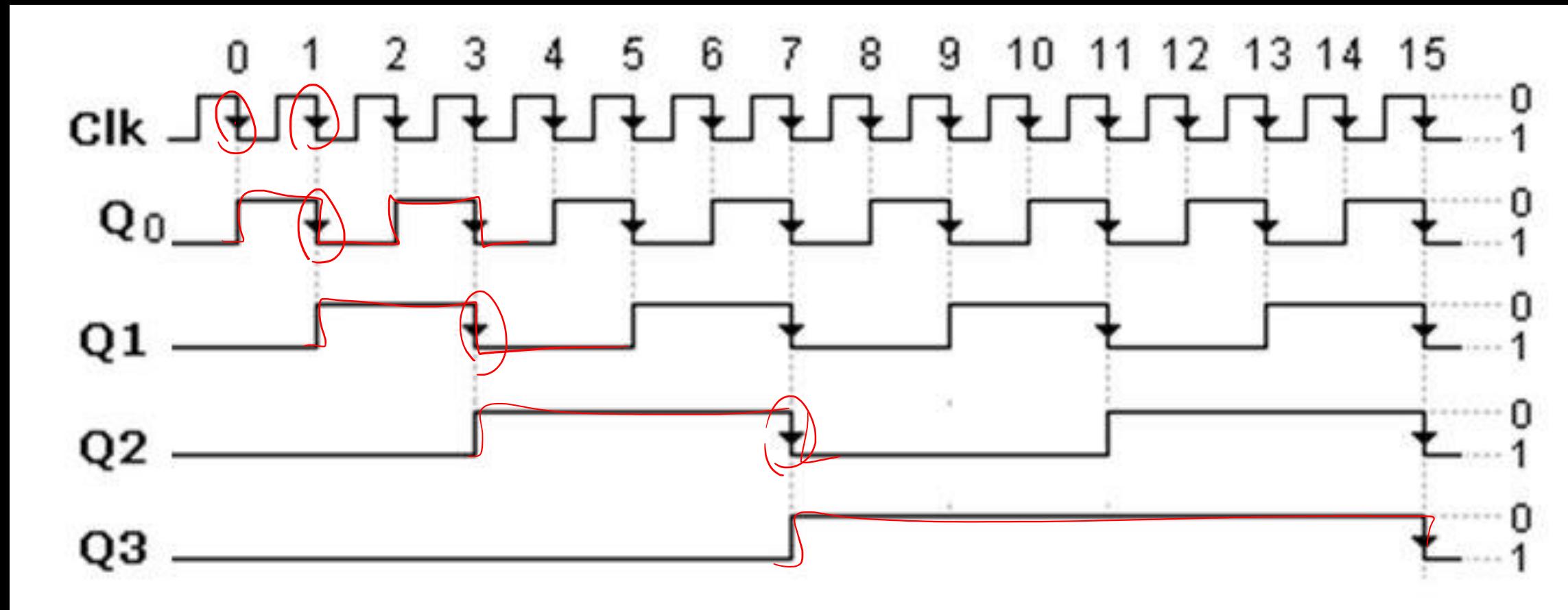
Instead of the ext. clock pulse, the output of first flip-flop acts as a clock pulse to the next flip flop, whose output is used as a clock to the next in line flip-flop and so on.

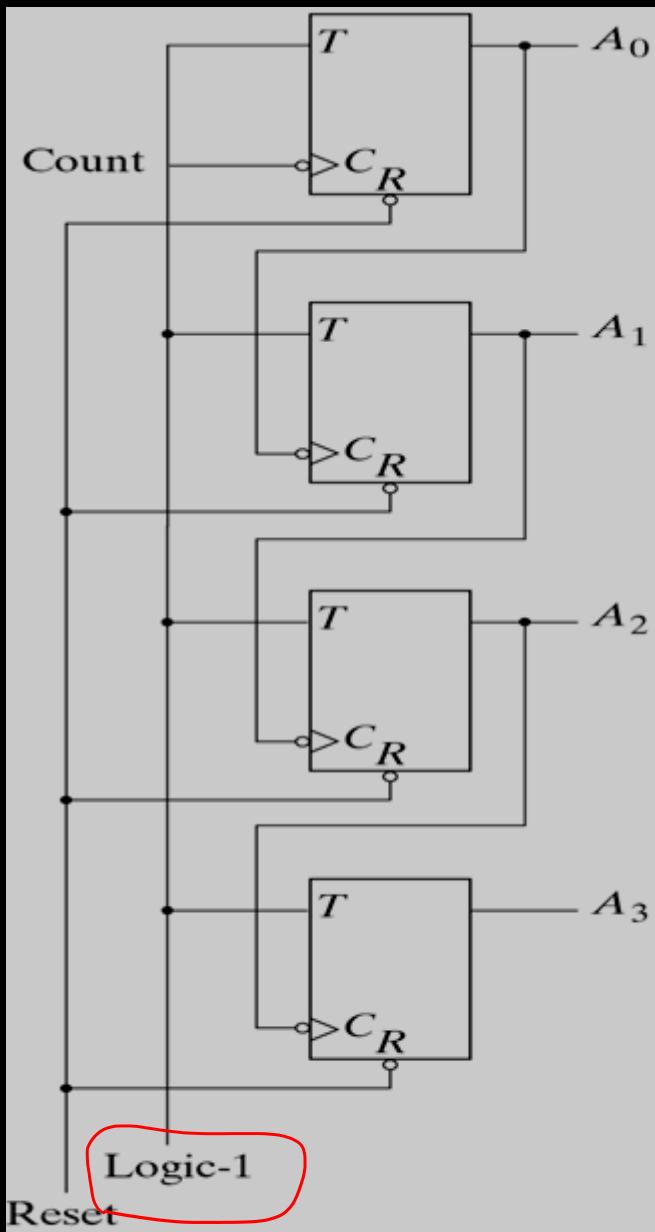
- The output of each FF is connected to the Clock input of the next FF in sequence.
- The FF holding the least significant bit receives the incoming clock pulses.
- The J and K inputs of all FFs are connected to a permanent logic 1.

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

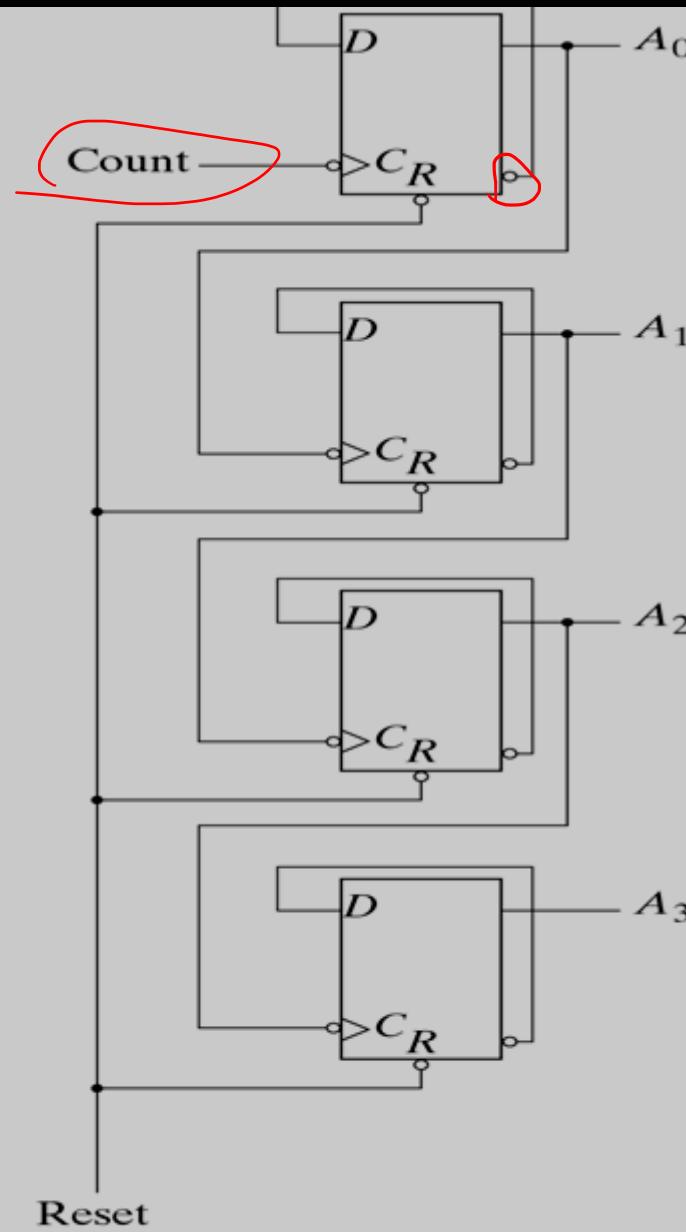
Upward Counting Sequence			
$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

# Timing diagram





(a) With T flip-flops



(b) With D flip-flops

## Design a Binary Down Counter

Approach:

*Preset  $\rightarrow 1111$*

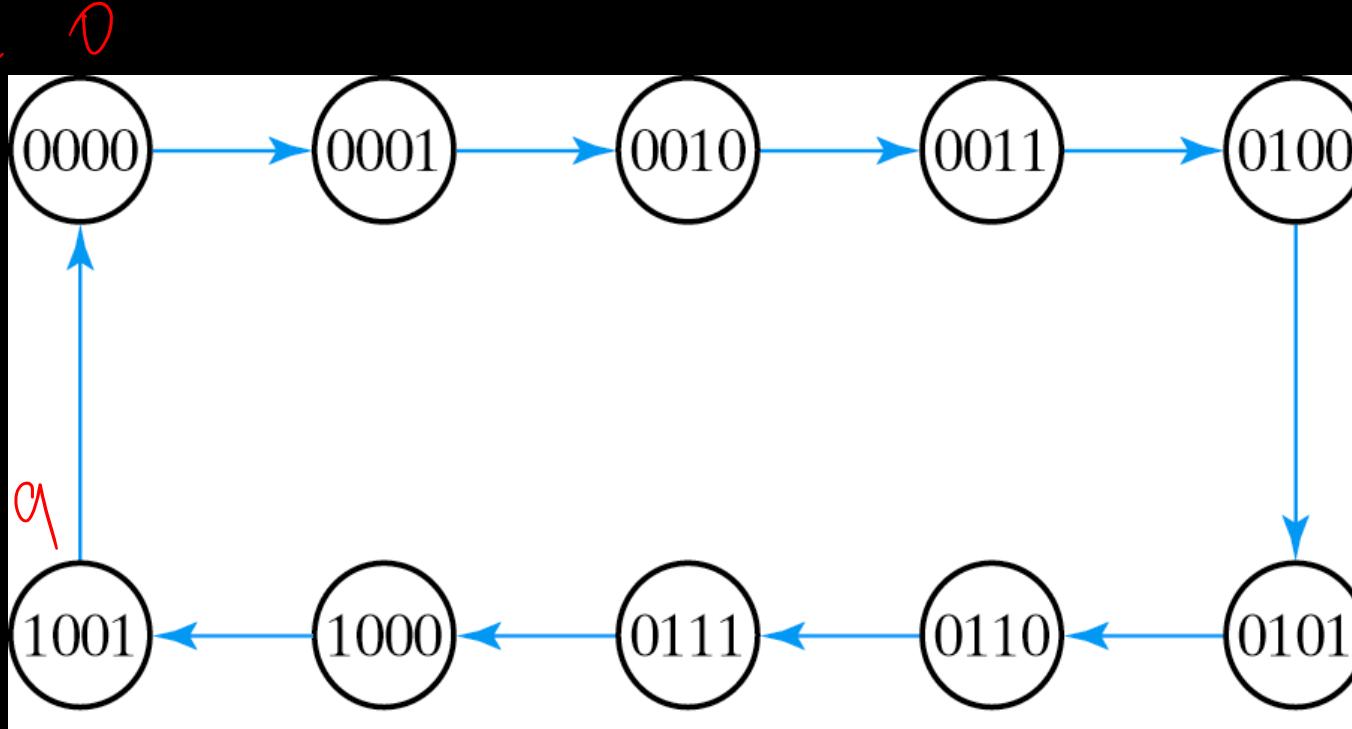
Use direct Set (S) signals (PRESET) instead of direct Reset (R), in order to start at 1111.

} Method 1: Change edge-triggering to positive

Method 2: Connect the complement output  $\bar{Q}_i$  of each FF to the C (CLOCK) input of the next FF in the sequence

*Decade Counter*

⑩

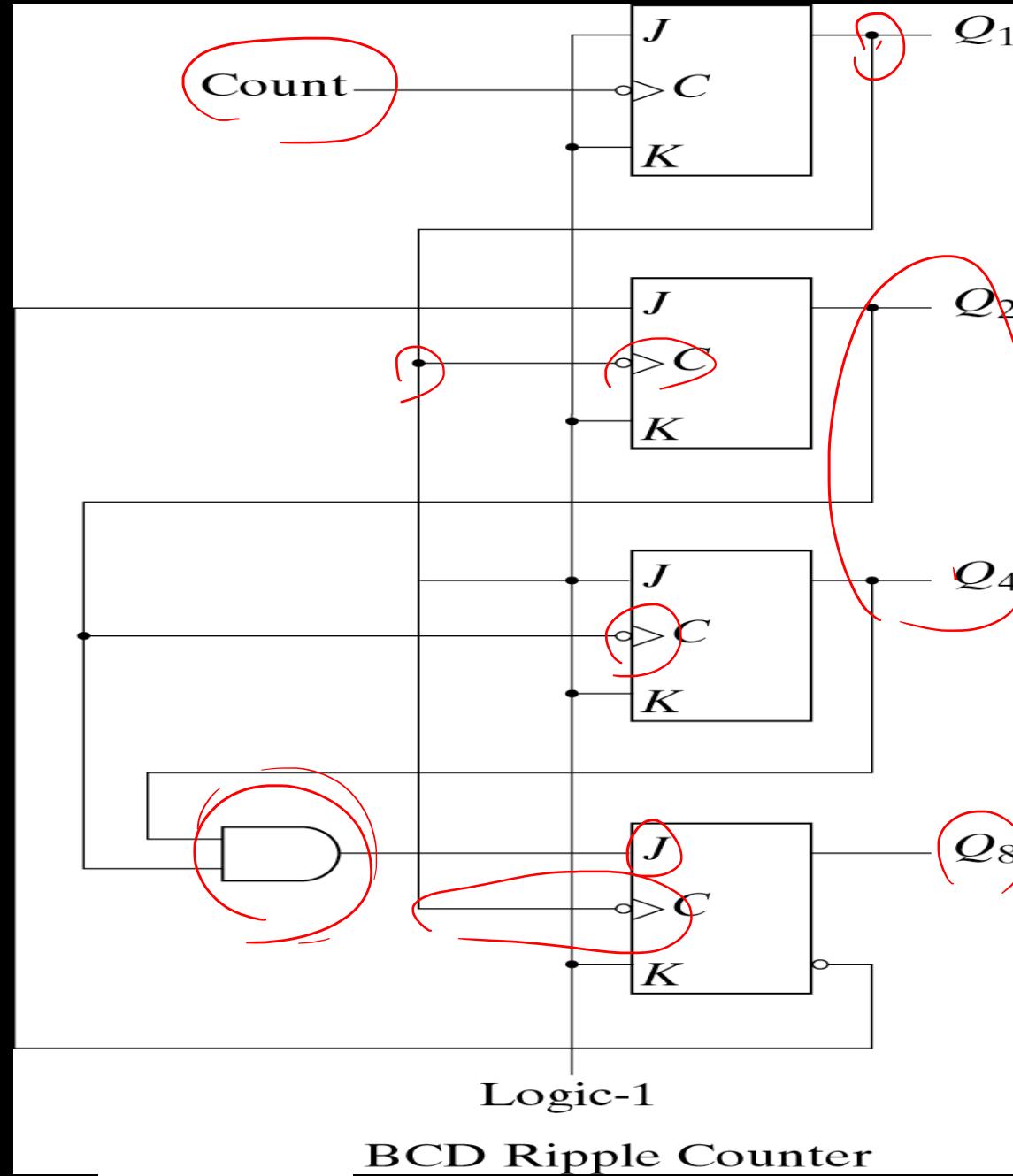


State Diagram of a Decimal BCD-Counter

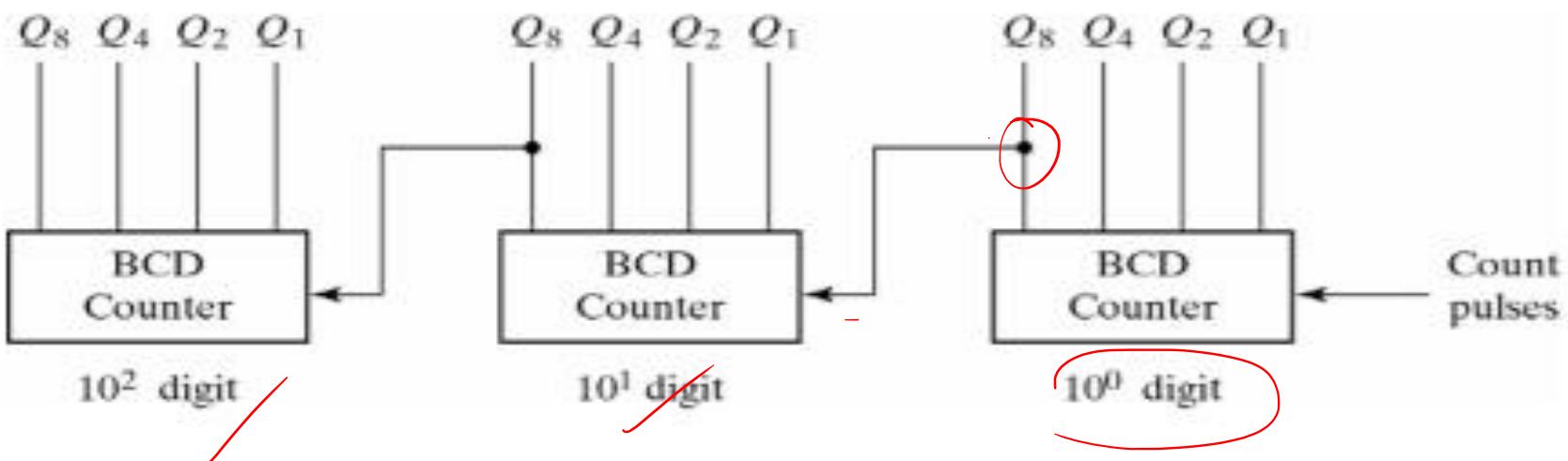
- Q1 complemented on every clock edge
- Q2 complemented when Q1 goes from 1 to 0. as long as  $Q8=0$ , when  $Q8=1$ , Q2 remains at 0 ✓
- Q4 complemented when Q2 goes from 1 to 0 ✓
- Q8 cleared (remains at zero) as long as  $Q4$  or  $Q2$  is 0  
and  $Q8$  complemented when  $Q4Q2 = 11$  and Q1 goes from 1 to 0.

0 0 0 0  
1 1 1 1

Q8	Q4	Q2	Q1
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
0	0	0	0



A multiple decade counter can be constructed by connecting BCD counters in cascade. A three-decade counter is shown below:



The inputs to the second and third decades come from  $Q_8$  of the previous decade. When  $Q_8$  in one decade goes from 1 to 0, it triggers the count for the next higher-order decade while its own goes from 9 to 0.

~~Modulo-n counter~~

A counter that goes through a repeated sequence of 'n' states.

Ex. Mod-3 counter will go through 3 states 00 to 01 to 10 to 00

divide by N counter

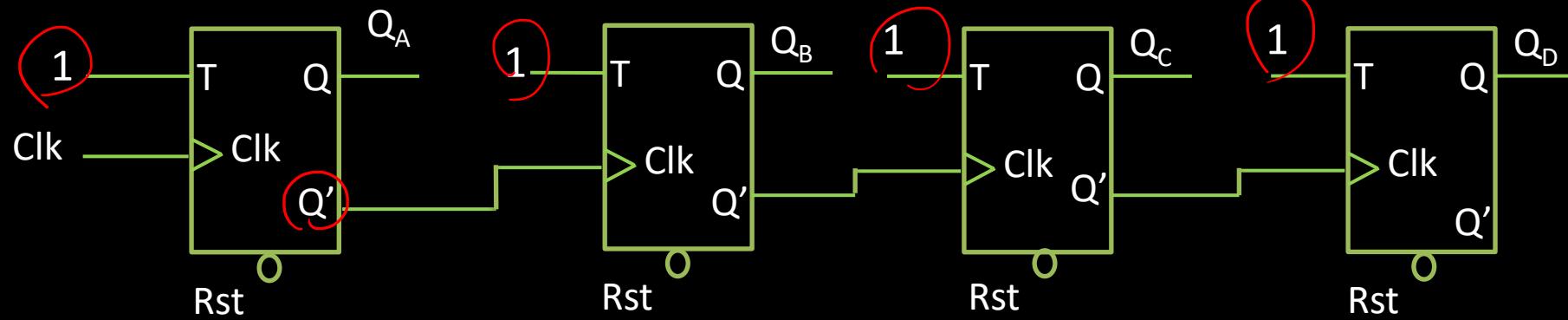
Mod - 10 Counter

## Mod Ripple counter

Can you design mod 10 counter counts till 9

0 to 9

1010  
0000



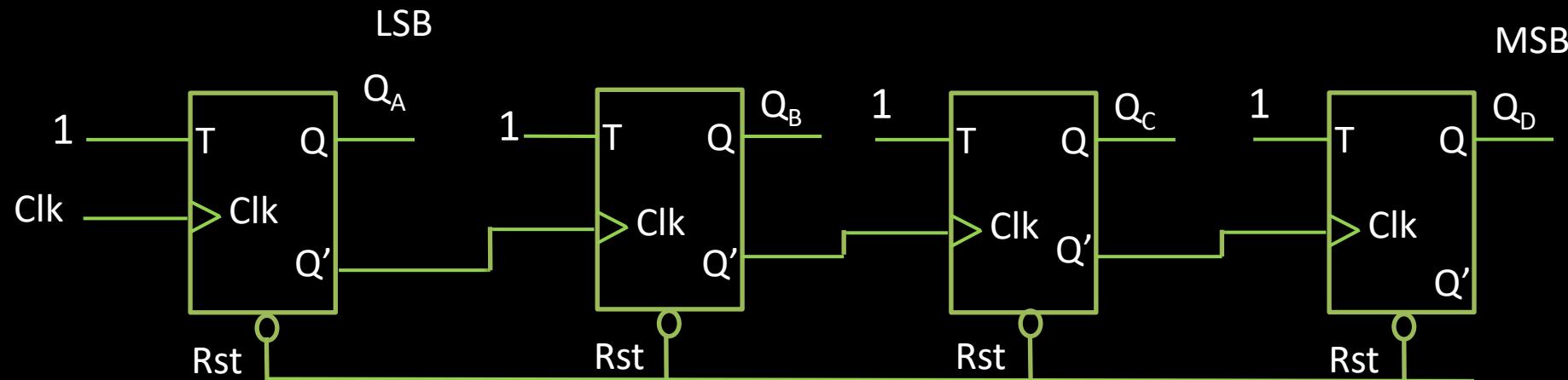
Counts from 0000 → 1111

As soon as 1010 reached all flip flops should be reset 0000

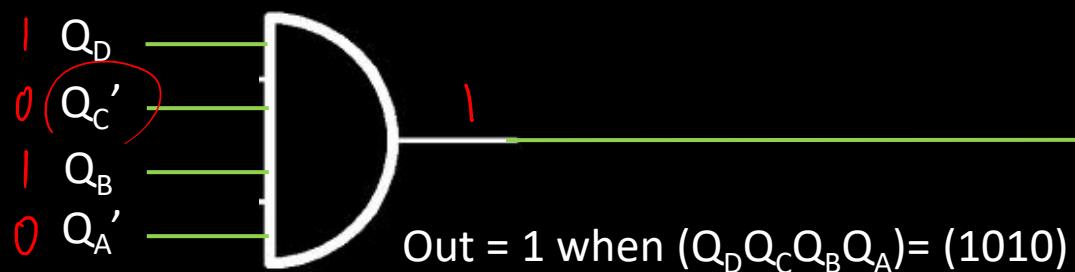
# Ripple Counters

mod -12

Mod Ripple counter



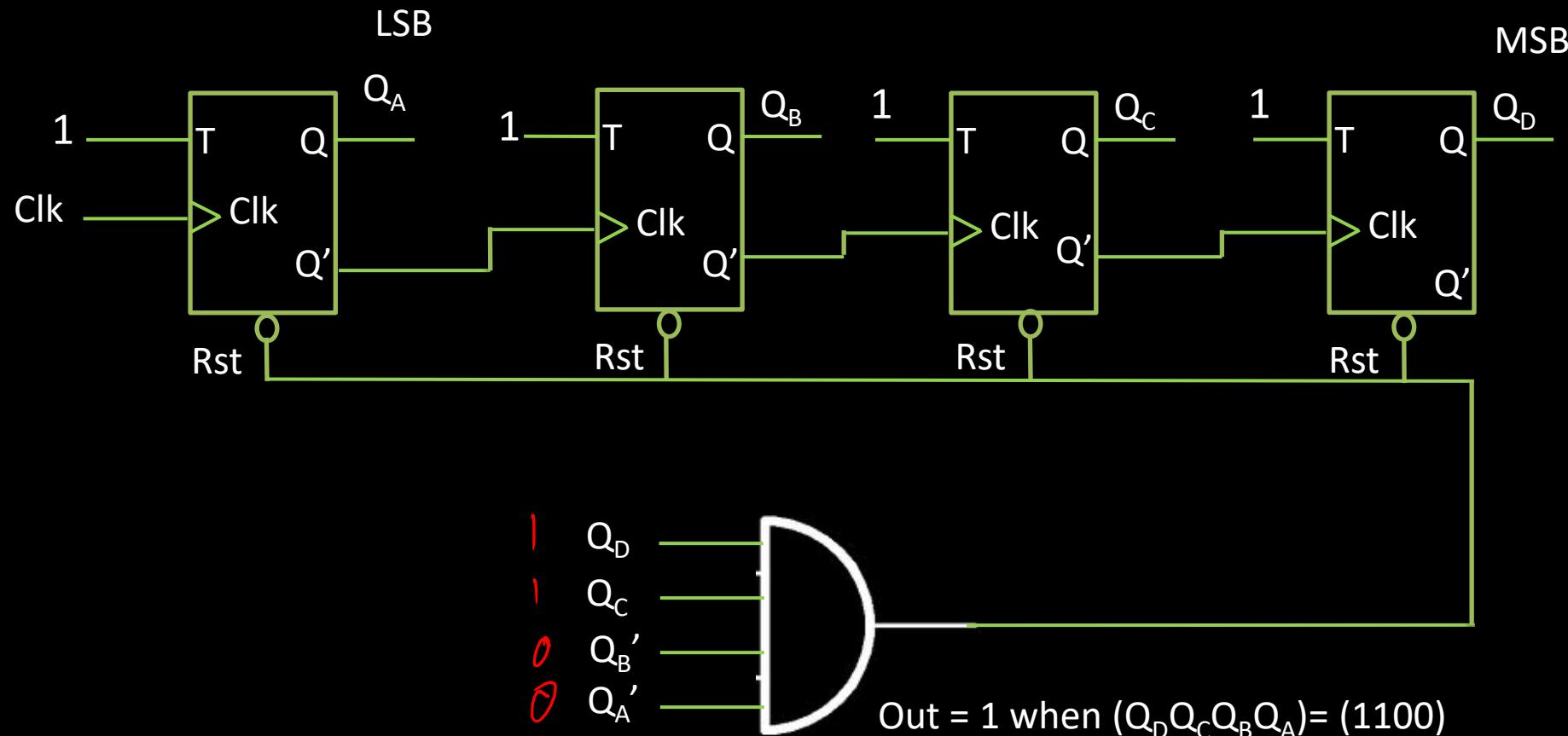
Mod 10 counter counts till 9



# Ripple Counters

Mod Ripple counter

Can you design Mod 12 counter



# Synchronous Counters

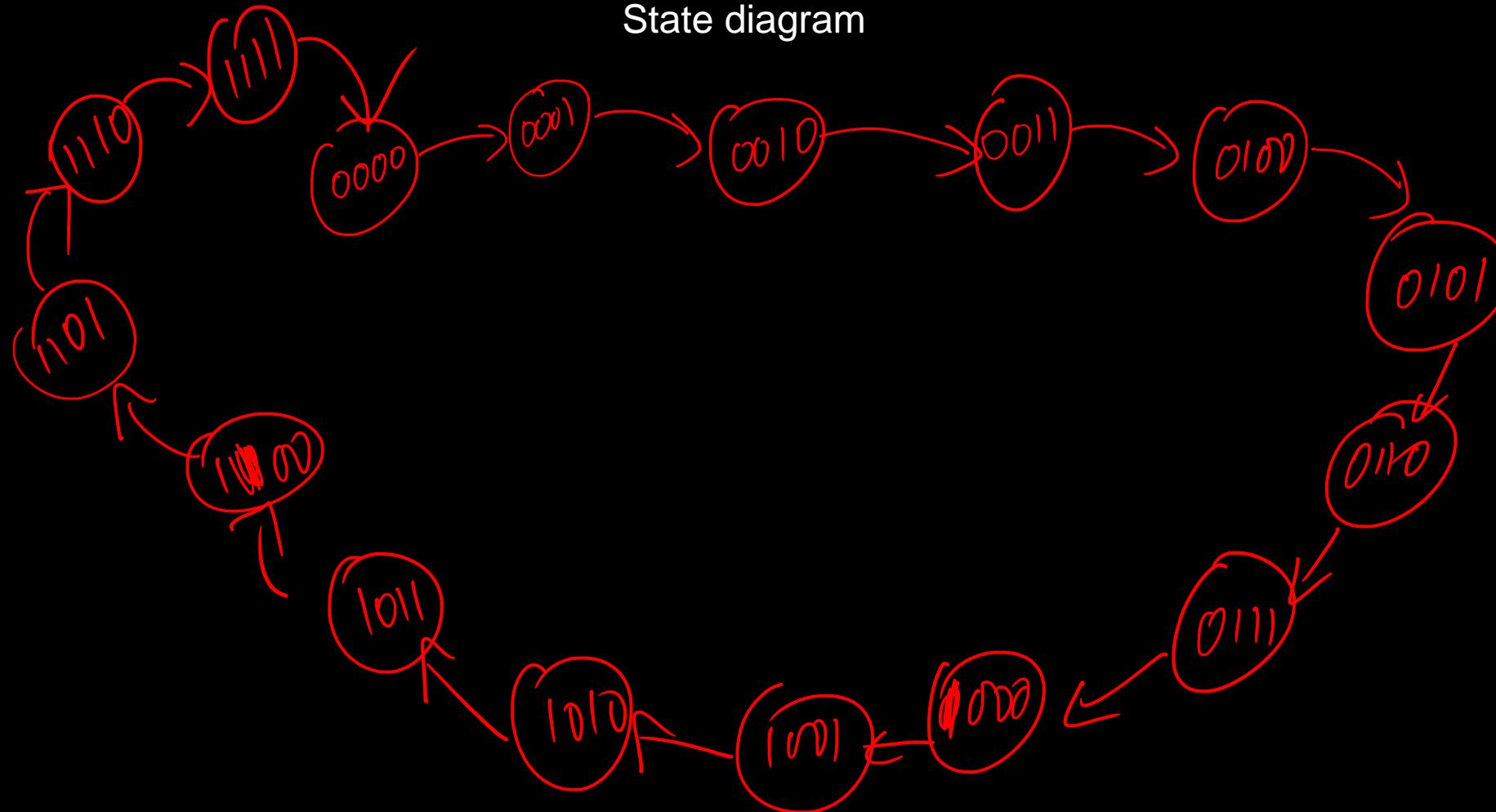
- The design procedure for a synchronous counter is the same as any other synchronous sequential circuit.
- The primary inputs of the circuit are the CLK and any control signals (EN, Load, etc).
- The primary outputs are the FF outputs (present state).
- Most efficient implementations usually use T-FFs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

We will examine JK, T and D flip-flop designs.

