Sequential Circuits

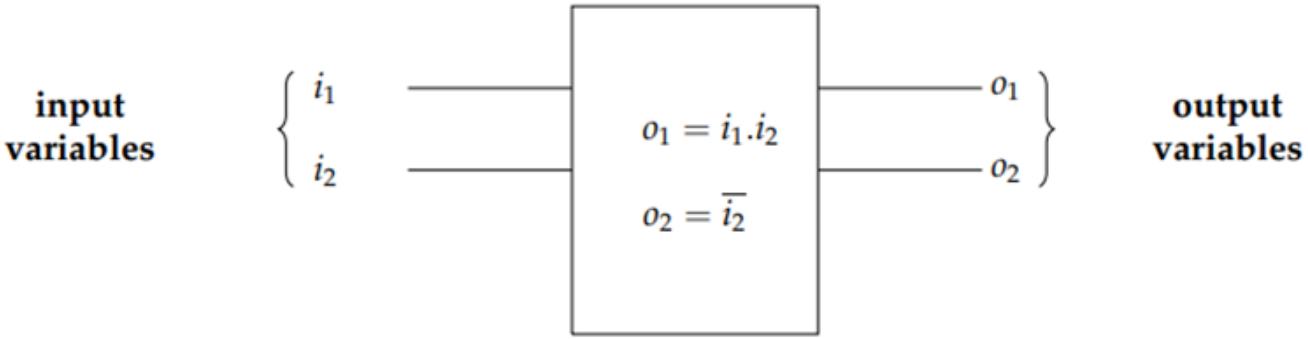
&

Combinational Circuits Vs Sequential Circuits

Next Topic:

Combinational Circuits Vs Sequential Circuits

Combinational Circuits



Combinational circuits:

- Consists only of logic gates
- The output are determined by the present value of input
- Circuit behavior specified by a set of Boolean functions, Truth-tables, K-maps
- We have learned techniques to analyze and synthesize such circuits
- Examples: Adder, Multiplexers, Encoders, Decoders, etc.

Digital Circuits

Combinational Circuit

Ex: So far
- logic gates
- mux, Decoder, encoder, Adder

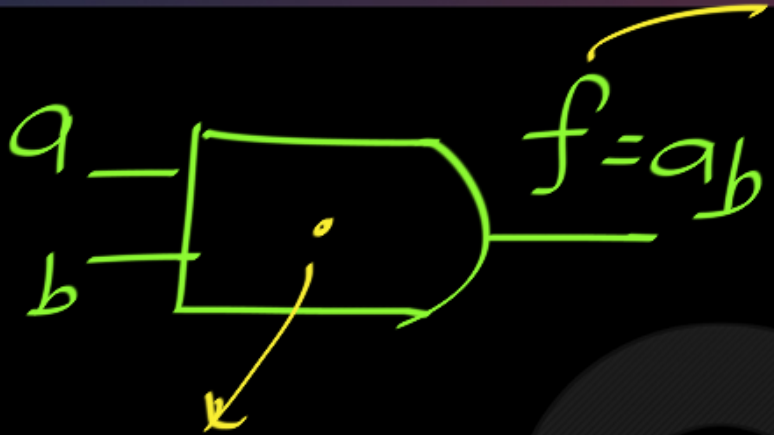
Sequential Circuits

Now onwards:
- Counter ; Finite State machines
- Register ;
- flip flop, latch

Combinational Circuits :

Digital Circuits whose output(s) Depend only on Current inputs.

(Current o/p Does not Depend on Previous outputs / inputs.)



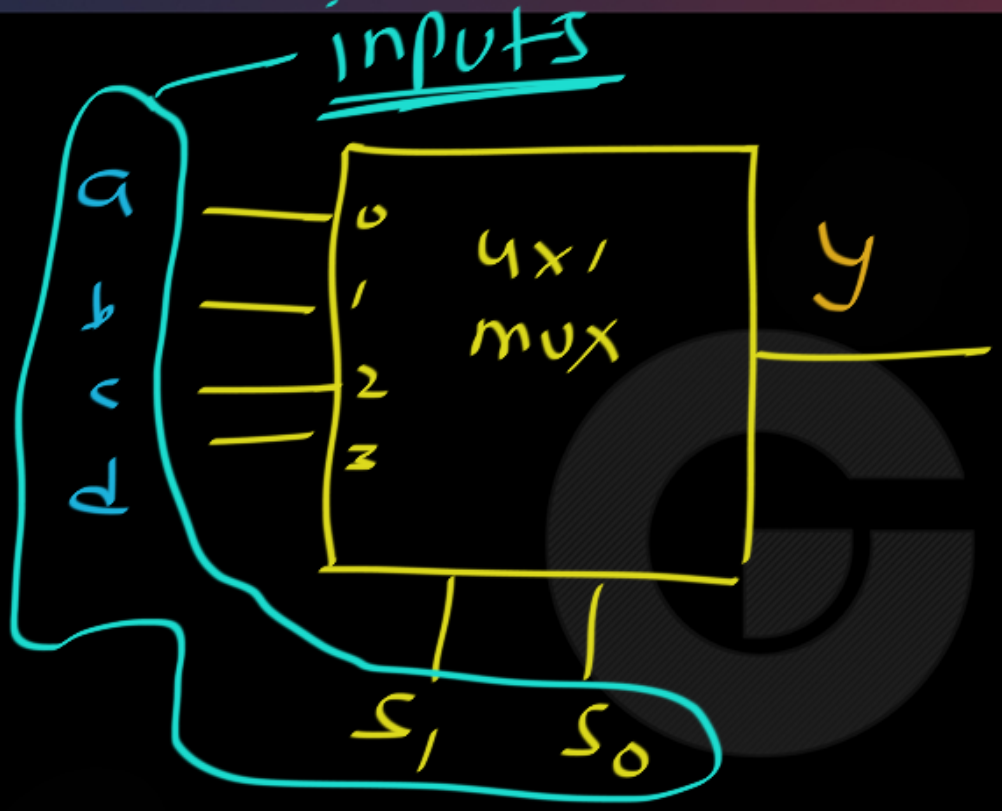
AND gate

logic gate

||

Comb. Circuit

time	Current inputs	output
t = 0	a = 0, b = 0	f = 0
t = 1	a = 1, b = 0	f = 0
t = 2	a = 1, b = 1	f = 1



$y = f(S_1, S_0, a, b, c, d)$

S_1 S_0	y	time
0 0	b	t=0
0 1	d	t=1
1 0	c	t=10
1 0	c	
1 0	c	

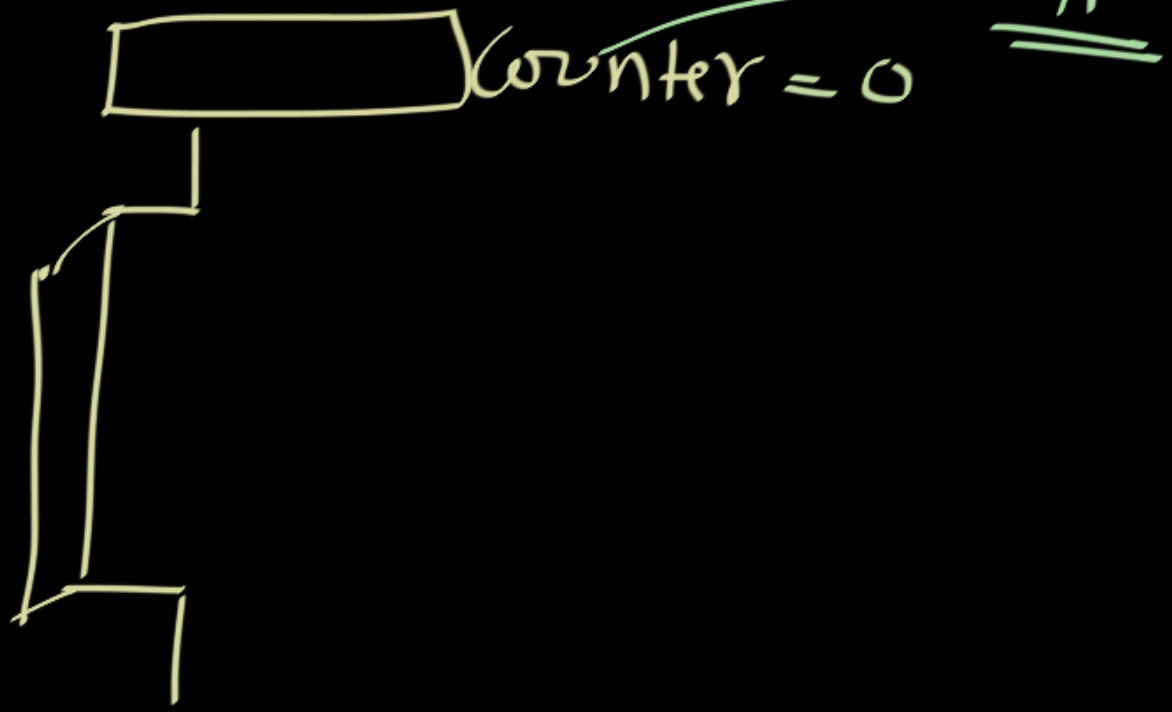
Combinational Circuits :

"Living in the Present."