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



State diagram



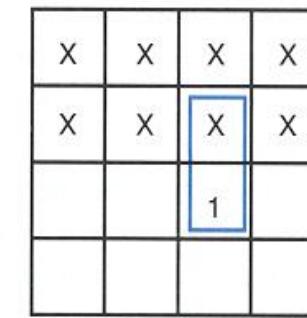
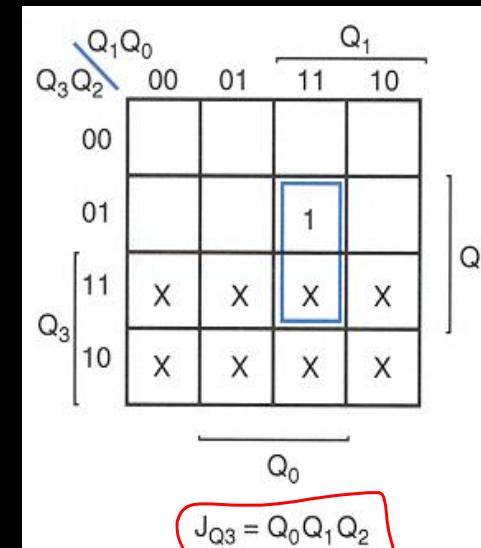
# J-K Flip Flop Design of a 4-bit Binary Up Counter



Present state				Next state				Flip-flop inputs							
$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_{Q3}$	$K_{Q3}$	$J_{Q2}$	$K_{Q2}$	$J_{Q1}$	$K_{Q1}$	$J_{Q0}$	$K_{Q0}$
0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
0	0	0	1	0	0	1	0	0	x	0	x	1	x	x	1
0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
0	0	1	1	0	1	0	0	0	x	1	x	x	1	x	1
0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
0	1	0	1	0	1	1	0	0	x	x	0	1	x	x	1
0	1	1	0	0	1	1	1	0	x	x	0	x	0	1	x
0	1	1	1	1	0	0	0	1	x	x	1	x	1	x	1
1	0	0	0	1	0	0	1	x	0	0	x	0	x	1	x
1	0	0	1	1	0	1	0	x	0	0	x	1	x	x	1
1	0	1	0	1	0	1	1	x	0	0	x	x	0	1	x
1	0	1	1	1	1	0	0	x	0	1	x	x	1	x	1
1	1	0	0	1	1	0	1	x	0	x	0	0	x	1	x
1	1	0	1	1	1	1	0	x	0	x	0	1	x	x	1
1	1	1	0	1	1	1	1	x	0	x	0	x	0	1	x
1	1	1	1	0	0	0	0	x	1	x	1	x	1	x	1

# Synchronous Binary Counters: J-K Flip Flop Design of a Binary Up Counter

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



# Synchronous Binary Counters:

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

Present state				Next state				Flip-flop control	
$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_{Q2}$	$K_{Q2}$
0	0	0	0	0	0	0	1	0	$\times$
0	0	0	1	0	0	1	0	0	$\times$
0	0	1	0	0	0	1	1	0	$\times$
0	0	1	1	0	1	0	0	1	$\times$
0	1	0	0	0	1	0	1	$\times$	0
0	1	0	1	0	1	1	0	$\times$	0
0	1	1	0	0	1	1	1	$\times$	0
0	1	1	1	1	0	0	0	$\times$	1
1	0	0	0	1	0	0	1	0	$\times$
1	0	0	1	1	0	1	0	0	$\times$
1	0	1	0	1	0	1	1	0	$\times$
1	0	1	1	1	1	0	0	1	$\times$
1	1	0	0	1	1	0	1	$\times$	0
1	1	0	1	1	1	1	0	$\times$	0
1	1	1	0	1	1	1	1	$\times$	0
1	1	1	1	0	0	0	0	$\times$	1

			1	
X	X	X	X	
X	X	X	X	
		1		

$J_{Q2} = Q_0 Q_1$

X	X	1	X
		1	
		1	
X	X	X	X

$K_{Q2} = Q_0 Q_1$

# Synchronous Binary Counters:

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

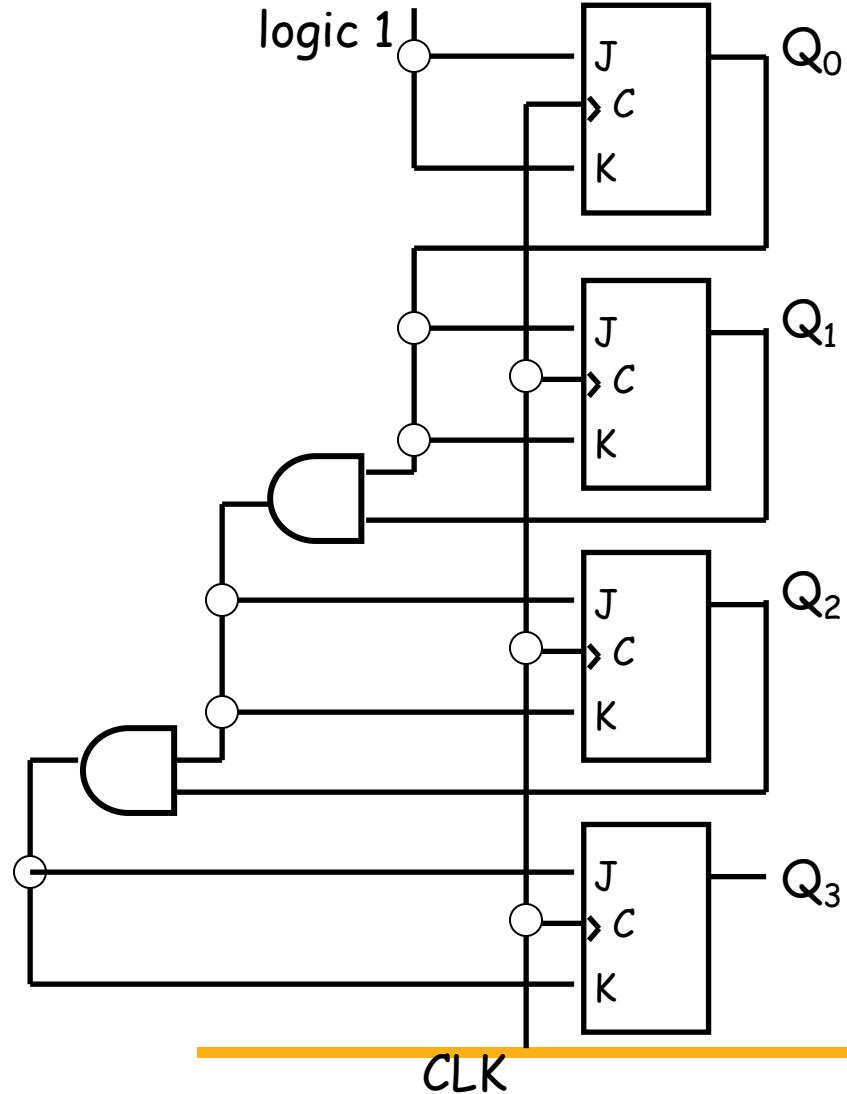
Present state				Next state				J-K inputs	
$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_{Q1}$	$K_{Q1}$
0	0	0	0	0	0	0	1	0	X
0	0	0	1	0	0	1	0	1	X
0	0	1	0	0	0	1	1	X	0
0	0	1	1	0	1	0	0	X	1
0	1	0	0	0	1	0	1	0	X
0	1	0	1	0	1	1	0	1	X
0	1	1	0	0	1	1	1	X	0
0	1	1	1	1	0	0	0	X	1
1	0	0	0	1	0	0	1	0	X
1	0	0	1	1	0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

	1	X	X
	1	X	X
	1	X	X
	1	X	X

$$J_{Q1} = Q_0$$

X	X	1	
X	X	1	
X	X	1	
X	X	1	

$$K_{Q1} = Q_0$$





# CS/ECE/EEE/INSTR F215:Digital Design



## Lecture 30: *Counters\_2*

*Fri, 19 Nov 2021*

**BITS** Pilani  
Hyderabad Campus

**Dr. R. N. Ponnalagu, EEE**

*I find that the harder I work,  
the more luck I seem to have.*

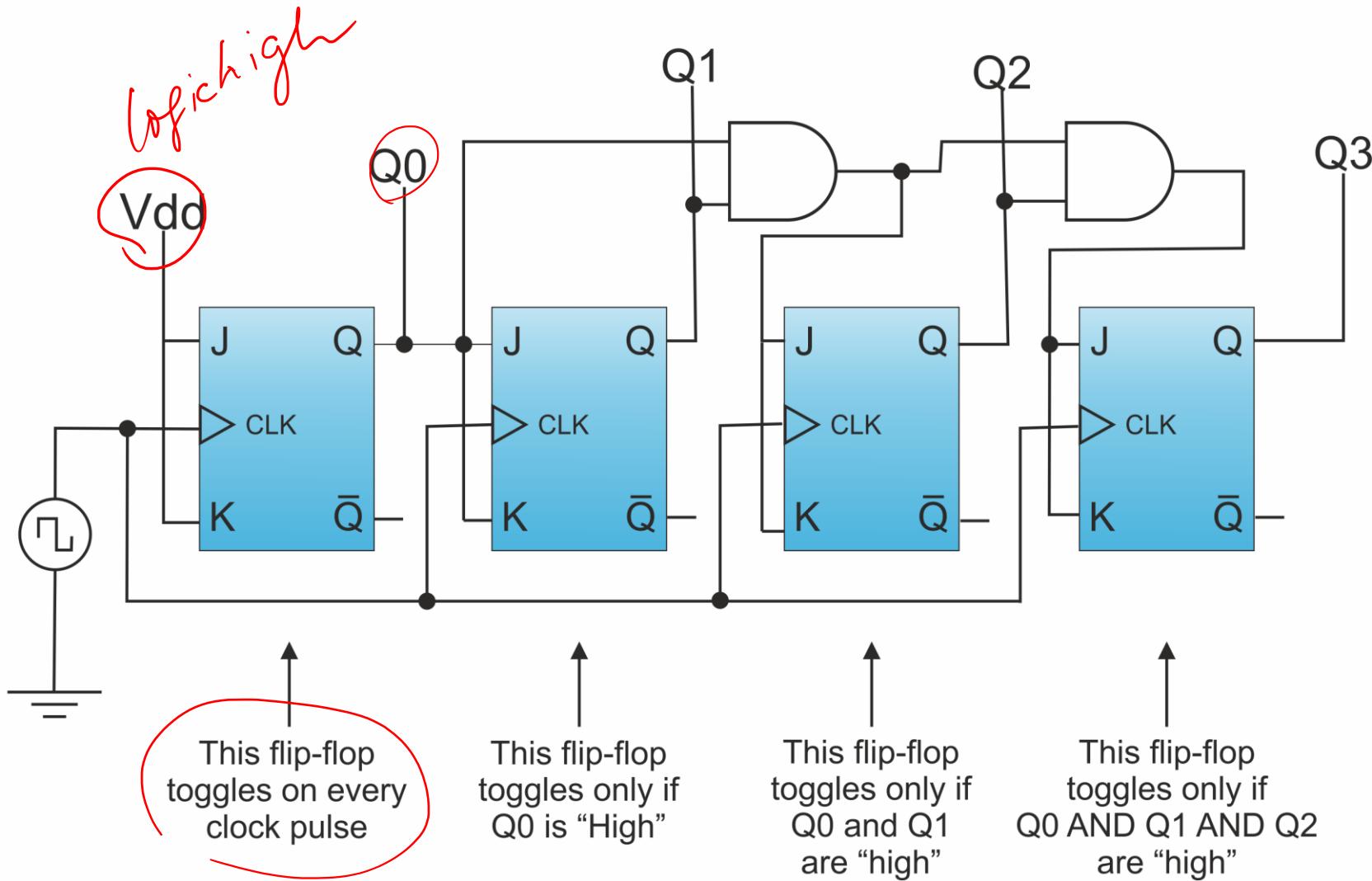
*- Thomas Jefferson*

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



Present state				Next state				Flip-flop inputs							
$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_{Q3}$	$K_{Q3}$	$J_{Q2}$	$K_{Q2}$	$J_{Q1}$	$K_{Q1}$	$J_{Q0}$	$K_{Q0}$
0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
0	0	0	1	0	0	1	0	0	x	0	x	1	x	x	1
0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
0	0	1	1	0	1	0	0	0	x	1	x	x	1	x	1
0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
0	1	0	1	0	1	1	0	0	x	x	0	1	x	x	1
0	1	1	0	0	1	1	1	0	x	x	0	x	0	1	x
0	1	1	1	1	0	0	0	1	x	x	1	x	1	x	1
1	0	0	0	1	0	0	1	x	0	0	x	0	x	1	x
1	0	0	1	1	0	1	0	x	0	0	x	1	x	x	1
1	0	1	0	1	0	1	1	x	0	0	x	x	0	1	x
1	0	1	1	1	1	0	0	x	0	1	x	x	1	x	1
1	1	0	0	1	1	0	1	x	0	x	0	0	x	1	x
1	1	0	1	1	1	1	0	x	0	x	0	1	x	x	1
1	1	1	0	1	1	1	1	x	0	x	0	x	0	1	x
1	1	1	1	0	0	0	0	x	1	x	1	x	1	x	1

## 4-BIT SYNCHRONOUS “UP” COUNTER



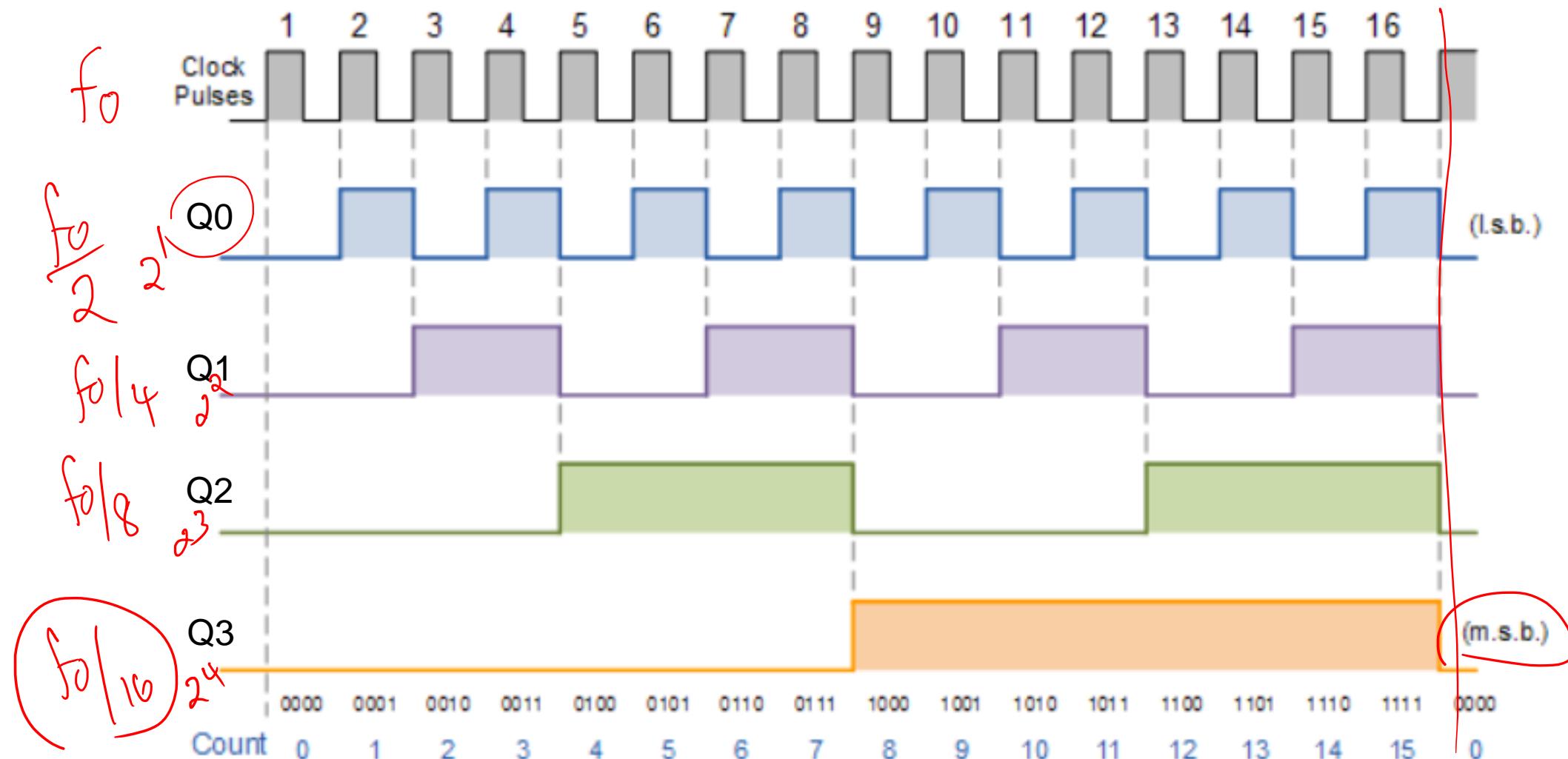
$$J_{Q_0} = 1$$
$$K_{Q_0} = 1$$

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

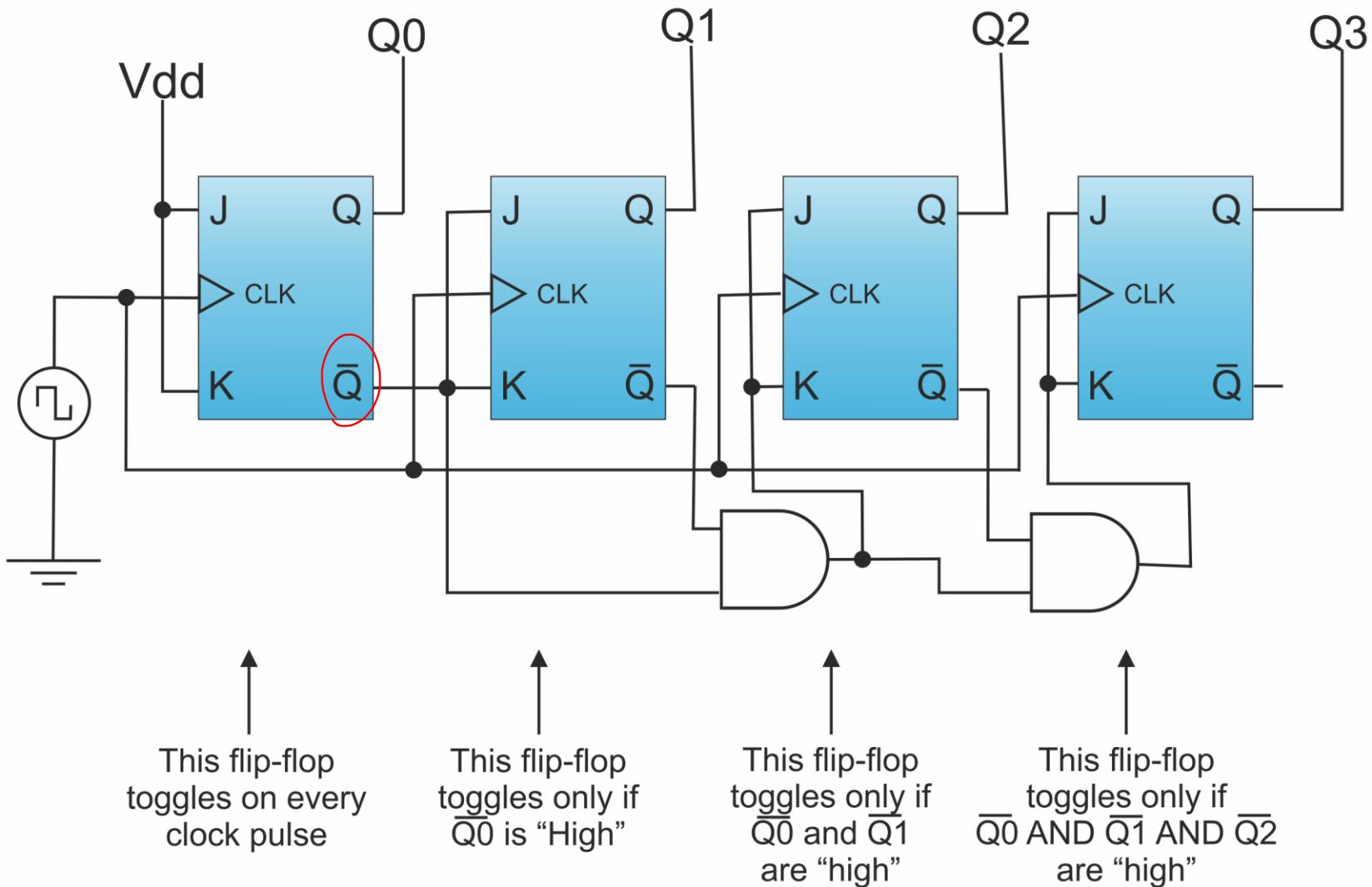
$$J_{Q_2} = Q_0 Q_1$$
$$K_{Q_2} = Q_0 Q_1$$

$$J_{Q_3} = Q_0 Q_1 Q_2$$
$$K_{Q_3} = Q_0 Q_1 Q_2$$

## 4-bit Synchronous Counter Waveform Timing Diagram



## 4-BIT SYNCHRONOUS “DOWN” COUNTER



## *"Down" count sequence*

1

1)

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

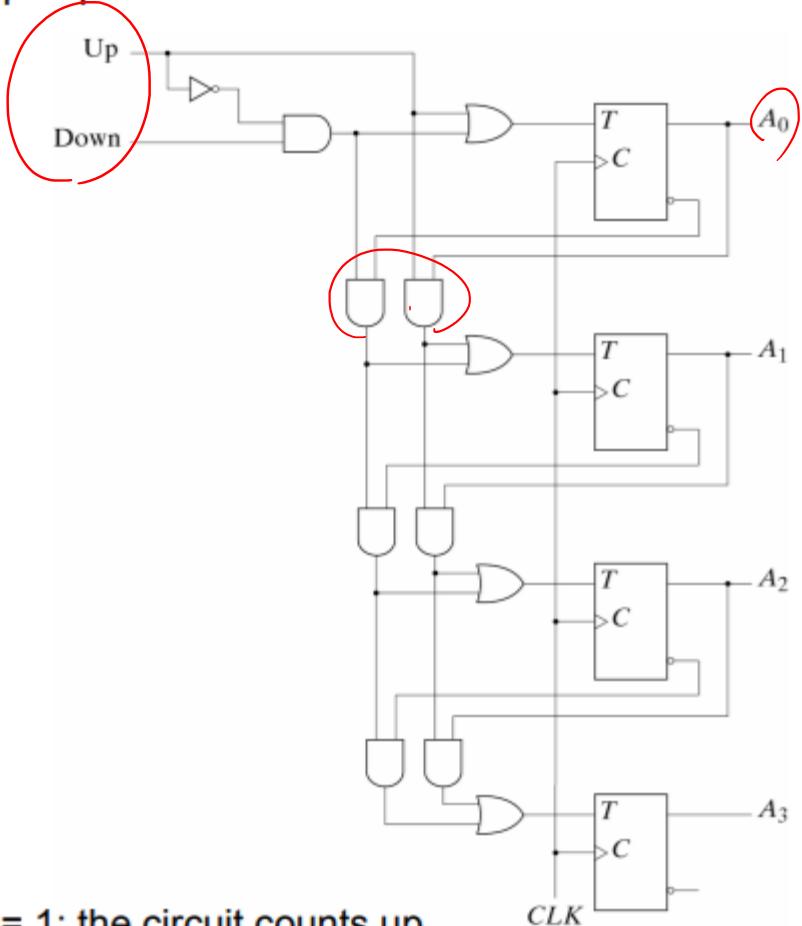
2

**1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0**

2

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

The circuit of a 4-bit up-down binary counter with  $T$  flip-flops is:



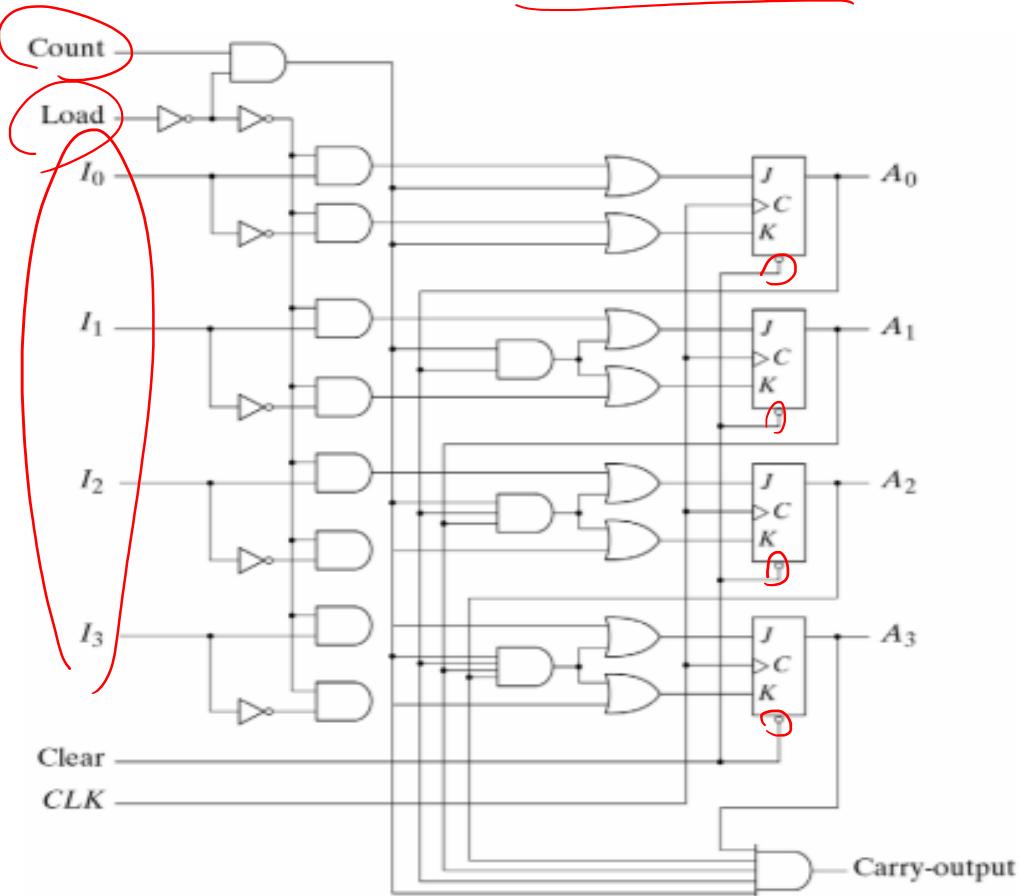
Up = 1; the circuit counts up.

Down = 1, Up = 0; the circuit counts down.

Up = 0, Down = 0; the circuit doesn't change state.

Up = 1, Down = 1, the circuit counts up.

## A 4-bit binary counter with parallel load capability:

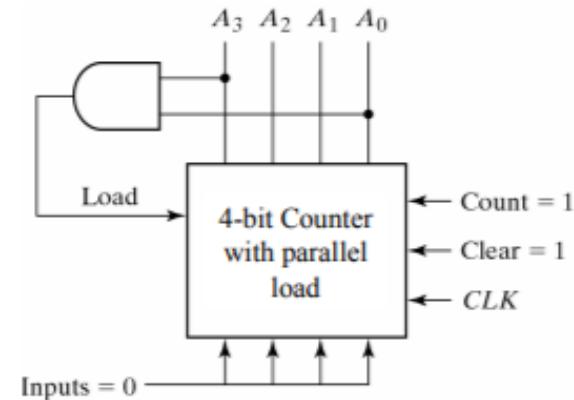


Its operation is summarized in the following table:

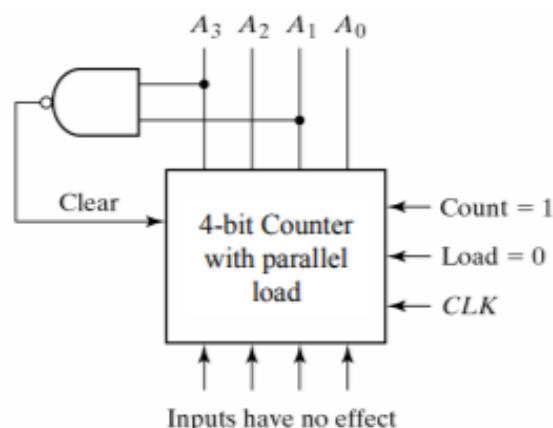
Active In	Clear	CLK	Load	Count	Function
0	X	X	X	X	Clear to 0
1	↑	↑	1	0	Load inputs
1	1	0	0	1	Count next binary state
1	1	0	0	0	No change

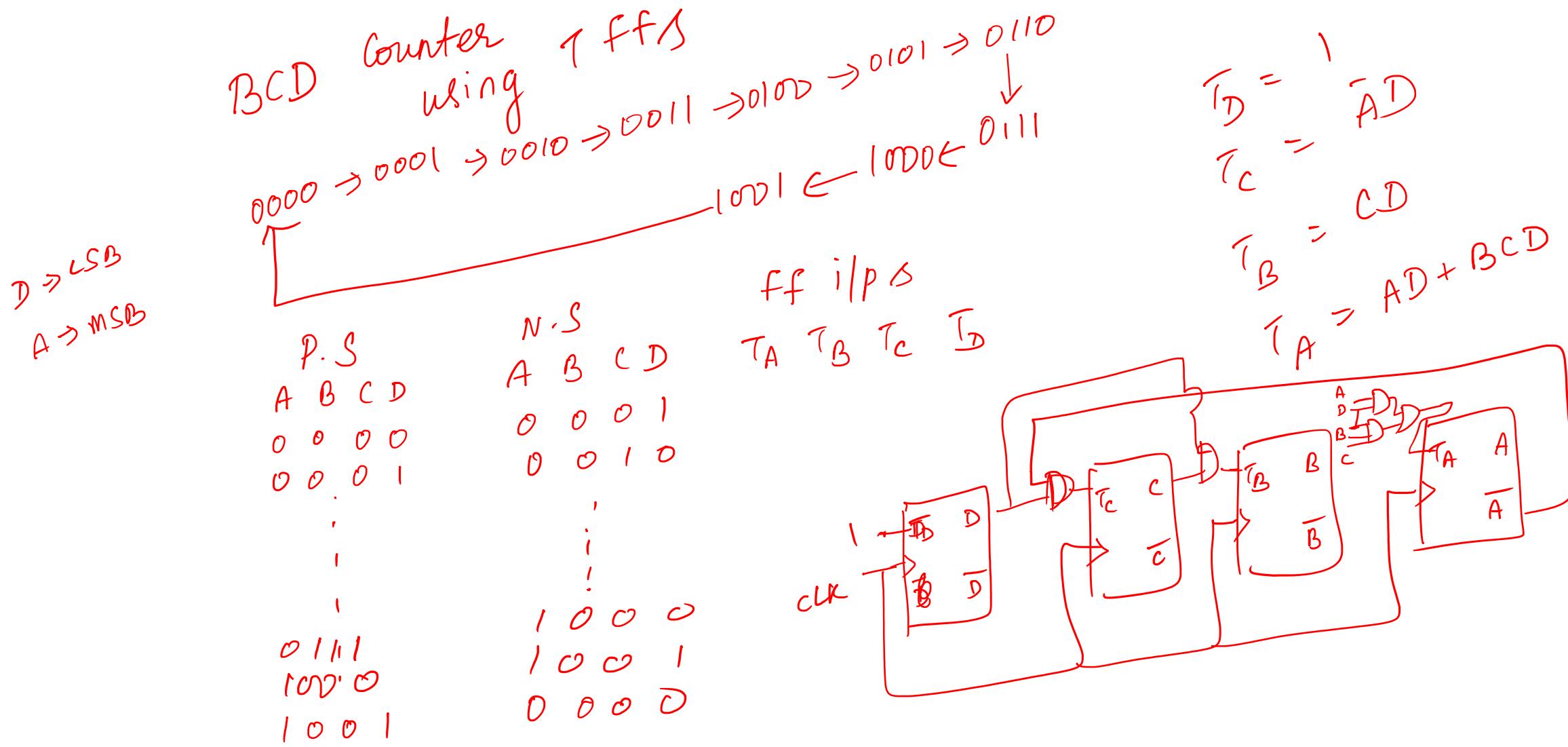
A counter with parallel load can be used to create any desired count sequence. For example, the 4-bit counter with parallel load shown previously can be used to generate a BCD count in two ways:

1. Using the load input:



2. Using the clear input:

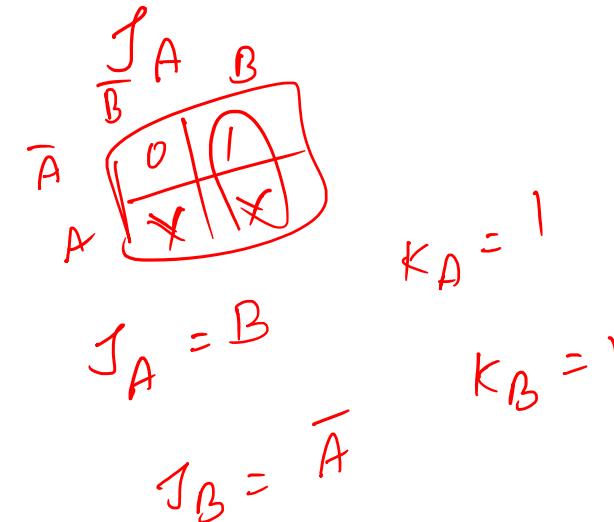




Mod-3 Counter

$0 \xrightarrow{ } 1 \xrightarrow{ } 2 \xrightarrow{ } 0$

P.S		N.S		ff i/p b			
A	B	A	B	J <sub>A</sub>	K <sub>A</sub>	J <sub>B</sub>	K <sub>B</sub>
0	0	0	1	0	x	x	1
0	1	1	0	x	1	0	x
1	0	0	0	x	x	x	x
1	1	x	x				



$$K_A = 1$$

$$K_B = 1$$

i/fbs

using

$$T_A = AC + BC$$

$$T_B = \bar{A}C$$

$$T_C = 1$$

Mod 6

0, 1, 2, 3, 4, 5

$0, 1, 2, 4, 5, \overbrace{6}^{\text{and repeats}}$   
 Design a counter to go through the sequence  $0, 1, 2, 4, 5, 6$  & repeat  
 Counters with unused states

P.S	N.S	$T_A$	$T_B$	$T_C$
A B C	A B C	0	0	1
0 0 0	0 0 1	0	1	1
0 0 1	0 1 0	0	1	0
0 1 0	1 0 0	1	x	x
<span style="border: 1px solid black; padding: 2px;">0 1 1</span>	x x x	x	0	0
1 0 0	1 0 1	0	1	1
1 0 1	1 1 0	0	1	1
1 1 0	0 0 0	x	x	x
<span style="border: 1px solid black; padding: 2px;">1 1 1</span>	x x x	x	x	x

unused states

self correcting counter

$$T_A = B$$

$$T_B = B + C$$

$$T_C = \bar{B}$$

$P.S \ A B C \Rightarrow 011$   
 $T_A = 1 \quad T_B = 1 \quad T_C = 0$

$N.S \ A B C = 101$

if P.S  $A B C \Rightarrow 111$   
 $T_A = 1 \quad T_B = 1 \quad T_c = 0$

$N.S \ A B C \Rightarrow 001$

0, 1, 3, 7, 6, 4 & repeat

0, 1, 3, 7, 6, 4, 0  
01 2 3 1 7, 6, 4,

P.S			N.S			ff i   PS			$T_A \quad T_B \quad T_C$		
A	B	C	A	B	C	0	0	0	1	0	0
0	0	0	0	0	1	x	x	x	0	0	0
0	0	1	0	1	1	1	0	0	1	0	0
<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>x</u>	<u>x</u>	<u>x</u>	<u>0</u>	<u>0</u>	<u>0</u>
0	1	1	0	0	0	110	x	x	0	1	0
1	0	0	1	0	0	x	x	x	0	0	1
1	0	1	1	0	0	0	0	0	0	1	0
1	1	0	1	0	0	0	0	0	0	0	1
1	1	1	<del>1</del>	<del>1</del>	<del>0</del>	<del>0</del>	<del>0</del>	<del>0</del>	<del>0</del>	<del>0</del>	<del>0</del>

$\bar{T}$  ffs

$T_A$		$T_B$		$T_C$	
$\bar{A}$	$\bar{B}C$	$\bar{B}C$	$BC$	$BC$	$\bar{C}$
0	0	0	1	X	
1	X	0	0	0	

$$\bar{T}_A = A \oplus B$$

$T_B$		$T_C$	
$\bar{A}$	$\bar{B}$	$\bar{C}$	$C$
0	0	0	X
0	X	0	1

$$\bar{T}_B = B \oplus C$$

$T_C$		$\bar{T}_C$	
$\bar{A}$	$\bar{B}$	$\bar{C}$	$C$
0	0	0	X
0	X	1	0

$$\bar{T}_C = A \oplus C$$

010 & 101

P.S  $ABC \Rightarrow 010$

$$\bar{T}_A = 1$$

$$\bar{T}_B = 1$$

$$\bar{T}_C = 1$$

N.S  $ABC \Rightarrow 101$

P.S  $ABC \Rightarrow 101$

$$\bar{T}_A = 1$$

$$\bar{T}_B = 1$$

$$\bar{T}_C = 1$$

N.S  $ABC \Rightarrow 010$



## CS/ECE/EEE/INSTR F215:Digital Design



### Lecture 31: Shift register counters and Introduction to ASM

*Thu, 25 Nov 2021*

**BITS** Pilani  
Hyderabad Campus

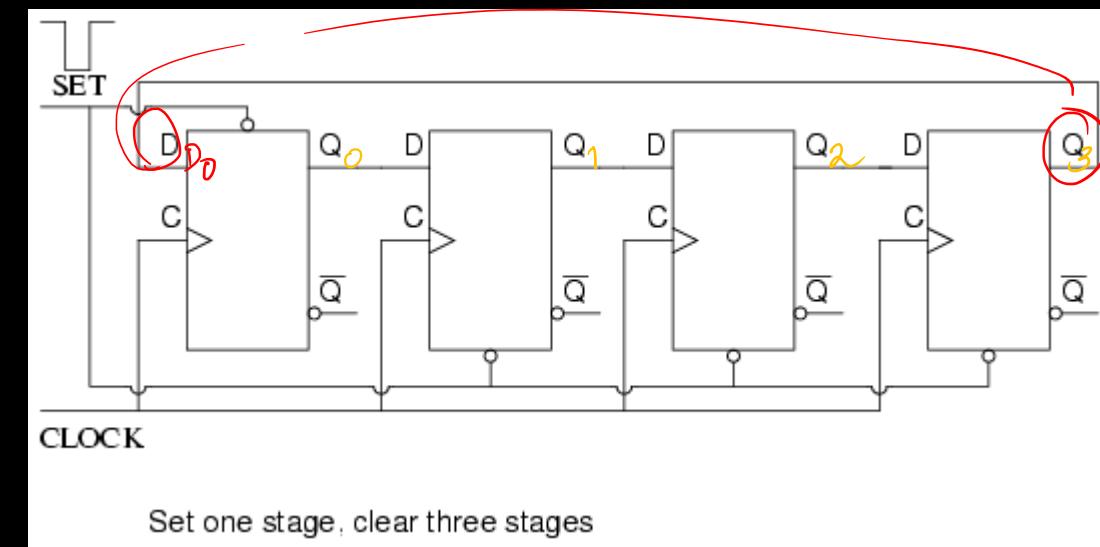
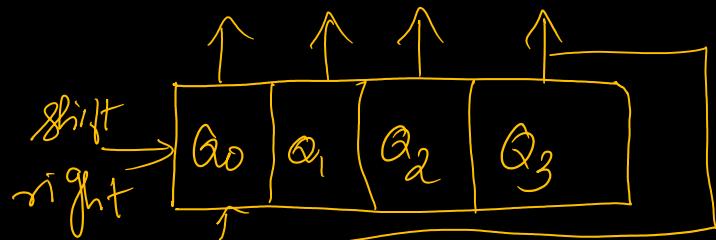
**Dr. R. N. Ponnalagu, EEE**

Stop thinking of  
Limitations

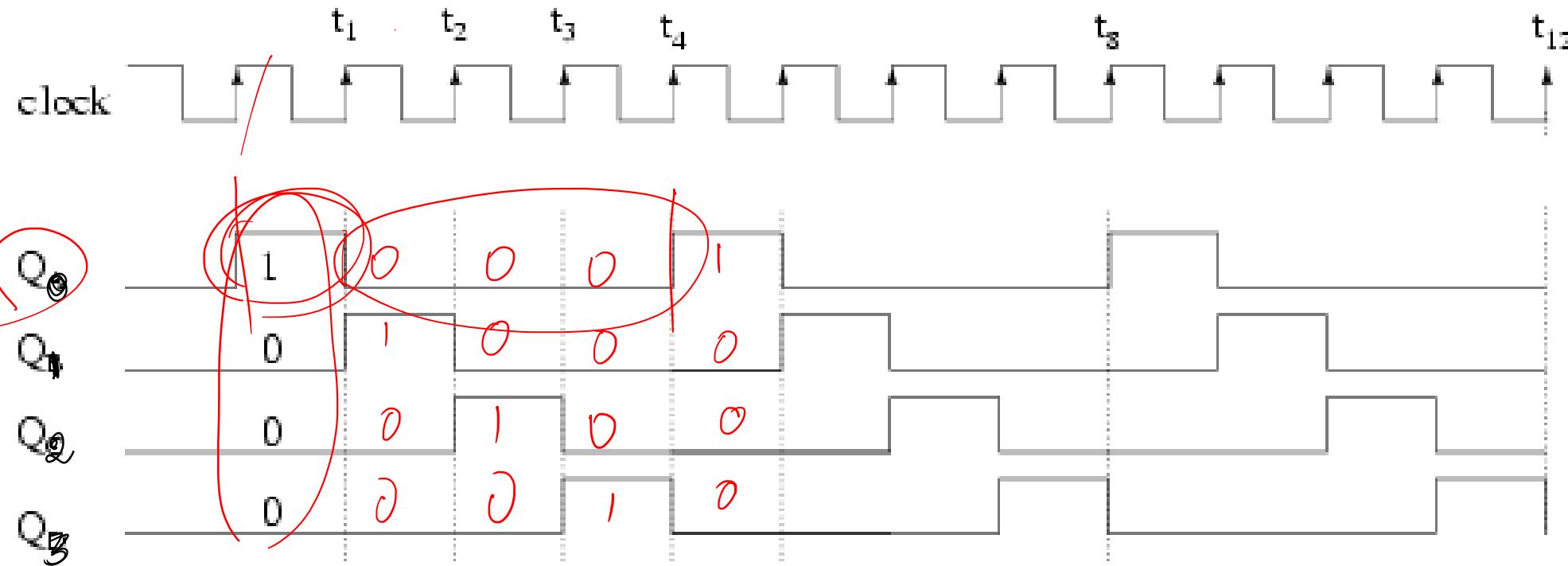
Start thinking of  
POSSIBILITIES

Shift register counter : Two types : 1. Ring Counter  
2. Johnson counter

These two shift register circuits go through prescribed sequence of states hence they are called as counters



CLK	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0



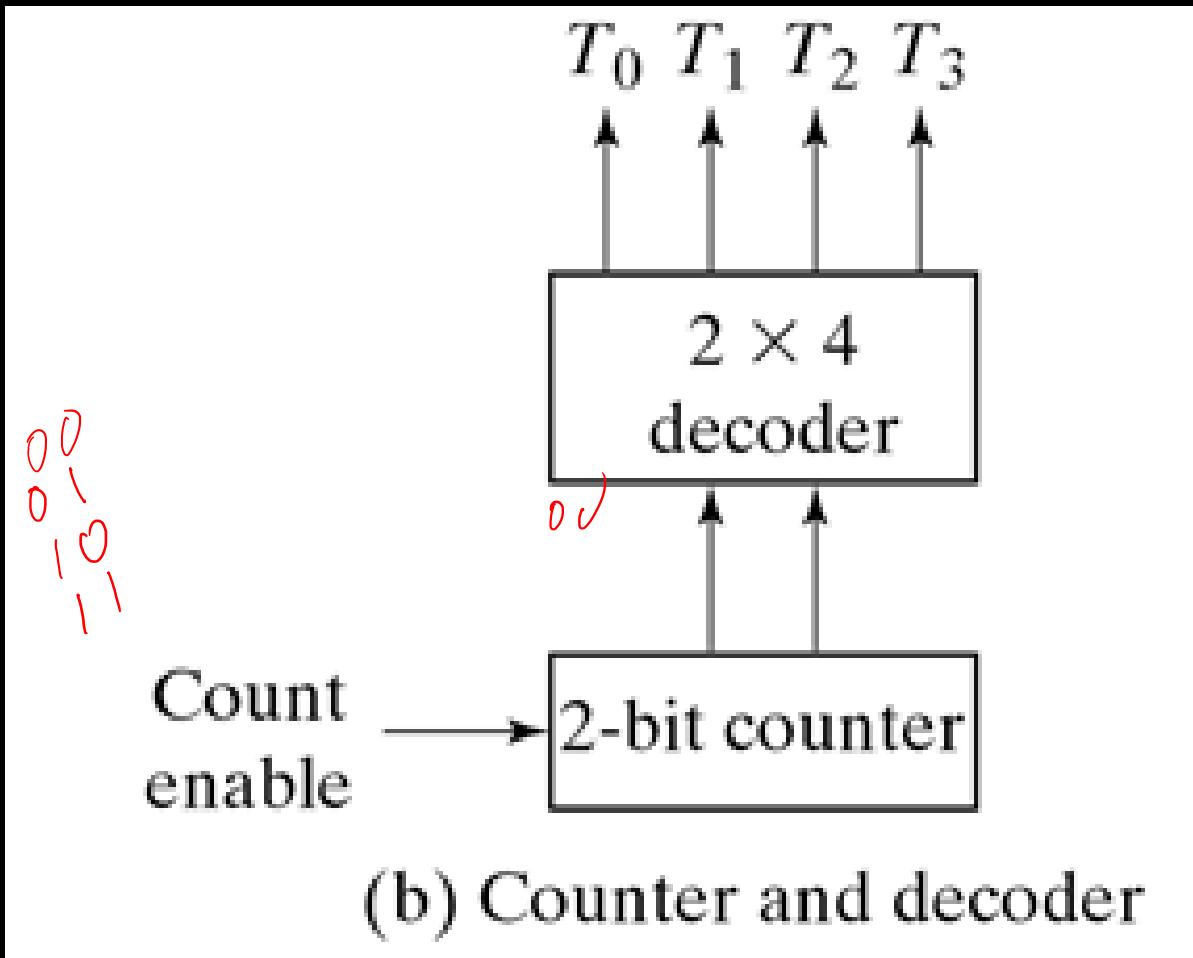
Load 1000 into 4-stage ring counter and shift

Application:  
Used for generation of timing signals

The circuit above is a divide by 4 counter. Comparing the clock input to any one of the outputs, shows a frequency ratio of 4:1

4 different states  $\Rightarrow$  Mod-4 counter  
 4 flip flops are used  $\rightarrow$  16 states are possible  
 out of which only 4 are utilized

Another method to generate 4 timing signal.

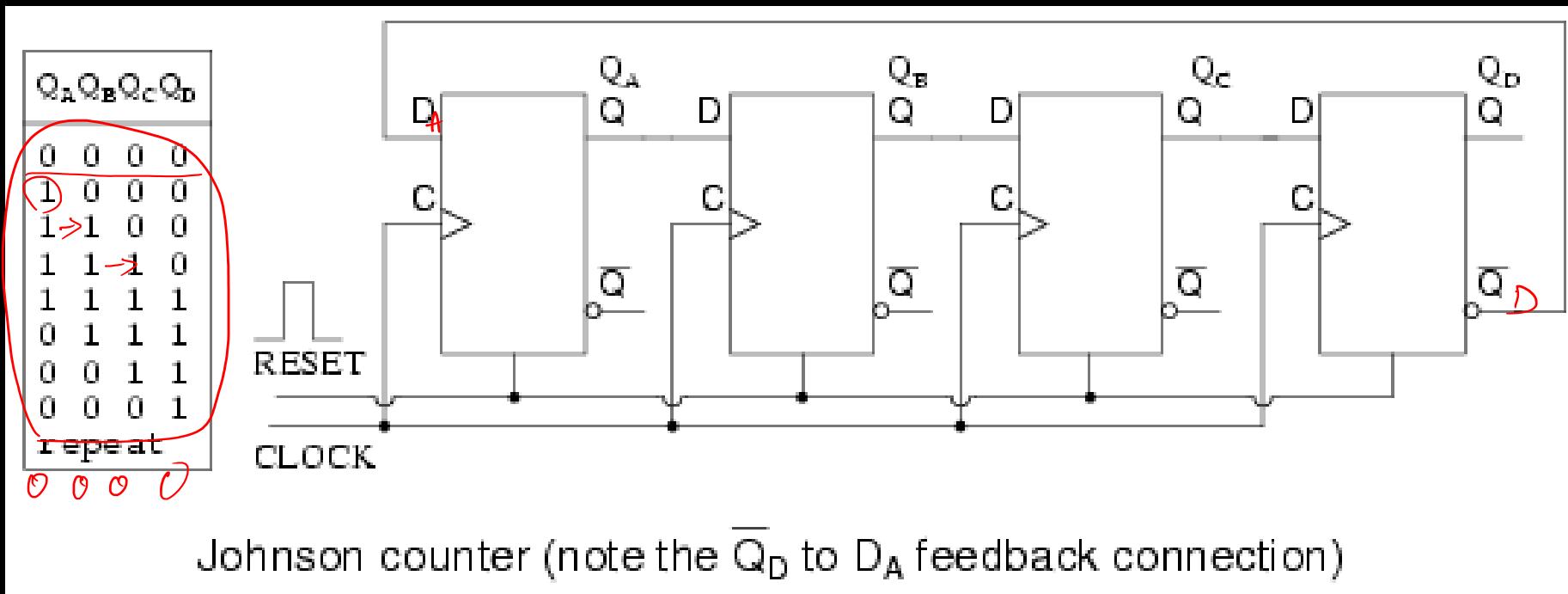


To generate  $(2^n)$  timing signals

1. shift register with  $2^n$  ffs  
or
2.  $n$ -bit binary counter with  
( $n$  to  $2^n$ ) decoder  
can be used.

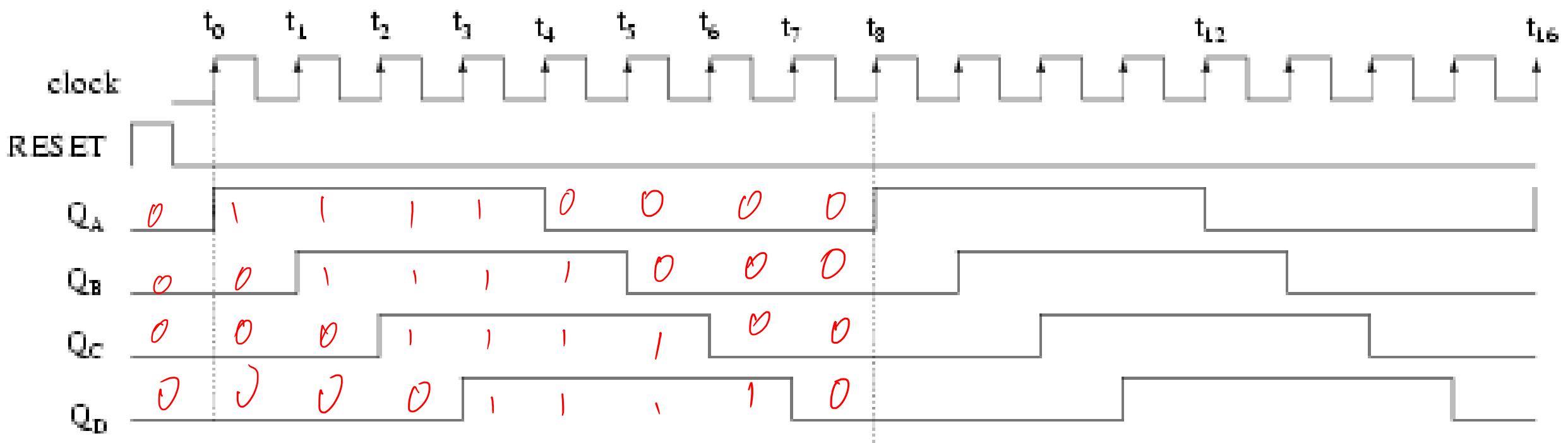
# Johnson counter

- A k-bit ring counter circulates a single bit and generates k-distinguishable states.
- No. of states can be doubled if shift register is connected as a switch tail ring counter.
- A switch tail ring counter is a circular shift register with the complemented output of the last FF connected to the input of the first FF.



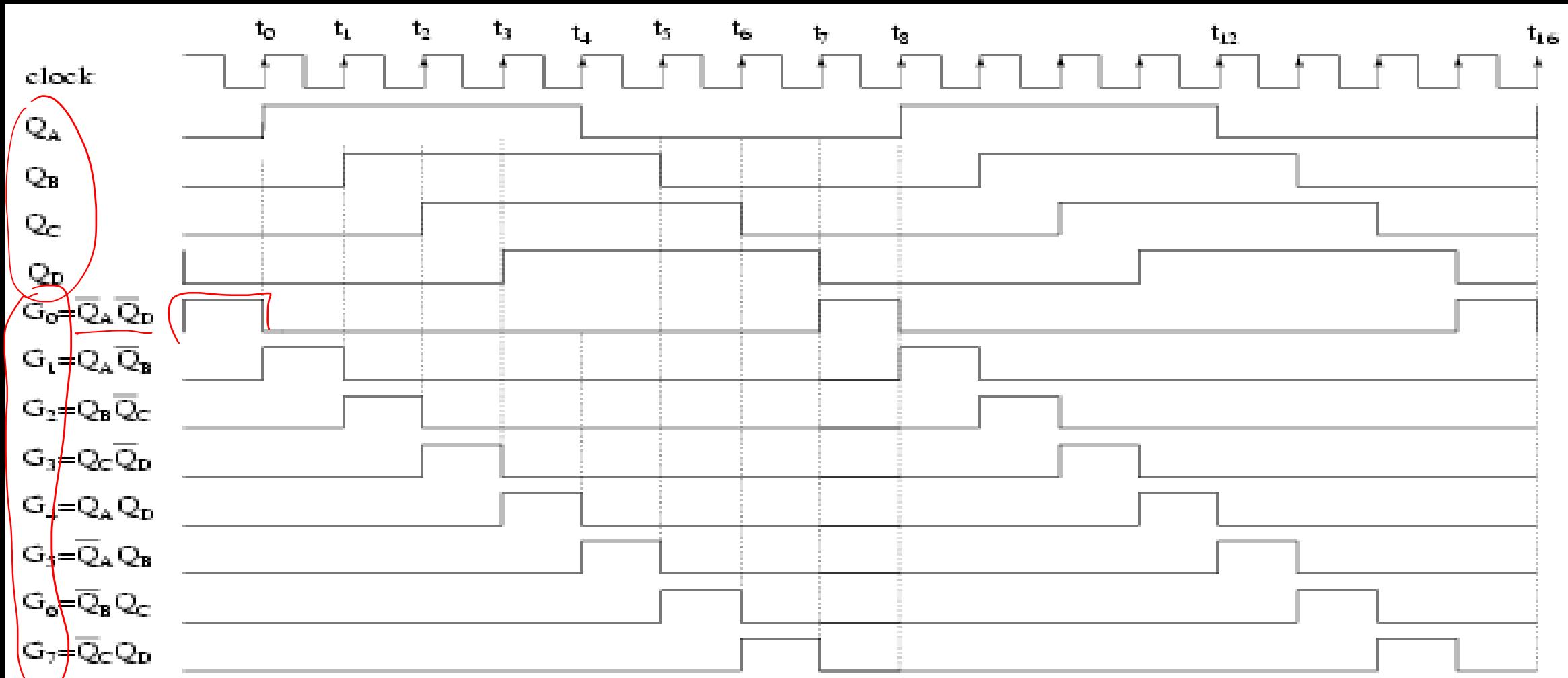
- Starting from cleared state 4 bit switch tail ring counter goes through 8 states

# Timing diagram



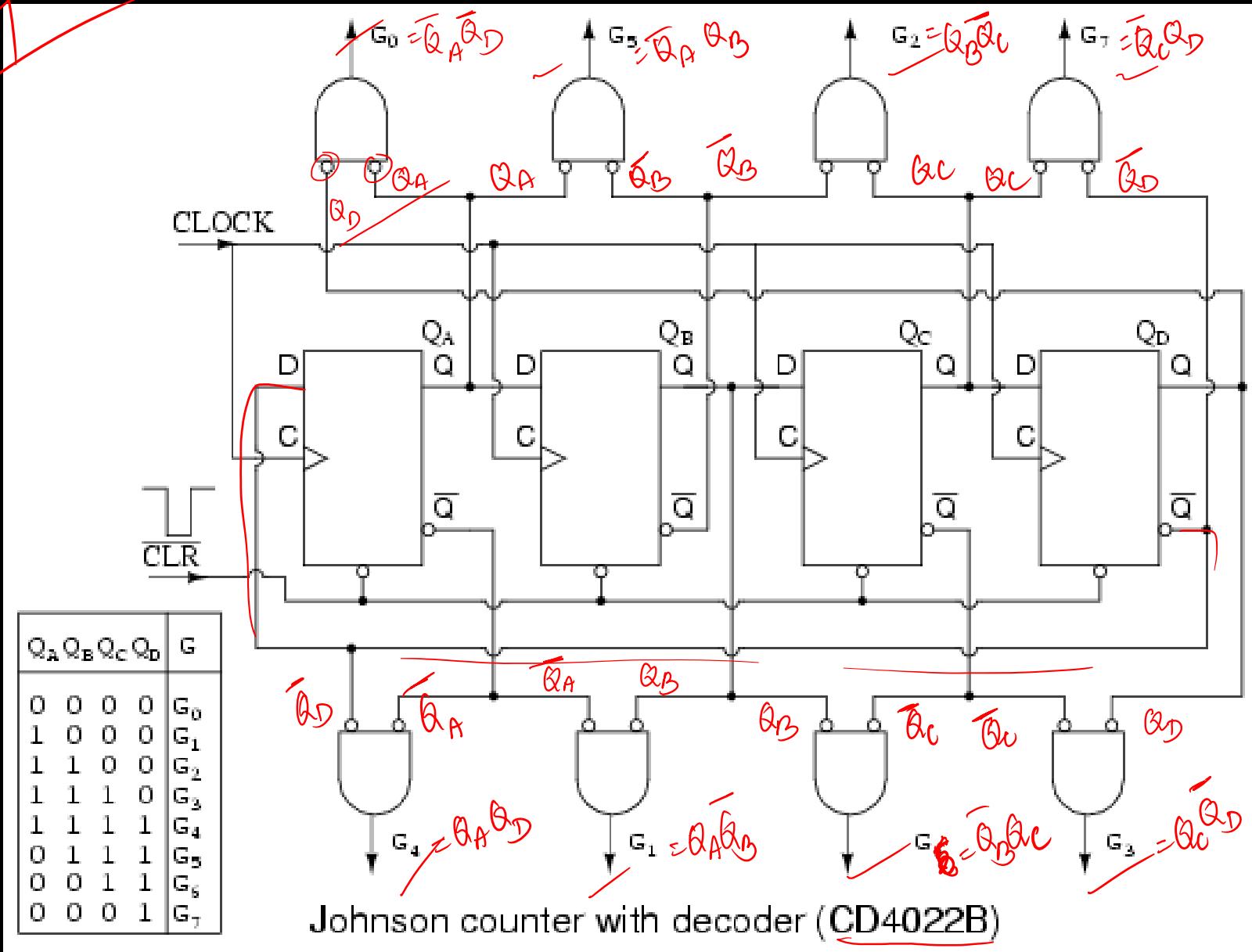
Four stage Johnson counter waveforms

four phased square waves  $Q_A$  to  $Q_D$  are decoded to eight signals ( $G_0$  to  $G_7$ )



Four stage (8-state) Johnson counter decoder waveforms

IC 7474



**A JOHNSON COUNTER is a k-bit switch tail ring counter with  
2k decoding gates to provide outputs for 2k timing signals**



2 ;/p AND gates

# **Algorithmic State Machines**

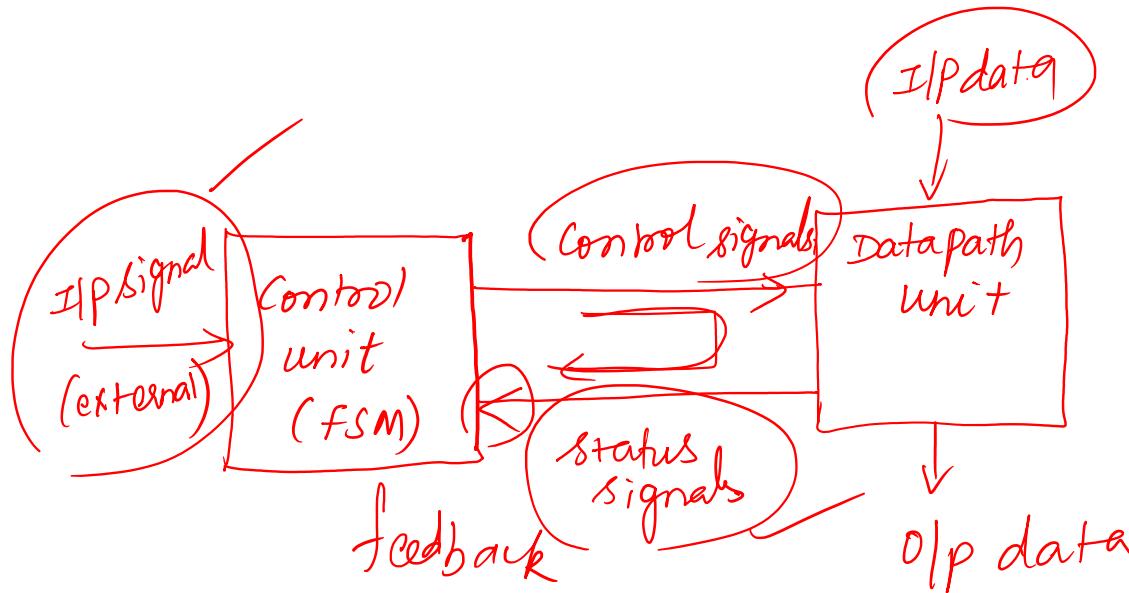
## **(ASM)**

# Digital System Logic Design

- Design of digital circuits performing data processing operations
- Design of control circuits to manipulate data operations

- Binary information stored in Digital Systems can be classified as Data or Control
- Data are discrete elements of information (binary words) manipulated by performing arithmetic, logic, shift and other similar data processing operations. Implemented using digital H/W components such as adders, multiplexers, decoders, registers, counters etc.
- Control information provides command signals that coordinate and execute the various operations in the data section to execute desired data processing tasks
- Various modules are interconnected to form digital system

# Relationship between control logic and data processing operations



Control and data path interaction

- Data path unit manipulates data in the registers according to system requirements.
- Control unit issues sequence of commands to data path unit
- Feedback from data path unit to control unit provides status conditions
- Control unit uses the status conditions and external input to determine the sequence of control signals that direct the operation of data path unit.

# FSM: Finite State Machine

- **FSM is a synchronous sequential circuit**
- **FSM produces the control commands for the system**
- **Control commands are functions of i) external inputs, ii) status signals and the iii) state of the machine**
- **Control commands are fed to data path unit**
- **Depending on present state, external inputs and status conditions of data path, FSM goes into its next state**

# Algorithm

- The control sequence and data path tasks of a digital system are specified by means of a hardware algorithm
- An algorithm consists of a finite no. of procedural steps that specify how to obtain solution to a problem
- A hardware algorithm is a procedure for solving the problem with a given piece of equipment
- Challenge is the formulation of hardware algorithm for achieving required objectives

# Algorithm

- A flow chart is the convenient way to specify the sequence of steps and decision paths for the algorithm
- Flowchart translates verbal instructions to an information diagram that depicts sequence of operations along with the conditions necessary for their execution.
- An algorithmic state machine (ASM) chart is a special purpose flowchart developed to define algorithms for execution on digital hardware
- Conventional flowchart describes procedural steps and decision paths of an algorithm in a sequential manner, without taking into consideration their time relationship
- ASM chart describes the sequence of events as well as the timing relationship between states of sequential controller and the events that occur while going from one state to next

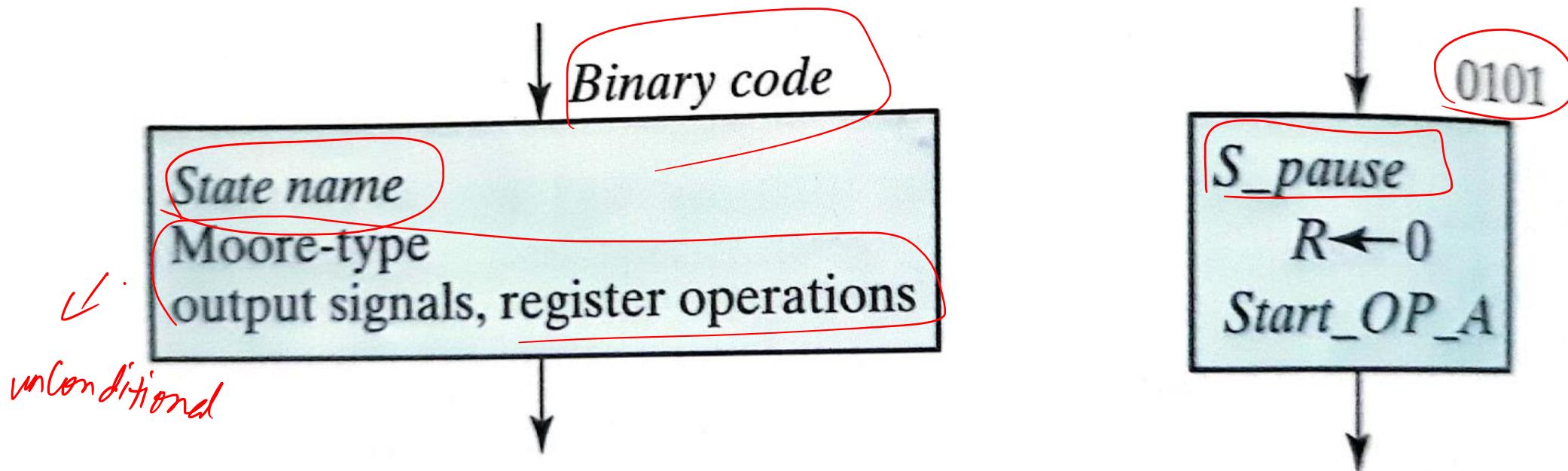
- Flowchart developed to design Digital Hardware Algorithms is **Algorithmic State Machines (ASM) chart**

## **ASM CHART**

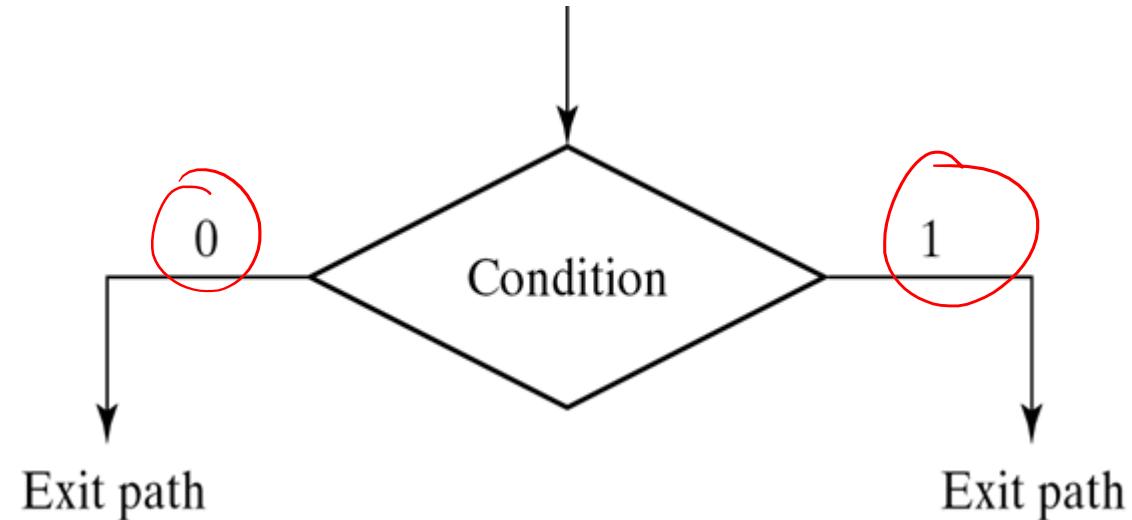
### **composed of three basic elements**

- The State box**
- The decision box**
- The conditional box**

- A state is indicated by the **state box** (Rectangle in shape) within which are written
  - i) register operations that occur when state transitions to next state are written **and/or**
  - ii) the names of the output signal that the control generates while being in the present state.
- State is given a symbolic name (**Upper left corner of the box**), **binary code assigned to state (upper right)**