Automatic Washing machine :-

20
19
18

Example :- Counter

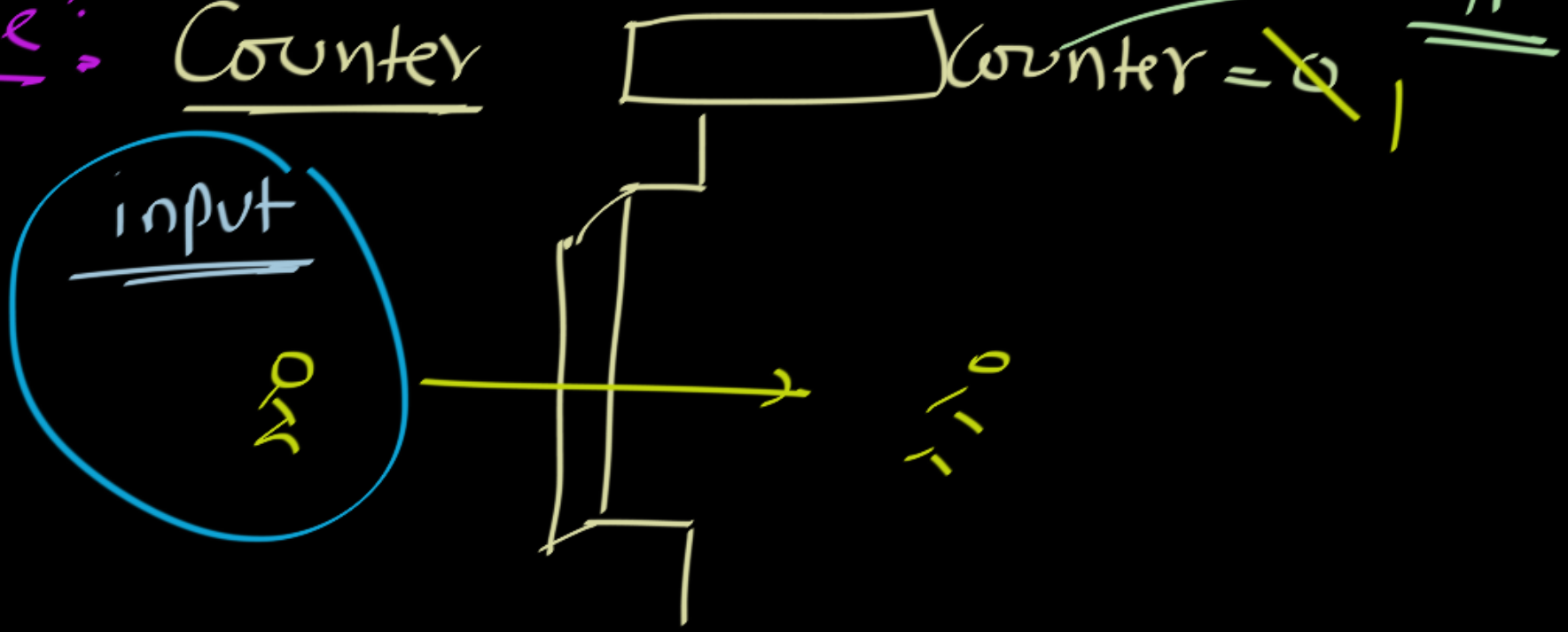


input

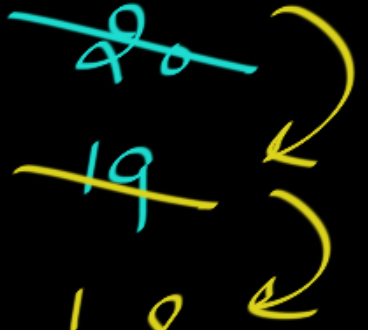
Automatic Washing machine :- timer

20 → 19 → 18

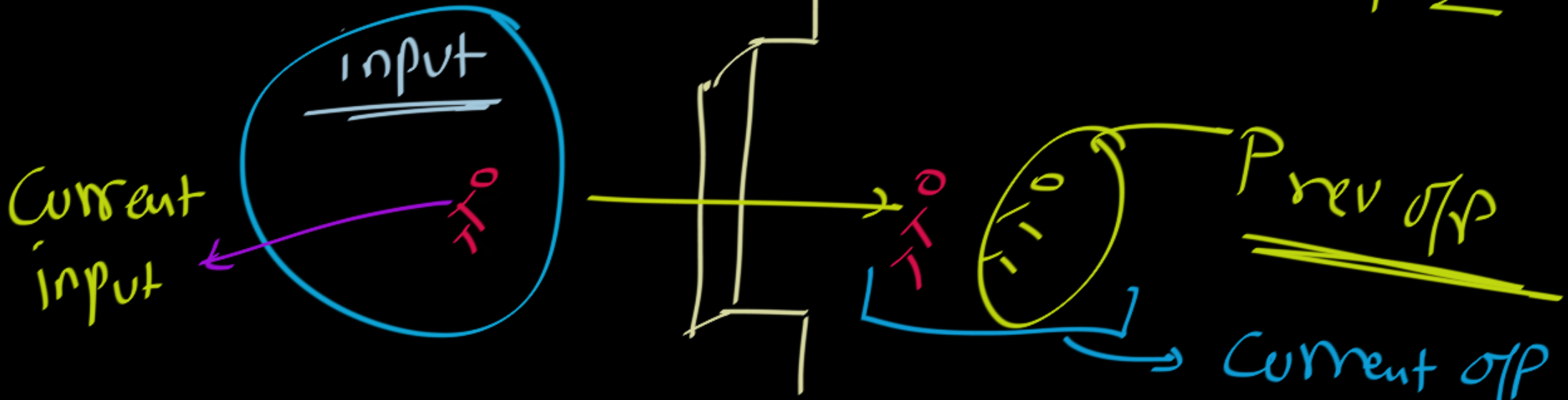
Example :- Counter



Automatic Washing machine :-



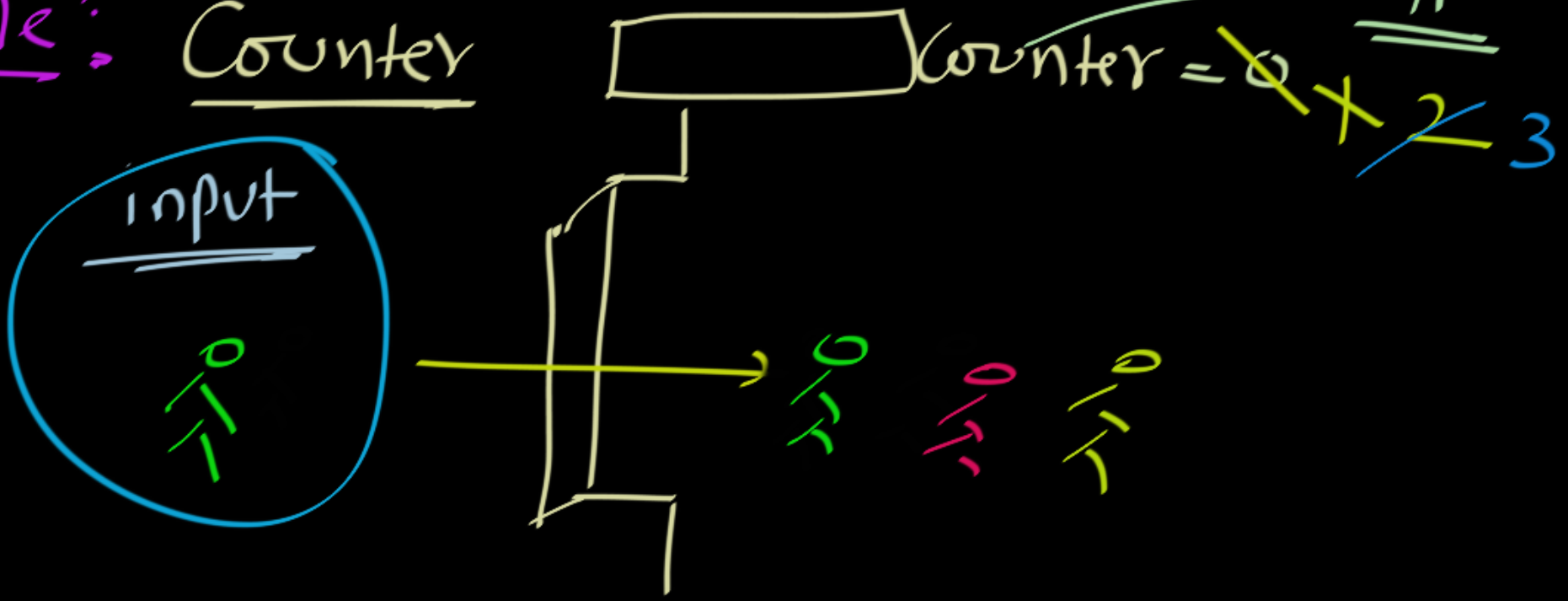
Example :- Counter



Automatic Washing machine :- timer

20 → 19 → 18

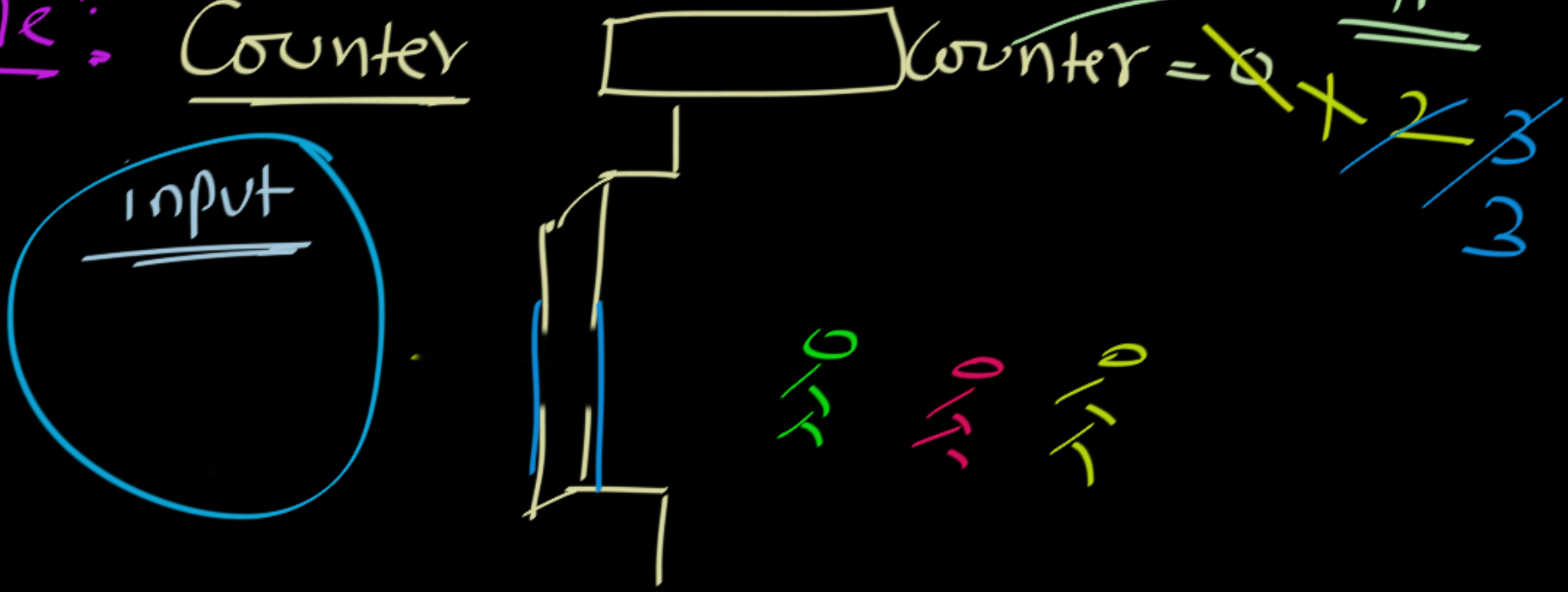
Example :- Counter



Automatic Washing machine :- timer

20 → 19 → 18

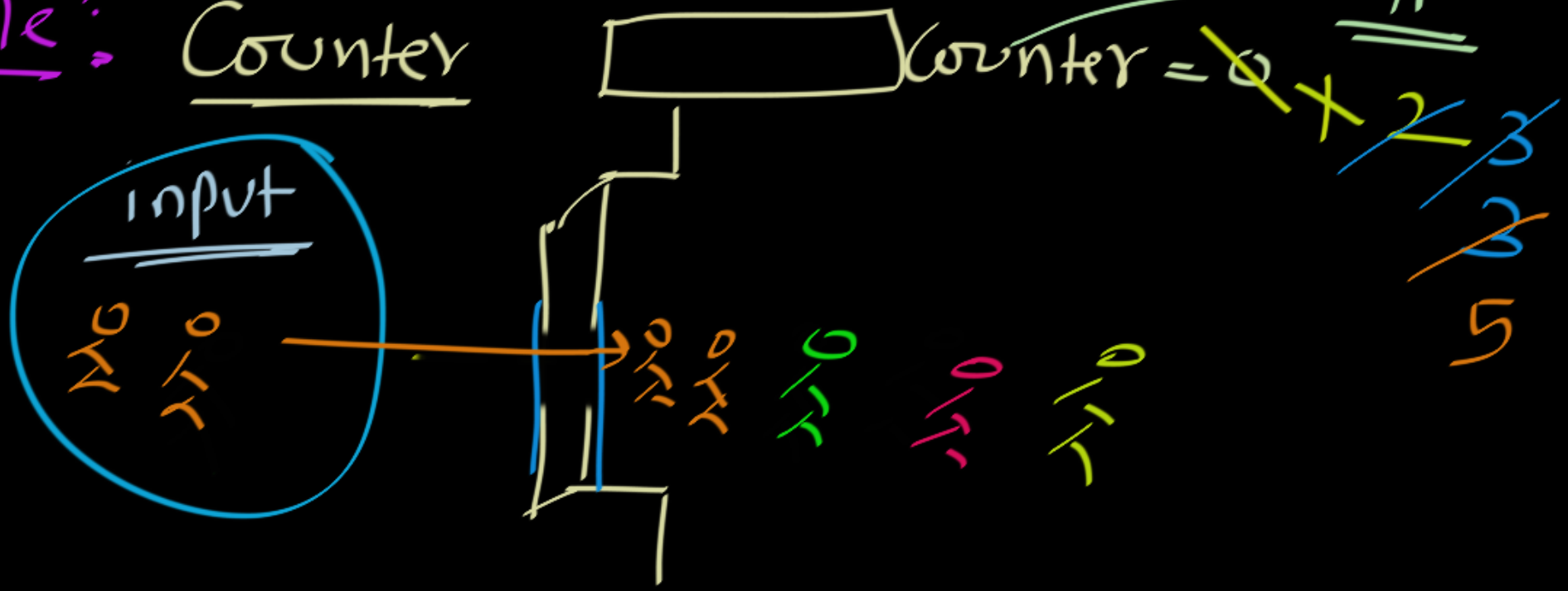
Example :- Counter



Automatic Washing machine :- timer

20 → 19 → 18

Example :- Counter



Counter :

current output = f (current input , prev. output)

Sequential Circuits

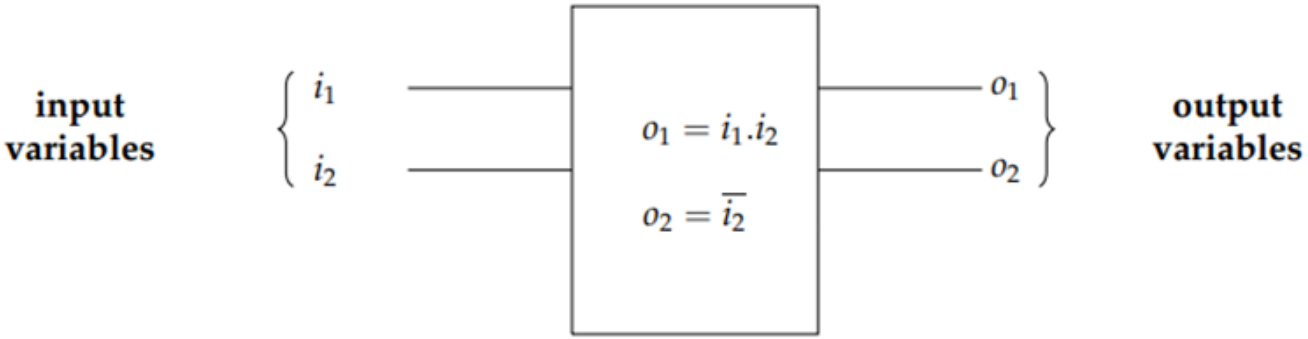
Sequential Logic

- The logic circuits discussed previously are known as *combinational*, in that the output depends only on the condition of the latest inputs
- However, we will now introduce a type of logic where the output depends not only on the latest inputs, but also on the condition of earlier inputs. These circuits are known as *sequential*, and implicitly they contain *memory* elements

Properties of Sequential Circuits

- So far we have seen Combinational Logic
  - the output(s) depends only on the current values of the input variables
- Here we will look at Sequential Logic circuits
  - the output(s) can depend on present and also past values of the input and the output variables

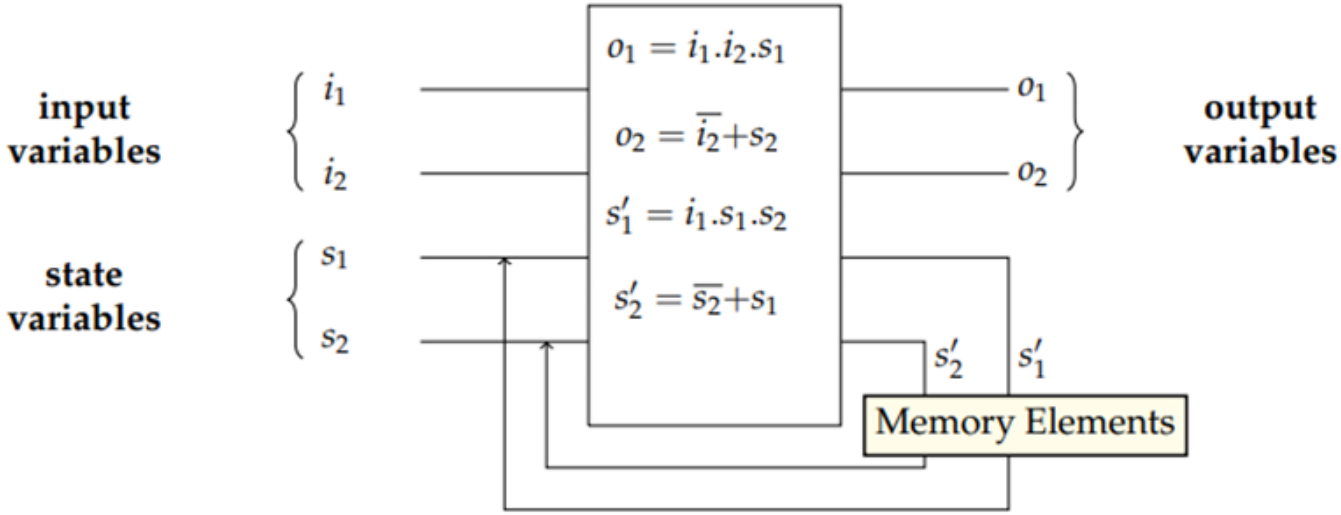
Combinational Circuits



Combinational circuits:

- Consists only of logic gates
- The output are determined by the present value of input
- Circuit behavior specified by a set of Boolean functions, Truth-tables, K-maps
- We have learned techniques to analyze and synthesize such circuits
- Examples: Adder, Multiplexers, Encoders, Decoders, etc.

Sequential Circuits



Sequential Circuits:

- Consists of logic gates and storage elements
- The output are determined by the present value of the input and the state of the storage elements
- In effect, the output may depend on past values of the input via the state of the storage elements

Real life examples of Seq. Ckts:

(1) Traffic lights:

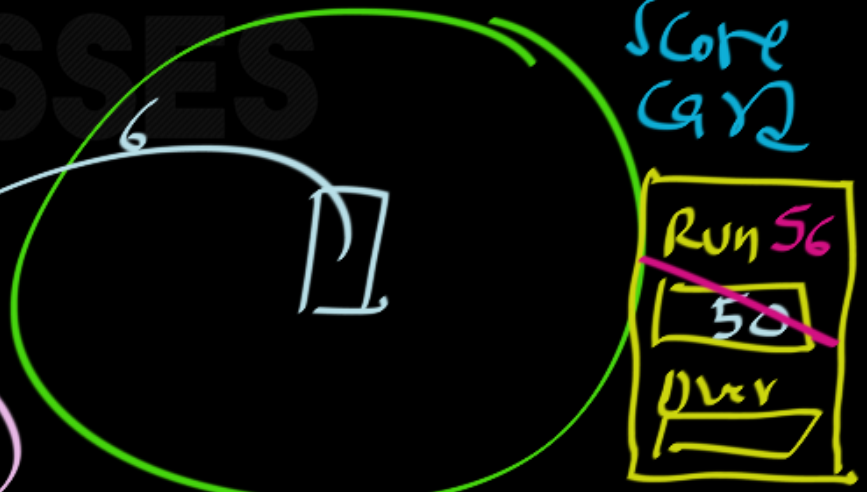
output / state

Real life examples of Seq. Ckts:

(2) Almost Every (useful) Circuit is Sequential Circuit.

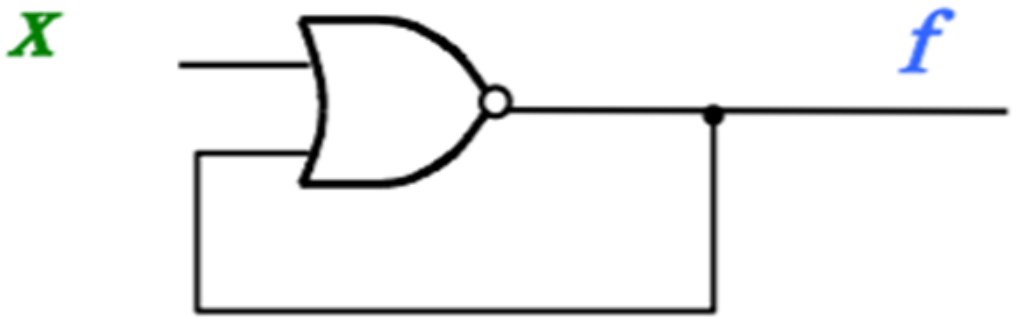
IISc Cricket:

Next output = f(Current input, Prev o/p)



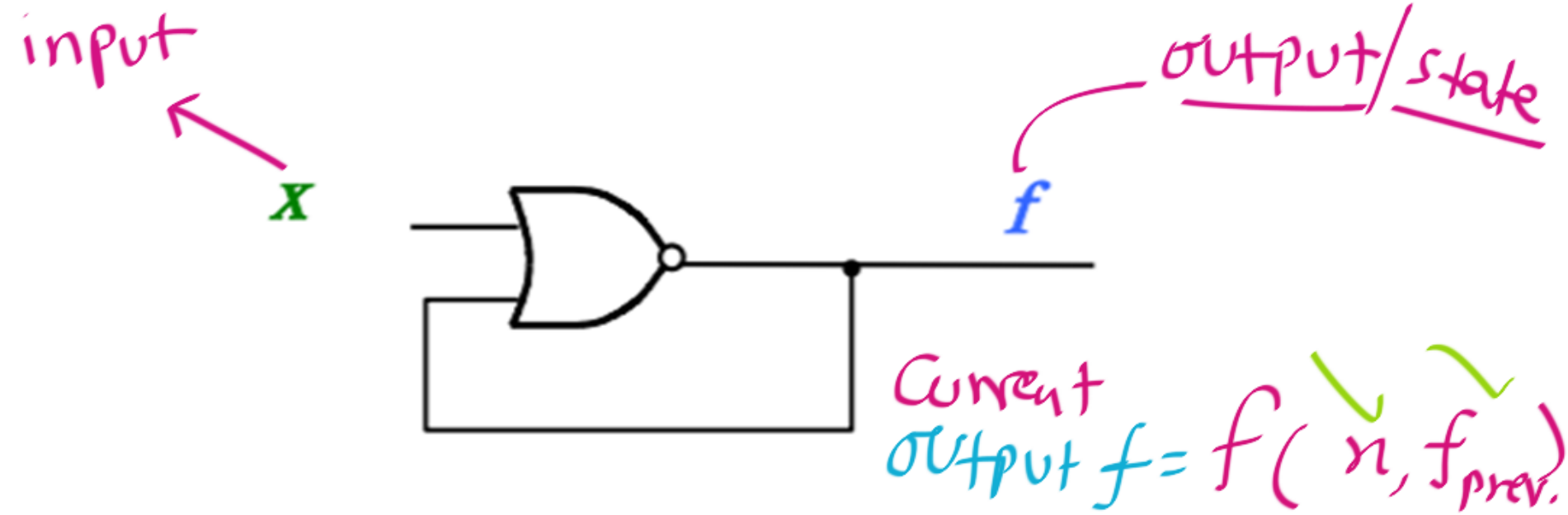
Is it Combinational or Sequential ?

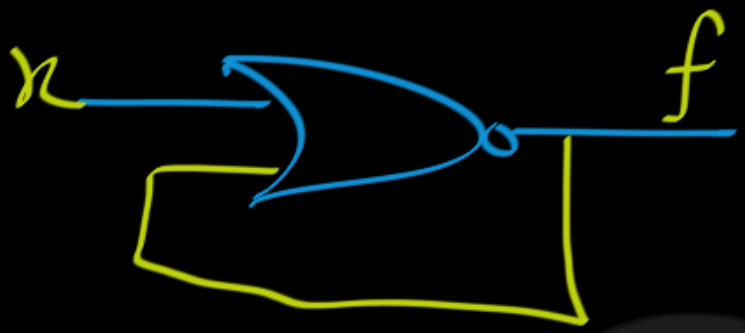
Let's Try to Analyze This Circuit



Seq. Ckt

Let's Try to Analyze This Circuit

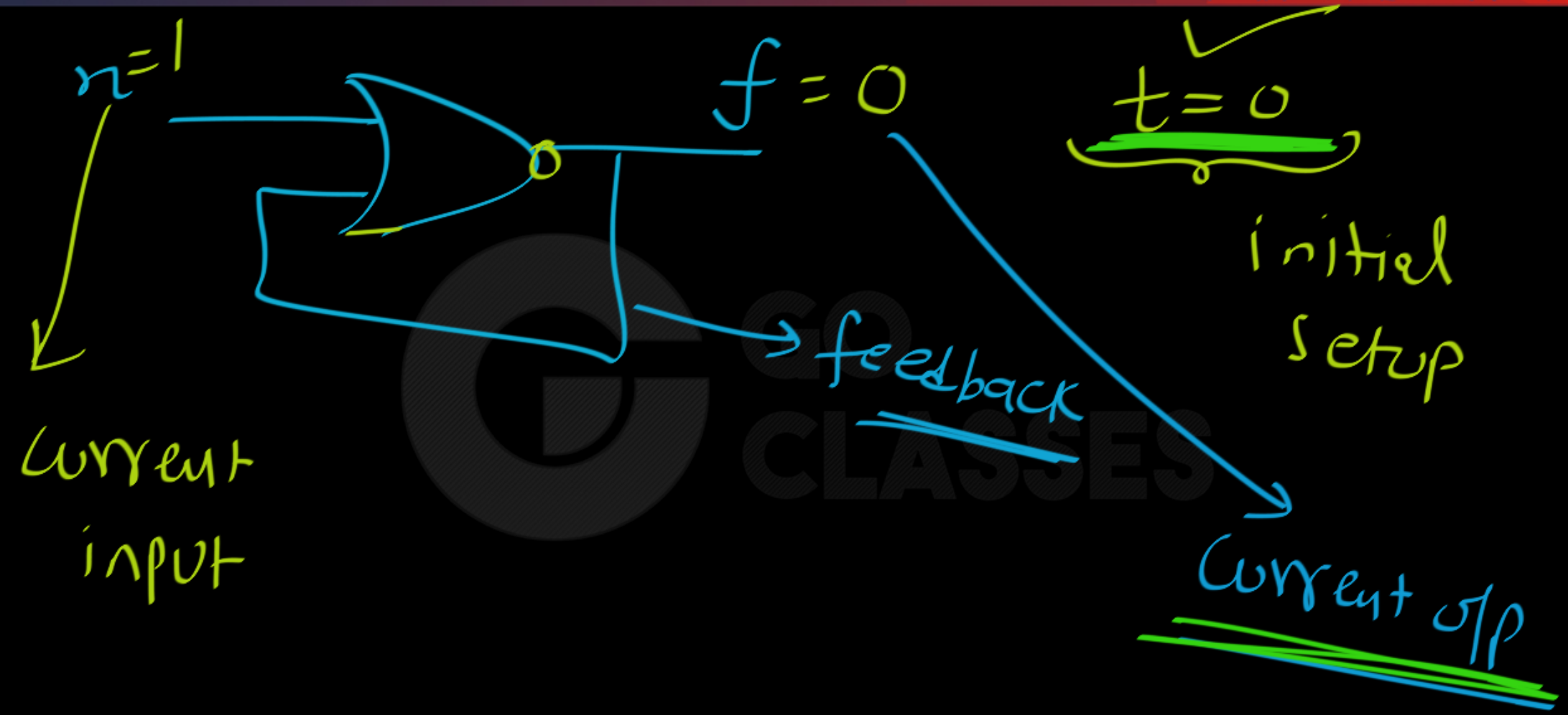


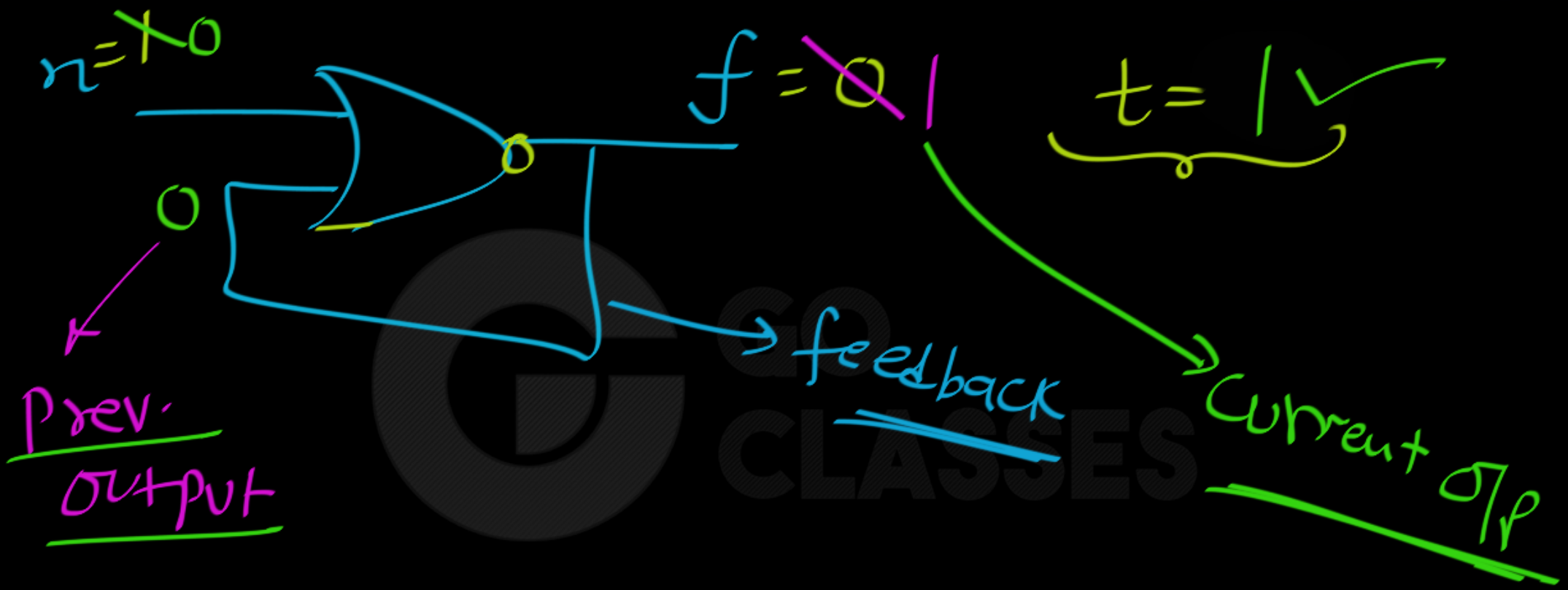


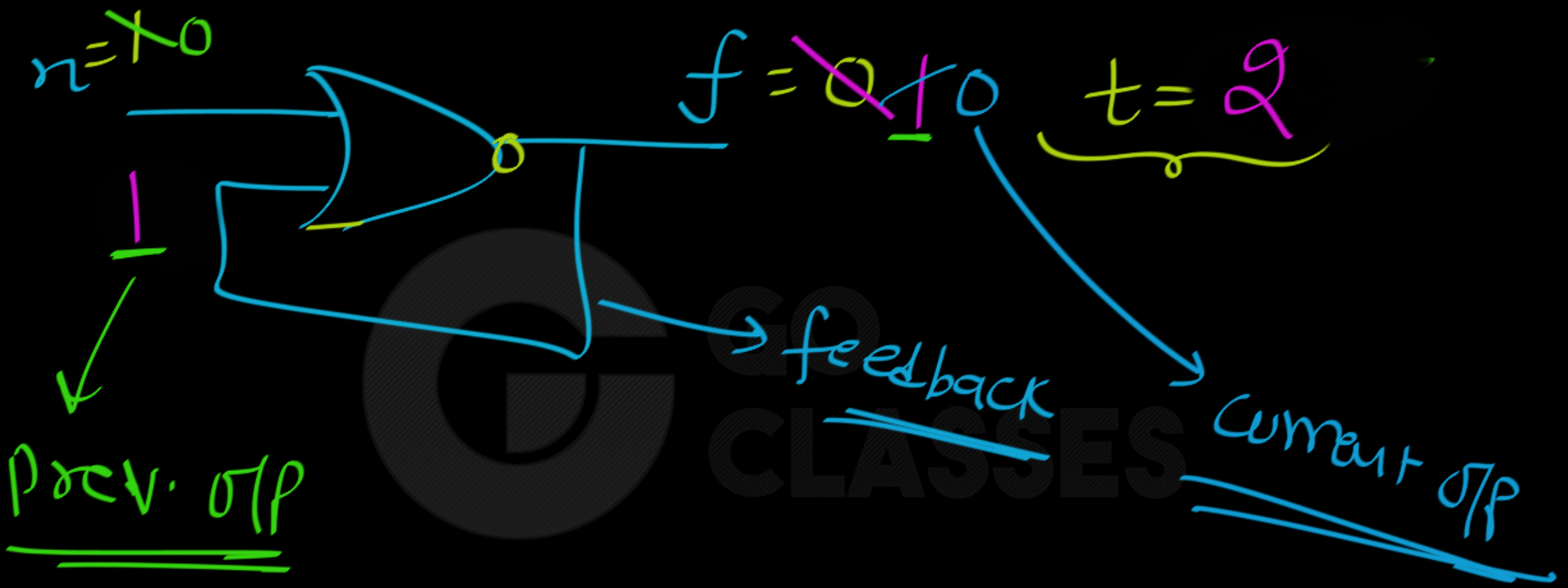
x	f	time
1	0	t = 0
0	1	t = 1
0	0	t = 2

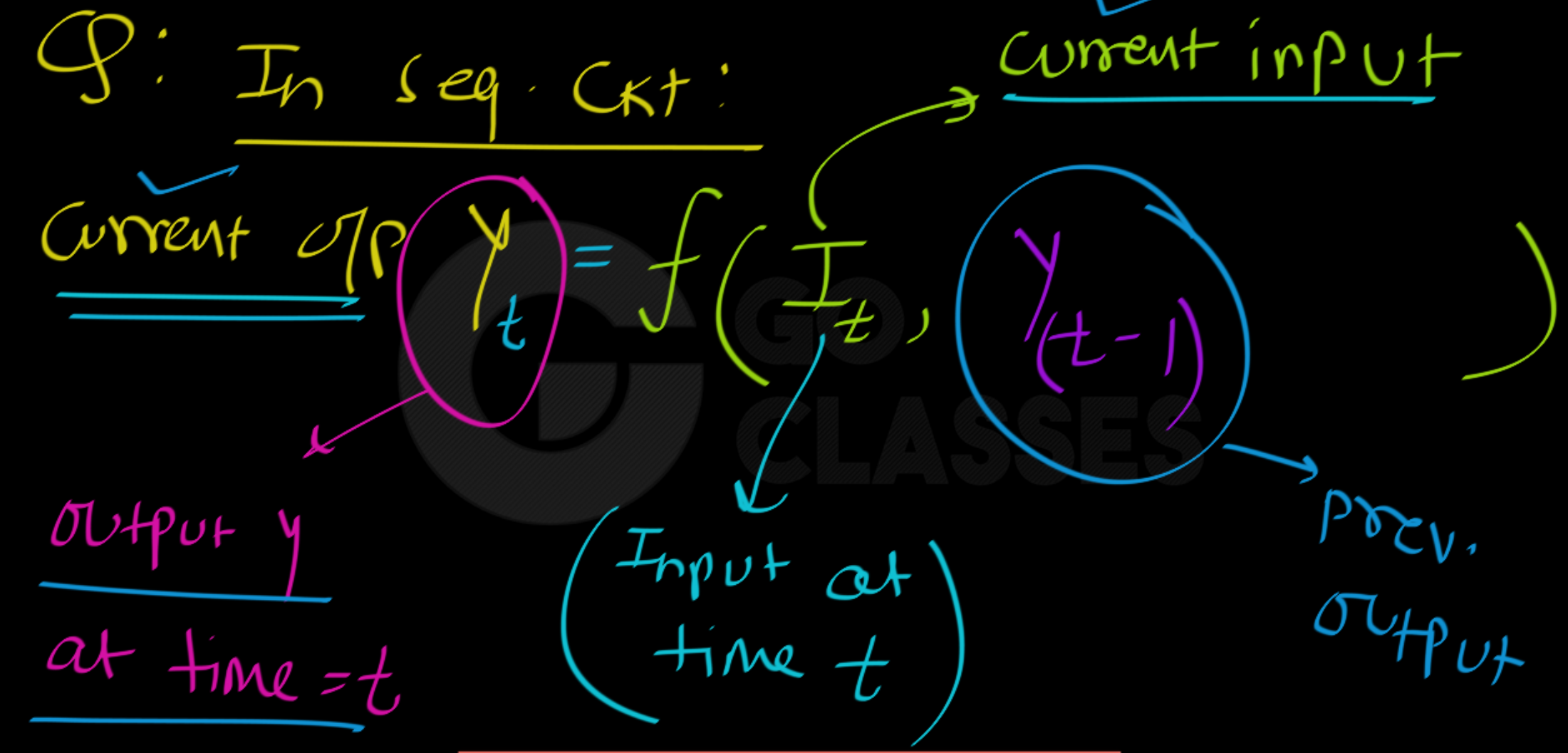
input same

O/P Different


x=1
f=0
t=0
initial setup
feedback
current input
current o/p


x=x0
0
f=01
t=1
Prev. output
feedback
Current o/p


x=0
f=0 1 0
t=2
1
feedback
prev. o/p
current o/p


Q: In Seq. Ckt:
Current input
Current o/p $y_t = f(I_t, y_{(t-1)})$
output y at time = t
(Input at time t)
prev. output

Q: for Seq. ckt:

(a) Current o/p Depends on Prev. output

(b) " " " " Prev. all o/ps.

Q: for Seq. ckt:

(a) Current o/p Depends on Prev. output

(b) " " " " Prev. all o/ps.

Y_t = o/p Y at time t

I_t = Input I at time t

Y_{t-x} = o/p Y at time t-x

$$Y_t = f(I_t, Y_{t-1})$$

$$Y_t = f(I_t, f(I_{t-1}, Y_{t-2}))$$

$$Y_t = f(I_t, f(I_{t-1}, f(I_{t-2}, Y_{t-3})))$$

Sequential Ckt: → State = output

Current o/p (state) = f(Current input ; Past inputs ; Past outputs)

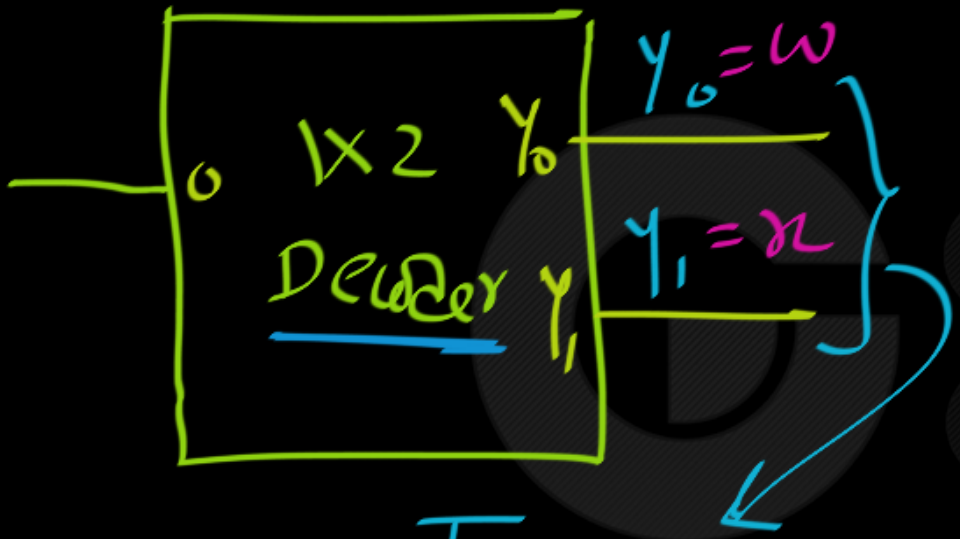
→ Depends on all prev. history

Small Small Points which are very important :

In Seq Circuit :

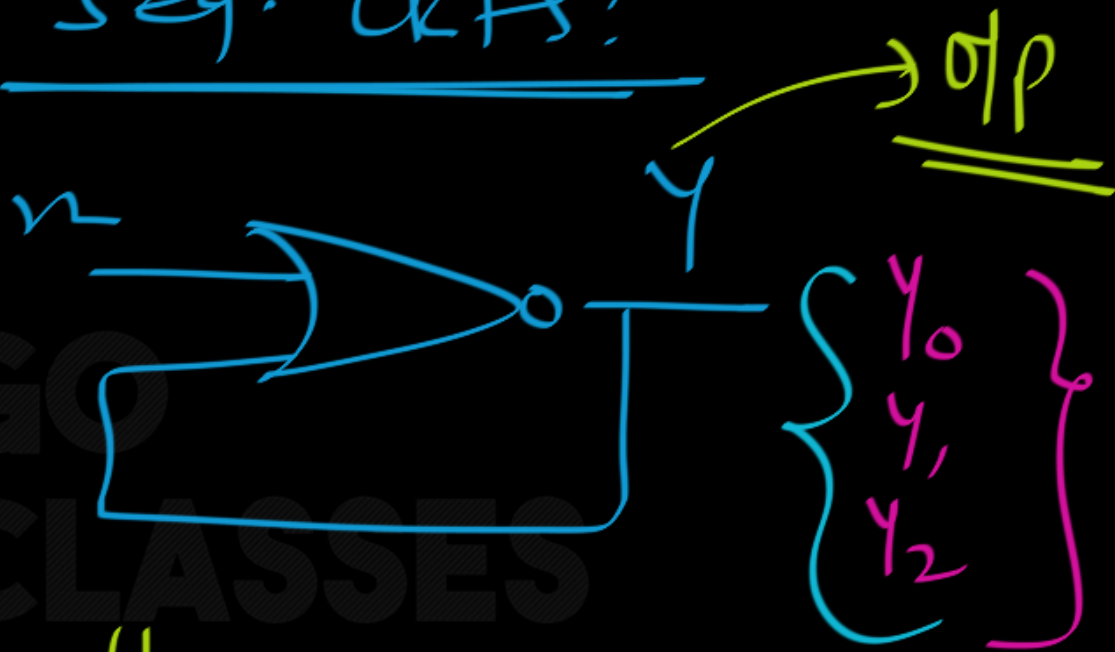
Y_0 = output Y at time $t=0$; Y_1 = output Y at time $t=1$

Combinational Ckts



Two Different o/p lines

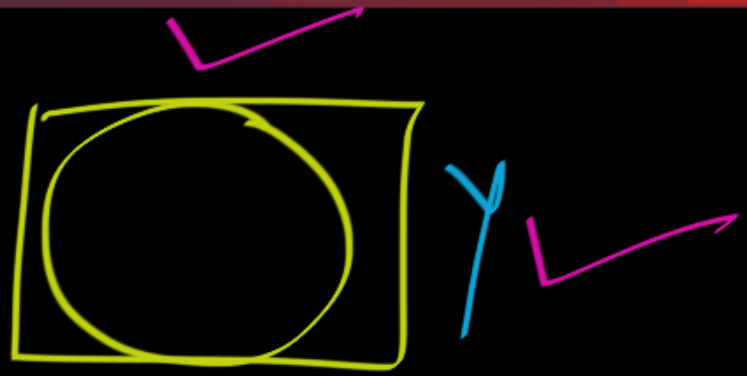
Seq. Ckts:



#output lines = 1

= Y

Traffic light:



Y_0 = Red

Y_1 = Yellow

Y_2 = Green

Human Growth: ⟶ analogous to Seq. Ckt.

age = 2 — idiot & cute

age = 10 — Headach & cute

age = 18 — intelligent & Not cute

age = 60 — Judge mental & cute

Different Conventions :

Author 1 :

Current o/p $= Y_t = Y(t)$

Prev o/p $= Y_{t-1} = Y(t-1)$

Next o/p $= Y_{t+1} = Y(t+1)$

Different Conventions :

Author 2 :

Next o/p = Y_n ← next o/p

Current o/p : Y

Q:

In Sequential Circuits, how to remember previous output?

Q:

In Sequential Circuits, how to remember previous output?

memory Circuit

Ans: Some Memory Device/element

What is the most Basic info. we need to store ? ⇒ single bit

Device D :⟶ Can store one bit (0/1)

To store 00101 ⇒

Next Topic:

Memory Element/Storage

Motivation for Sequential Circuits :

So far, our circuits have been converting inputs to outputs without storing any data.

To do more advanced things (e.g., to build computers), we need components that can store data. Can we make a component that "remembers" using the components that we know already?

One bit Storage Device

we need to create.

What does "Remembering Information" mean?

What does "Remembering Information" mean?

Store and Retain until it is changed further.