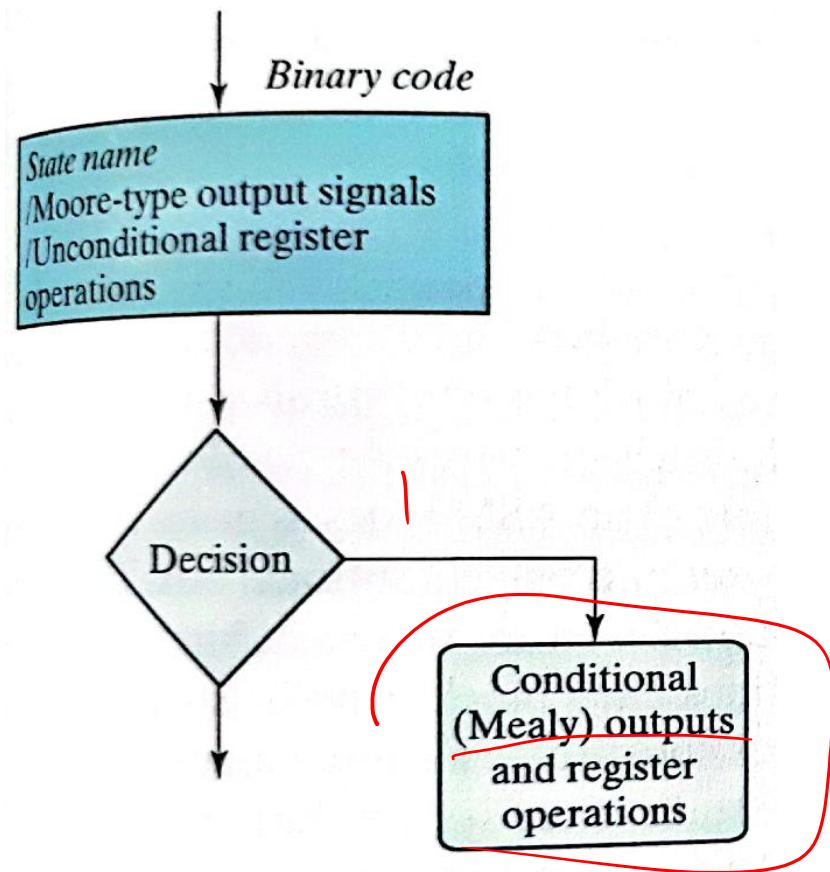
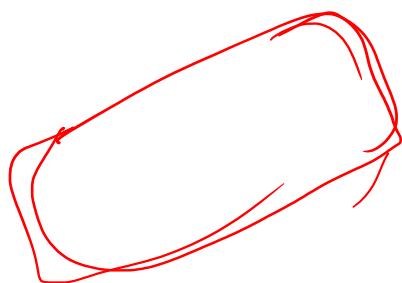


- **Decision box (diamond shaped) indicates the effect of an input on the control subsystem**
- **Condition to be tested is written inside the box**
- **One exit path is taken if the condition is true (label 1) and another if false ( label 0)**

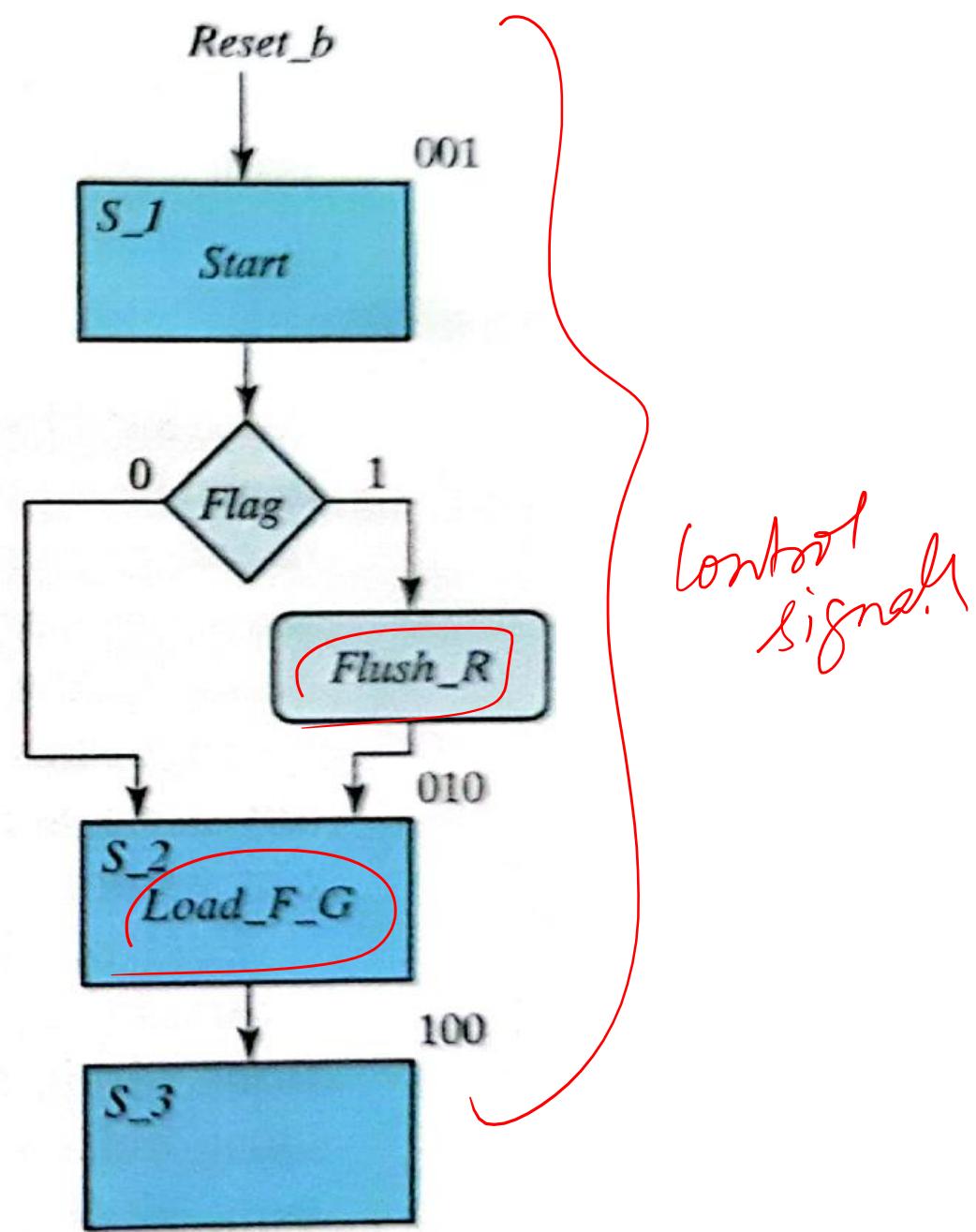
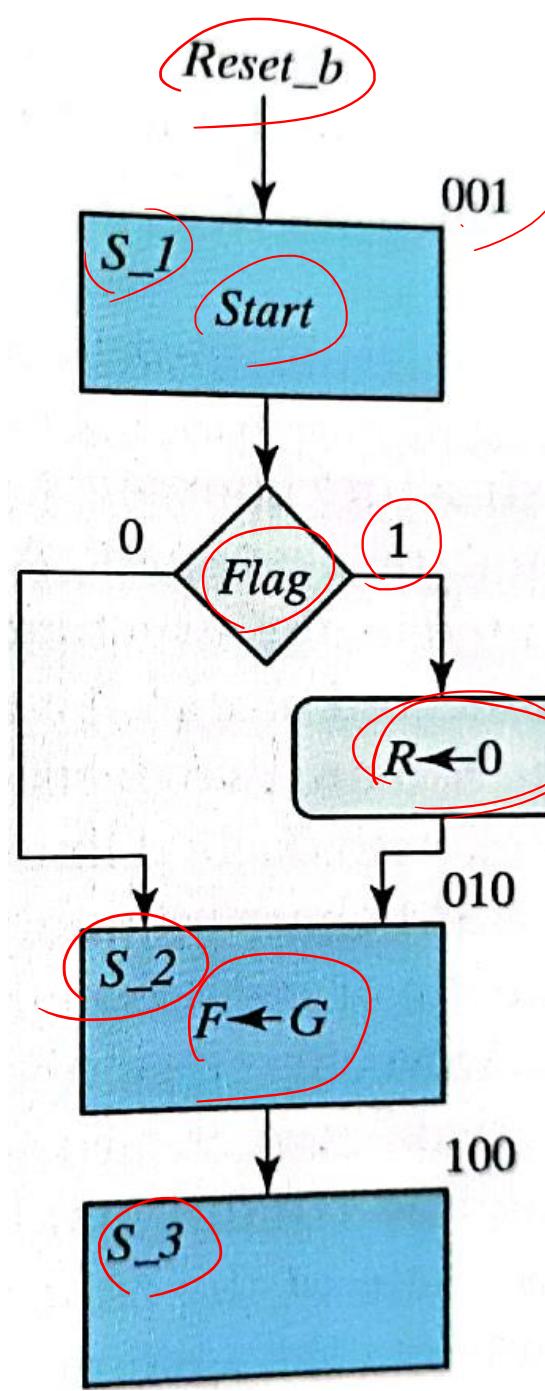


- **State box and decision box are similar to those used in conventional flowchart**

- The conditional box (Shape: rounded corners) is unique to ASM chart
- Input path to conditional box comes from one of exit paths of a decision box
- The outputs listed inside a conditional box are generated during a given state and the register operations are associated with a transition from the state



A SSM  
Chart





# CS/ECE/EEE/INSTR F215:Digital Design



## Lecture 32: ASM

*Sat, 27 Nov 2021*

**BITS** Pilani  
Hyderabad Campus

**Dr. R. N. Ponnalagu, EEE**

*Sometimes we are tested,  
Not to expose our weakness,  
But to discover our strengths.*

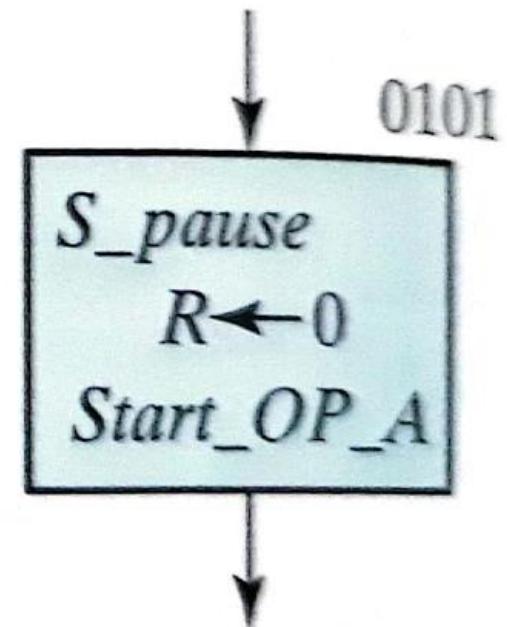
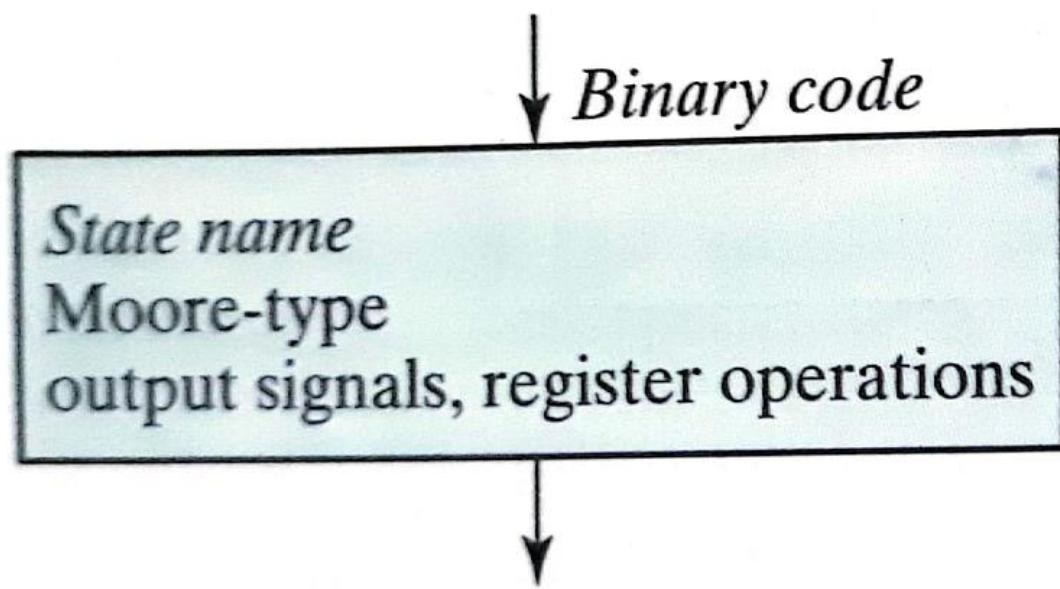
- Flowchart developed to design Digital Hardware Algorithms is **Algorithmic State Machines (ASM) chart**

## **ASM CHART**

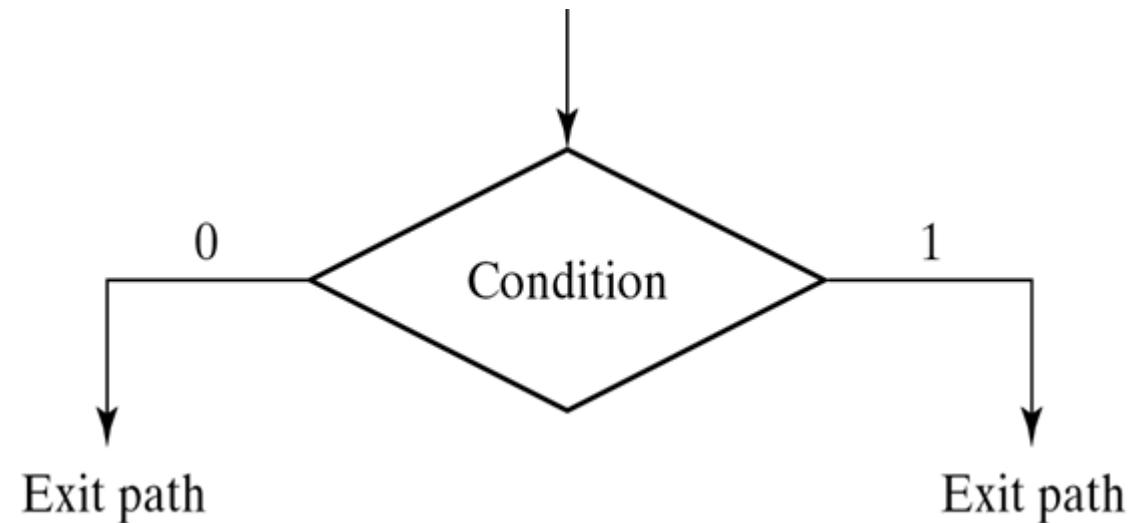
### **composed of three basic elements**

- The State box**
- The decision box**
- The conditional box**

- A state is indicated by the **state box** (Rectangle in shape) within which are written
  - i) register operations that occur when state transitions to next state are written **and/or**
  - ii) the names of the output signal that the control generates while being in the present state.
- State is given a symbolic name (**Upper left corner of the box**), binary code assigned to state (**upper right**)

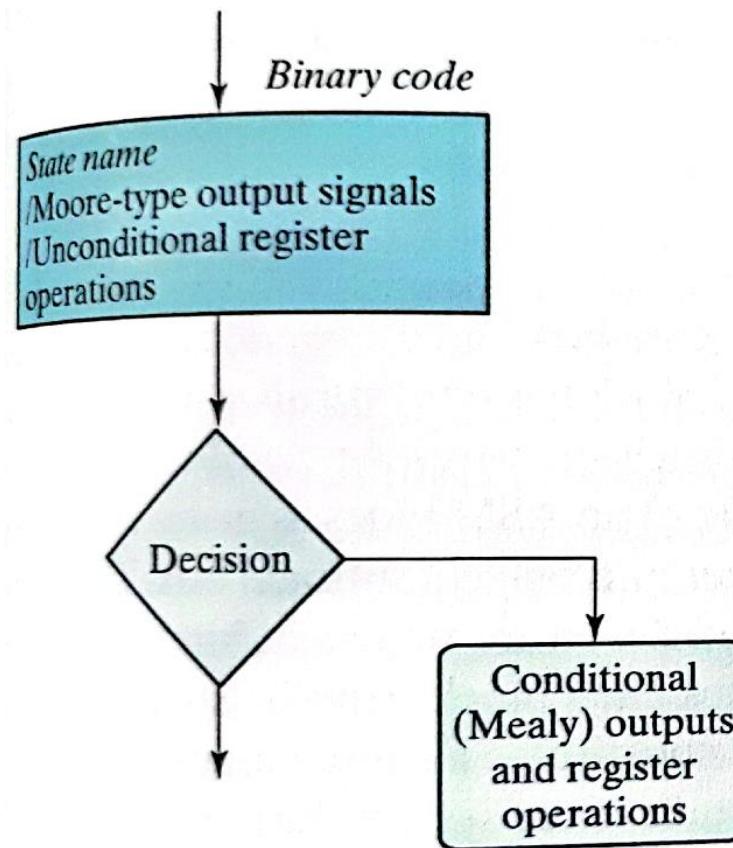


- **Decision box (diamond shaped) indicates the effect of an input on the control subsystem**
- **Condition to be tested is written inside the box**
- **One exit path is taken if the condition is true (label 1) and another if false ( label 0)**

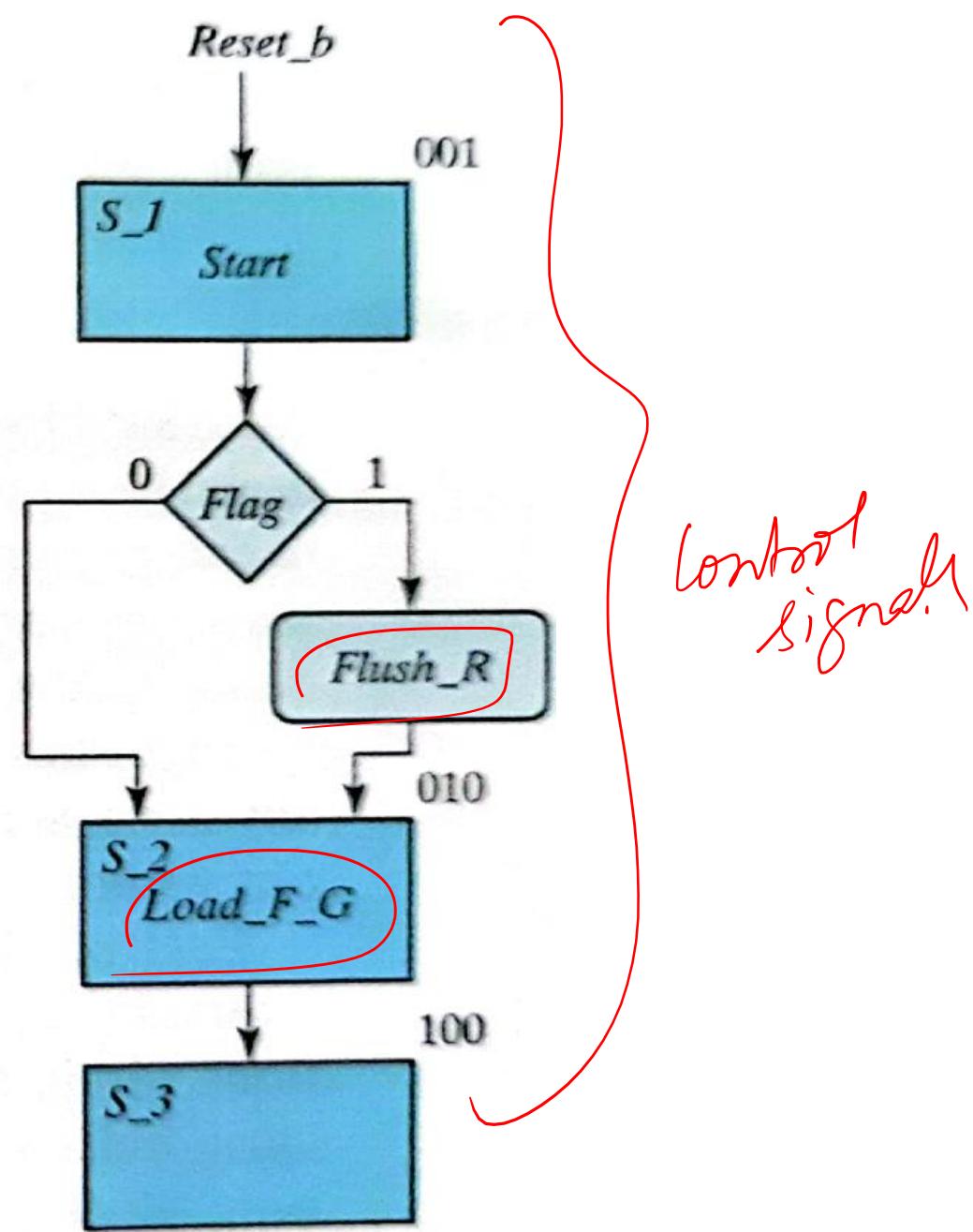
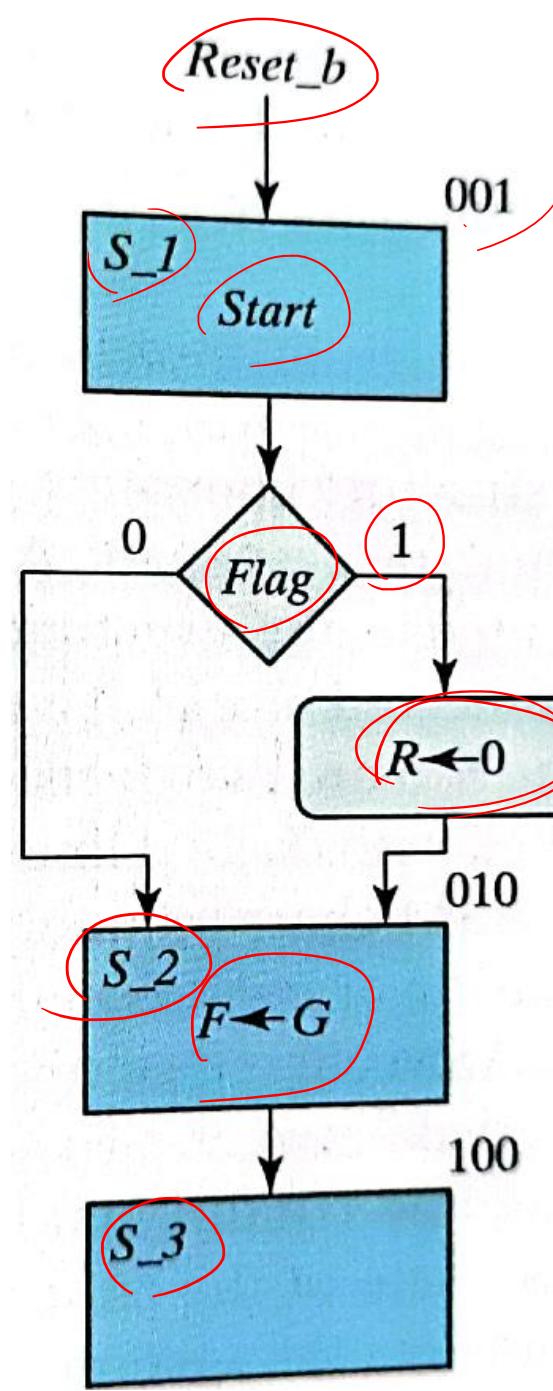


- **State box and decision box are similar to those used in conventional flowchart**

- The conditional box (Shape: rounded corners) is unique to ASM chart
- Input path to conditional box comes from one of exit paths of a decision box
- The outputs listed inside a conditional box are generated during a given state and the register operations are associated with a transition from the state

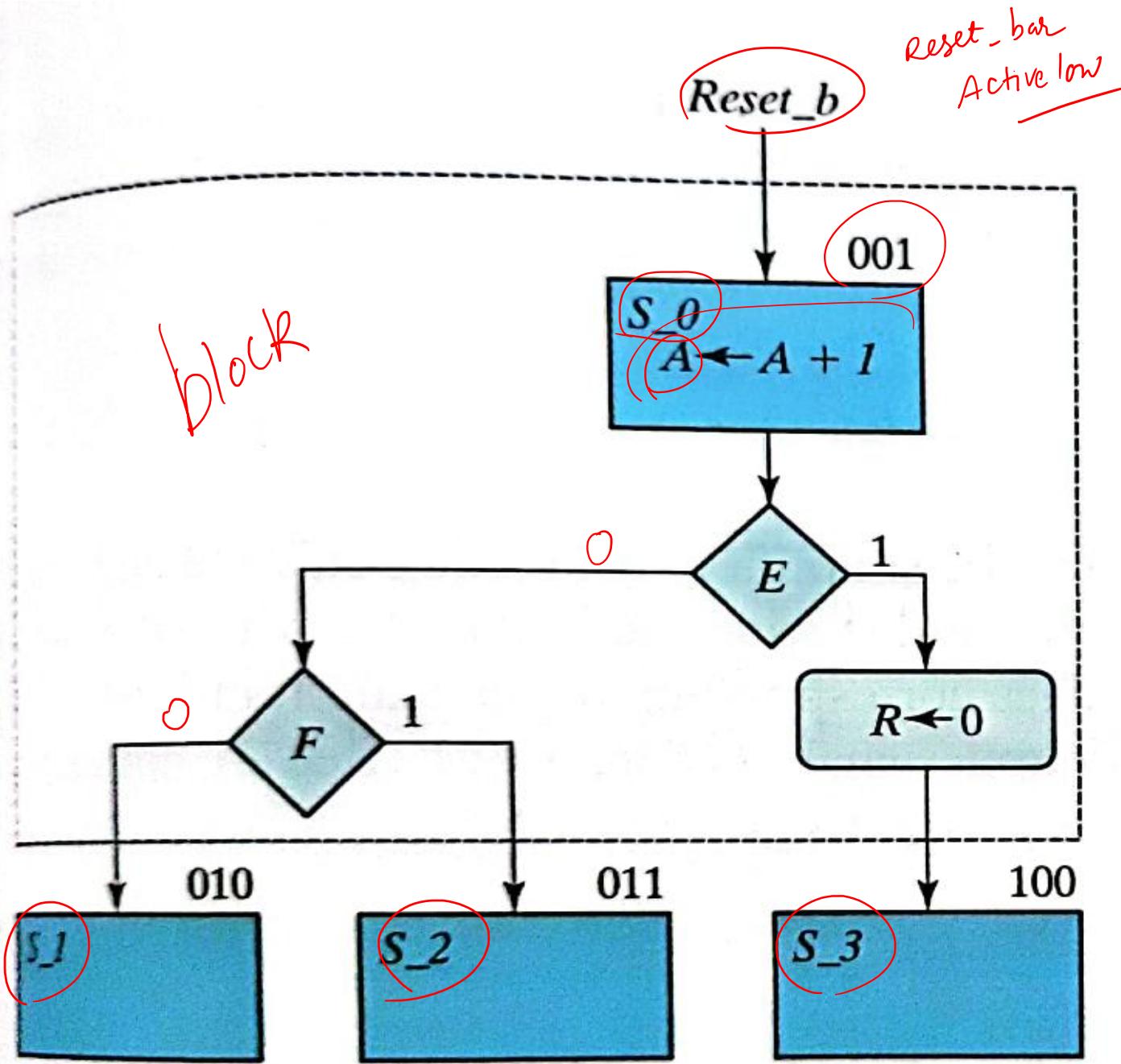


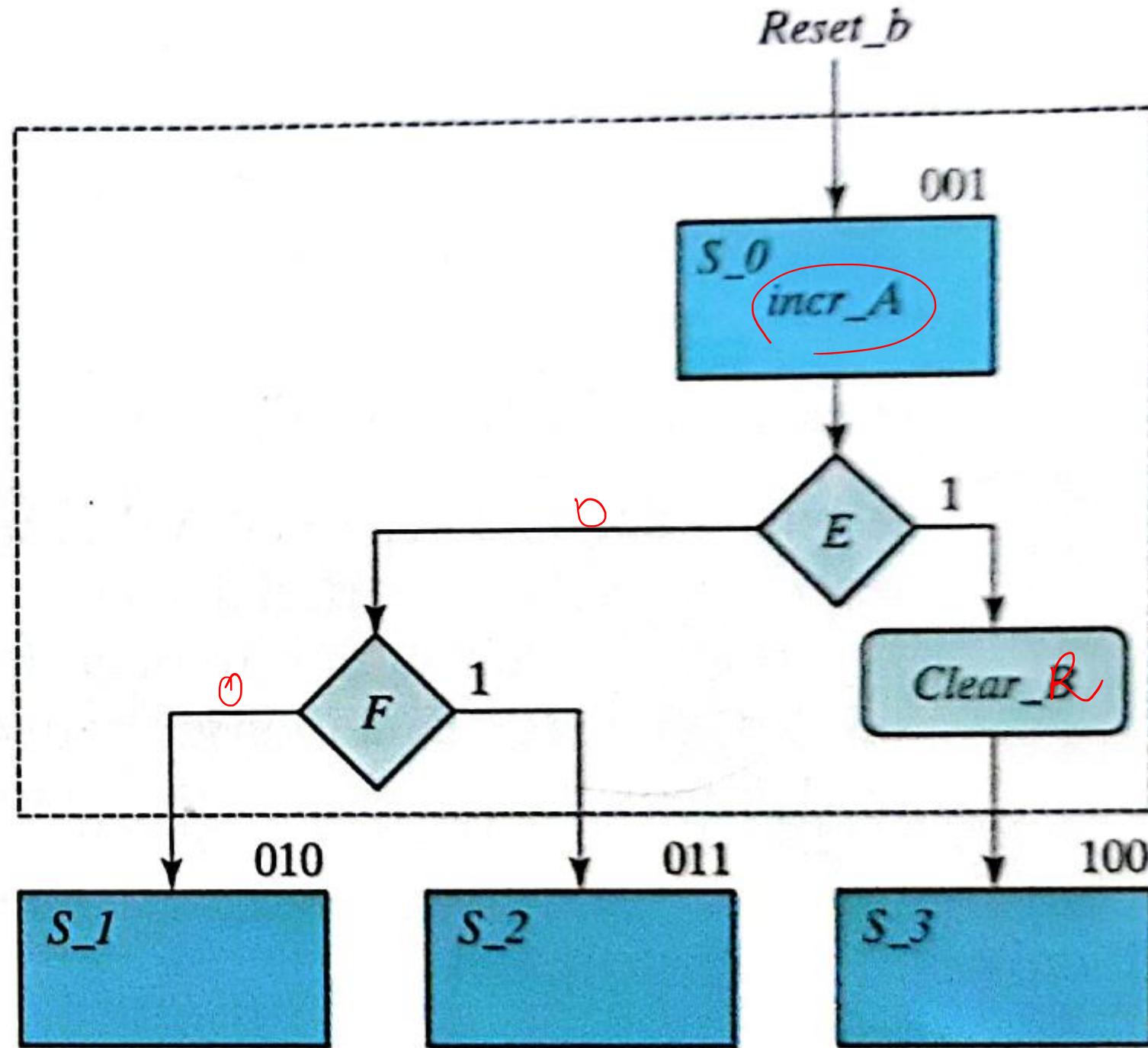
ASM  
Chart

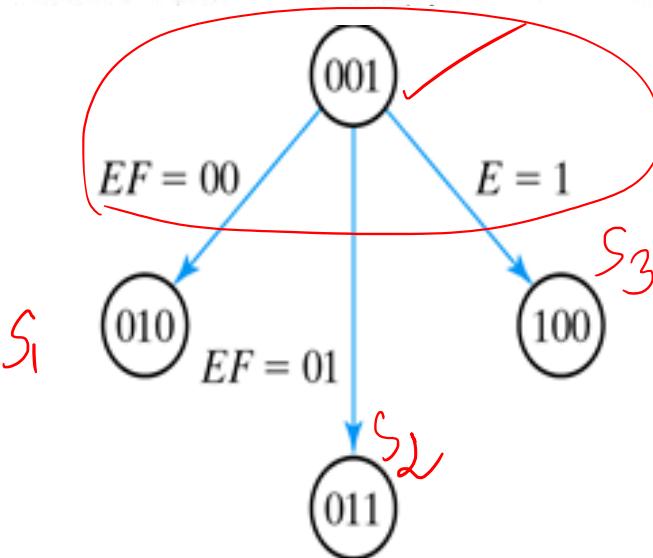
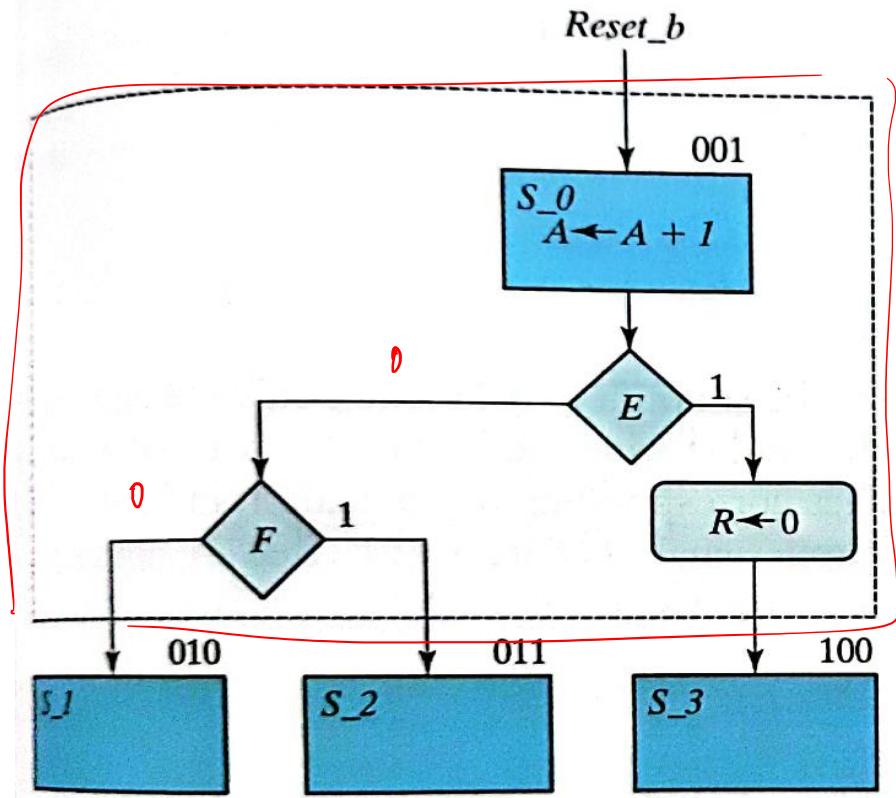


## ASM Block in ASM Chart

- A structure consisting of one state box and all the decision and conditional boxes connected to its exit path.
- An ASM block has one entrance and any number of exit paths represented by the structure of the decision boxes
- ASM Chart may have one or more interconnected blocks.
- Each block in the ASM chart describes the state of the system during one clock pulse. (interval between two successive active edges of clock)
- The operations within the state and conditional boxes are executed with a common clock pulse while the system transits from one state to next.
- Same clock pulse transfers system controller to next states.
- ASM chart is very similar to state diagram.