Device: D

$x = 1$ → 1

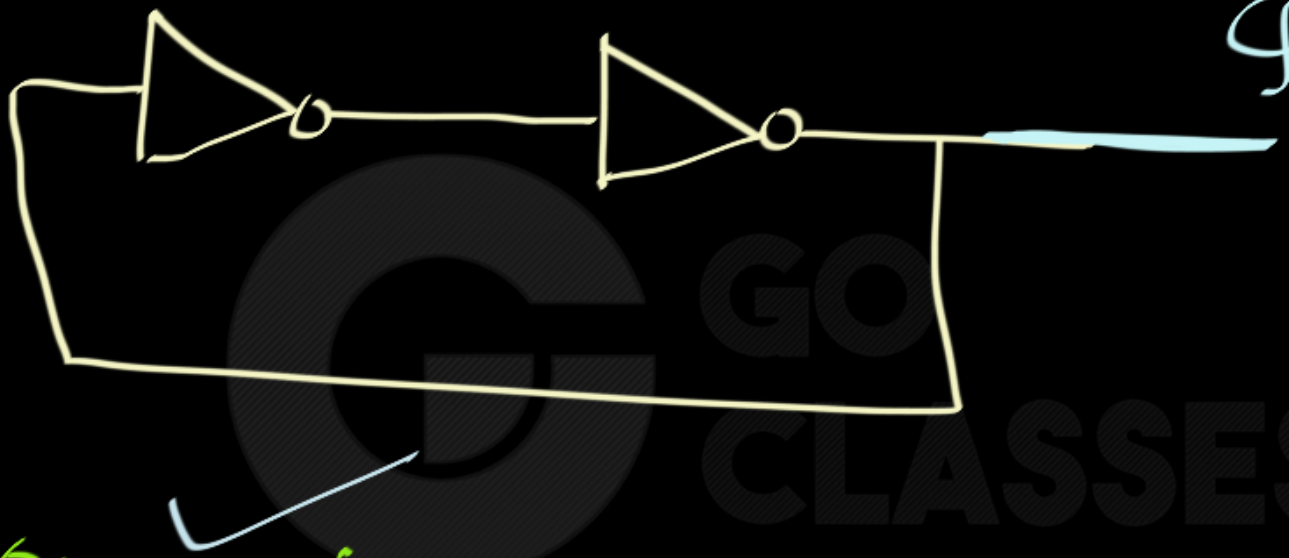
Removing signal : $x = 0$

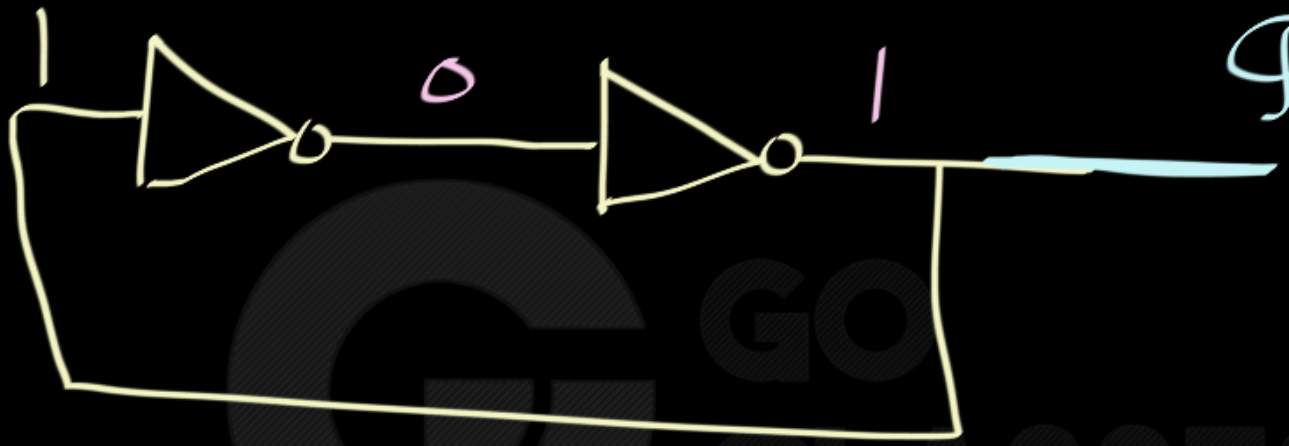
Absence of voltage

D 1

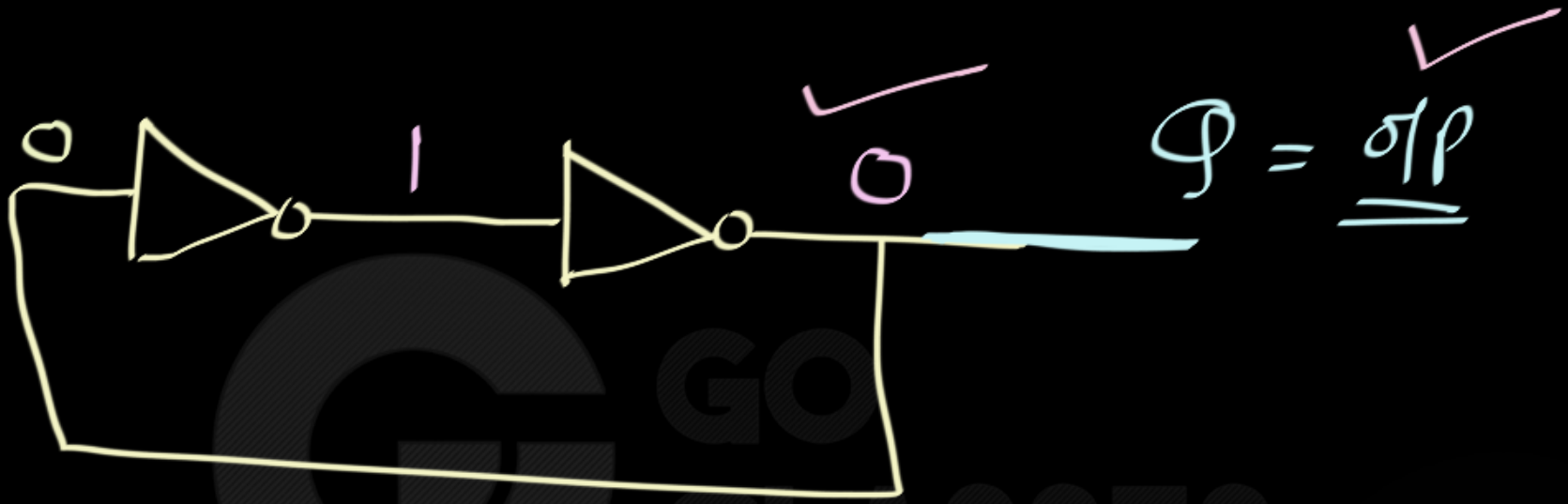
Next Topic:

Basic Memory Element/Device For One Bit Storage


Q = Q'
one bit Storage Device

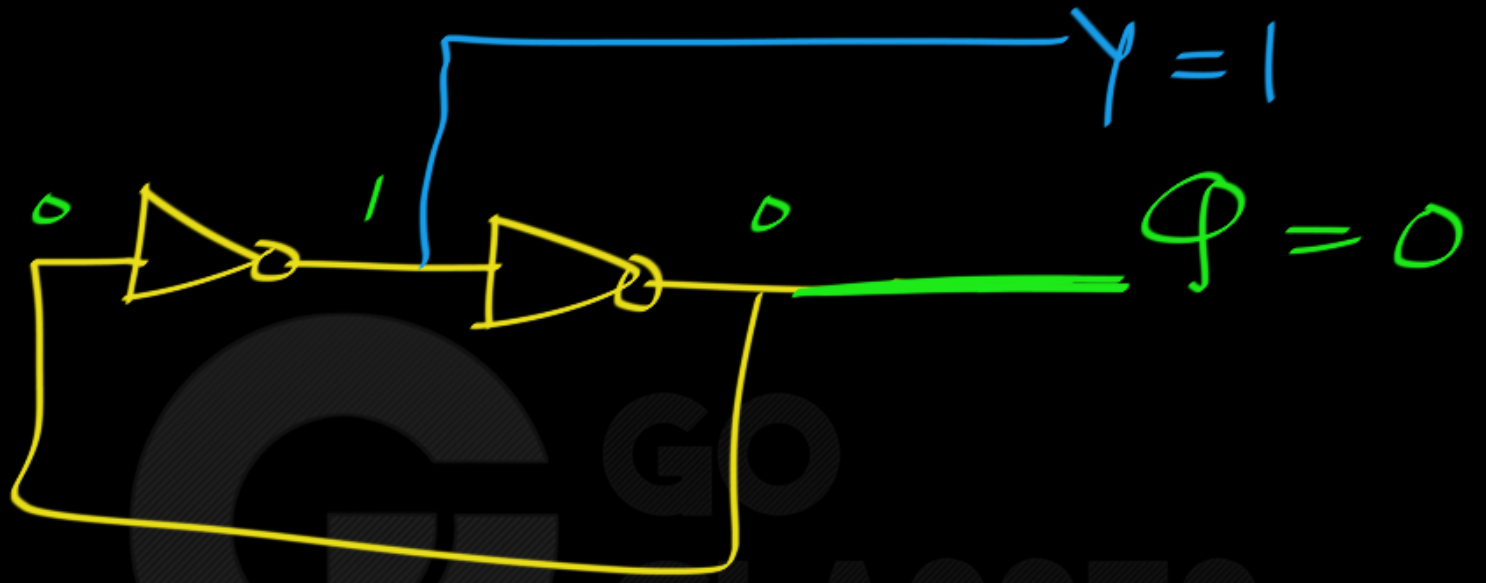


Storing 1, Retained



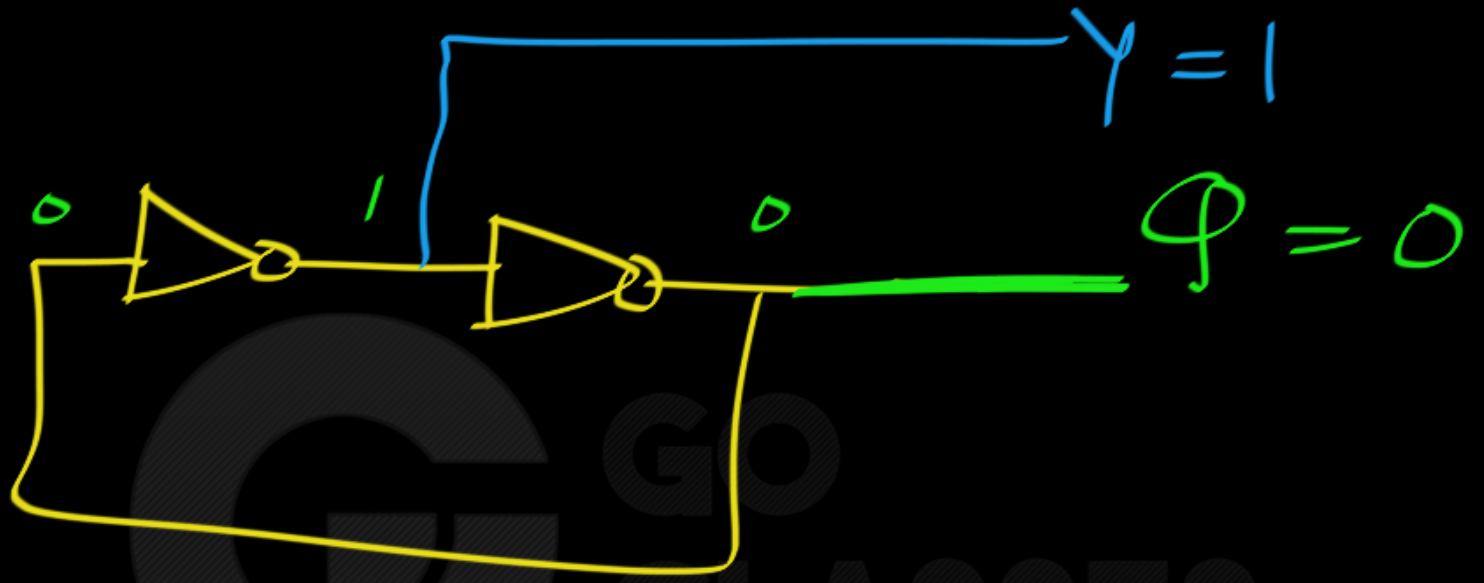
Storing 0, Retained

Q:



Can we say that this is a 2-bit Storage Device?

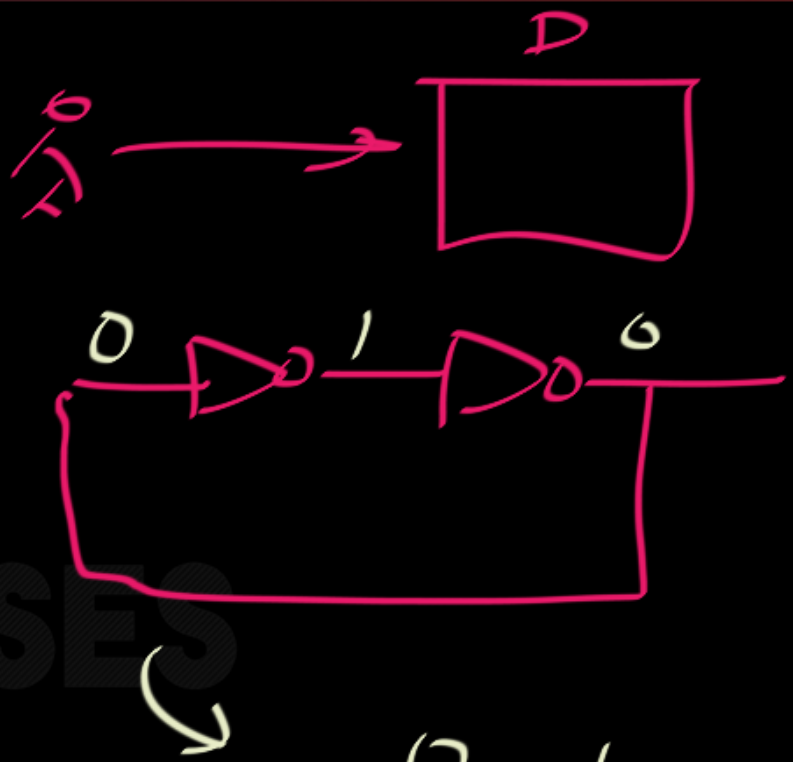
Q:

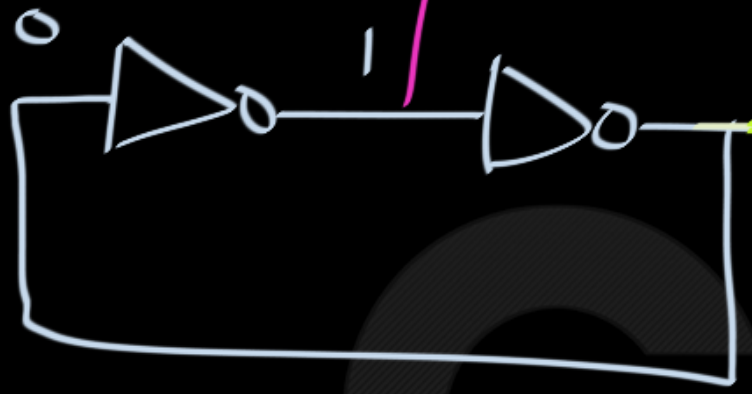


Can we say that this is a 2-bit Storage Device? No

2-bit Storage Device :

{ 0, 0
0, 1
1, 0
1, 1 }





$\bar{Q}$ o/p

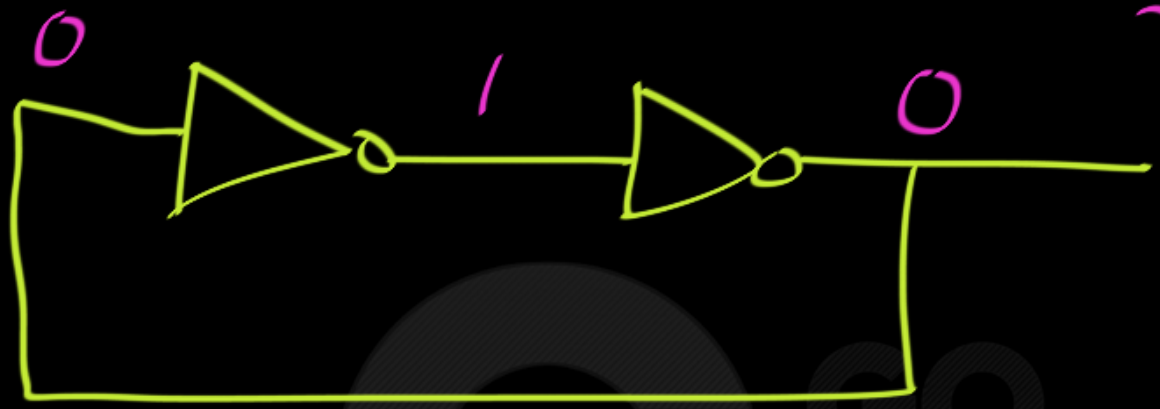
Q o/p

main o/p

Storing $Q=0$ But

$\bar{Q}$ we are getting

Complimentarily (freely)



fixed for indefinite time

"Cross-Coupled Inverter"

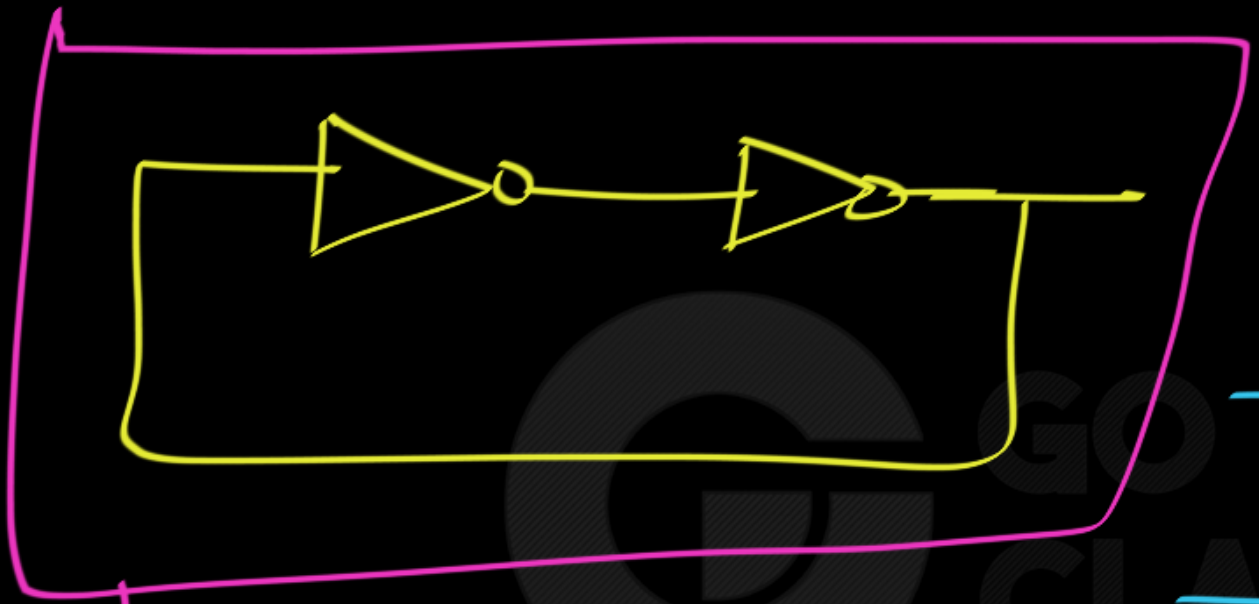
↳ We can Store & Retain one bit.

Kid Toy

3 Comps

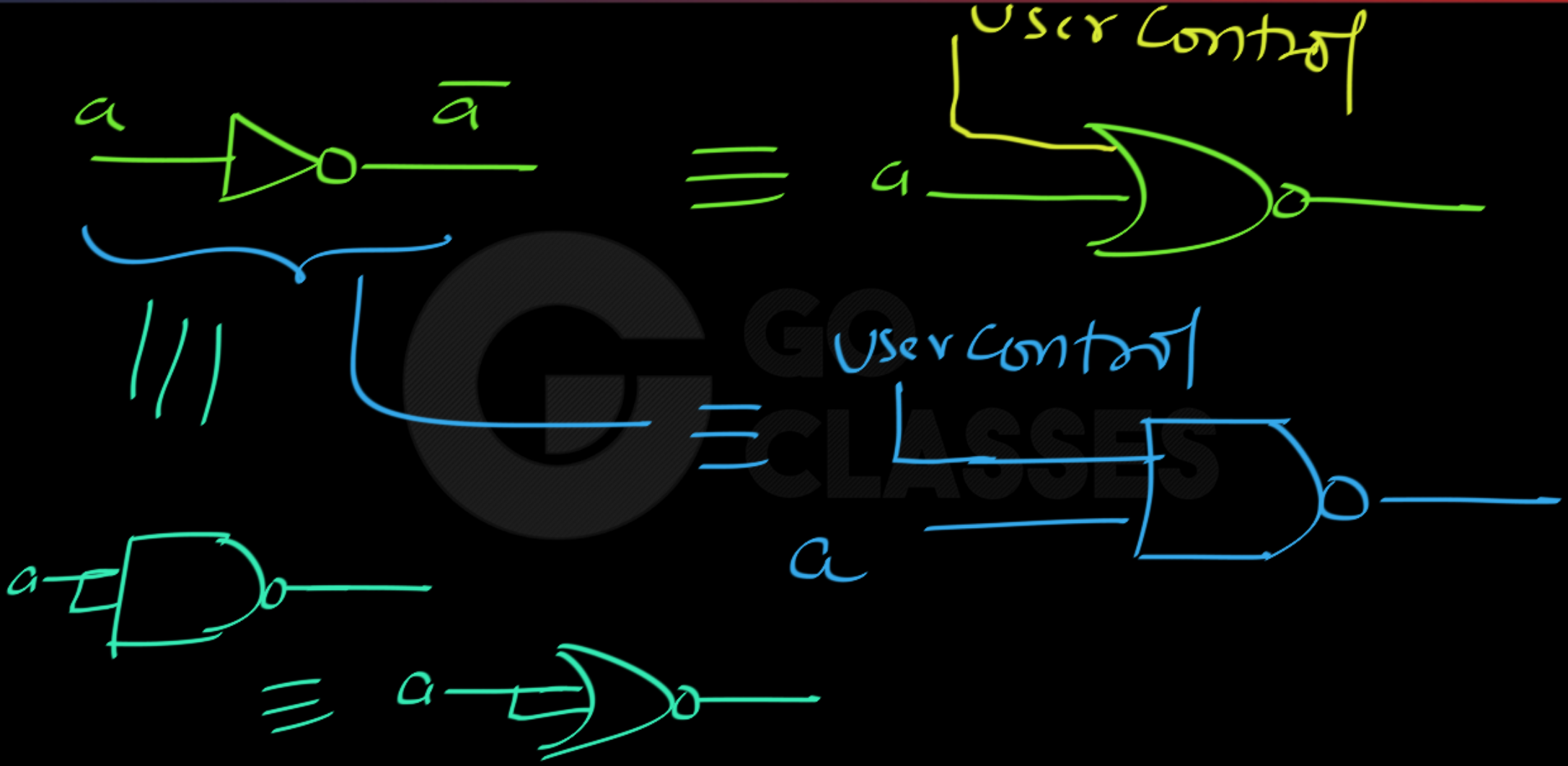
Pre-Programmed

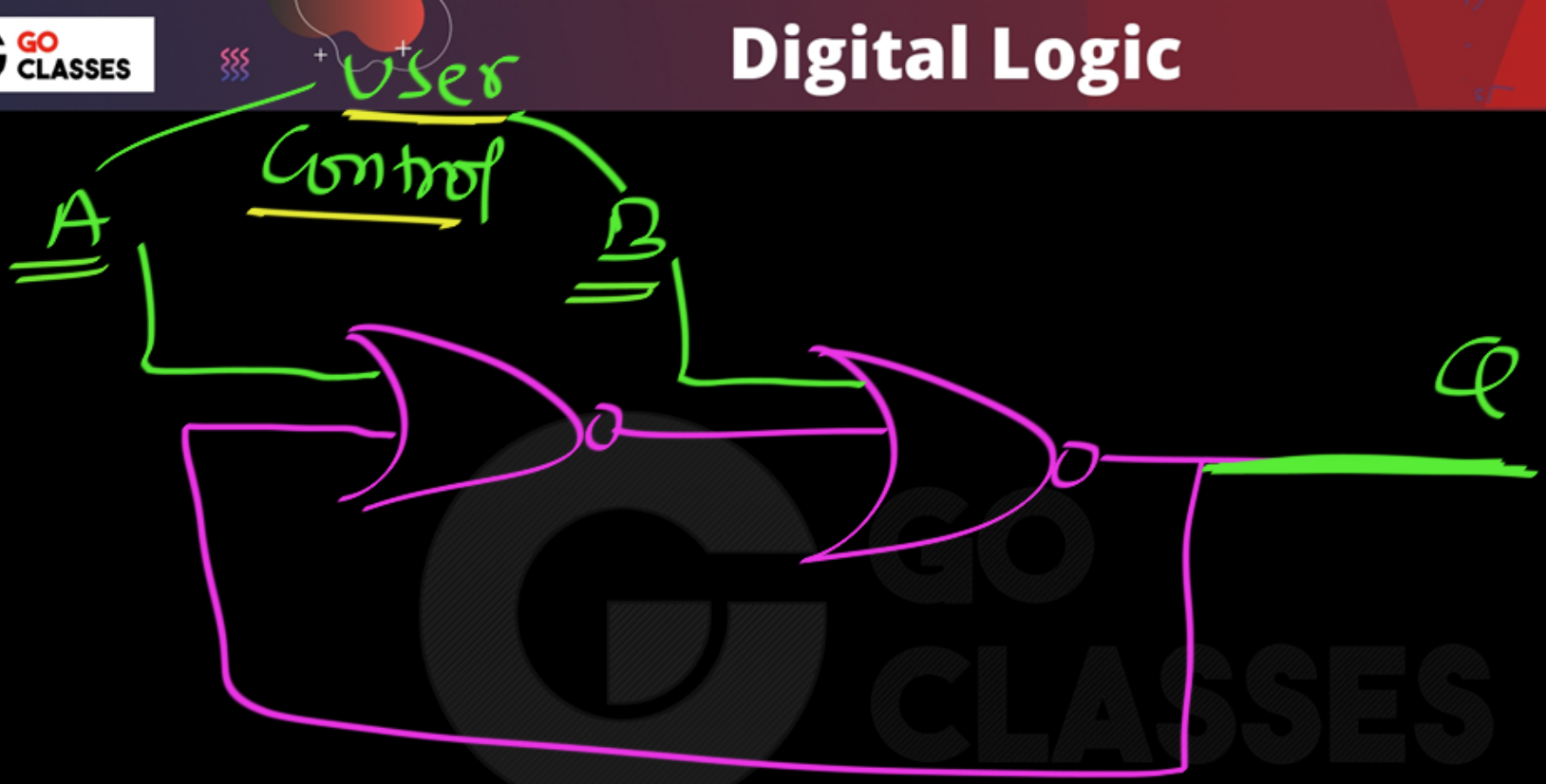
Not in User Control



No User Control.

→ Basic Idea to store one bit.


a
$\bar{a}$
User Control
a
User Control
a
a



Resume:

9:35

Store one bit :

(1)

Can we call it storage Device ? No

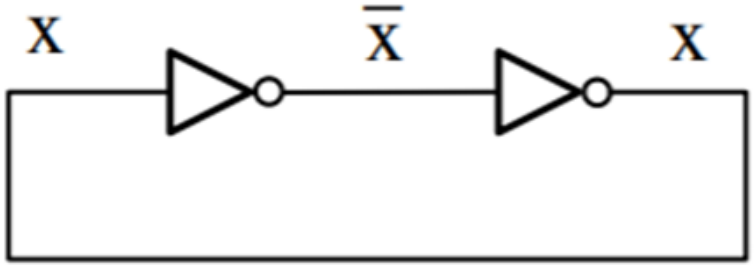
To store 0, Continuously $x=1$ input must be applies.

(2)

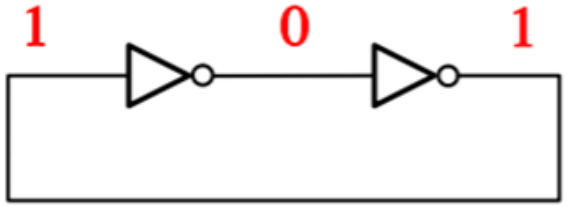
Cross Coupled inverter

Storing 0 & Retaining

A simple memory element with NOT Gates

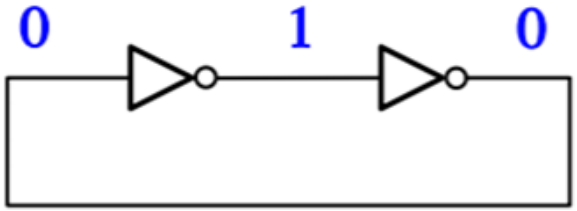


A simple memory element with NOT Gates



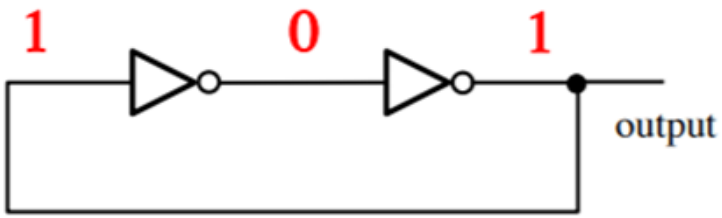
The circuit will stay in this state indefinitely.

A simple memory element with NOT Gates

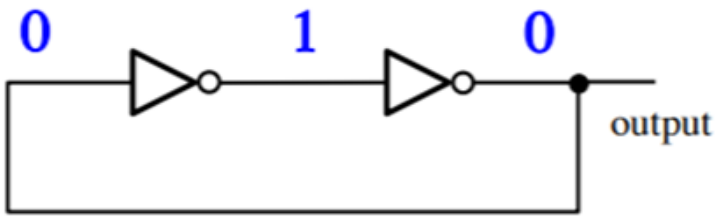


The circuit will stay in this state indefinitely.

This circuit can be in two possible states

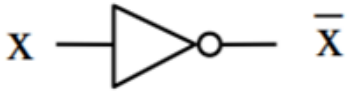


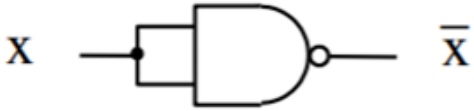
used to store a 1



used to store a 0

Building a NOT gate with a NAND gate





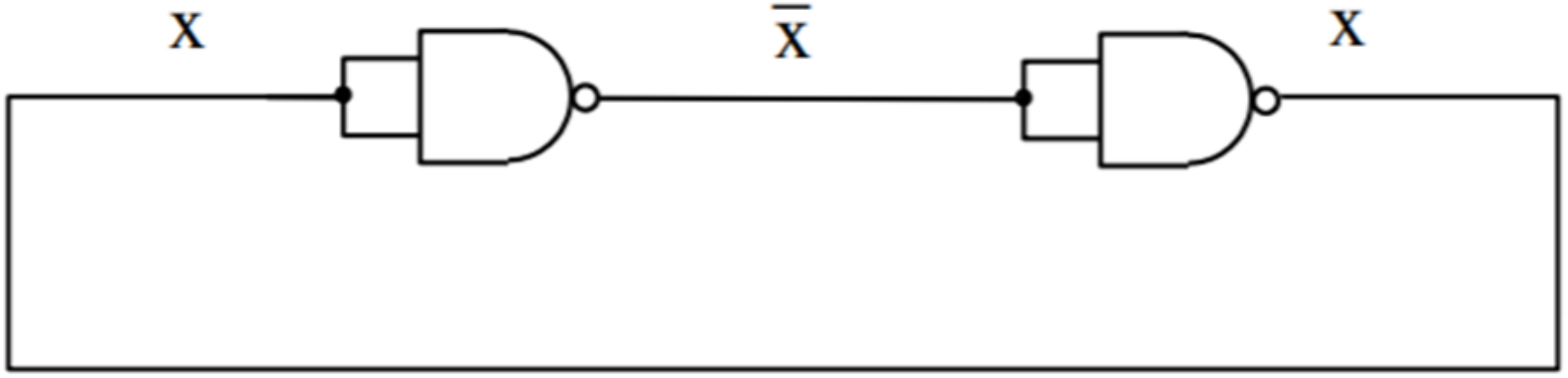
x	$\overline{x}$
0	1
1	0

x	x	f
0	0	1
1	1	0

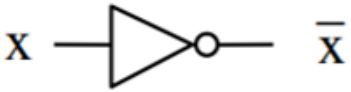
impossible combinations

Thus, the two truth tables are equal!

A simple memory element with NAND Gates



Building a NOT gate with a NOR gate



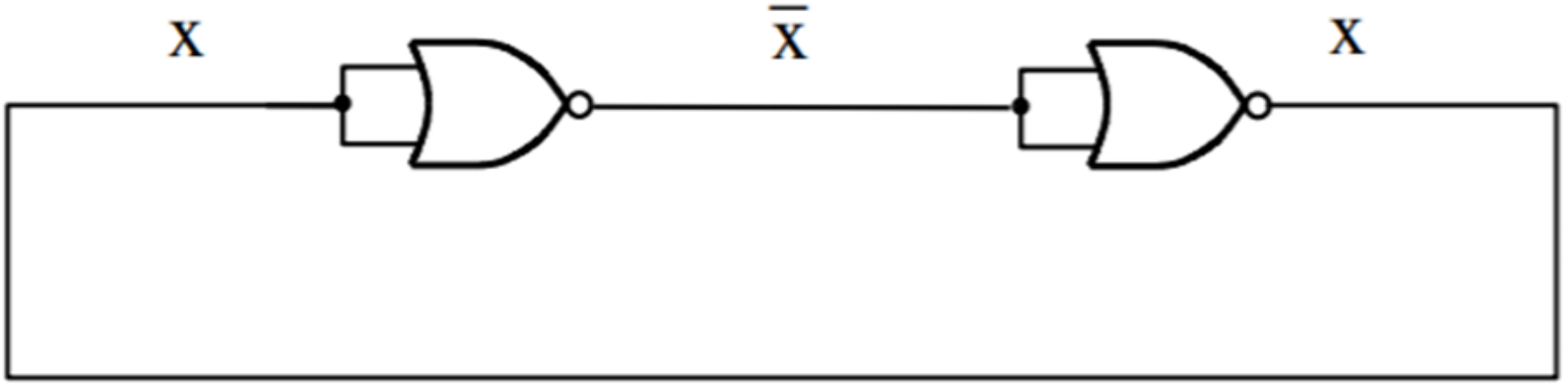
x	$\overline{x}$
0	1
1	0

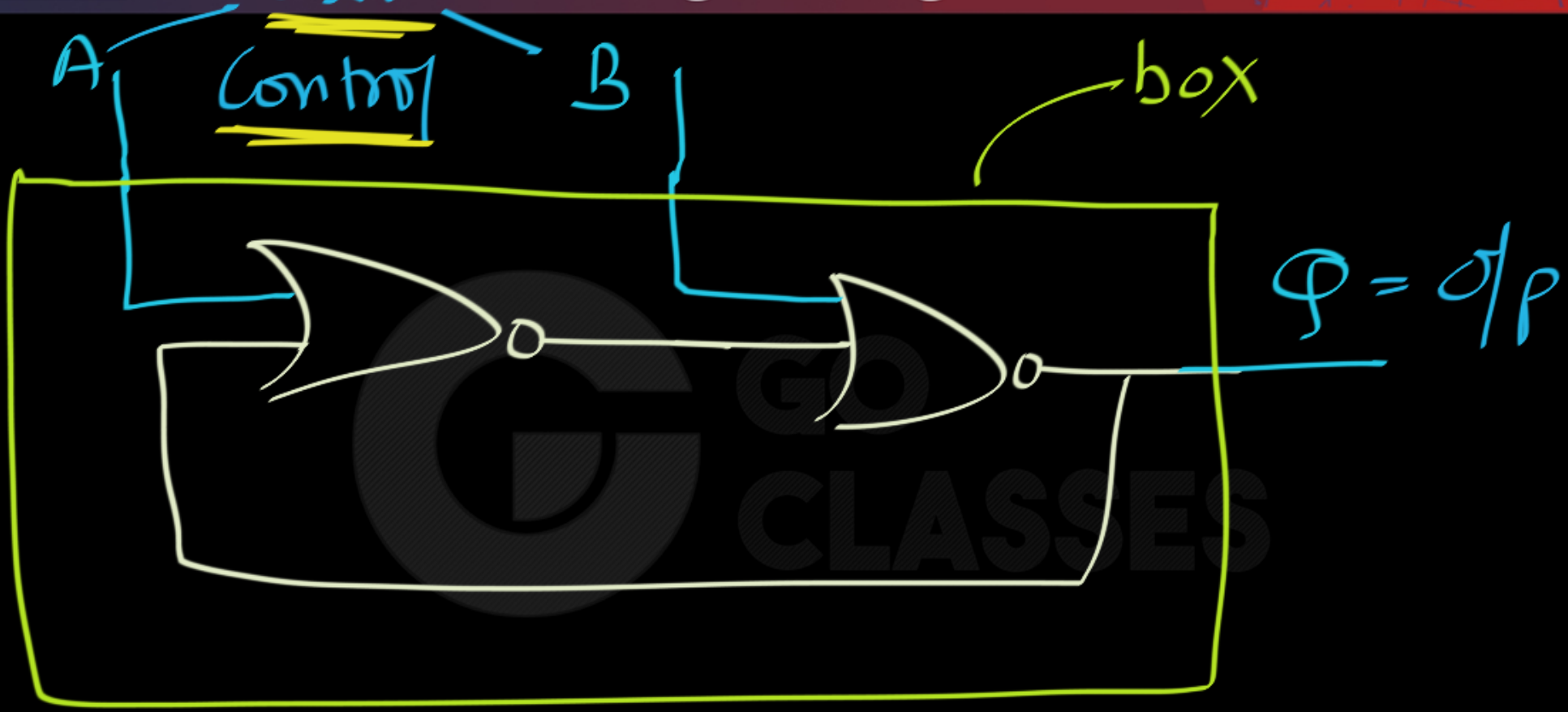
x	x	f
0	0	1
1	1	0

impossible combinations

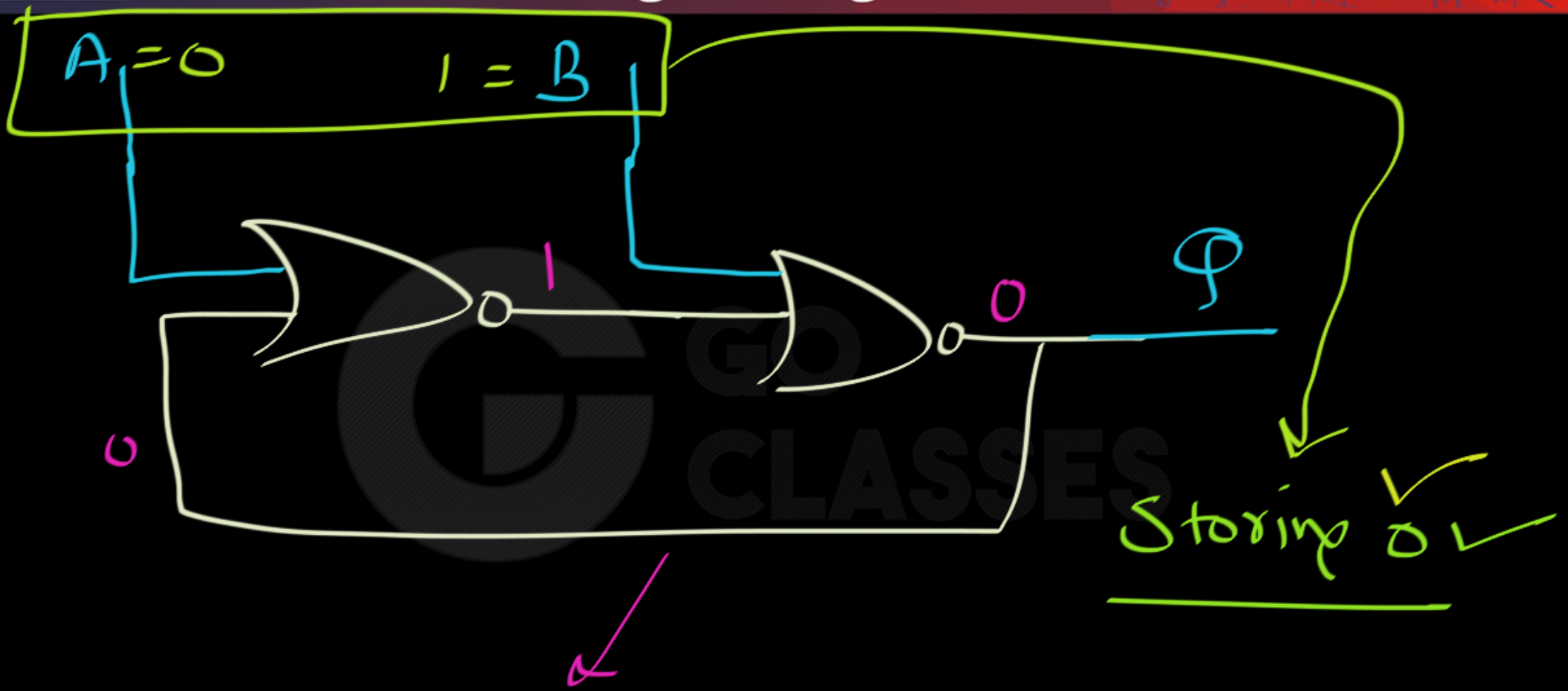
Thus, the two truth tables are equal!