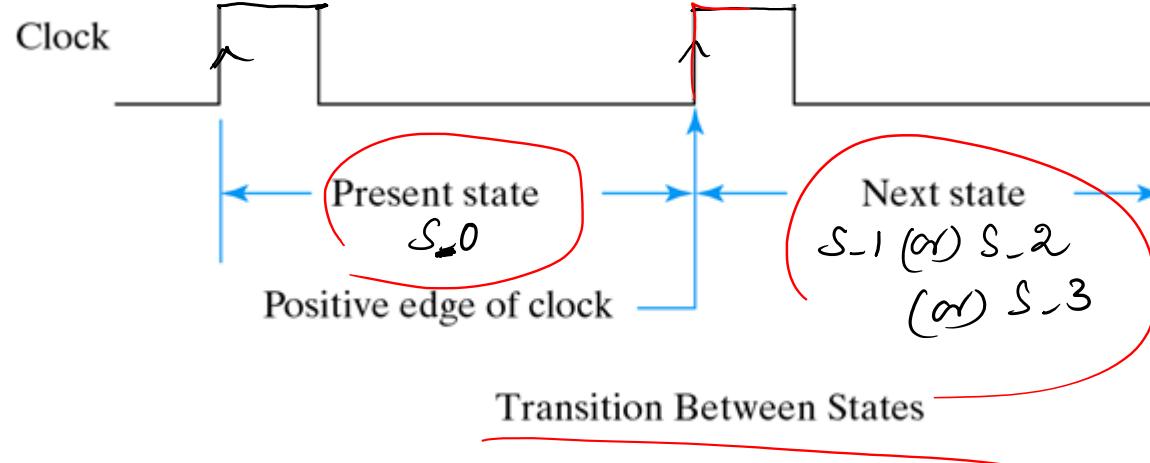






- Each **state block** is equivalent to a state in a sequential circuit.
- **Decision box** is equivalent to the binary information written along the directed lines that connects two states in a state diagram
- ASM chart can be easily converted to state diagram and then proceed with the design
- The unconditional and conditional operations that must be performed in the data path unit are not indicated in the state diagram of the controller

- In a digital system clock pulses are applied not only to registers of the data path but also to all the FFs in the state machine implementing control unit.
- Inputs are also synchronized to clock and change state in response to an edge transition (synchronous sequential circuit)
- Major difference between a conventional flowchart and an ASM chart is interpreting the timing relations among the various operations
- In conventional flowchart the listed operations follow one after the other in a sequence. No concept of timing and synchronization.
- In contrast, ASM chart consider the entire block as one unit.
- All operations within a block must occur in synchronism during clock edge transition.

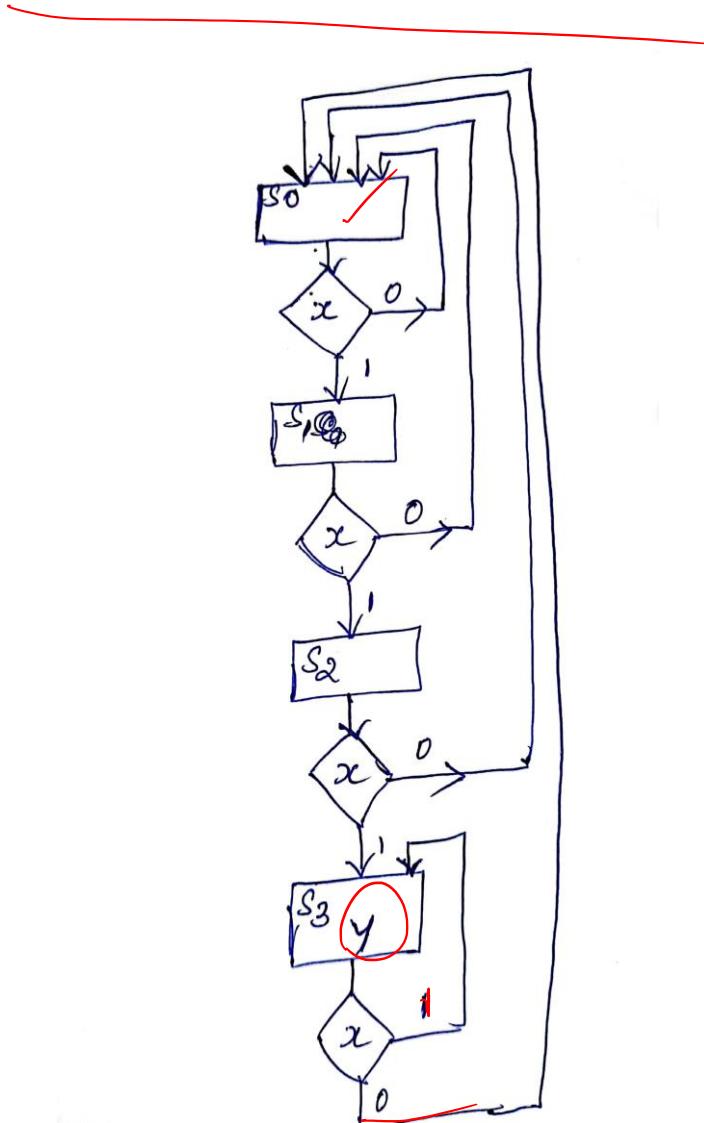
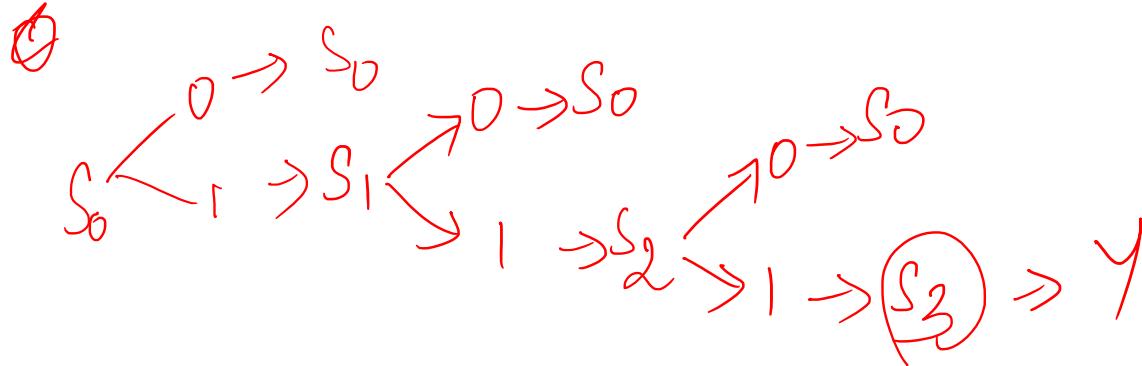


During transition

A is incremented, If  $E = 1$ , R is cleared

Depending on values of E and F control transferred to states  $S_1$  or  $S_2$  or  $S_3$

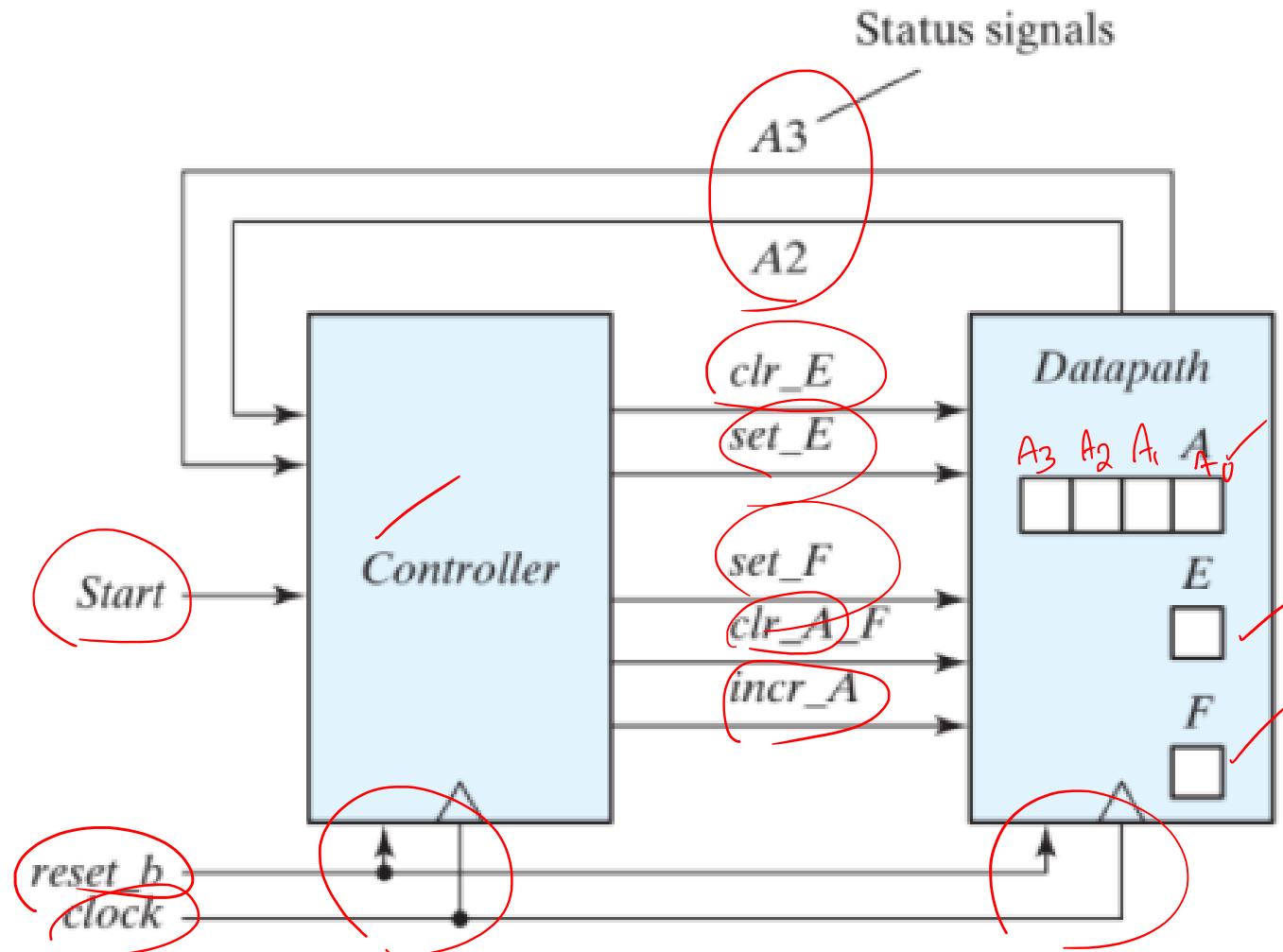
Draw an ASM chart for a synchronous state machine that is to monitor input  $x$  and assert output  $y$  after three consecutive 1s are received and remain asserted until a 0 is received.



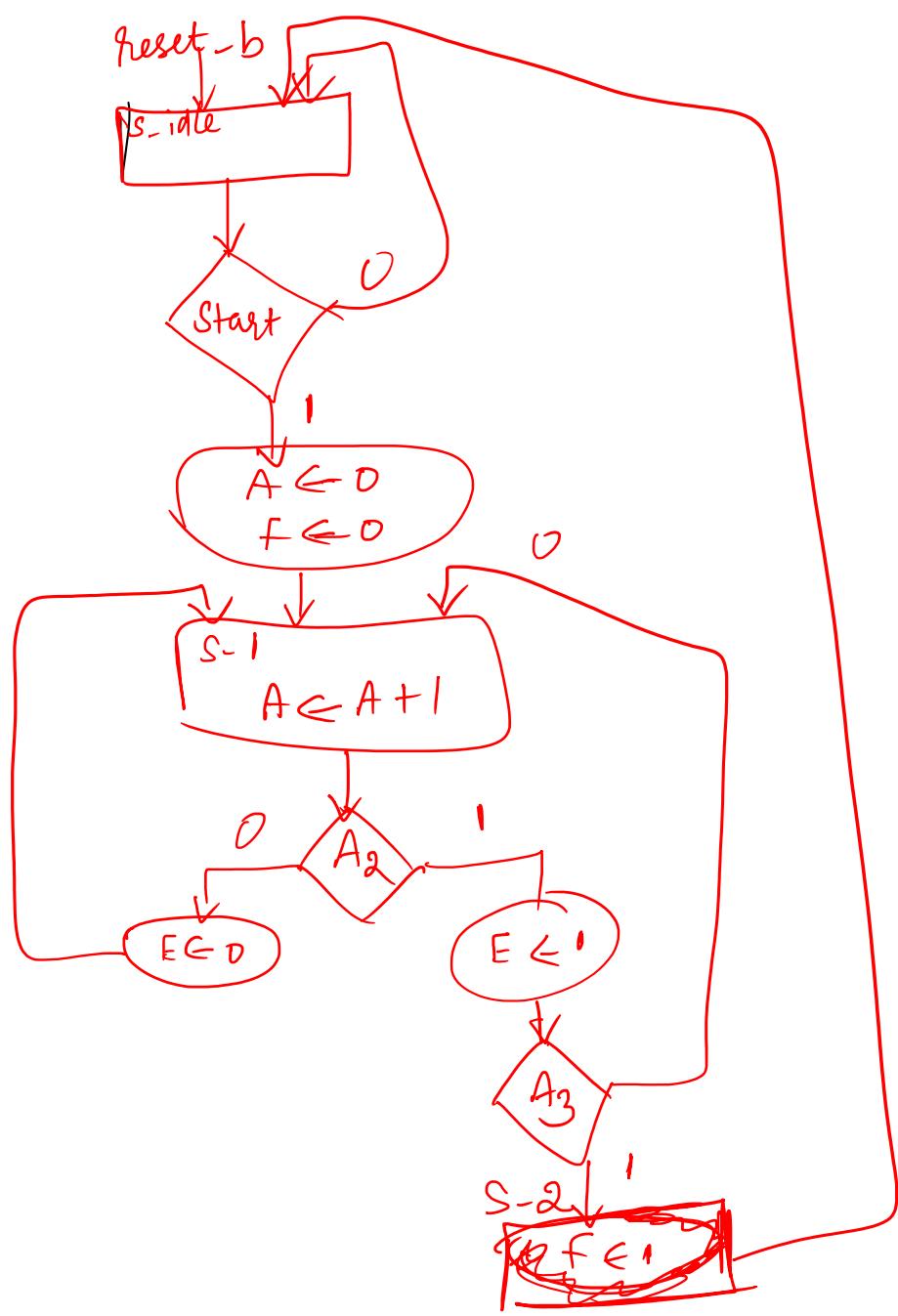
## Example

- Design a Digital system having two FFs **E** and **F**, one 4-bit binary counter **A** (individual FFs :~~A4A3A2A1~~).  
$$\begin{matrix} A_3 & A_2 & A_1 & A_0 \end{matrix}$$
- Initial state is the reset state: **S\_idle** (state reached by application of `reset_b`)
- A **Start signal** initiates the operation by clearing counter A and flip-flop F.
- Counter incremented by one starting from next clock pulse, continues to increment until operations are such that it stops.
- Counter bits **A2** and **A3** determine sequence of operations
  - ✓ If  $A2 = 0$ , E is cleared to 0 and count continues
  - ✓ If  $A2 = 1$ , E set to 1; then if  $A3=0$ , count continues,
  - ✓ but if  $A3 = 1$ , F is set to 1 on next clock pulse and system stops counting.
- If Start = 0 system remains in initial state, but if Start = 1 cycle repeats.

# Block diagram of the design



- When no operations the system is in initial state  $S_{idle}$  awaiting for Start signal
- When input Start = 1, state changes to  $S_1$  and the counter A and flip-flop F are cleared. The register operations occur unconditionally (Moore type).
- Register A is incremented at every clock edge while machine is state  $S_1$ .
- In state  $S_1$  when the counter is incremented with every clock pulse and at the same time, one of three operations occur during clock transition either
  - ✓ E is cleared and control stays in state  $S_1$  ( $A2 = 0$ ); or
  - ✓ E is set and control stays in state  $S_1$  ( $A2A3 = 10$ ); or
  - ✓ E is set and control goes to state  $S_2$  ( $A2A3 = 11$ ).
- Hence block with state  $S_1$  has two decision and two conditional boxes
- In state  $S_2$ , a Moore type control signal is asserted to set flipflop F and state changes to  $S_{idle}$ .



# Sequence of operations

Counter

A3 A2 A1 A0

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 0 1

Flipflops

E

1

0

0

0

F

0

0

0

0

1

0

1

0

1

0

0

0

0

1

0

Conditions

A2=0, A3=0

State

S\_1

#2  
0 0 1 1  
0 0 0 0  
0 0 0 0  
0 1 0 1

A2=1, A3=0

A2=1, A3=0

A2=1, A3=1

S\_2  
S\_idle

TABLE 6.5

*Sequence of Operations for Design Example*

Counter				Flip-Flops		Conditions	State
$A_3$	$A_2$	$A_1$	$A_0$	$E$	$F$		
0	0	0	0	1	0	$A_2 = 0, A_3 = 0$	$S\_1$
0	0	0	1	0	0		
0	0	1	0	0	0		
0	0	1	1	0	0		
0	1	0	0	0	0	$A_2 = 1, A_3 = 0$	
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	1	0		
1	0	0	0	1	0	$A_2 = 0, A_3 = 1$	
1	0	0	1	0	0		
1	0	1	0	0	0		
1	0	1	1	0	0		
1	1	0	0	0	0	$A_2 = 1, A_3 = 1$	$S\_2$
1	1	0	1	1	0		
1	1	0	1	1	1	$S\_idle$	



## CS/ECE/EEE/INSTR F215:Digital Design



### Lecture 33: ASM\_2

*Thu, 02 Dec 2021*

**BITS Pilani**  
Hyderabad Campus

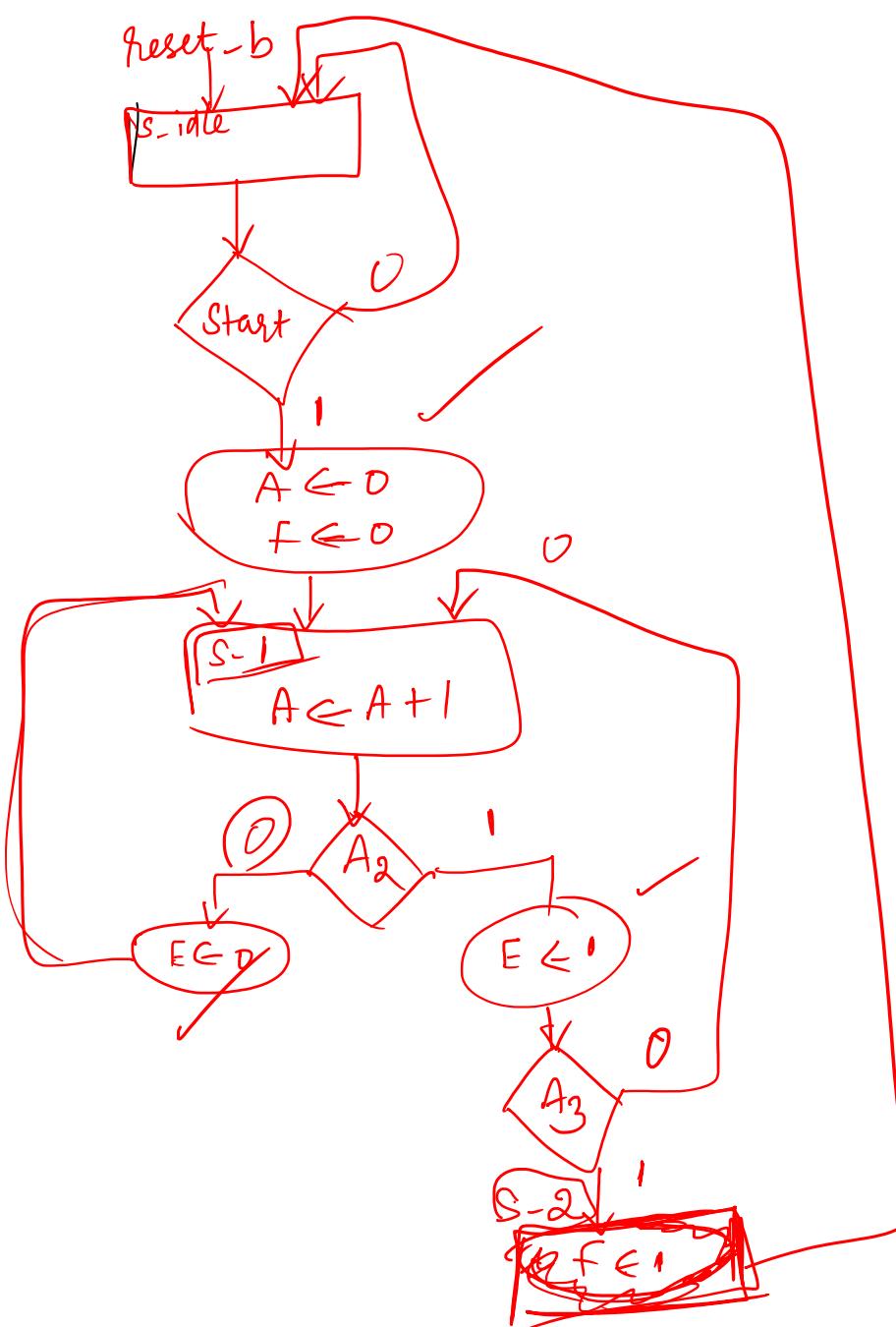
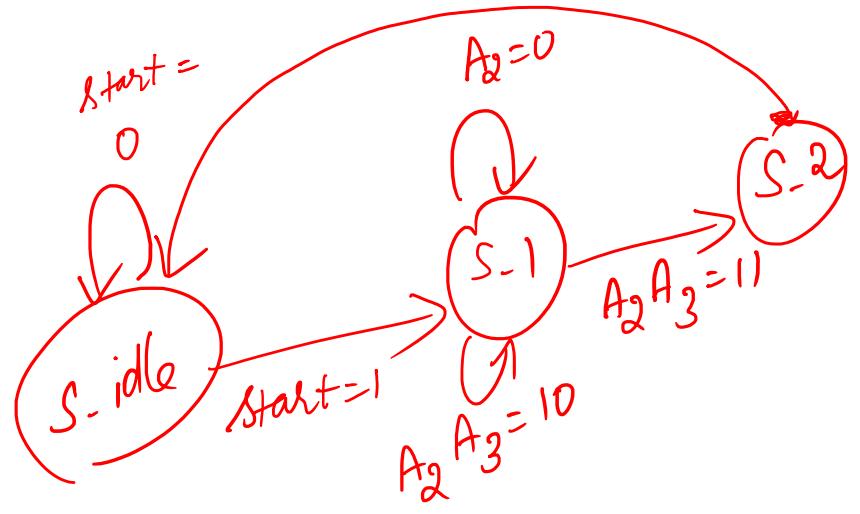
**Dr. R. N. Ponnalagu, EEE**

*“Today could be one day  
closer to success  
A little progress each day adds  
up to big results”.*

## Example

- Design a Digital system having two FFs **E** and **F**, one 4-bit binary counter **A** (individual FFs :A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>).
- Initial state is the reset state: **S\_idle** (state reached by application of reset\_b)
- A **Start signal** initiates the operation by clearing counter A and flip-flop F.
- Counter incremented by one starting from next clock pulse, continues to increment until operations are such that it stops.
- Counter bits A<sub>2</sub> and A<sub>3</sub> determine sequence of operations
  - ✓ If A<sub>2</sub> = 0, E is cleared to 0 and count continues
  - ✓ If A<sub>2</sub> = 1, E set to 1; then if A<sub>3</sub>=0, count continues,
  - ✓ but if A<sub>3</sub> = 1, F is set to 1 on next clock pulse and system stops counting.
- If Start = 0 system remains in initial state, but if Start = 1 cycle repeats.

- When no operations the system is in initial state  $S_{idle}$  awaiting for  $Start$  signal
- When input  $Start = 1$ , state changes to  $S_1$  and the counter A and flip-flop F are cleared. The register operations occur unconditionally (Moore type).
- Register A is incremented at every clock edge while machine is state  $S_1$ .
- In state  $S_1$  when the counter is incremented with every clock pulse and at the same time, one of three operations occur during clock transition either
  - ✓ E is cleared and control stays in state  $S_1$  ( $A2 = 0$ ); or
  - ✓ E is set and control stays in state  $S_1$  ( $A2A3 = 10$ ); or
  - ✓ E is set and control goes to state  $S_2$  ( $A2A3 = 11$ ).
- Hence block with state  $S_1$  has two decision and two conditional boxes
- In state  $S_2$ , a Moore type control signal is asserted to set flipflop F and state changes to  $S_{idle}$ .



# Sequence of operations

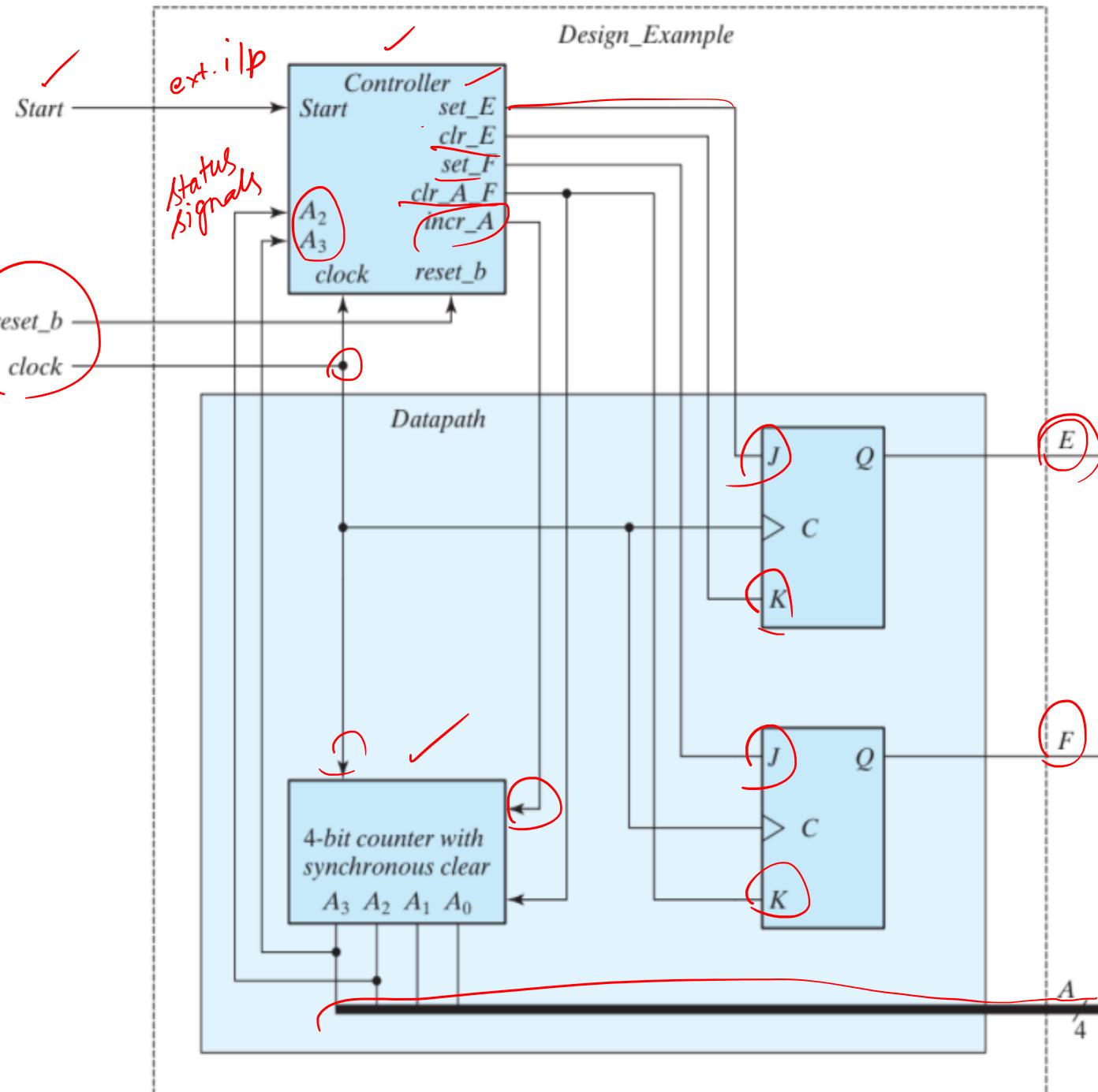
Counter	Flipflops				Conditions	State
A3 A2 A1 A0	E	F				
0 0 0 0	1	0			<u>A2= 0, A3=0</u>	<u>S_1</u>
0 0 0 1	0	0				
0 0 1 0	0	0				
0 0 1 1	0	0				
0 1 0 0	0	0			<u>A2=1, A3= 0</u>	
0 1 0 1	1	0				
0 1 1 0	1	0				
0 1 1 1	1	0				
1 0 0 0	1	0			<u>A2=1, A3= 0</u>	
1 0 0 1	0	0				
1 0 1 0	0	0				
1 0 1 1	0	0				
1 1 0 0	0	0			<u>A2=1, A3= 1</u>	
1 1 0 1	1	0				<u>S_2</u>
1 1 0 1	1	1				<u>S_idle</u>

TABLE 8.5

*Sequence of Operations for Design Example*

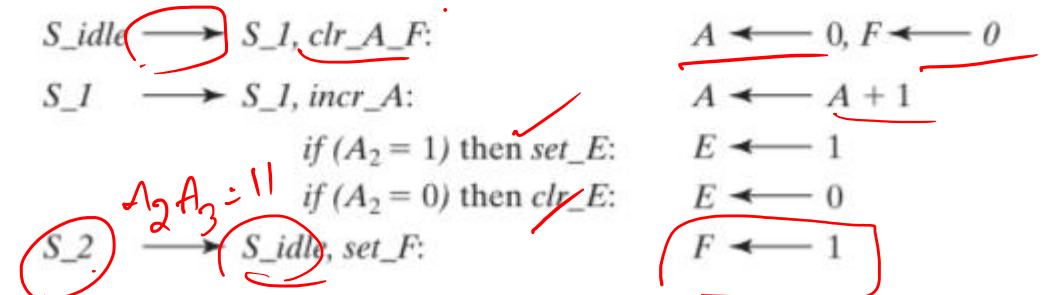
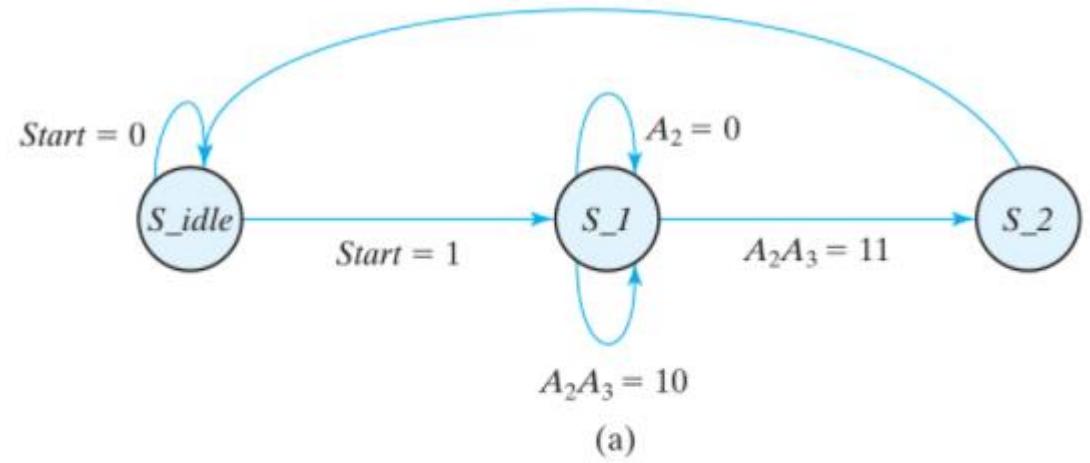
Counter				Flip-Flops		Conditions	State
<b><math>A_3</math></b>	<b><math>A_2</math></b>	<b><math>A_1</math></b>	<b><math>A_0</math></b>	<b><math>E</math></b>	<b><math>F</math></b>		
0	0	0	0	1	0	$A_2 = 0, A_3 = 0$	<i>S_1</i>
0	0	0	1	0	0		
0	0	1	0	0	0		
0	0	1	1	0	0		
<hr/>							
0	1	0	0	0	0	$A_2 = 1, A_3 = 0$	
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	1	0		
<hr/>							
1	0	0	0	1	0	$A_2 = 0, A_3 = 1$	
1	0	0	1	0	0		
1	0	1	0	0	0		
1	0	1	1	0	0		
<hr/>							
1	1	0	0	0	0	$A_2 = 1, A_3 = 1$	
<hr/>							
1	1	0	1	1	0		<i>S_2</i>
<hr/>							
1	1	0	1	1	1		<i>S_idle</i>

- The data path unit consists of a four-bit binary counter and two JK flip-flops.
- The counter is incremented with every clock pulse when the controller state is  $S_1$ .
- It is cleared only when control is at state  $S_{\text{idle}}$  and Start is equal to 1.
- The logic for the signal  $\text{clr}_A.F$  will be included in the controller and requires an AND gate to guarantee that both conditions are present.
- Similarly, we can anticipate that the controller will use AND gates to form signals  $\text{set}_E$  and  $\text{clr}_E$ .
- Depending on whether the controller is in state  $S_1$  and whether  $A_2$  is asserted,  $\text{set}_F$  controls flip-flop F and is asserted unconditionally during state  $S_2$ .
- Note that all flip-flops and registers, including the flip-flops in the control unit, use a common clock.



Design of logic circuit of the controller

# State diagram



# State Table

Present-State Symbol	Present State		Inputs		Next State		Outputs					
	$G_1$	$G_0$	Start	$A_2$	$A_3$	$G_1$	$G_0$	set_E	clr_E	set_F	clr_A_F	incr_A
$S_{idle}$	0	0	0	X	X	0	0	0	0	0	0	0
$S_{idle}$	0	0	1	X	X	0	1	0	0	0	1	0
$S_1$	0	1	X	0	X	0	1	0	1	0	0	1
$S_1$	0	1	X	1	0	0	1	1	0	0	0	1
$S_1$	0	1	X	1	1	1	1	1	0	0	0	1
$S_2$	1	1	X	X	X	0	0	0	0	1	0	0

$S_{idle} \rightarrow 00$   
 $S_1 \rightarrow 01$   
 $S_2 \rightarrow 11$

~~$S_{idle} = S_1 + S_2 D_{G_1} + D_{G_0}$~~

$D_{G_1}$	$D_{G_0}$
0	0
0	1
0	1
0	1
0	0
0	0

$$S_1 A_2 = \overline{G_1} G_0 A_2$$

$$S_1 \overline{A_2} = \overline{G_1} \overline{G_0} \overline{A_2}$$

Set\_E

clr\_E

Set\_F

clr\_A\_F

incr\_A =  $S_1 = \overline{G_1} G_0$

$$D_{G_1} S_1 A_2 A_3$$

$$D_{G_0} = S_{idle} Start + S_1$$

Set

$$D_{G_1} = S_1 A_2 A_3$$

$$D_{G_0} = Start S_{idle} + S_1$$

$$set_E = S_1 A_2$$

$$clr_E = S_1 A_2'$$

$$set_F = S_2$$

$$clr_A_F = Start S_{idle}$$

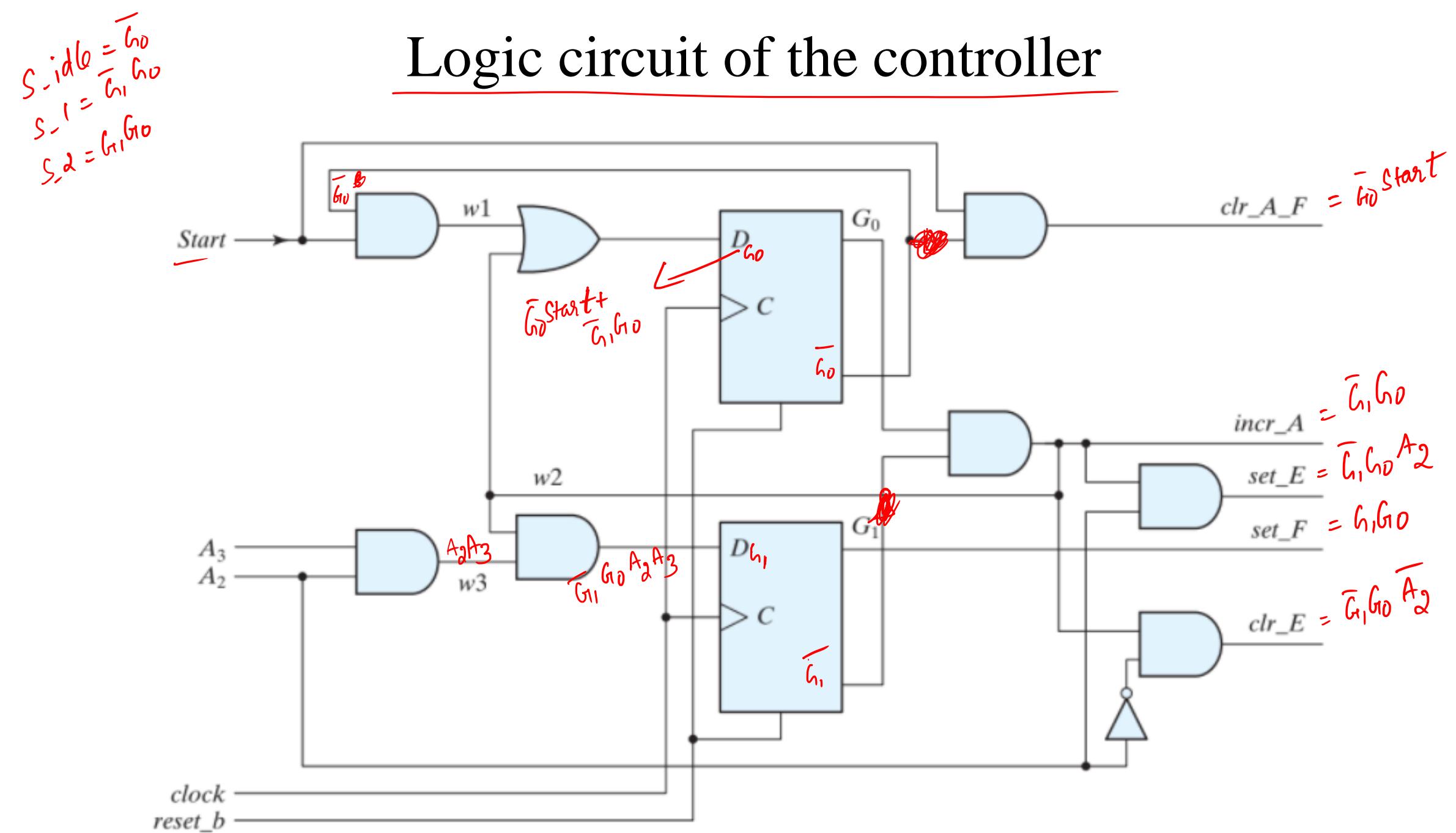
$$incr_A = S_1$$

$$S_2 = G_1 G_0$$

$$S_1 = \overline{G_1} \overline{G_0}$$

$$S_{idle} = \overline{G_1} \overline{G_0} \overline{G_1} \overline{G_0}$$

# Logic circuit of the controller



# BINARY MULTIPLIER

To Multiply two unsigned binary numbers.

Sequential Multiplier

Uses One adder and a shift register.

Less hardware but takes more clock cycles to complete the operation

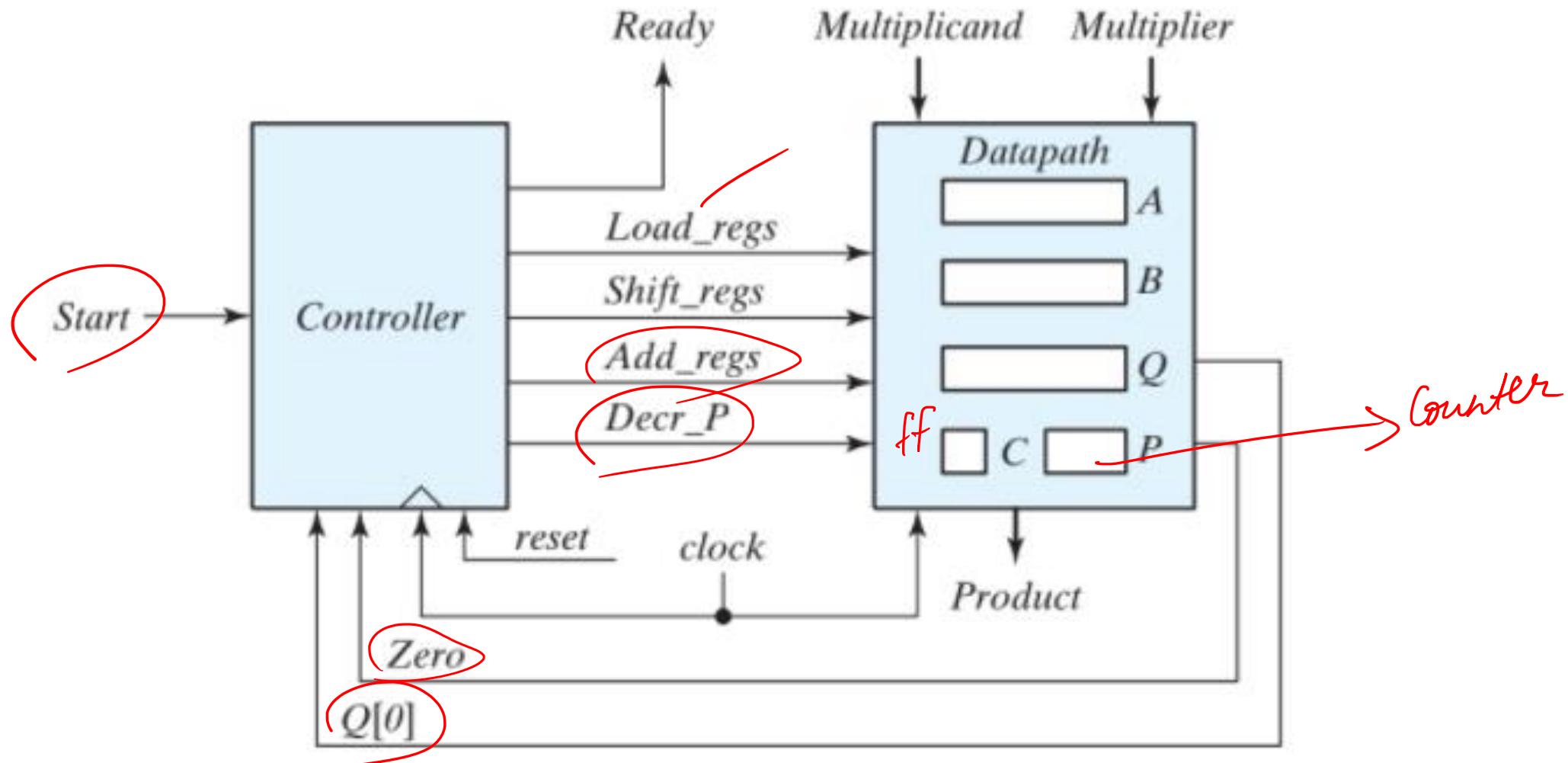
The process consists of successively adding and shifting copies of the multiplicand.

The product obtained from the multiplication of two binary numbers of n bits each can have up to 2 n bits.

multiply the two binary numbers 10111 and 10011:

$$\begin{array}{r} 23 & 10111 \text{ multiplicand} \\ 19 & \underline{10011} \text{ multiplier} \\ & 10111 \\ & \underline{10111} \\ & 00000 \\ & \underline{00000} \\ & 00000 \\ & \underline{10111} \\ & 110110101 \text{ product} \\ \hline 437 & \end{array}$$

# Block diagram





# CS/ECE/EEE/INSTR F215:Digital Design



## Lecture 34: ASM\_3

*Sat, 04 Dec 2021*

**BITS Pilani**  
Hyderabad Campus

**Dr. R. N. Ponnalagu, EEE**

# Mistakes

*Easy to judge when others do it*

*Difficult to realize when we do it.*

*Never Lose Heart*

*Remember*

*The best view comes  
after the hardest climb*

# BINARY MULTIPLIER

To Multiply two unsigned binary numbers.

Sequential Multiplier

Uses One adder and a shift register.

Less hardware but takes more clock cycles to complete the operation

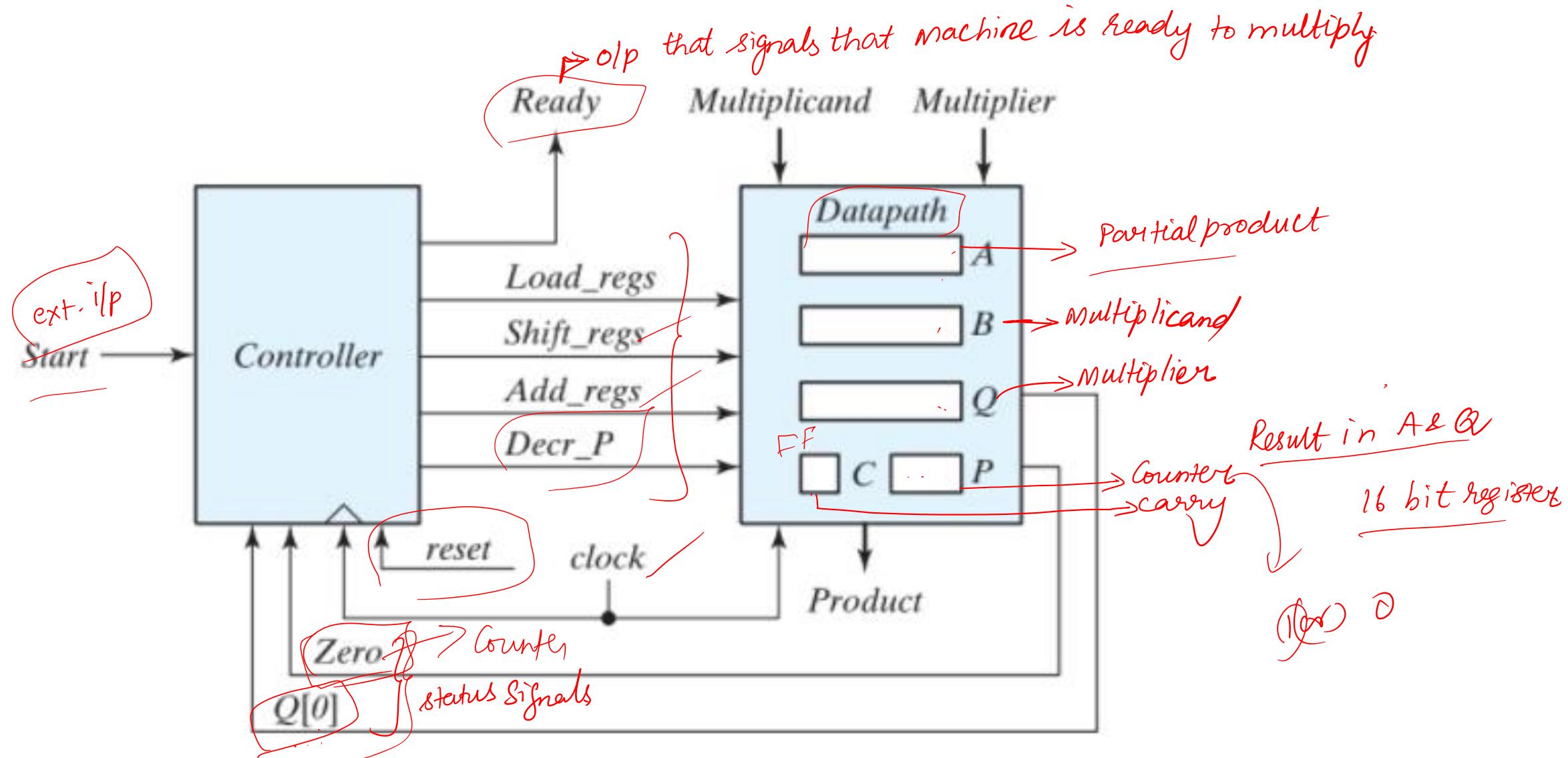
The process consists of successively adding and shifting copies of the multiplicand.

multiply the two binary numbers 10111 and 10011:

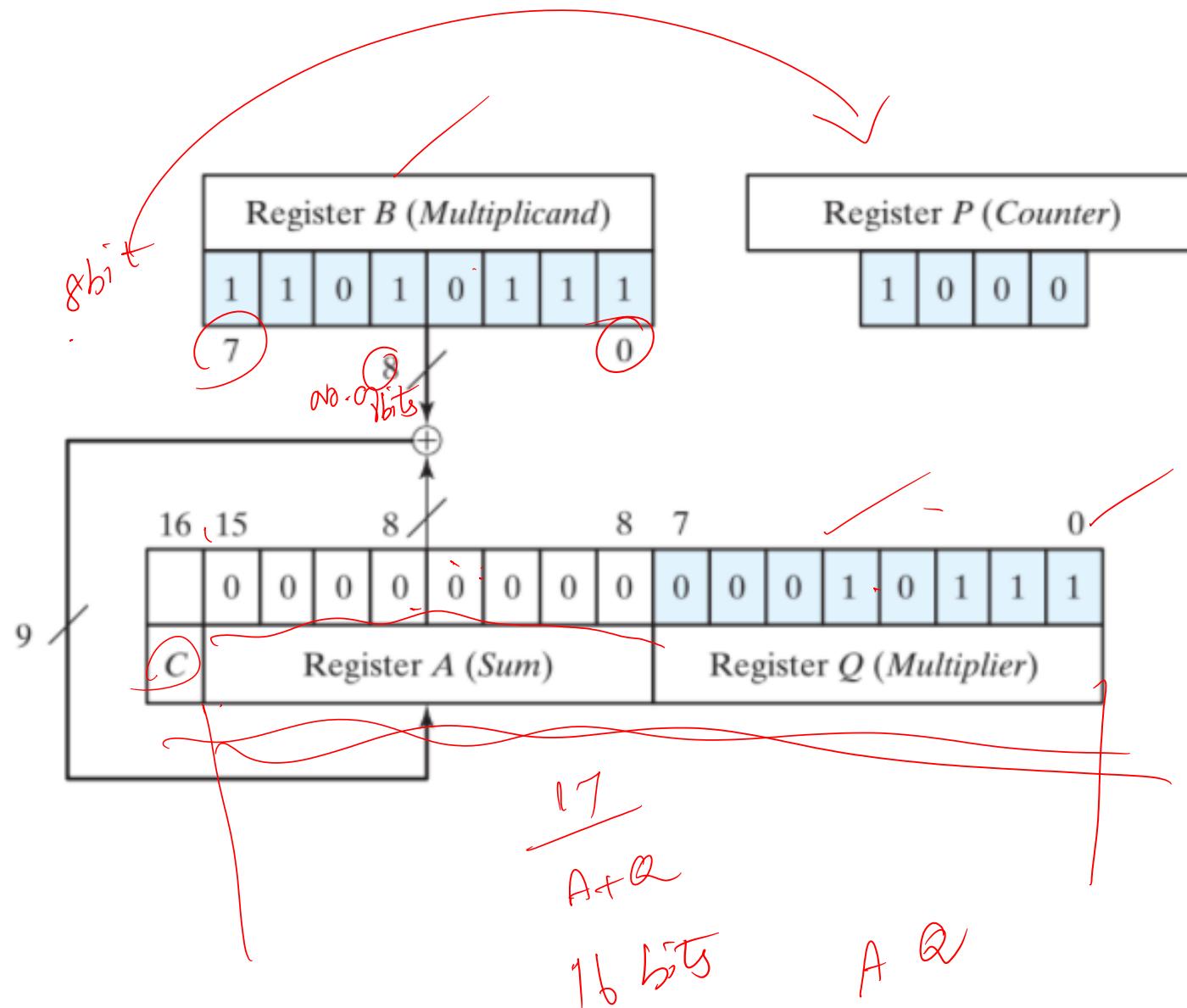
The product obtained from the multiplication of two binary numbers of n bits each can have up to 2 n bits.

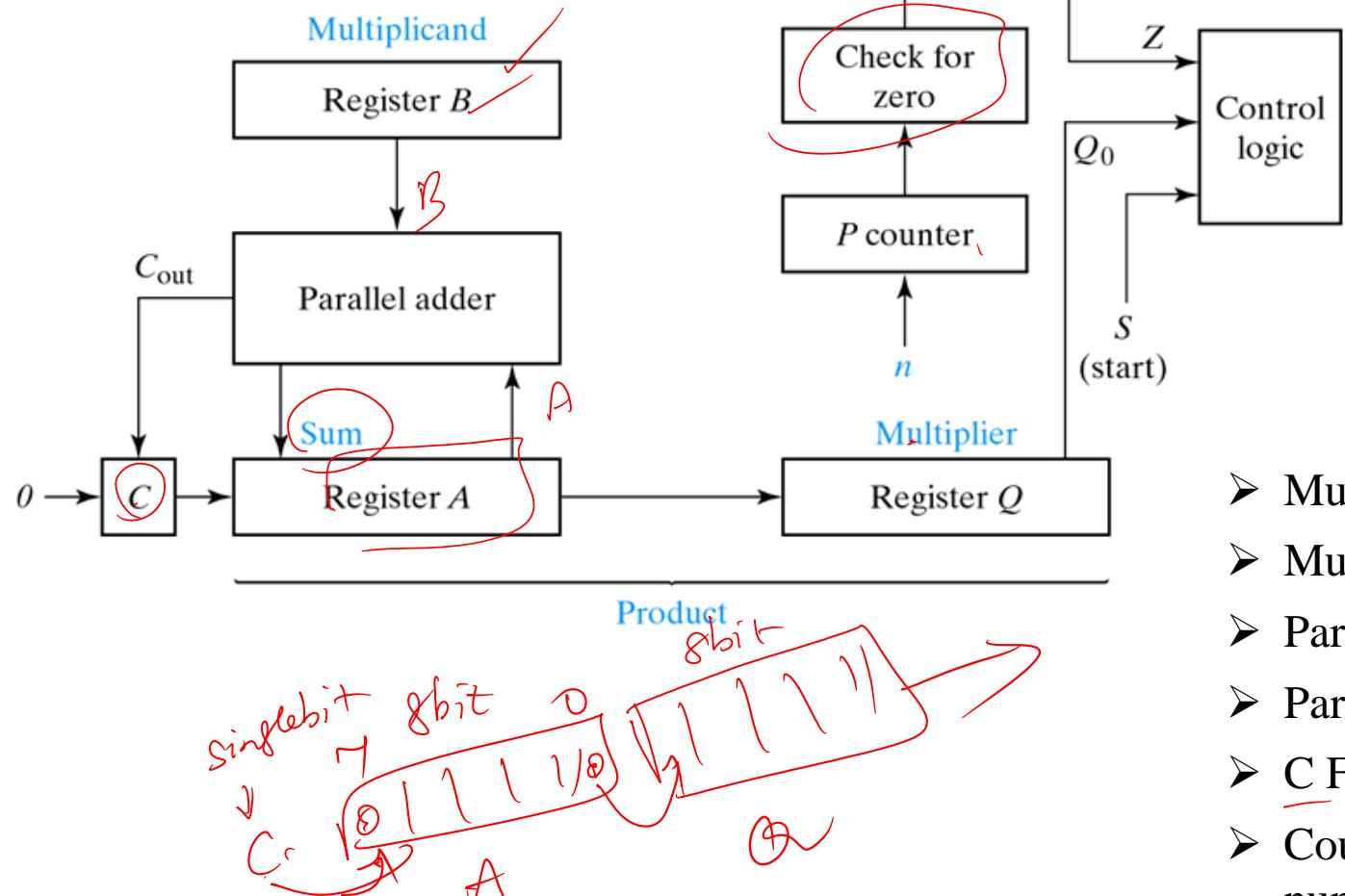
$$\begin{array}{r} 23 & 10111 \text{ multiplicand} \\ \underline{19} & \cancel{10011} \text{ multiplier} \\ & \cancel{10111} \\ & 10111 \\ & \underline{00000} \\ & 00000 \\ & 10111 \\ \hline 437 & \underline{0110110101} \text{ product} \end{array}$$

# Block diagram of Sequential Multiplier



# Register configuration of the data path unit



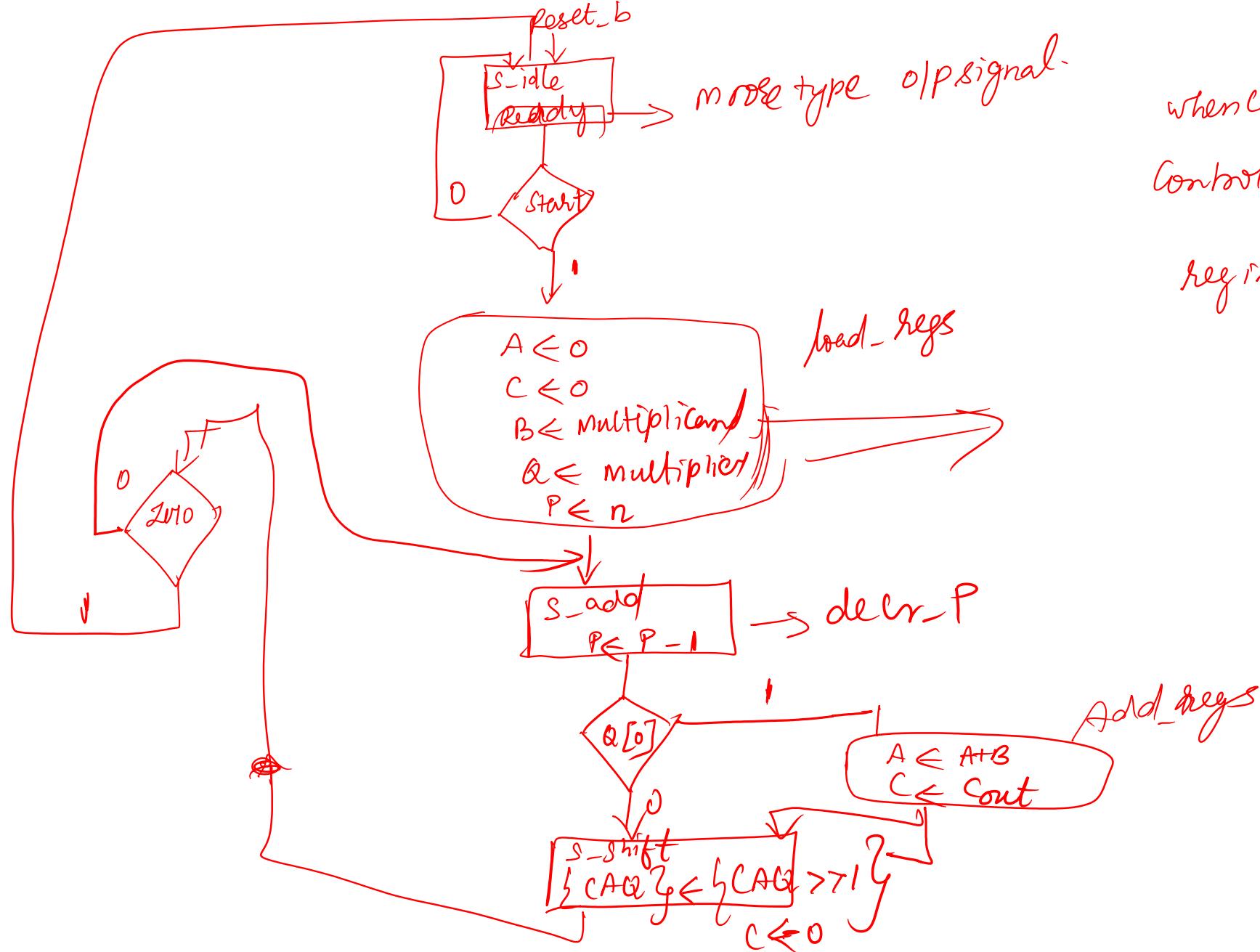


- Multiplicand in Register B
- Multiplier in register Q
- Partial product formed in register A and stored in A and Q
- Parallel adder adds the contents of register B to A
- C Flip-flop stores the carry resulting from addition
- Counter P initially set to hold a binary number equal to the number of bits in the multiplier
- Counter P is decremented after formation of each partial product
- When Counter is zero, the product is formed in the double register A and Q and the process stops

- Initial state S-idle until start signal (external input) is 1.
- After Ready signal from controller Machine starts to perform multiplication.
- Sum of A and B forms the ‘n’ most significant bits of the partial product which is transferred to A from parallel adder
- Output carry after addition (whether a 0 or 1) is transferred to C flip-flop C A Q  $\rightarrow$
- The partial formed in register A and stored in A and Q are shifted to right
- LSB of A is shifted to MSB of Q, carry C is shifted into MSB of A and a 0 is shifted into C FF
- After shifting right, one bit of the partial product is transferred into Q while the multiplier bits in Q are shifted one position right
- LSB of register Q , Q[0] holds the bit of the multiplier that must be inspected next
- Control logic determines whether to add or not on the basis of Q[0]
- Q[0] from multiplier register & Zero signal from the counter P are status signals for control unit
- Based on Start, Q[0] and Zero signal controller generates five output control signals, Ready, Load\_regs, Add\_regs, Shift\_regs, Decr\_P

- Based on Start, Q[0] and Zero signal controller generates five output control signals, Ready, Load\_regs, Add\_regs, Shift\_regs, Decr\_P

# ASM Chart



when clock is available  
 Control transfers to  
s\_add  
 register A & FF C  
 are cleared

## Numerical Example For Binary Multiplier

**Multiplicand  $B = \underline{10111}_2 = 17_H = 23_{10}$**

Multiplier in  $Q$

$Q_0 = 1$ ; add  $B$

First partial product

Shift right  $CAQ$

$Q_0 = 1$ ; add  $B$

Second partial product

Shift right  $CAQ$

$Q_0 = 0$ ; shift right  $CAQ$

$Q_0 = 0$ ; shift right  $CAQ$

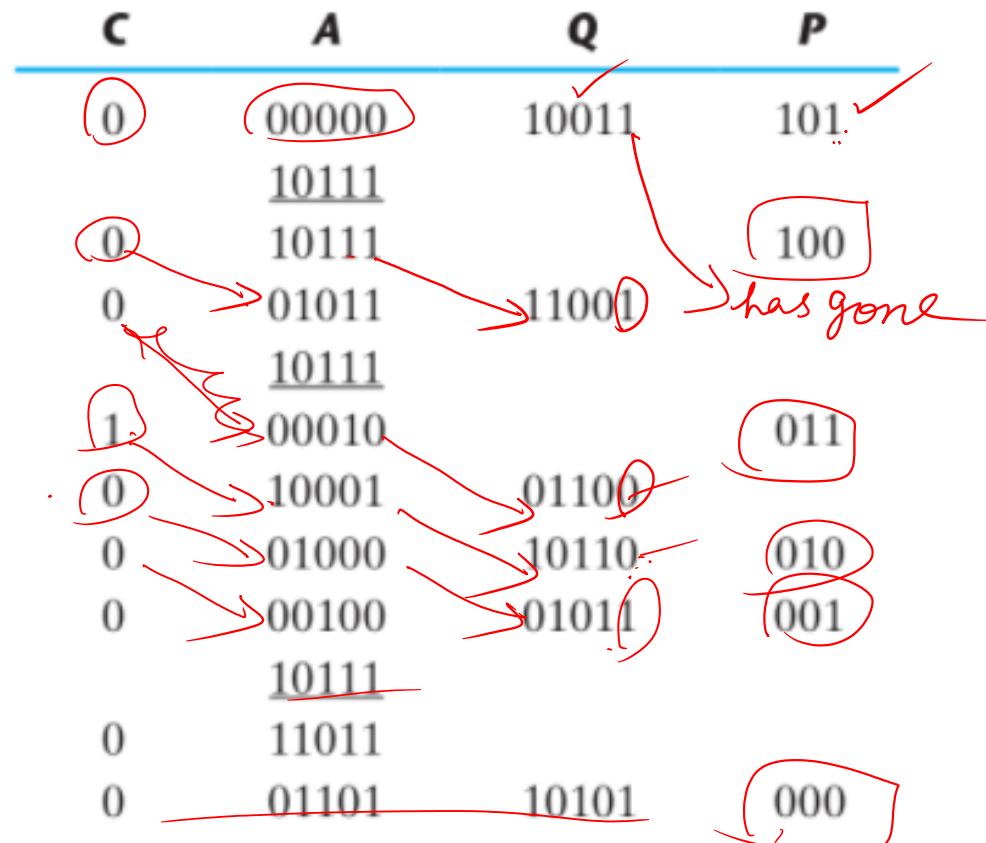
$Q_0 = 1$ ; add  $B$

Fifth partial product

Shift right  $CAQ$

Final product in  $AQ = \underline{\underline{0110110101}}_2 = 1b5_H$

**Multiplier  $Q = \underline{\underline{10011}}_2 = 13_H = 19_{10}$**

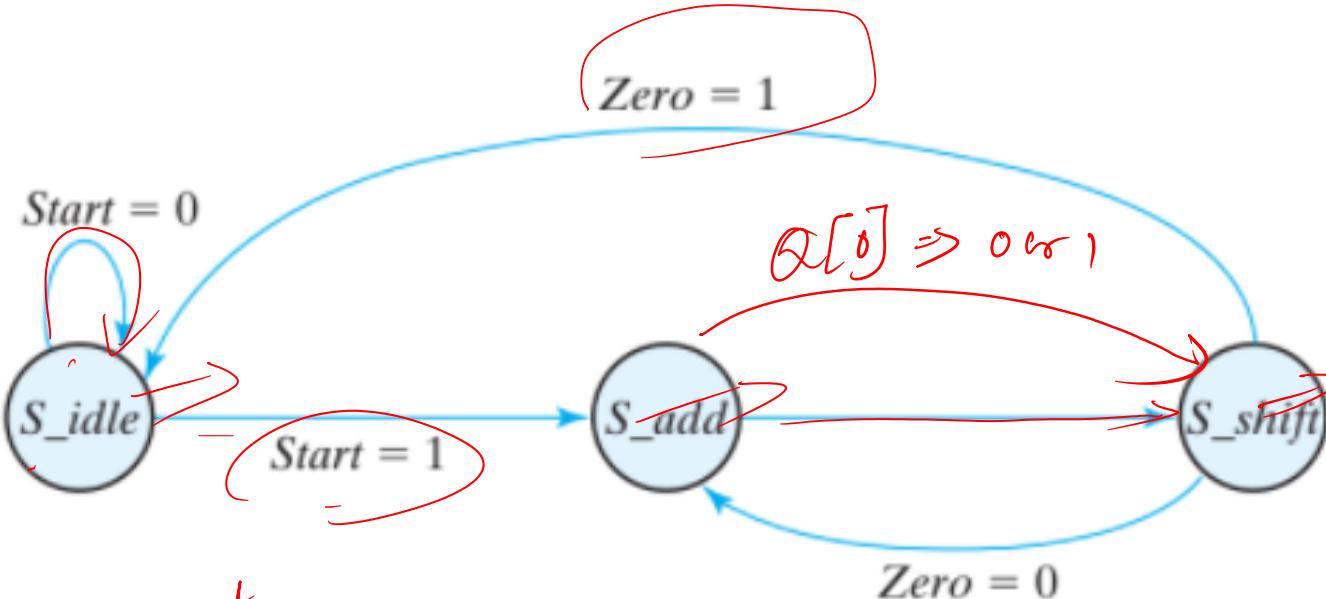


3 States

start

zero

$Q[0]$



State assignment

Binary

Gray code

one-hot assignment

	Binary	Gray	One-hot	
$s_{idle}$	00	00	001	3 bits
$s_{add}$	01	01	010	3 states $\underline{3 \text{ FFs}}$
$s_{shift}$	10	11	100	
	2 FFs	2 FFs	3 FFs	
specialized methods for control logic design				

2 methods

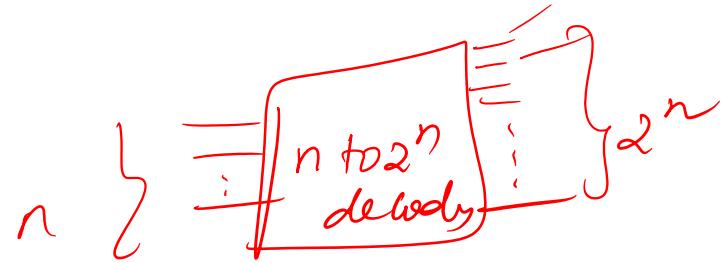
- 1) sequence register & a decoder
- 2) one ff per state  $\rightarrow$  one hot assignment