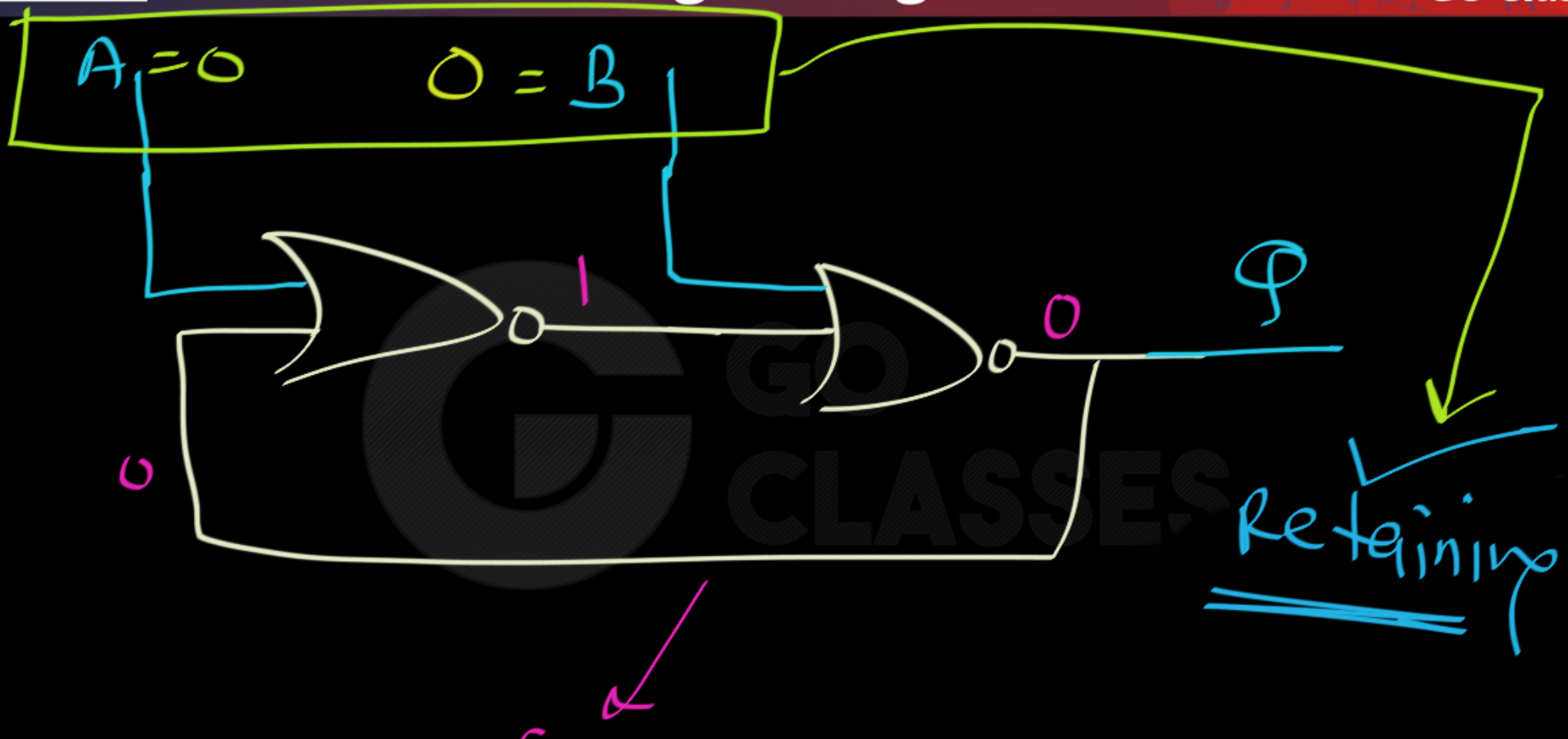
A simple memory element with NOR Gates

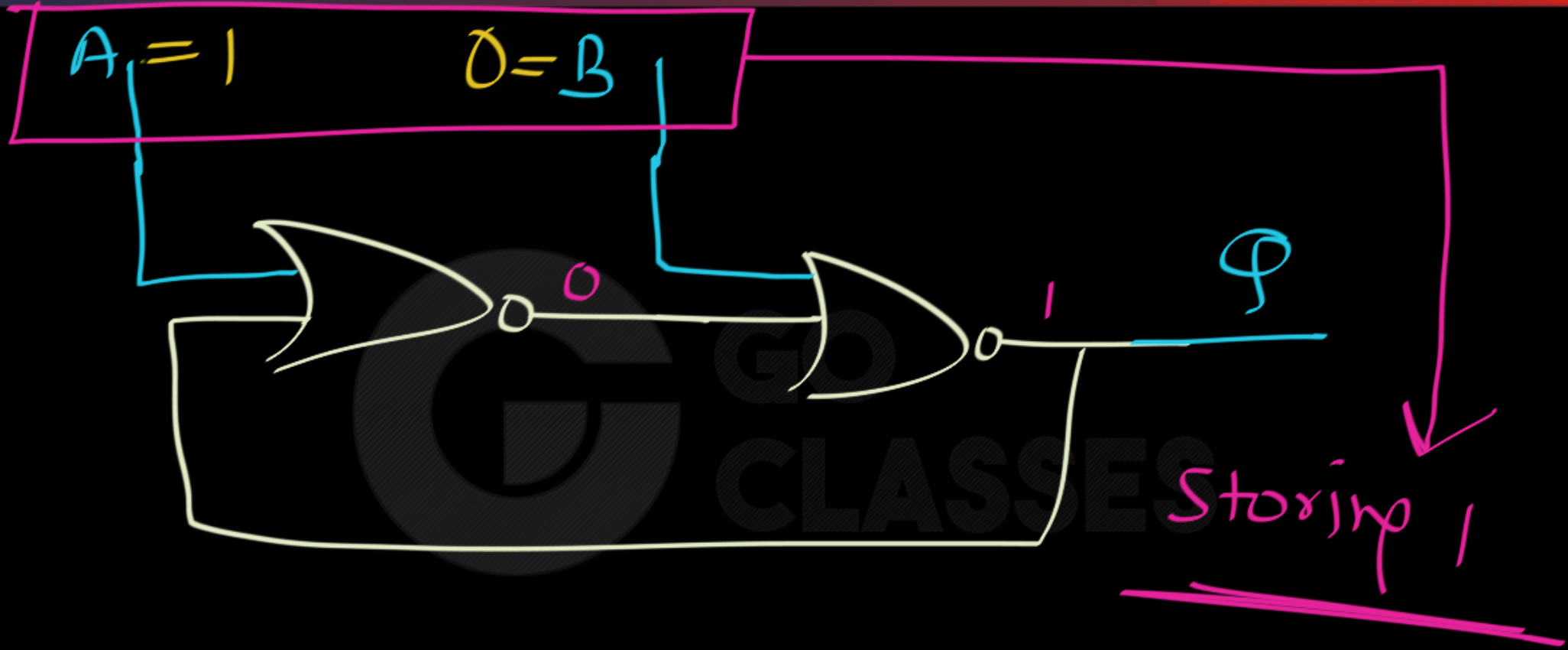


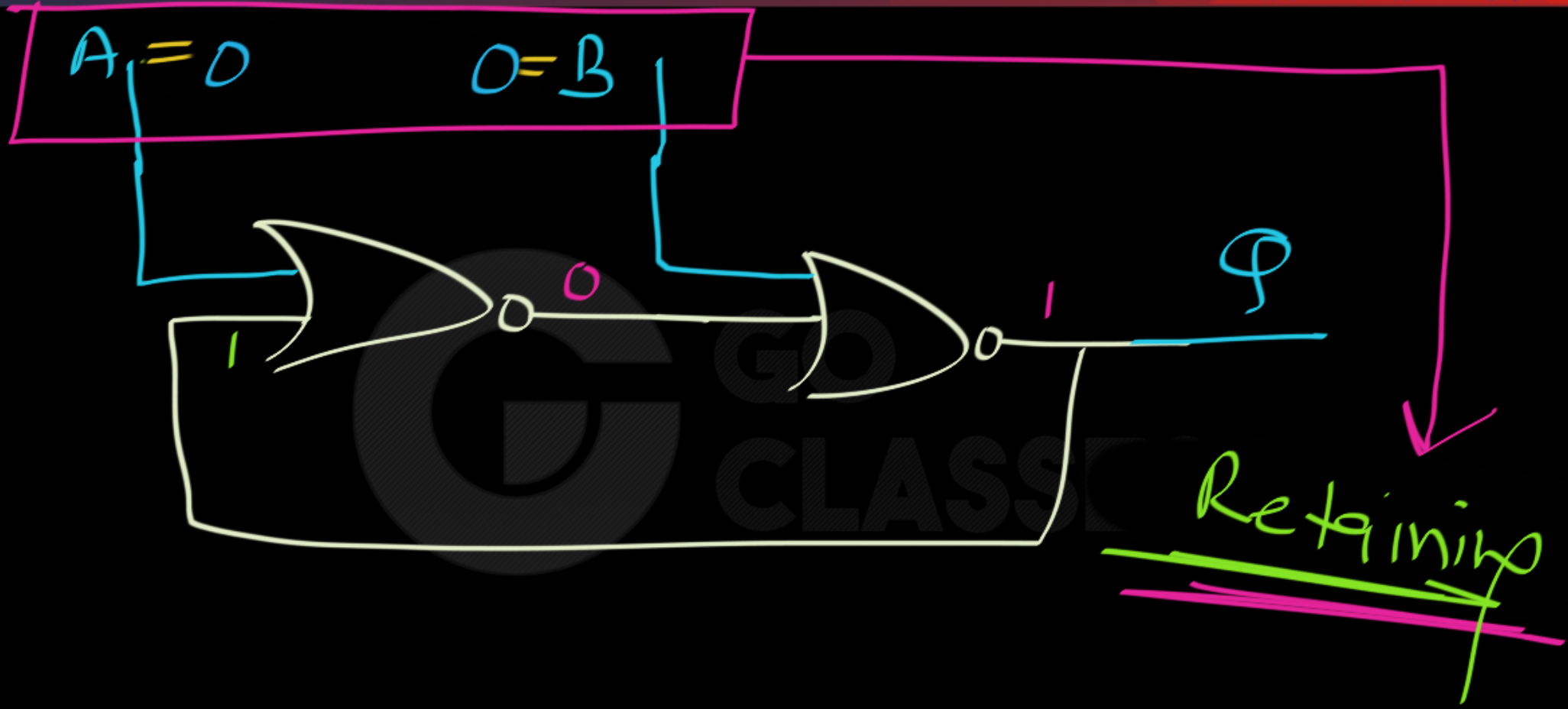

A
Control
B
box
Q = o/p

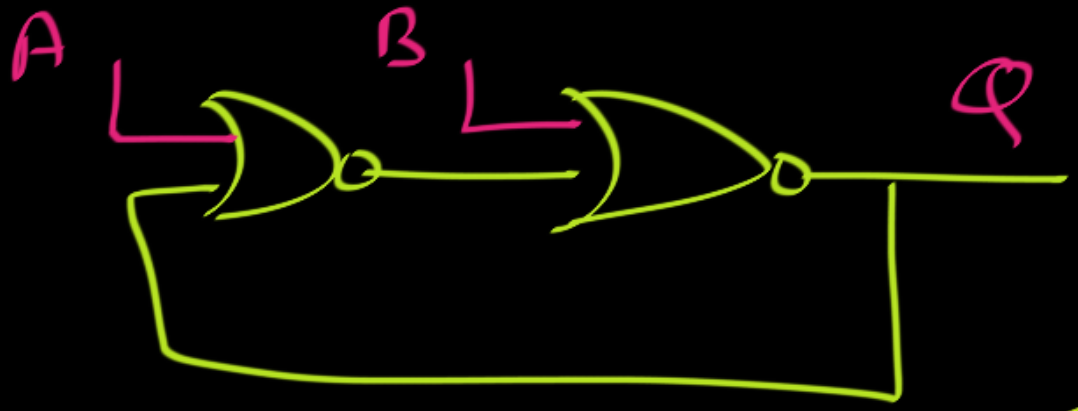

1
0


A = 0
1 = B
1
0
Q
0
Storing 0
feedback


A=0
0=B
1
0
Q
0
Retaining
feedback


A = 1
B = 0
0
1
Q
Storing 1


A=0
O=B
1
0
1
Q
Retaining



Next o/p

A	B	Q_n	time	
0	1	0	0	Storing 0
0	0	Q=0	1	Retaining
1	0	1 ✓	2	Storing 1
0	0	1	3	Retaining

Store 1 : $A = 1, \quad B = 0$

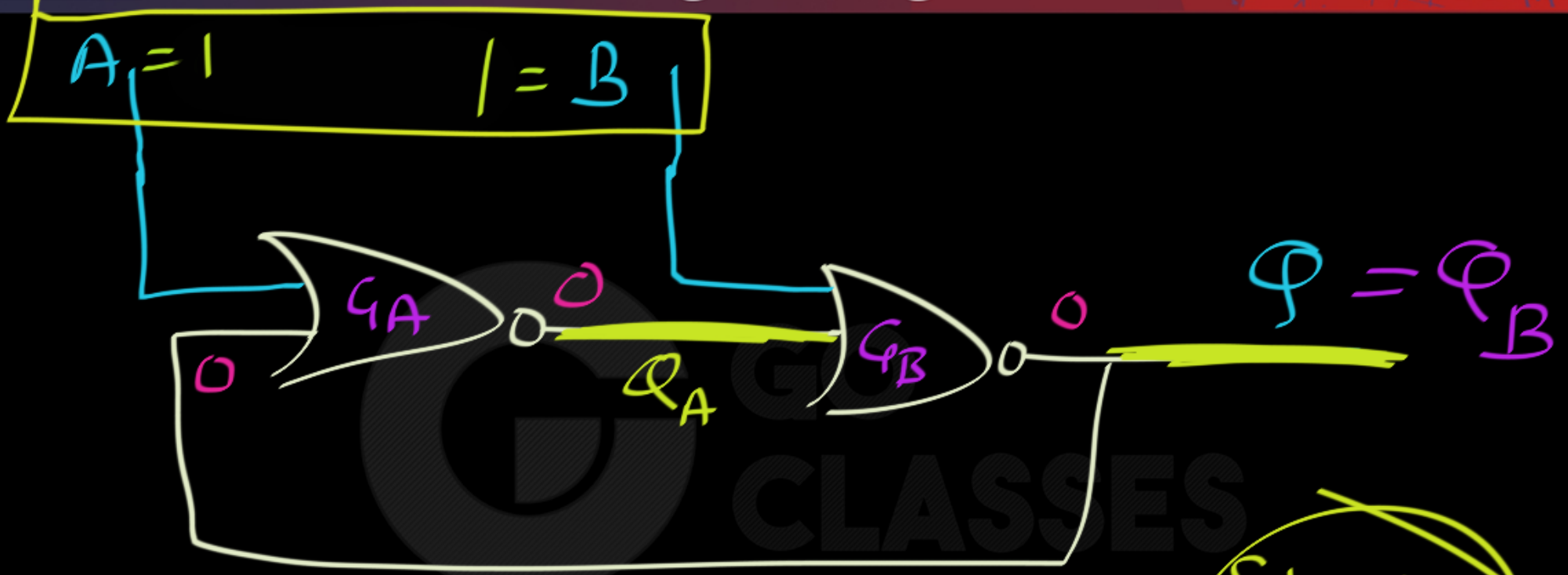
Store 0 : $A = 0, \quad B = 1$

Retain : $A = 0, \quad B = 0$

What happens when

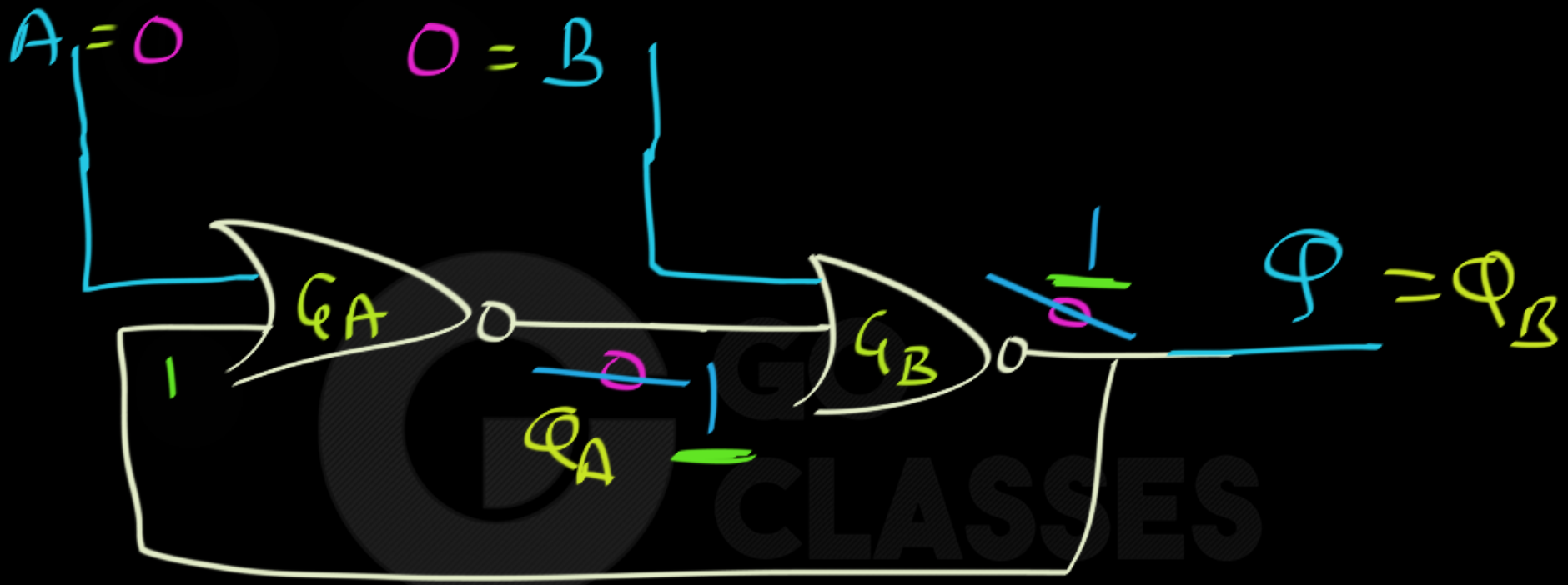
A = 1
B = 1

} ?