## Sequence register & decoder

ASMchart 3 states      3 ips

Binary state assignment    00, 01, 10

2 FFs for the register and 2 to 4 line decoder



Next State  $G_1 = D_{G_1}$   
 when Present state is 01  $\rightarrow S\_add$   
 $D_{G_1} = T_1$

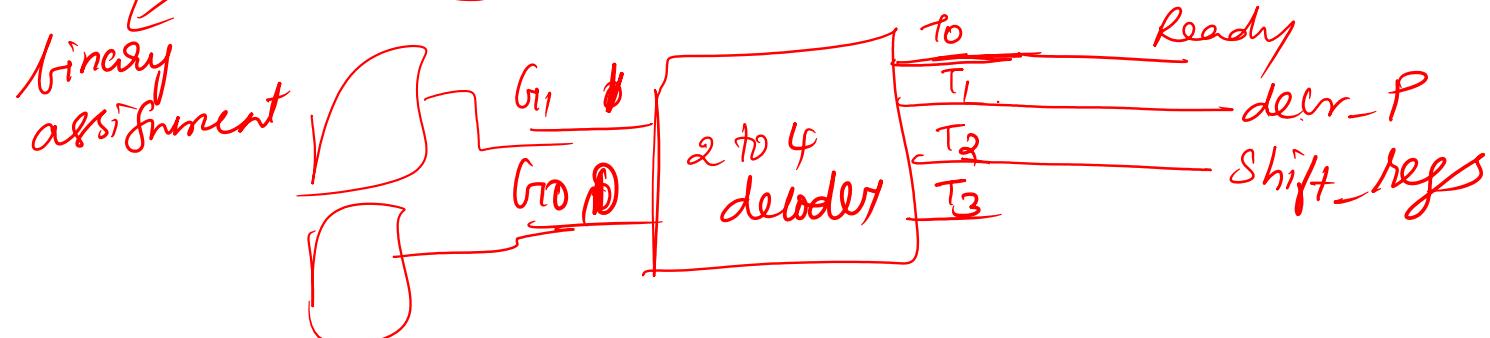
2 FFs for register

State Table for Control Circuit

Next State  $G_0 \leftarrow D_{G_0}$   
 $D_{G_0} = S\_idle \& Start + S\_Shift \& Zero = 0$   
 $= T_0 \text{ Start} + T_2 \overline{\text{Zero}}$

Mosse type  
 D type  
 FF

Present State	Inputs			Next State		O/P						
Present-State Symbol	$G_1$	$G_0$	Start	$Q[0]$	Zero	$G_1$	$G_0$	Ready	Load_regs	Decr_P	Add_regs	Shift_regs
$S\_idle$	0	0	0	X	X	0	0	1	0	0	0	0
$S\_idle$	0	0	1	X	X	0	1	1	1	0	0	0
$S\_add$	0	1	X	0	X	1	0	0	0	1	0	0
$S\_add$	0	1	X	1	X	1	0	0	0	1	1	0
$S\_shift$	1	0	X	X	0	0	1	0	0	0	0	1
$S\_shift$	1	0	X	X	1	0	0	0	0	0	0	1



Ready = 1  
 when state is  
 $S\_idle$

State Table for Control Circuit

Present-State Symbol	Present State		Inputs		Next State		Ready	Load_regs	Decr_P	Add_regs	Shift_regs
	G <sub>1</sub>	G <sub>0</sub>	Start	Q[0]	Zero	G <sub>1</sub>	G <sub>0</sub>				
S_idle	0	0	0	X	X	0	0	1	0	0	0
S_idle	0	0	1	X	X	0	1	1	0	0	0
S_add	0	1	X	0	X	1	0	0	0	1	0
S_add	0	1	X	1	X	1	0	0	0	1	0
S_shift	1	0	X	X	0	0	1	0	0	0	1
S_shift	1	0	X	X	1	0	0	0	0	0	1

$D_{610} = S\_idle \cdot Start + S\_shift \cdot \overline{Zero}$   
 $= T_0 \cdot Start + T_2 \cdot \overline{Zero}$

Load\_regs = state  $\rightarrow$  S\_idle  
 if P  $\rightarrow$  Start

Load\_regs = T<sub>0</sub> Start

Add\_regs  $\leftarrow$  state S\_add  $\wedge$  Q[0] = 1

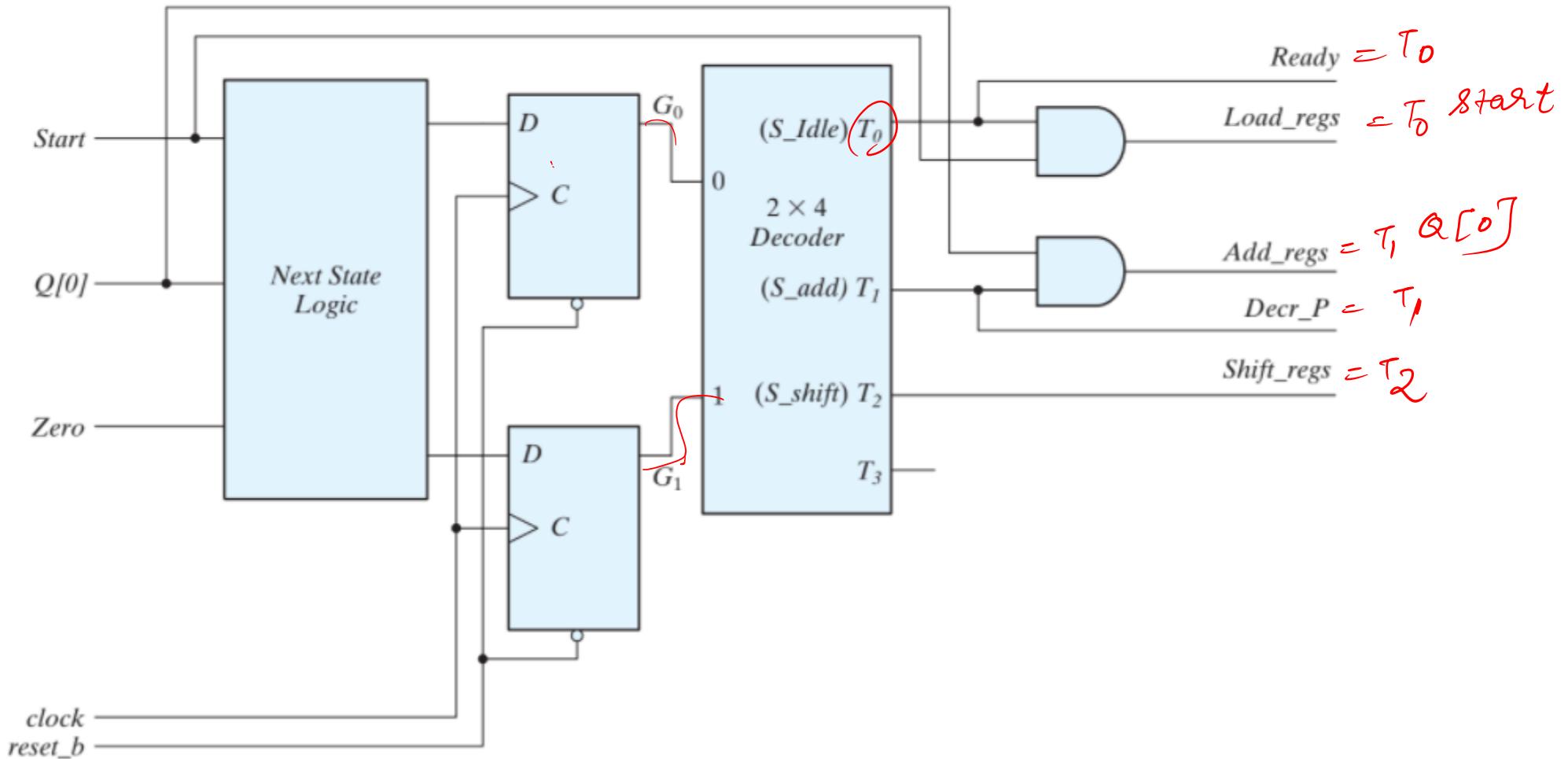
Add\_regs = T<sub>1</sub> Q[0]

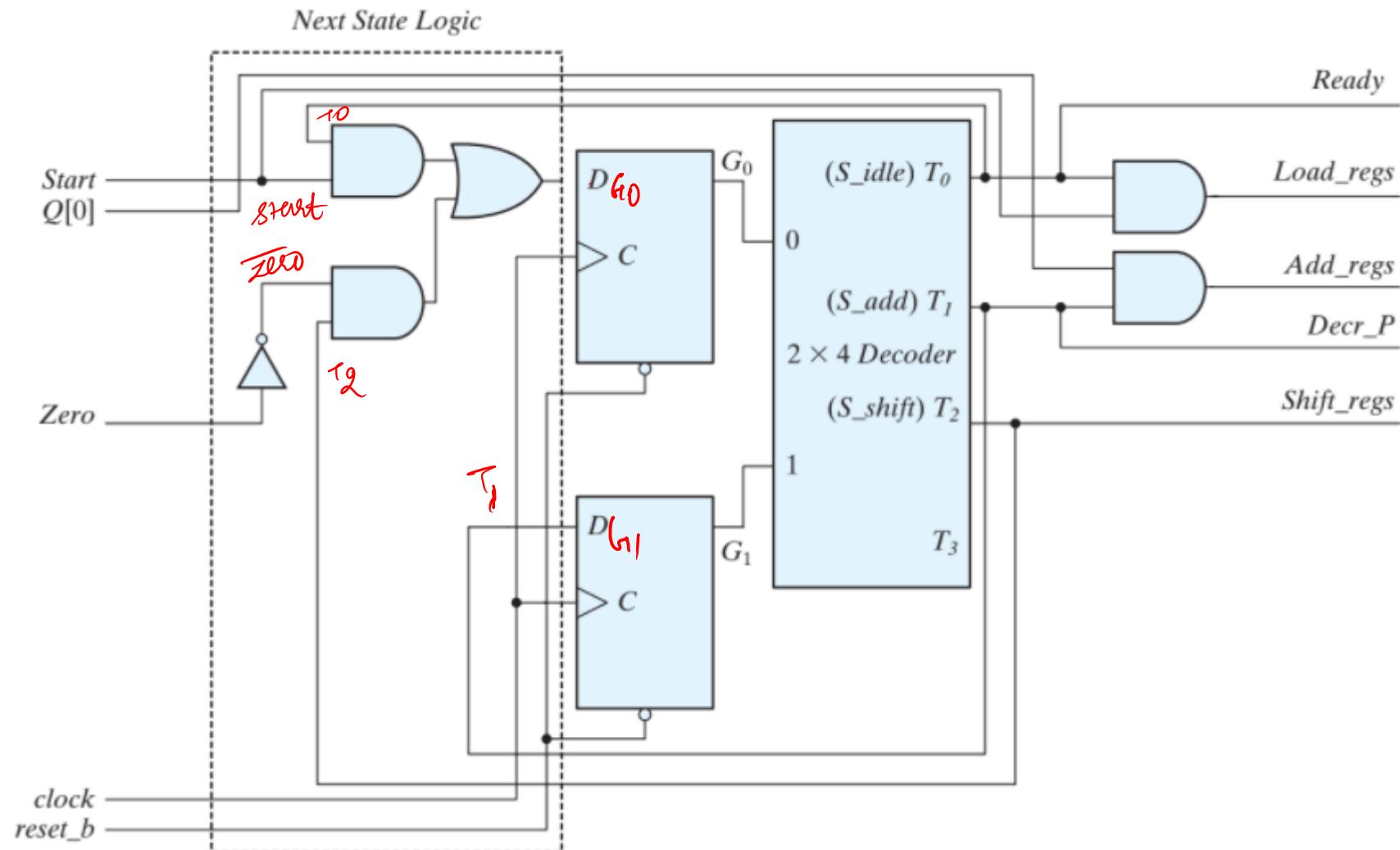
Ready = T<sub>0</sub>

Decr\_P = T<sub>1</sub>

Shift\_Regs = T<sub>2</sub>

D<sub>611</sub> = T<sub>1</sub>





one hot design

- just by seeing  
state dgm we can  
design the controller  
in case of one hot design

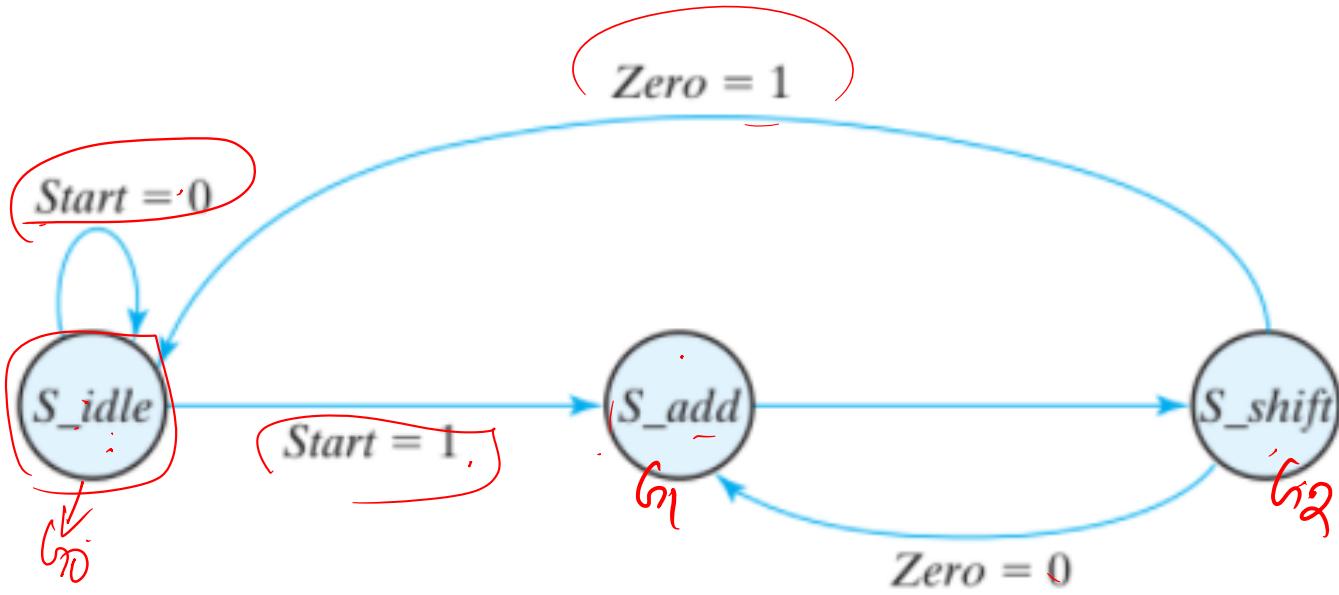
one FF / state

3 FFs      S\_idle      S\_add  
↓            001            010  
3 D FFs  
~~3 op's~~      G<sub>0</sub>, G<sub>1</sub>, & G<sub>2</sub>.

S\_Shift  
R00



001  
010  
010



$$D_{G_0} = 1 \quad ?$$

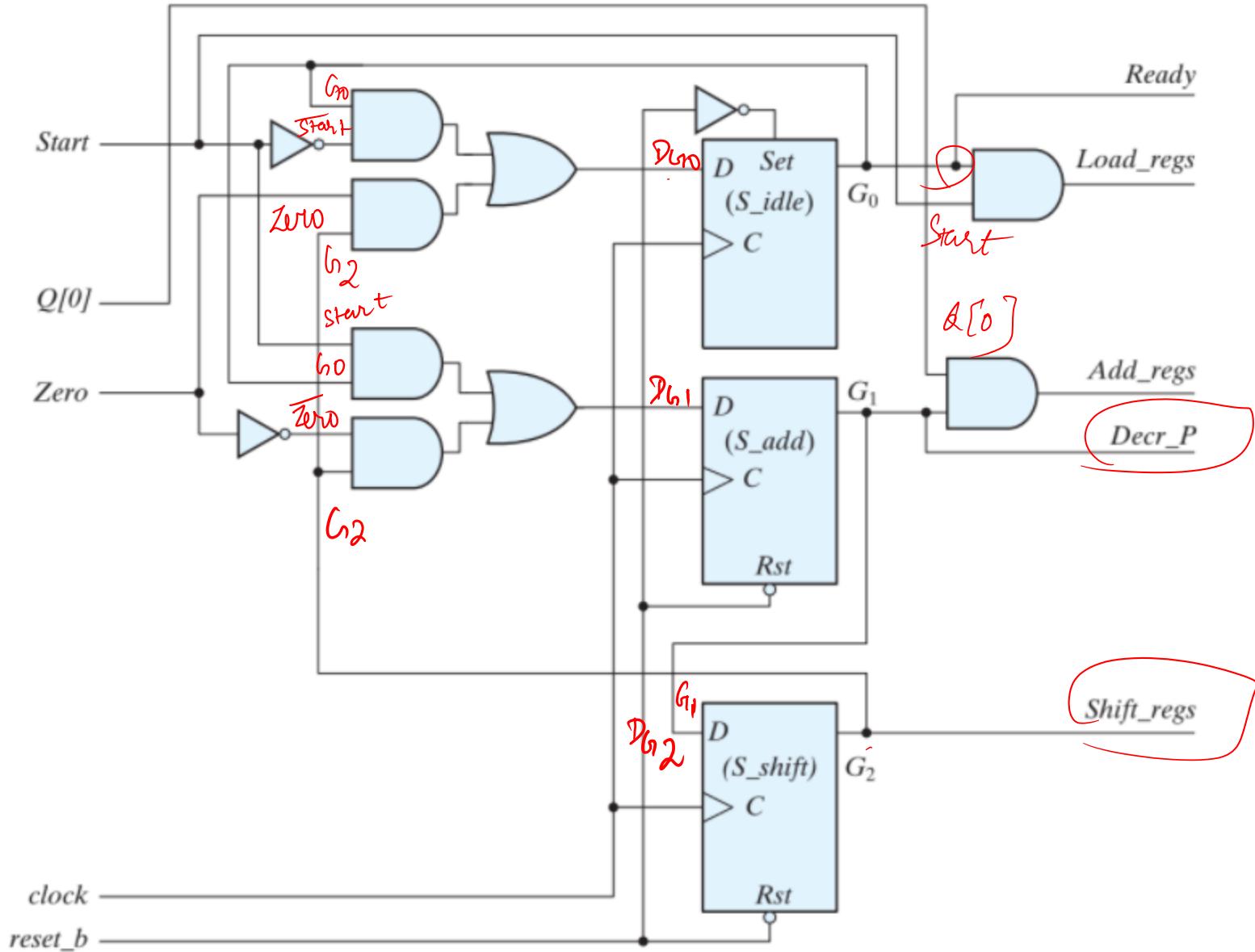
$$D_{G_1} = ?$$

$$D_{G_2} = 1$$

$$D_{G_0} = G_0 \overline{Start} + G_2 \overline{Zero}$$

$$D_{G_1} = G_0 Start + G_2 \overline{Zero}$$

$$D_{G_2} = G_1$$



# Digital Logic Families

# Digital Logic Families

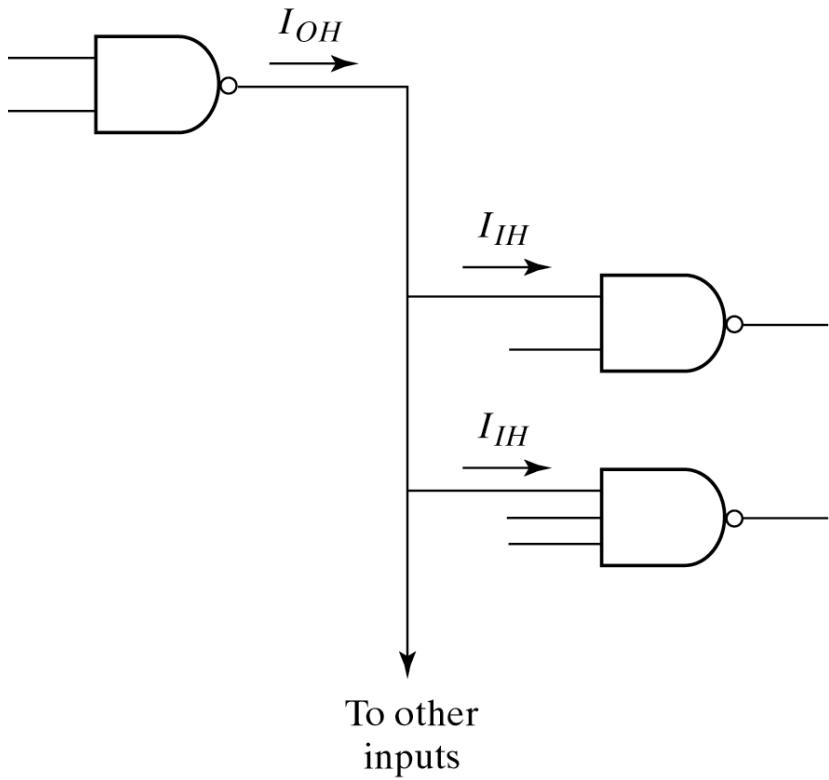
- ICs are also classified based on their specific circuit technology, known as *digital logic family*.
- Each family has its own basic electronic components (NAND, NOR, and NOT gates), used to build complex digital circuits.
- Various digital logic families have been introduced and used over the years.

# Digital Logic Families (in chronological order)

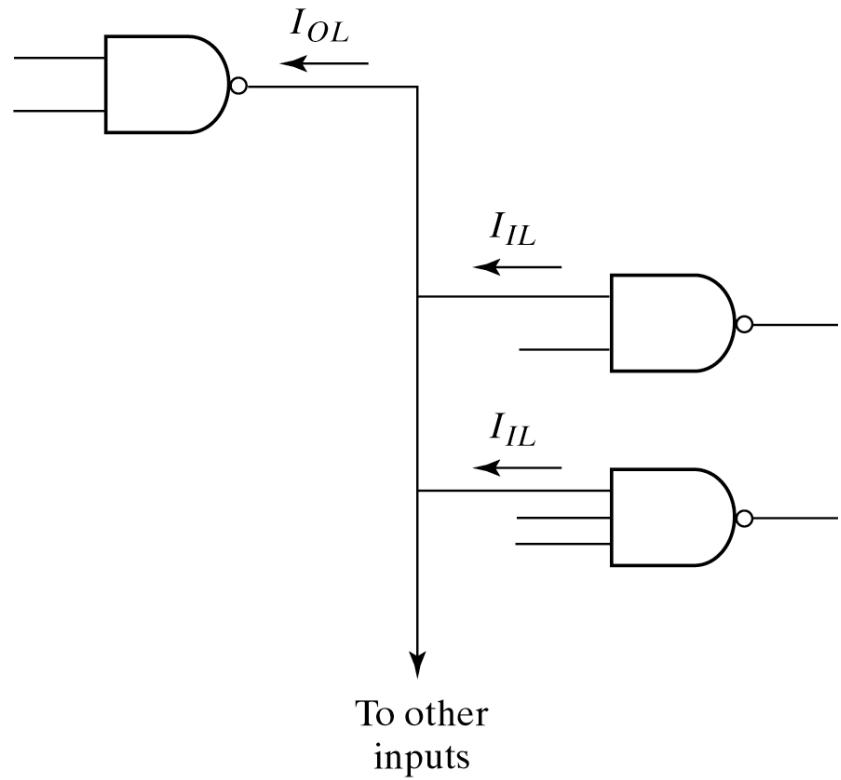
- RTL: Resistor-Transistor Logic
  - DTL: Diode-Transistor Logic
  - TTL: Transistor-Transistor Logic
  - ECL: Emitter-coupled Logic
  - MOS: Metal-Oxide Semiconductor
  - CMOS: Complementary MOS
    - Low power dissipation, currently the **MOST DOMINANT**
  - BiCMOS: Bipolar CMOS
    - CMOS and TTL for additional current/speed
  - GaAs: Gallium-Arsenide
- earliest,  
now obsolete
- widely used
- high-speed operation
- compact
- very high-speed operation

# Defining Characteristics of Digital Logic Families

- *Fan-in*: # of gate inputs.
- *Fan-out*: # of standard loads a gate's output can drive.
- *Noise margin*: max external noise tolerated.
- *Power dissipation*: power consumed by the gate (dissipated as heat).
- *Propagation delay*: time required for an input signal change to be observed at an output line.

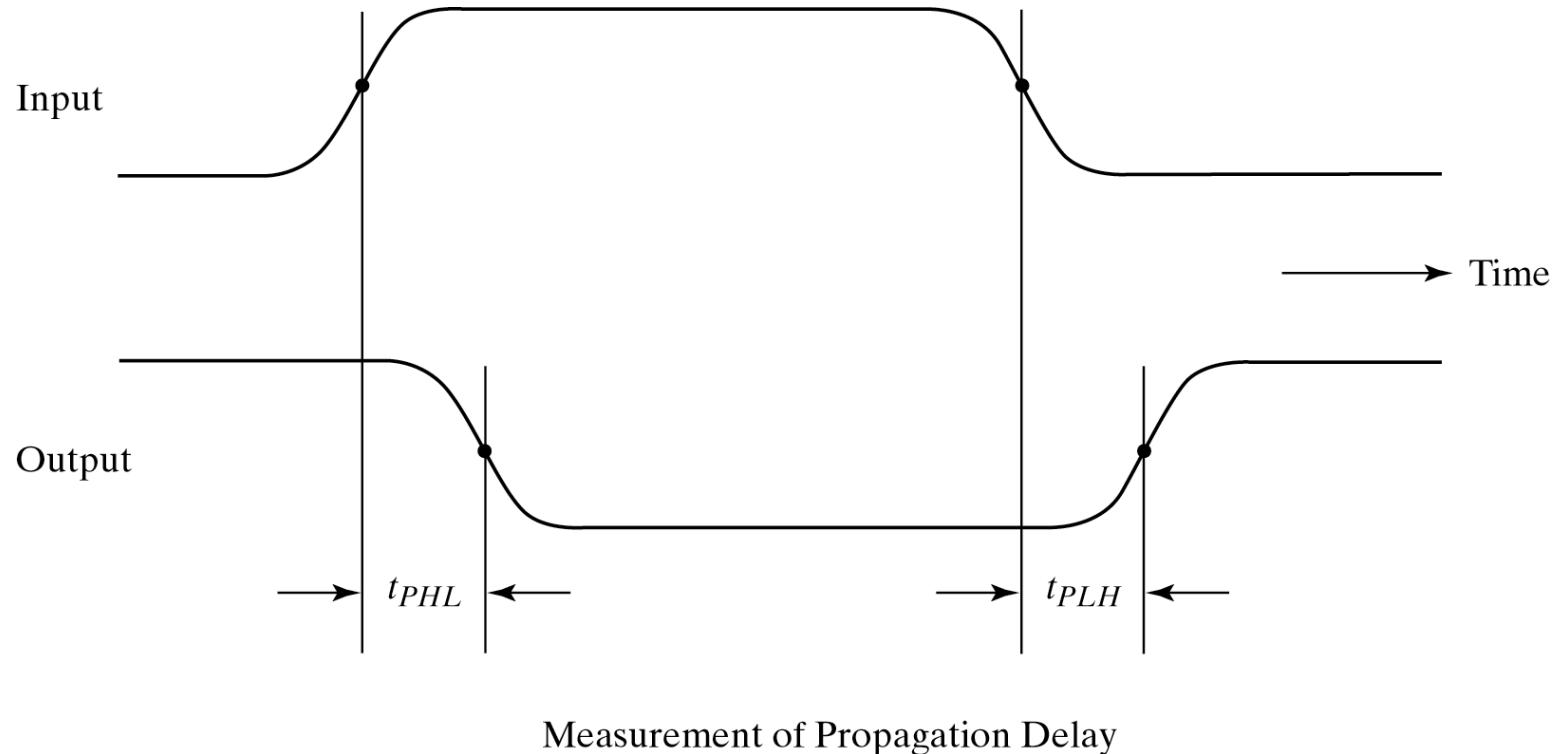


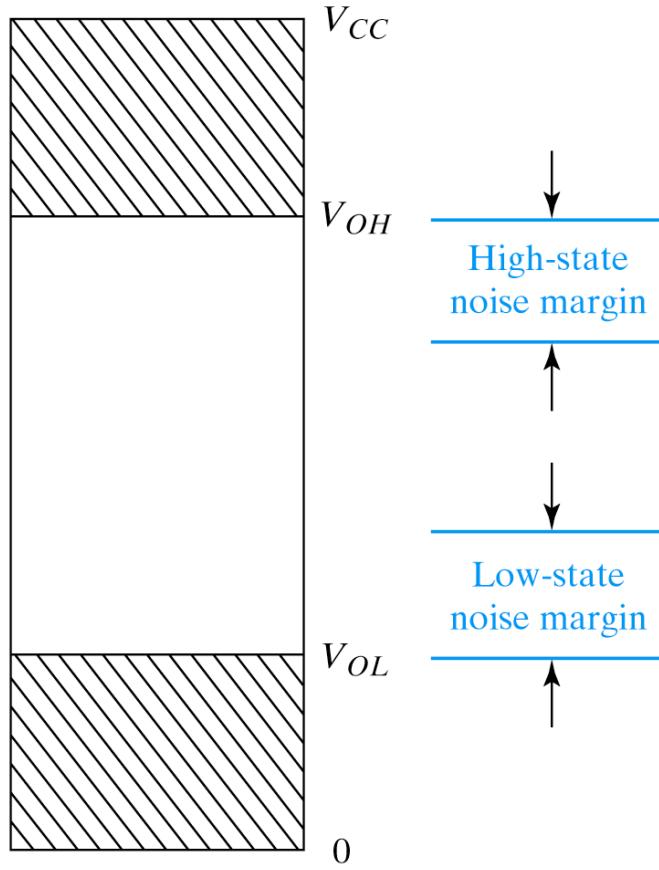
(a) High-level output



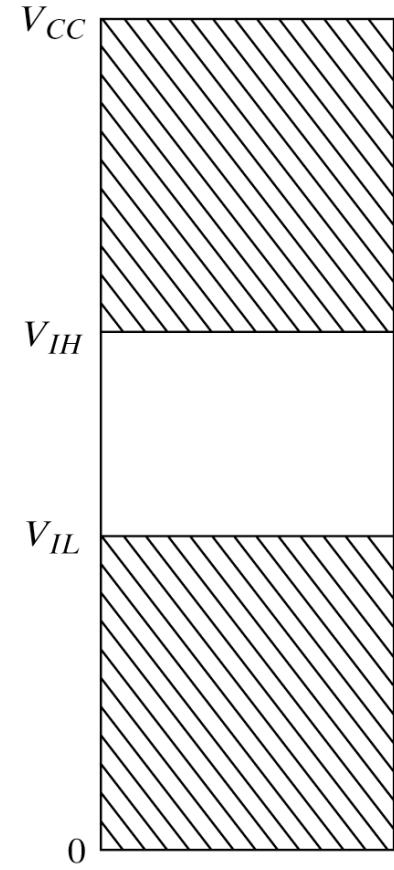
(b) Low-level output

Fan-Out Computation





(a) Output voltage range



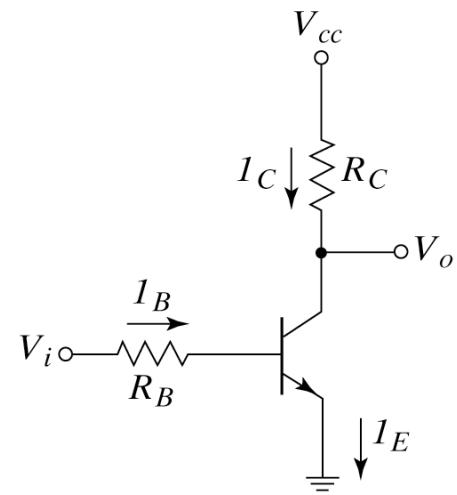
(b) Input voltage range

Signals for Evaluating Noise Margin

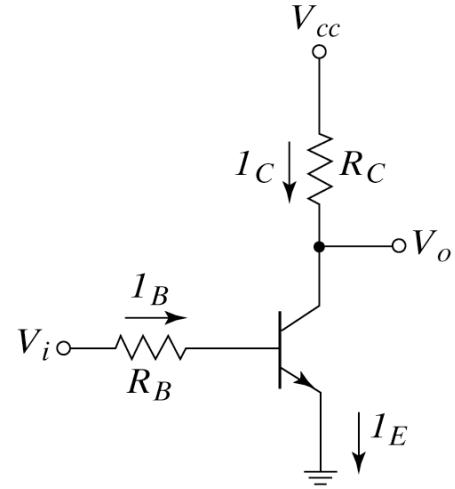
# **Power Dissipation ( $P_D$ )**

- Expressed in Milliwatts
- $P_D = V_{CC} * I_{CC}$
- $I_{CC(\text{avg})} = (I_{CCH} + I_{CCL}) / 2$

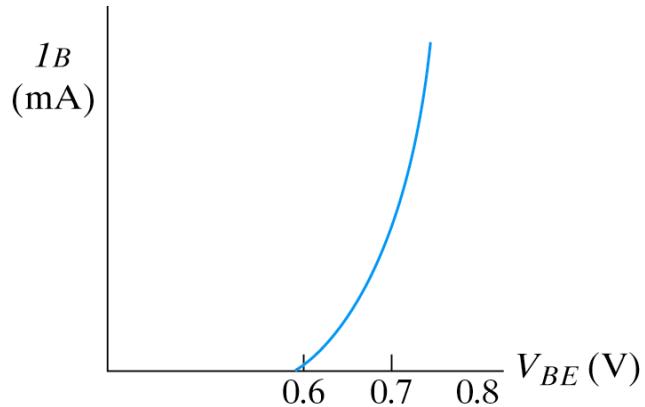
# **TTL LOGIC FAMILY**



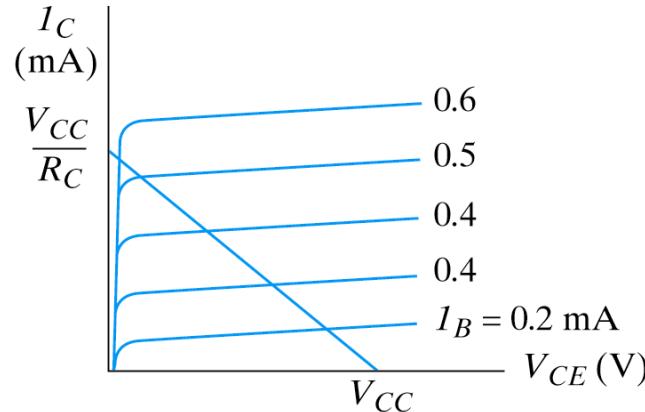
(a) Inverter circuit



(a) Inverter circuit

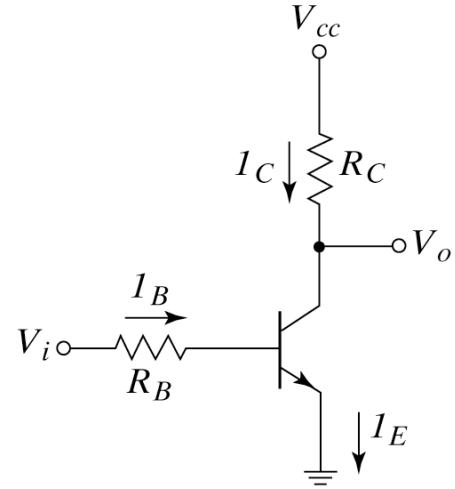


(b) Transistor-base characteristic



(c) Transistor-collector characteristic

Silicon *n*p*n* Transistor Characteristics



(a) Inverter circuit

$$R_C = 1\text{k}\Omega, R_B = 22\text{k}\Omega, \beta = 50$$

$V_{CC} = 5\text{V}$ , find  $V_o$  for

$V_i = 0.2\text{V}$  and  $V_i = 5\text{V}$

# BJT Characteristics

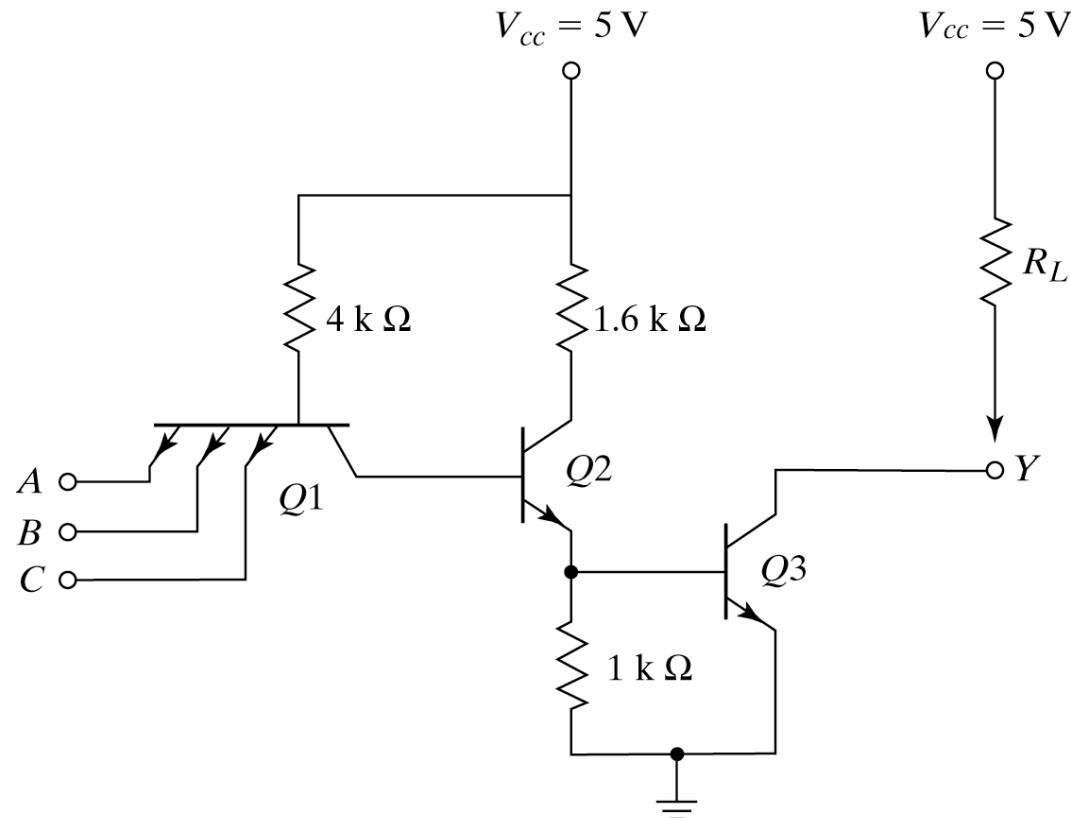
- Base-Emitter voltage less than 0.6V ;  $I_B = 0$   
: Cut-Off region
- Base-Emitter voltage more than 0.6V,  
transistor starts conducting - active region  
 $I_C = \beta I_B$
- Maximum collector current  $I_C = V_{CC}/R_C$

- In the cut-off region  $V_{BE} < 0.6V$ ,  
 $V_{CE}$  – open circuit,  $I_C$ ,  $I_B$  negligible
- In the active region  $V_{BE}$  about 0.7  
 $V_{CE}$  wide range and  $I_C = \beta I_B$
- In the saturation region  $V_{BE}$  hardly changes,  $V_{CE}= 0.2V$

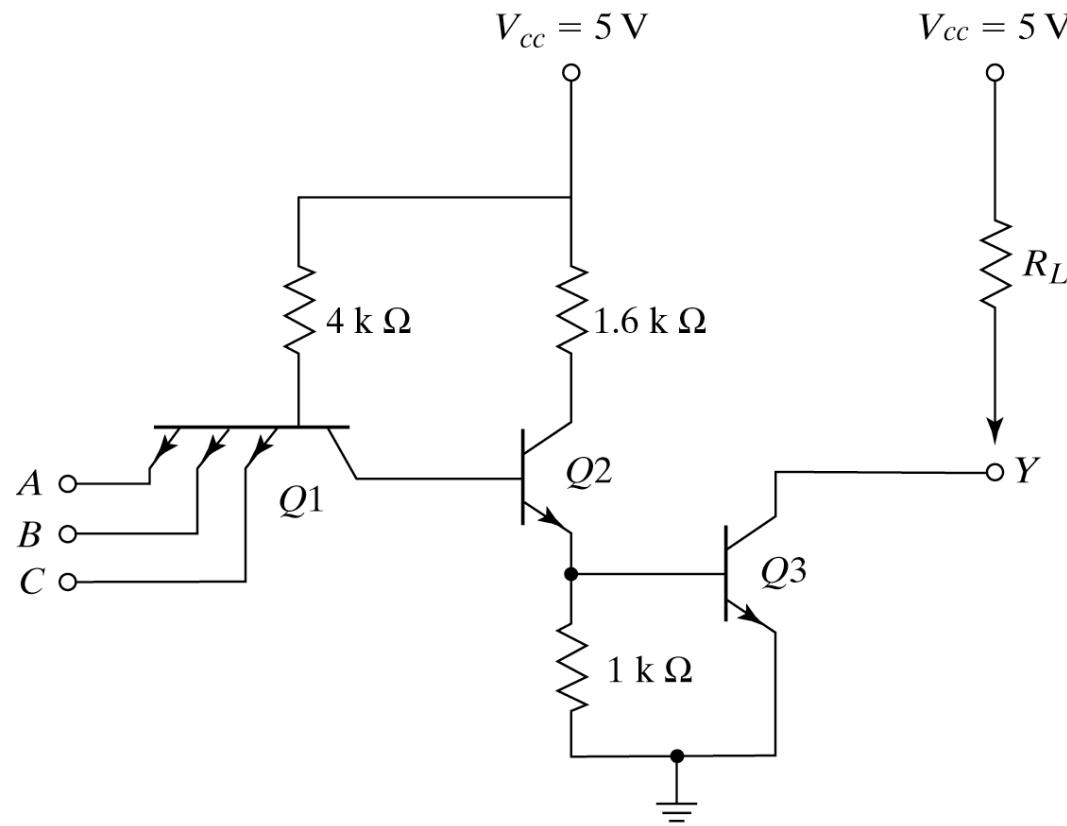
TTL Series name	Prefix
<b>Standard</b>	<b>74</b>
<b>Low-power</b>	<b>74L</b>
<b>High-speed</b>	<b>74H</b>
<b>Schottky</b>	<b>74S</b>
<b>Low-power Schottky</b>	<b>74LS</b>
<b>Advanced Schottky</b>	<b>74AS</b>
<b>Advanced Low power- Schottky</b>	<b>74ALS</b>
<b>Fast</b>	<b>74F</b>

# **Three Types of TTL gates**

- Open - collector output**
- Totem- pole output**
- Three- state output**

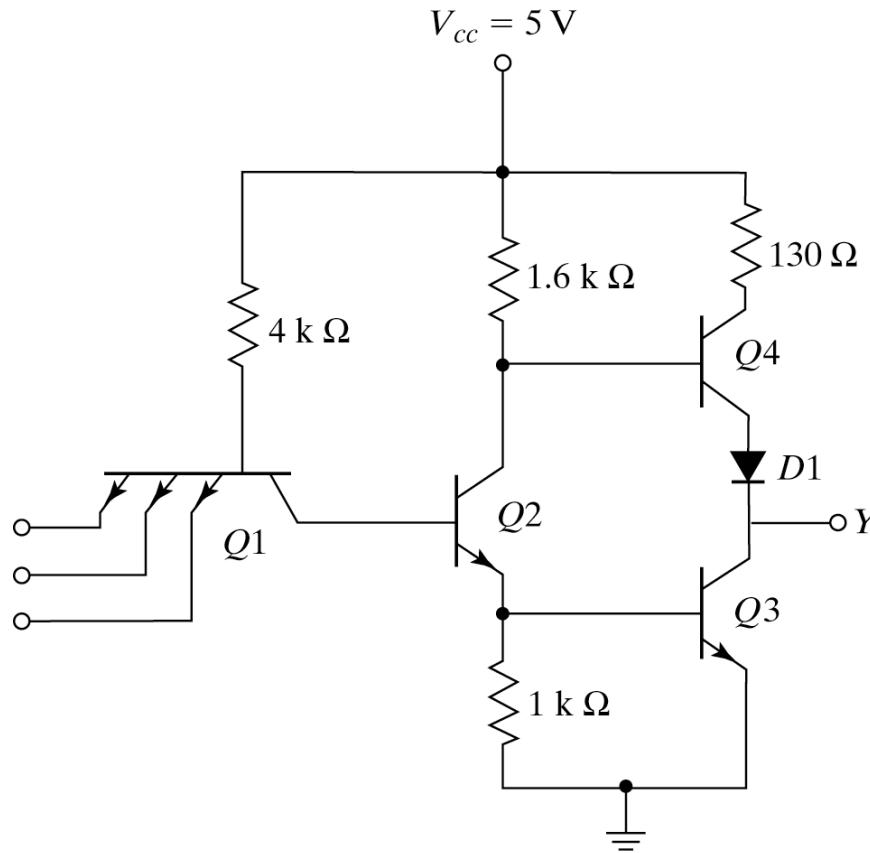


Open-Collector TTL Gate



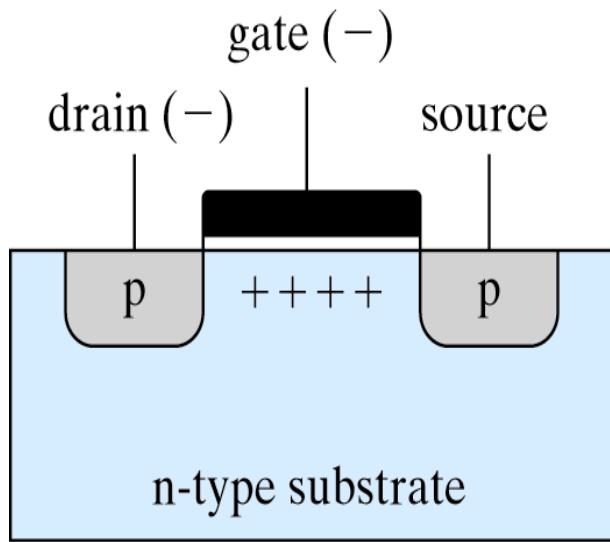
Open-Collector TTL Gate

- **Output impedance of a gate is resistive plus Capacitive load**
- **For output low to high transition C charges exponentially through RC**
- **R is  $R_L$  ( external) in open collector**
- **With active pull-up delay can be reduced**

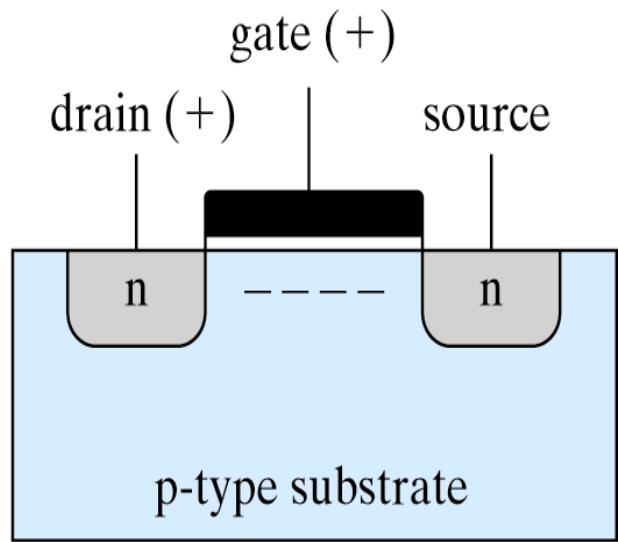


TTL Gate with Totem-Pole Output

# MOS Transistor

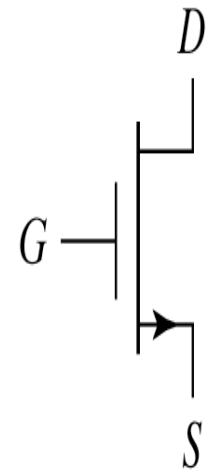
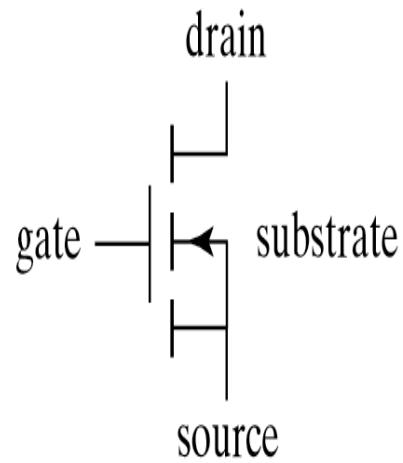
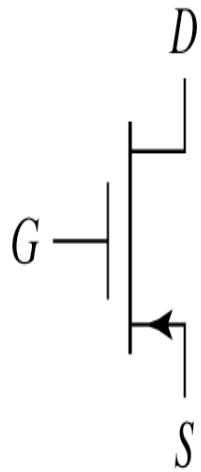
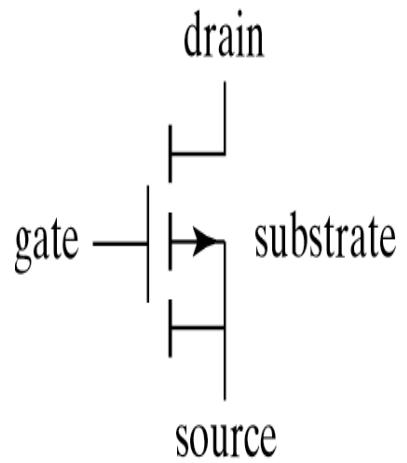


(a) p-channel



(b) n-channel

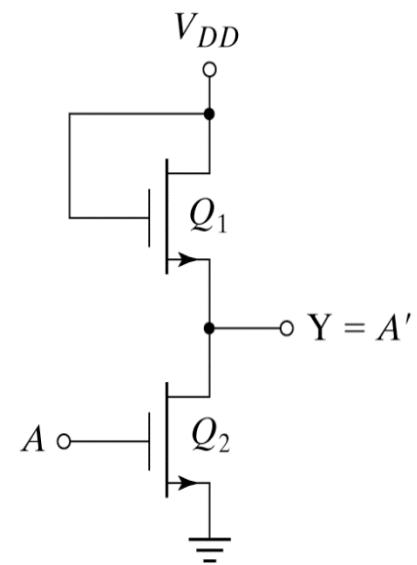
Basic Structure of MOS Transistor



(a) p-channel

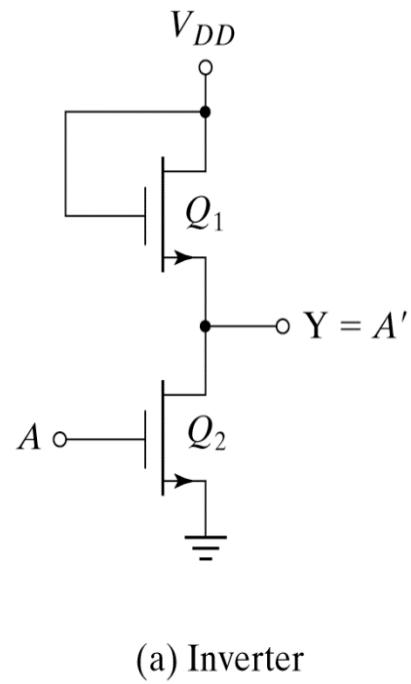
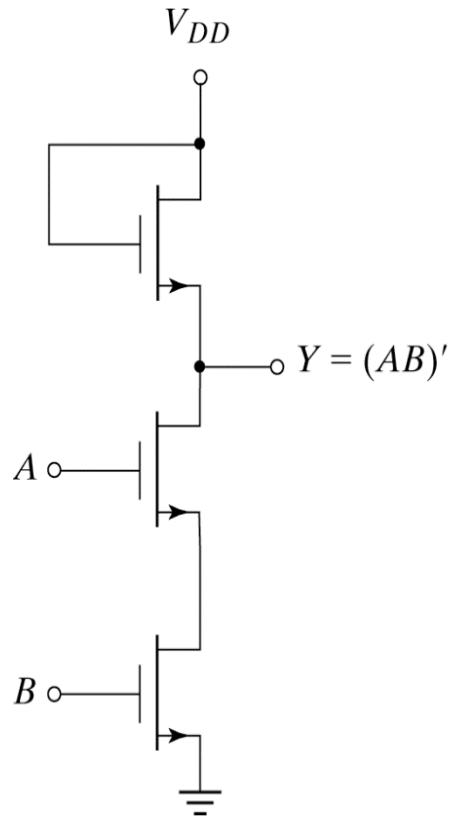
(b) n-channel

Symbols for MOS Transistors

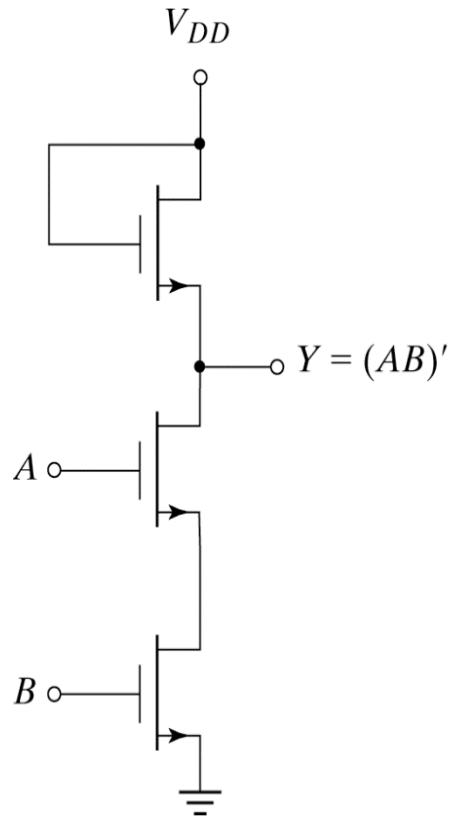


(a) Inverter

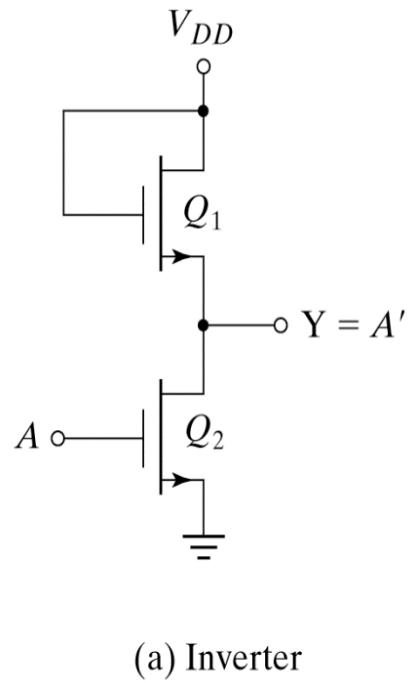
*n*-channel MOS Logic Circuits



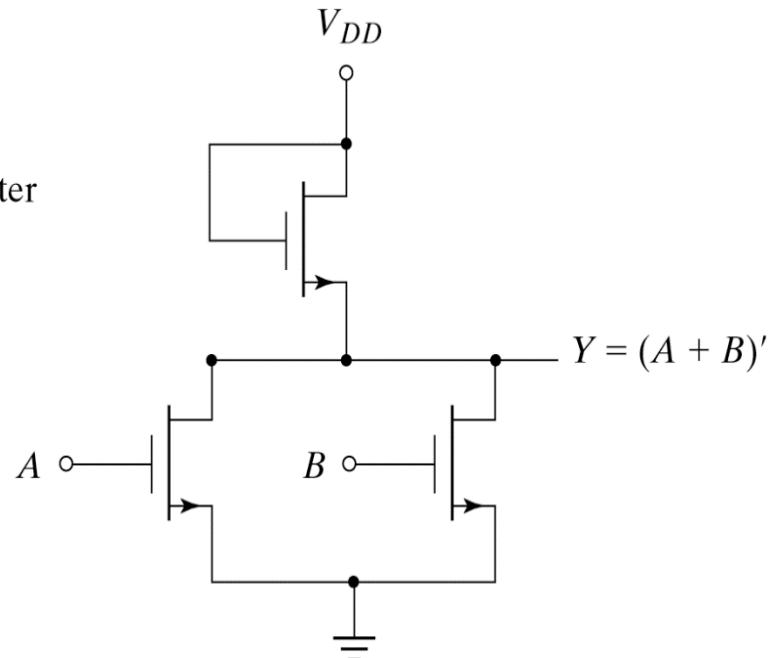
*n*-channel MOS Logic Circuits



(b) NAND gate



(a) Inverter

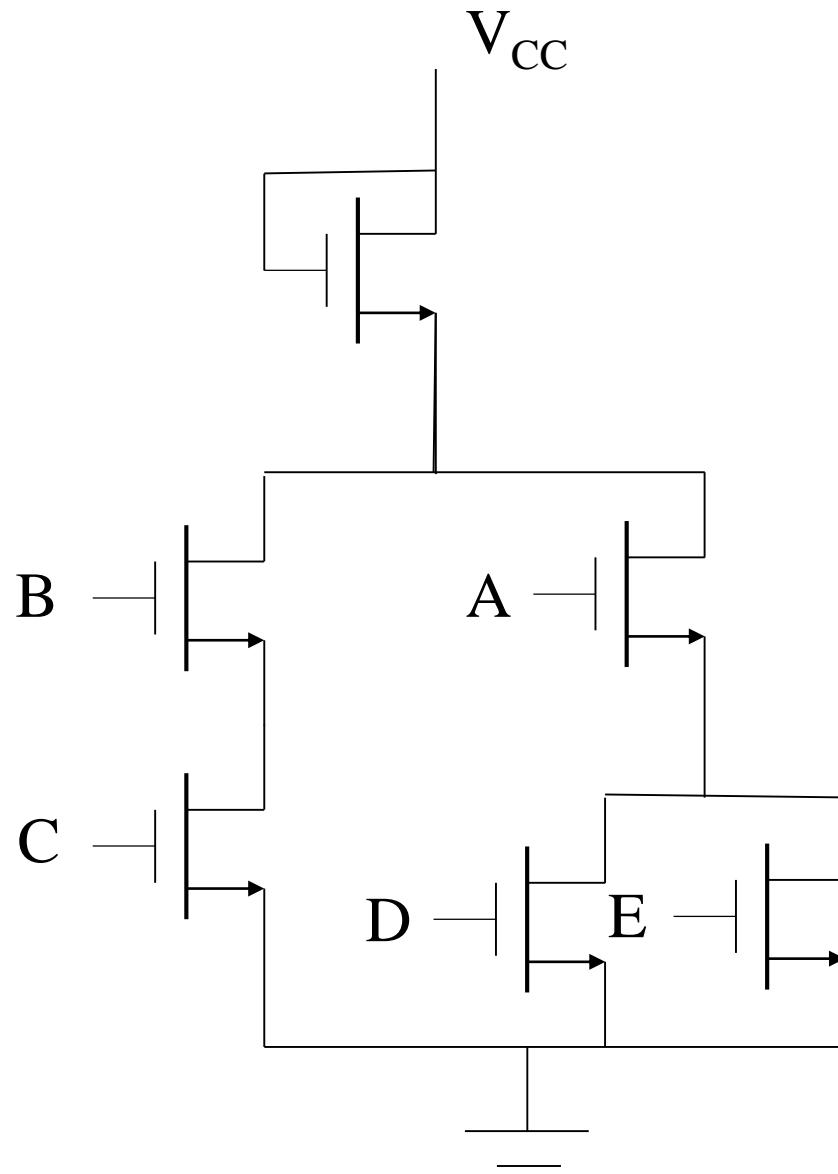


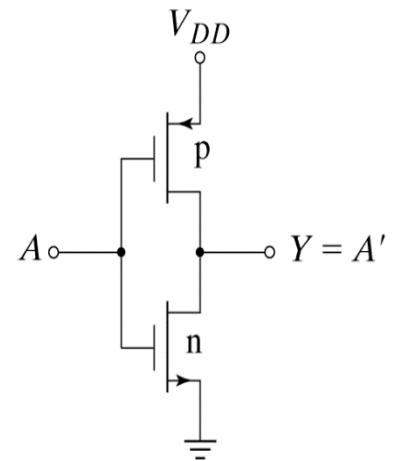
(c) NOR gate

*n*-channel MOS Logic Circuits

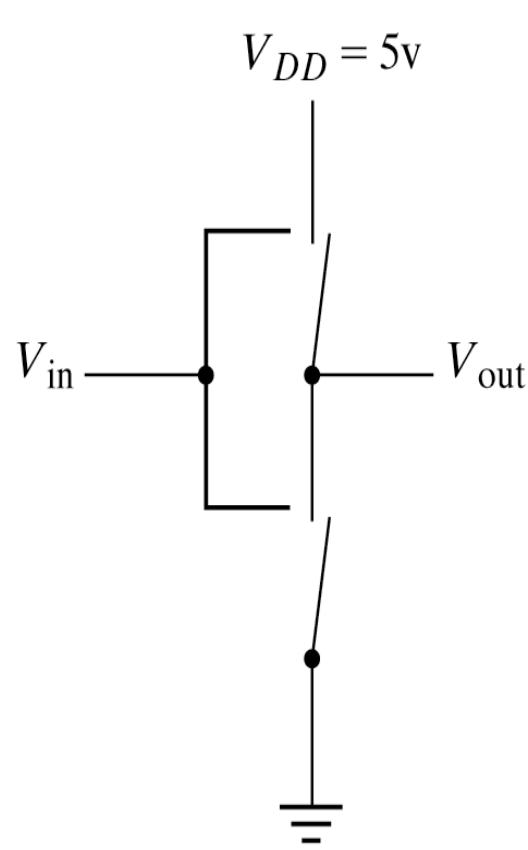
# Design the logic Using NMOS

$$Z = (A(D+E) + BC)',$$

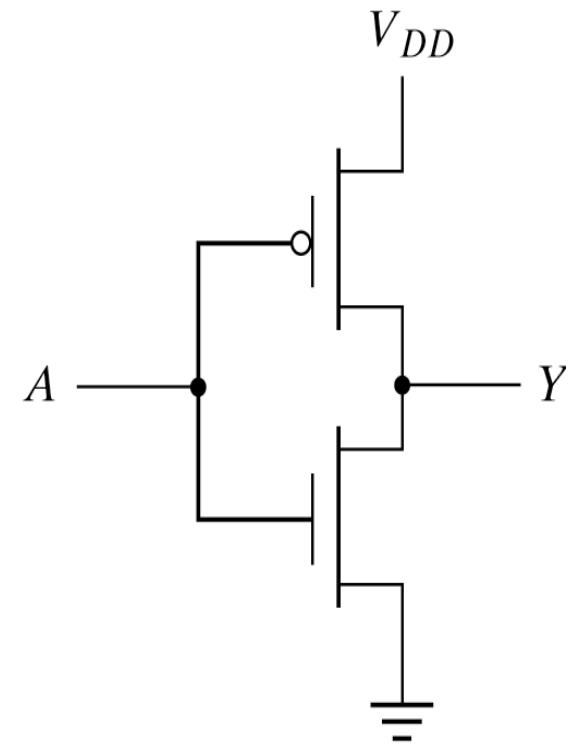




(a) Inverter

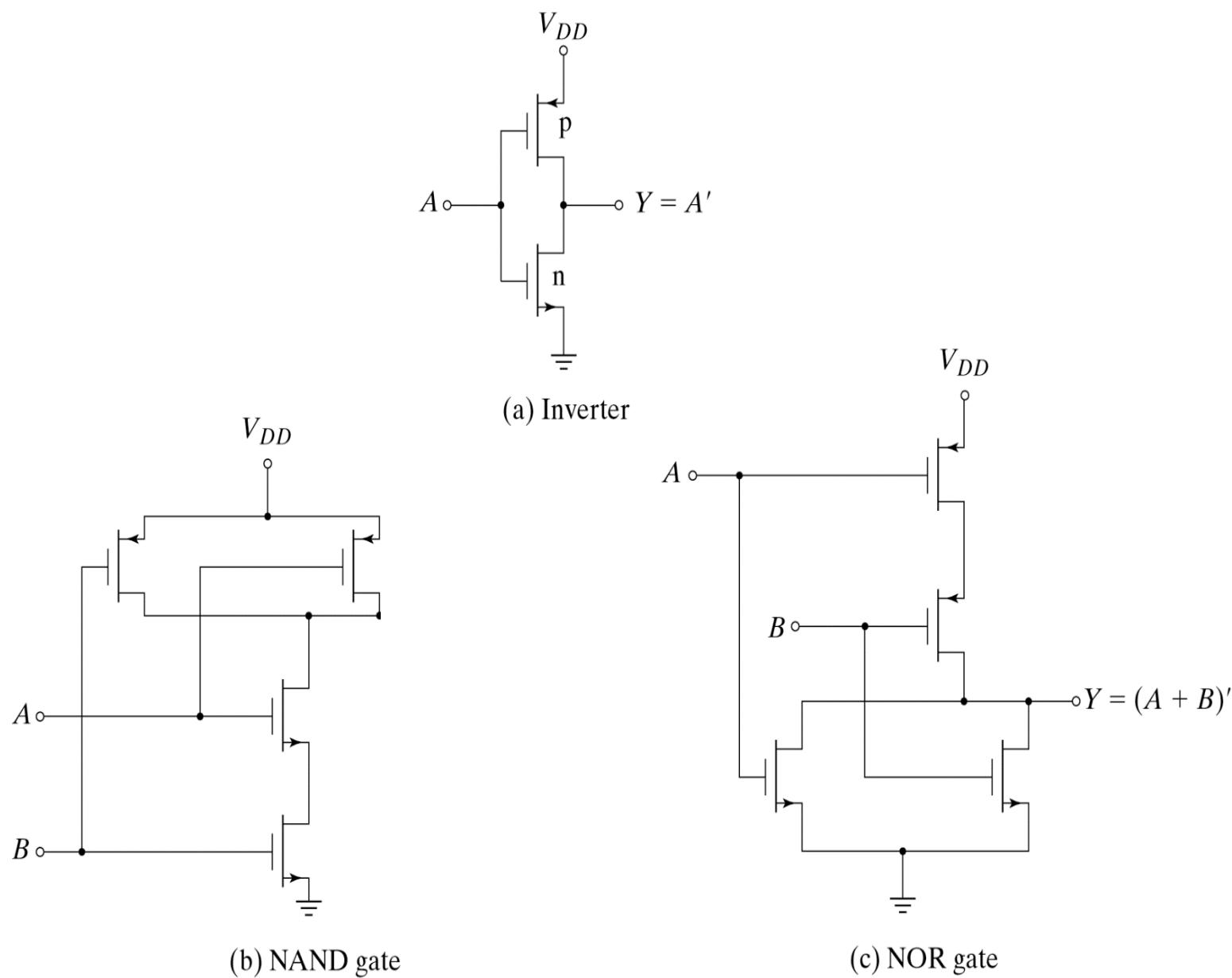


(a) Switch model



(b) Logical model

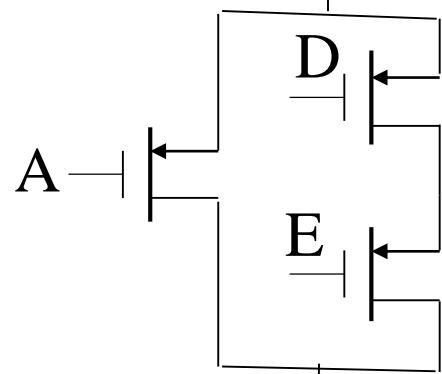
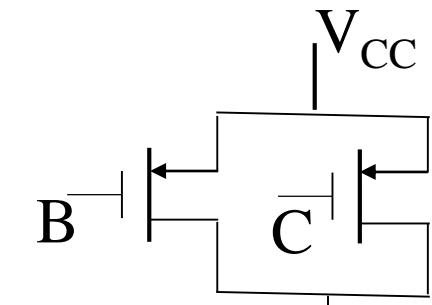
*CMOS* inverter



CMOS Logic Circuits

# Design the logic Using CMOS

$$Z = (A(D+E) + BC)',$$



$$Z = (A(D+E) + BC)'$$