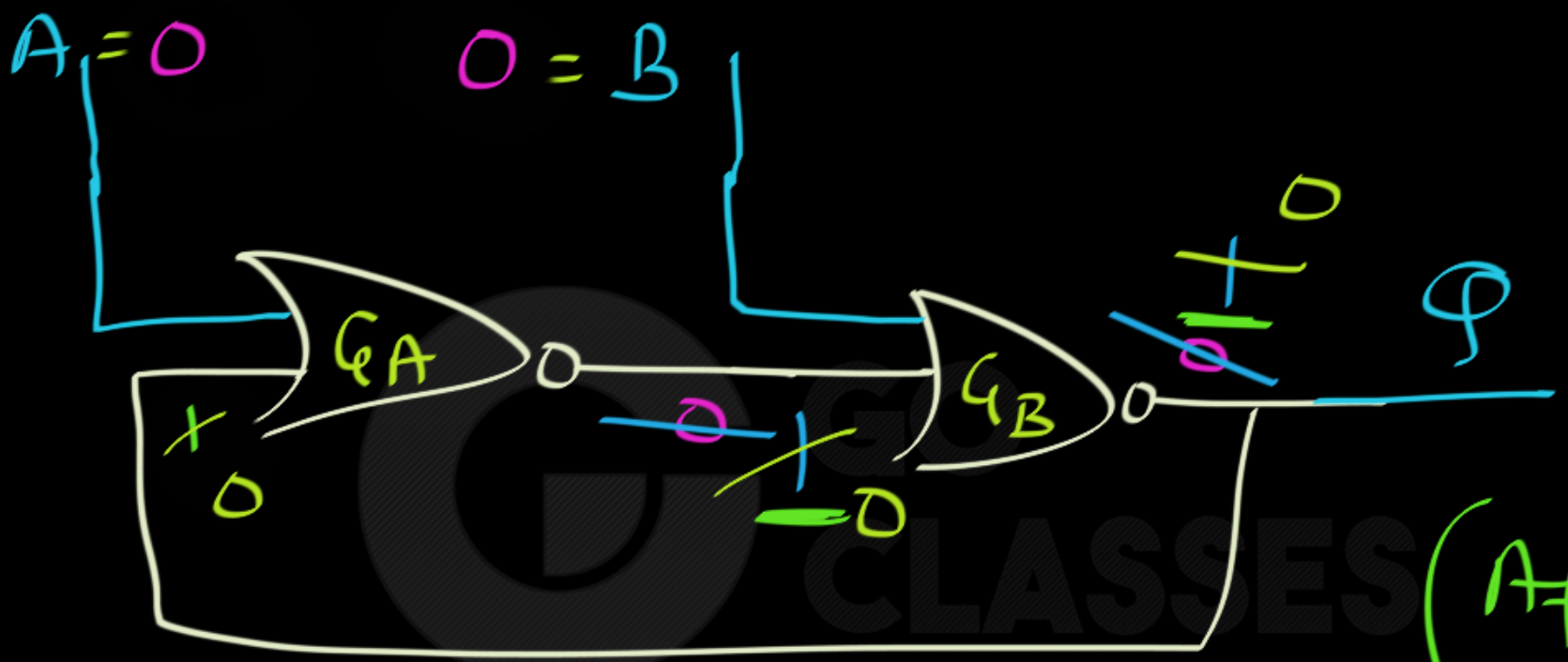


Output $Q = 0$ ✓

Stable output



If G_A, G_B have equal speeds :



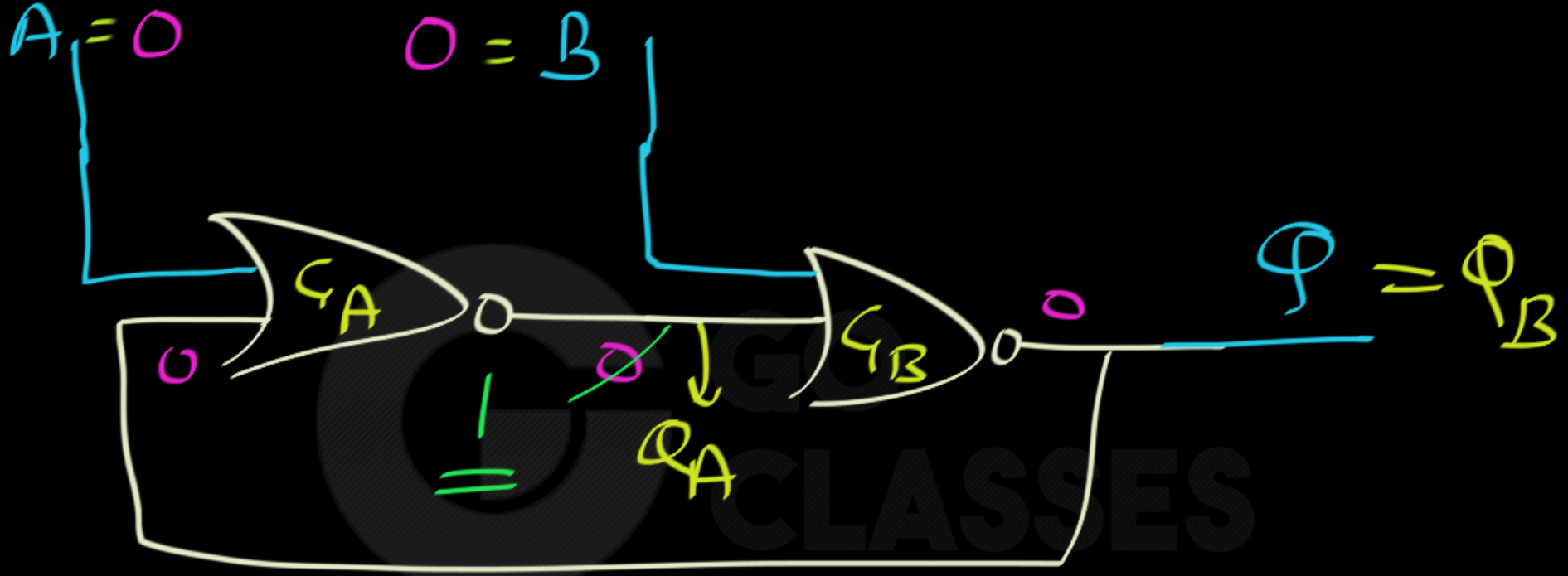
If G_A, G_B have equal speeds :

(After 1 second)

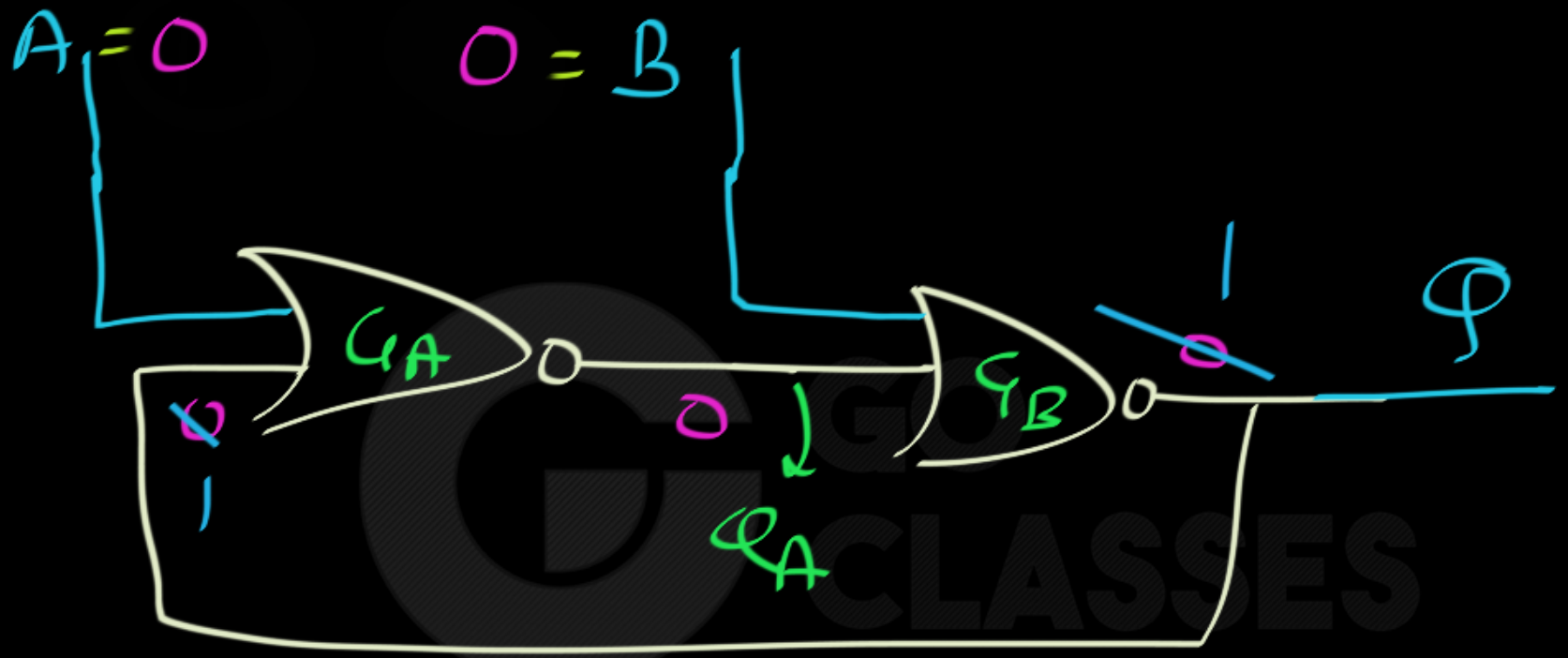
If G_A, G_B have equal speeds:

Q_A	Q_B
0	0
1	1
0	0
1	1

Not Stable o/p

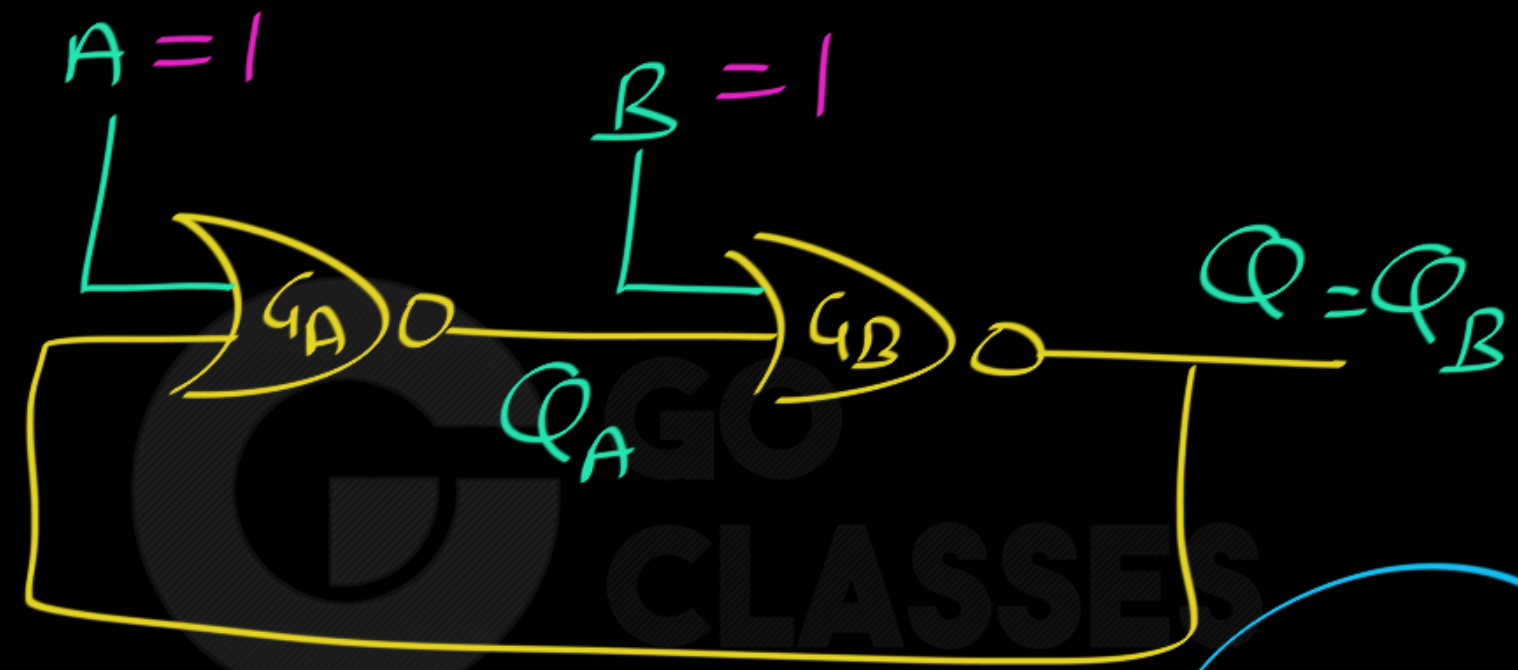


If G_A speed > G_B speed



If G_B speed > G_A speed

Note:



A	B	Q_A	Q_B
1	1	0	0

Stable O/P

But After $A = B = 1$

If You Remove input signal

(If You want to Retain)

(If You make $A = 0, B = 0$

then we Don't know what will happen.

Analogy:

Dinner: milk + curd

forbidden

Next Day: Uncertain behaviour of stomach.

So, $A=1, B=1$ is forbidden.

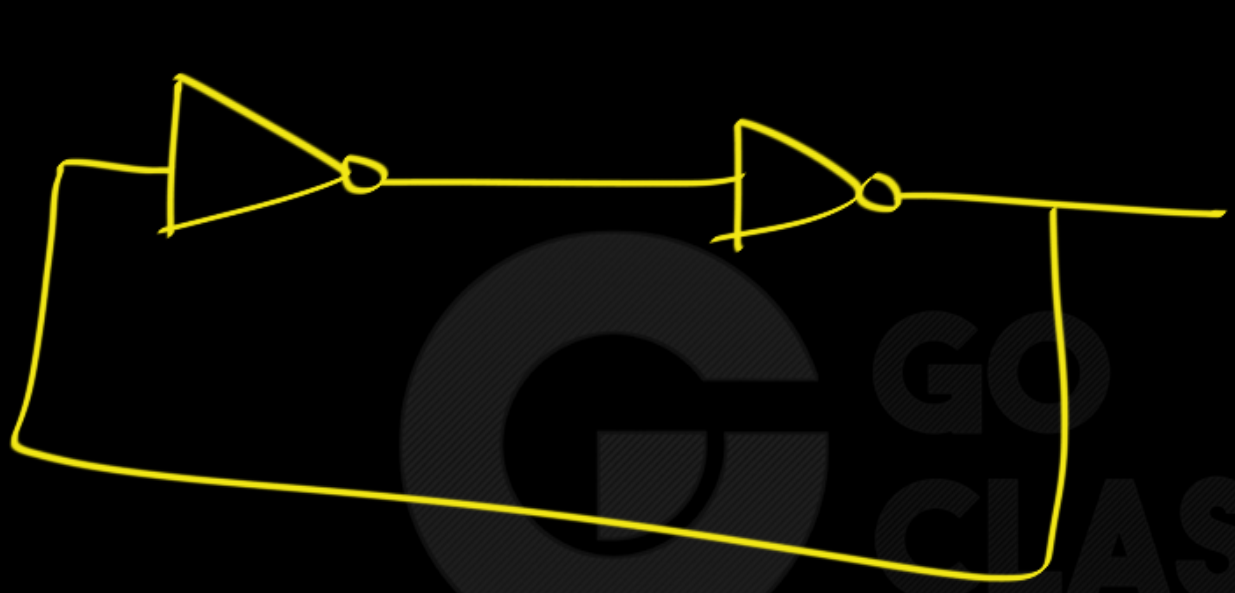
$\Downarrow$

$\Rightarrow$ o/p $\Rightarrow$ stable Q_A Q_B : 0 0

Problem:

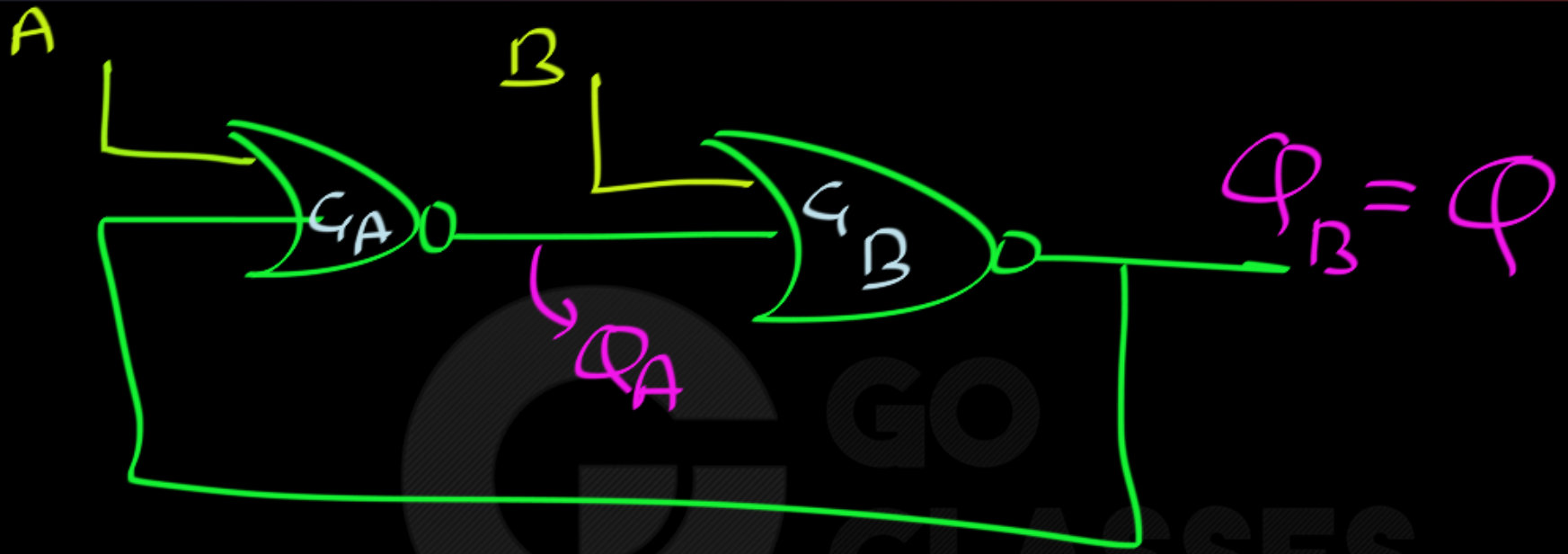
$A=1, B=1 \longrightarrow A=0, B=0$

Uncertain o/p



No User Control

→ One bit Storage Device


A
B
G_A
G_B
Q_A
$Q_B = Q$

A	B	Q_n → next o/p
0	1	0
1	0	1
0	0	Q (current o/p)
1	1	0 (forbidden)

Retain / unchanged

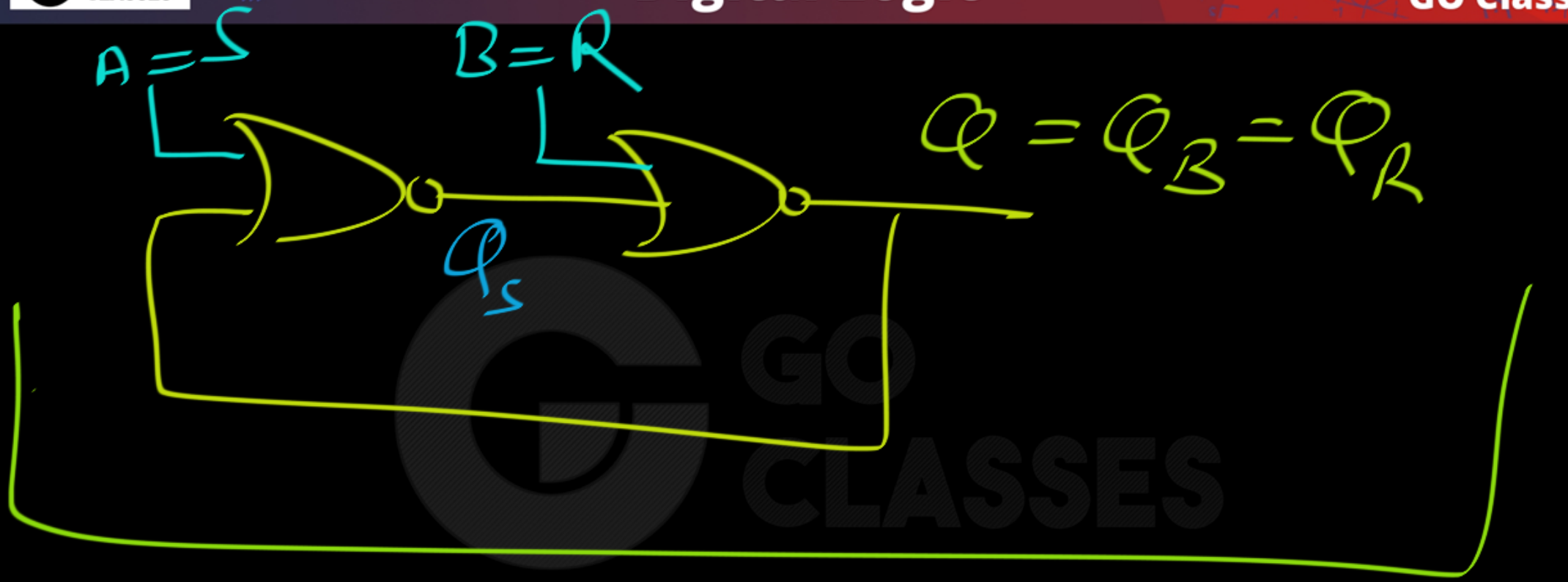
In Digital Ckts:

- making $o/p = 1$ — Set (inp)
- making $o/p = 0$ — Reset (inp)

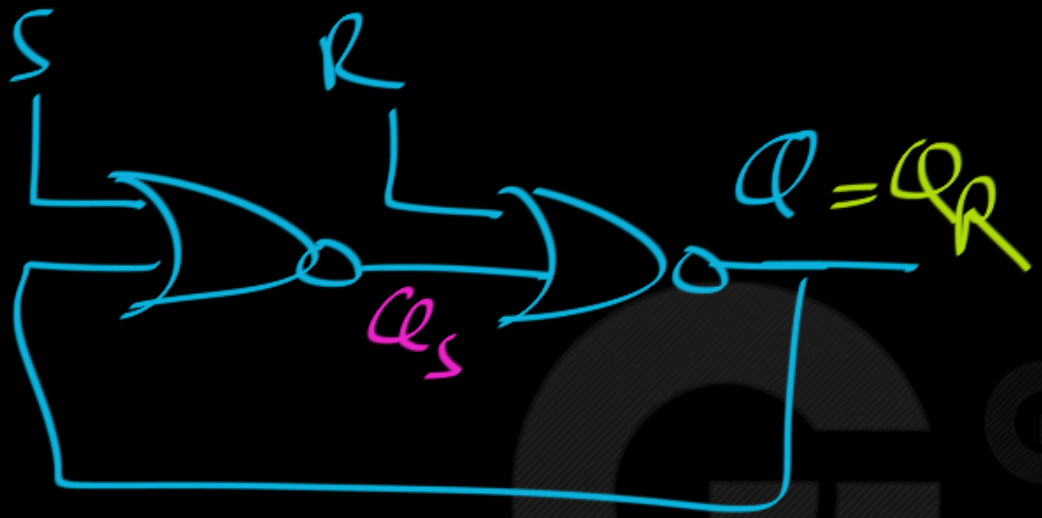
A=S	B=R	Q_n → next o/p	
0	1	0	(Reset)
1	0	1	(set)
0	0	Q	Retain / unchanged
1	1	0	(forbidden)

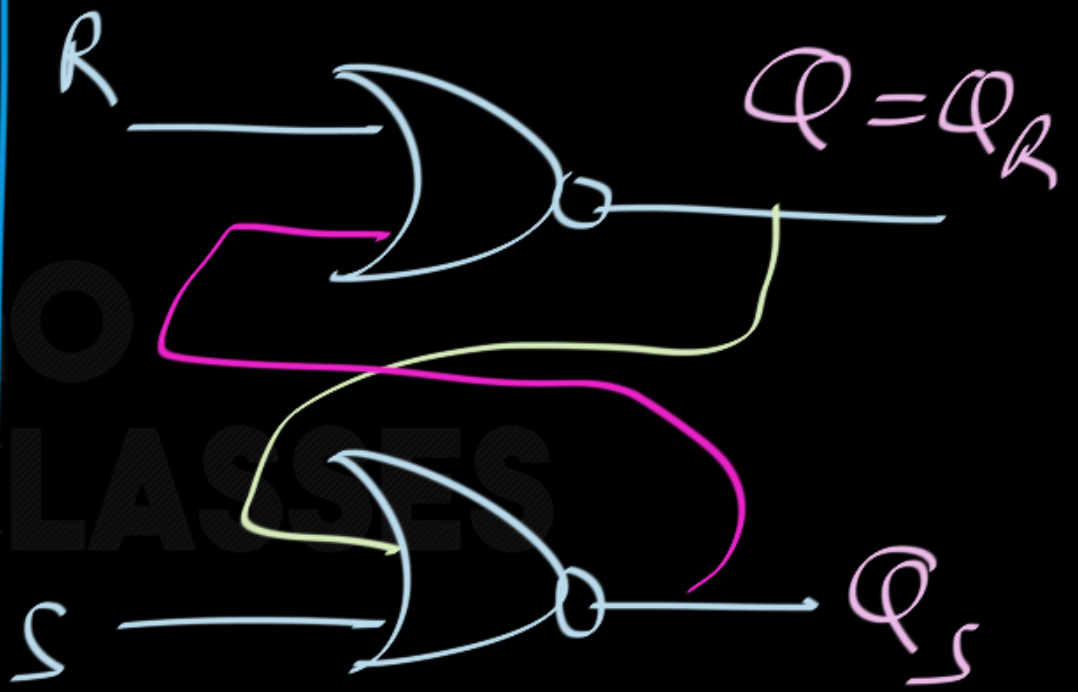
Current o/p

SR latch



SR latch (Set-Reset latch)

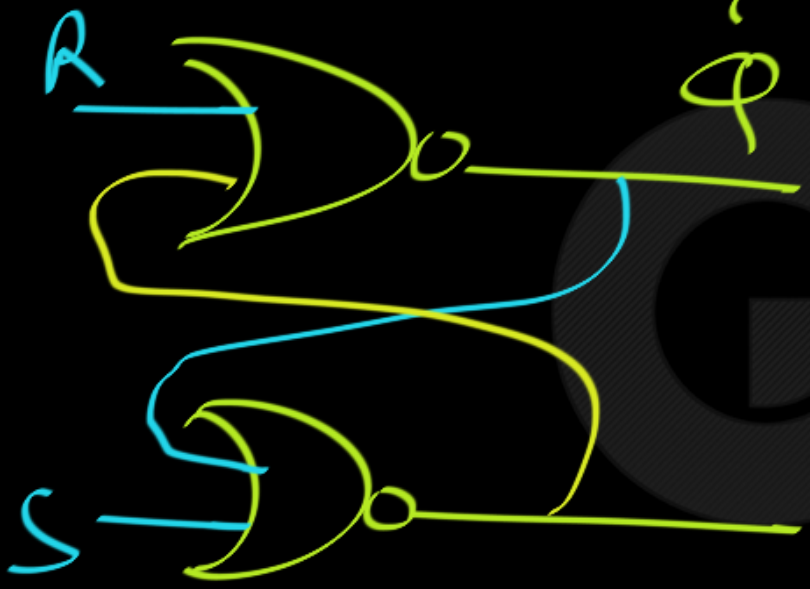




SR latch

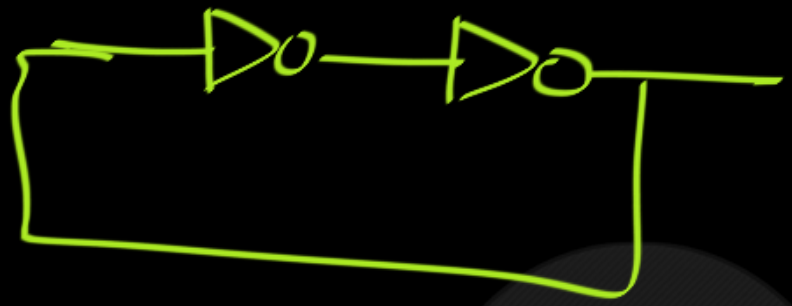
SR latch:

main o/p

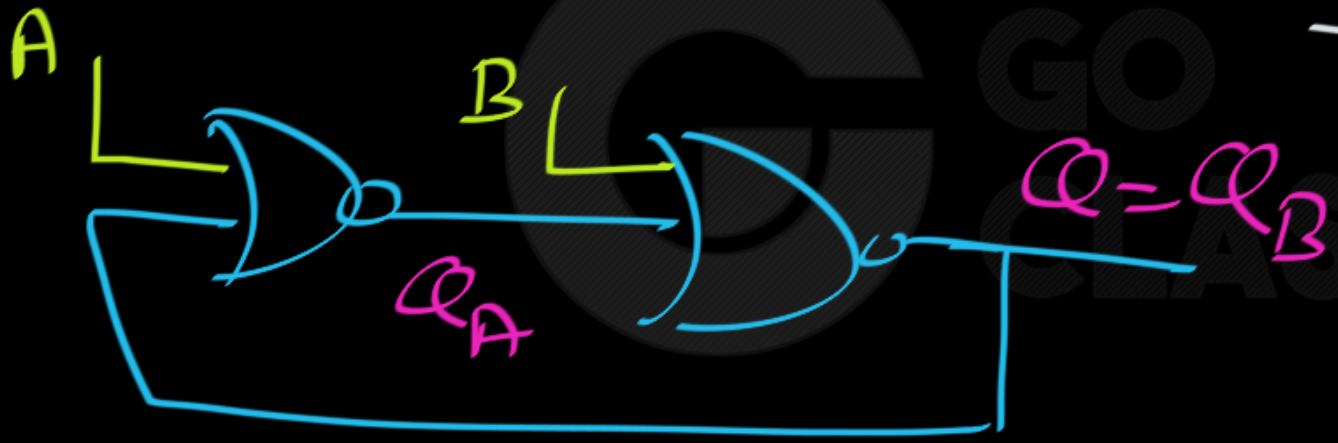


S	R	Q_n	time
1	0	1	0
0	0	Q=1	1
0	1	0	2
0	0	Q=0	3
1	1	0	4
0	0	UnDefined	5

Problem



No User Control



Analysis

$A=0, B=1$ R Reset

$A=1, B=0$ Set

$A=0, B=0$ Retain

S

SR latch:

Set ($Q_p = 1$)

Reset ($Q_p = 0$)

S	R	Q_{t+1} → Next o/p
0	1	0
1	0	1
0	0	Q_t
1	1	0 But is forbidden

Q:

Why $S=1, R=1$ is forbidden in SR latch ?

(A) Uncertain o/p

(2) Unstable o/p

(3) None of these

Q:

Why S = 1, R = 1 is forbidden in SR latch ?

output = 0

Stable

(A) Uncertain o/p

(2) Unstable o/p

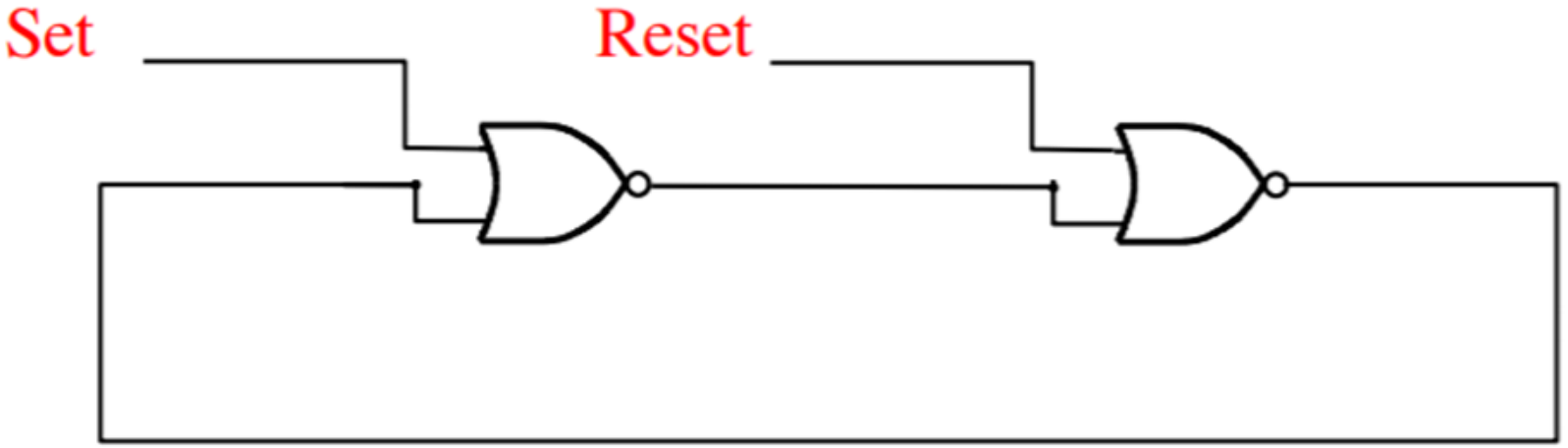
(3) None of these

Why $S=1, R=1$ is forbidden in SR latch ?

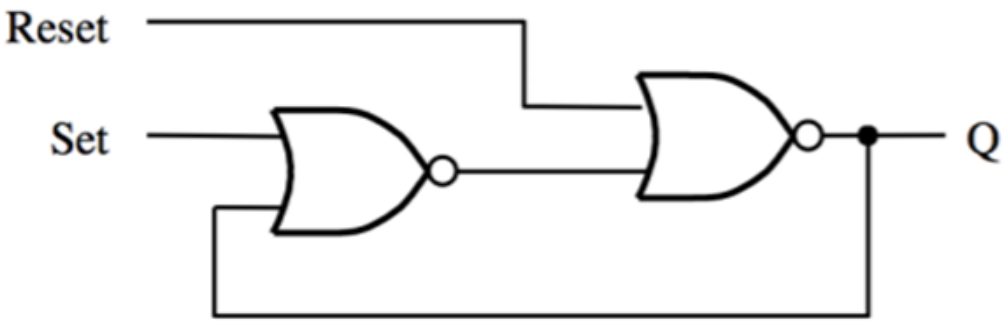
Reason: After $S=1, R=1$

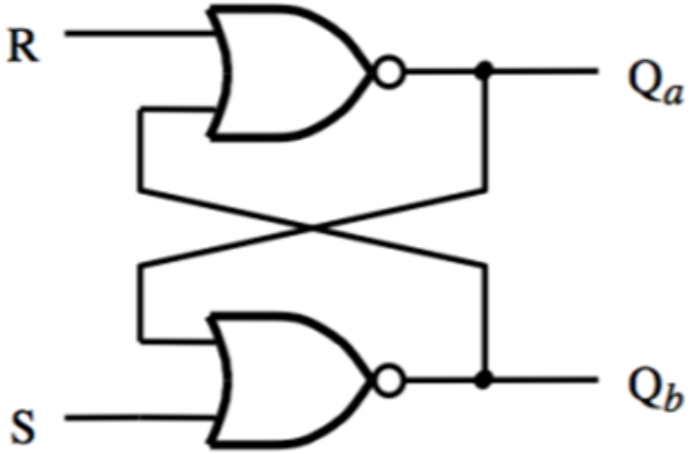
If we make $S=0, R=0 \Rightarrow$ then uncertain o/p

A simple memory element with NOR Gates



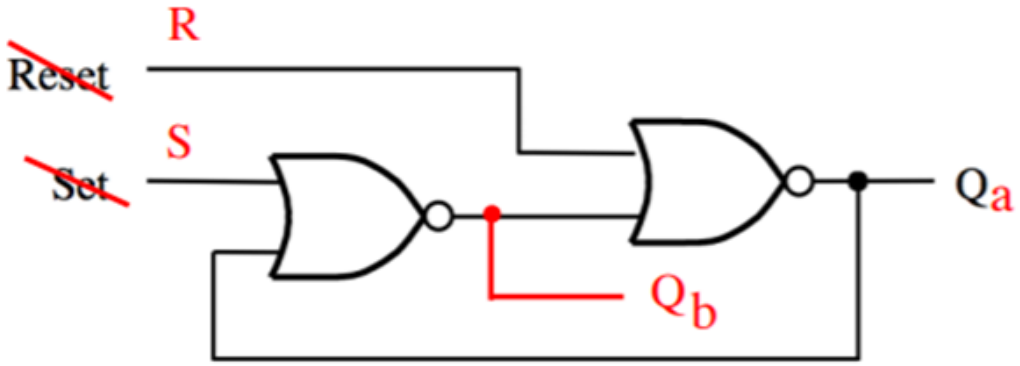
Two Different Ways to Draw the Same Circuit

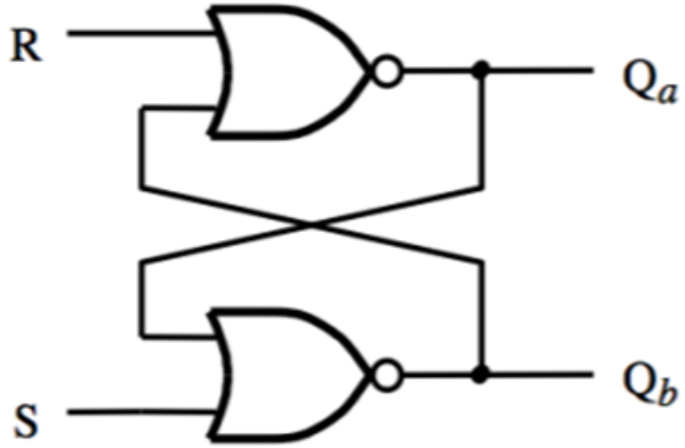




Two Different Ways to Draw the Same Circuit

SR latch

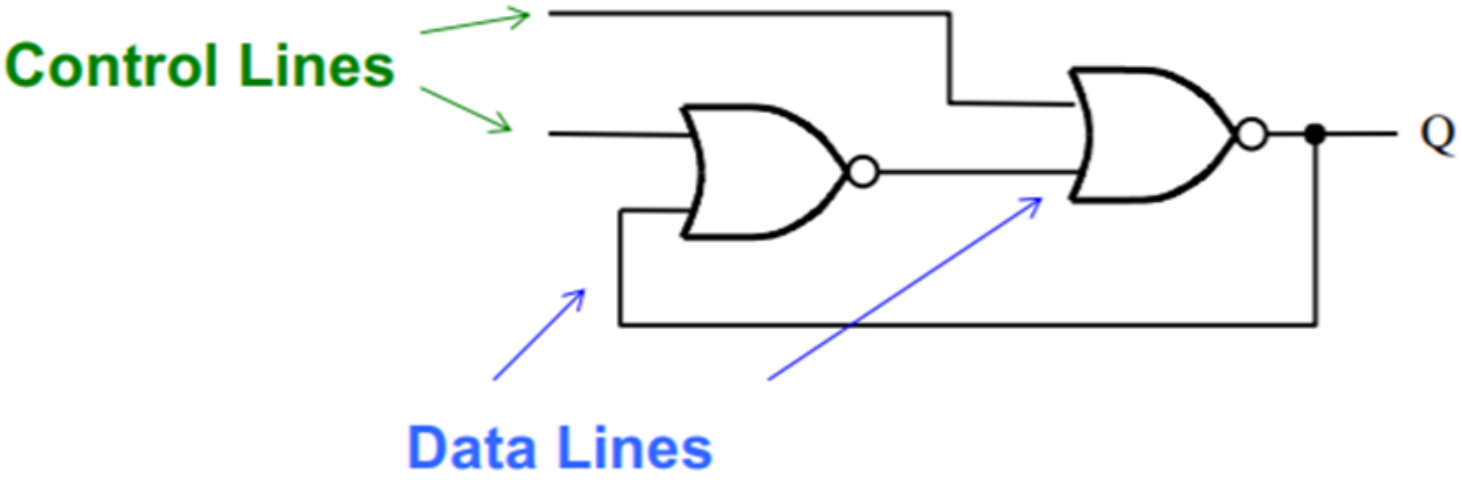




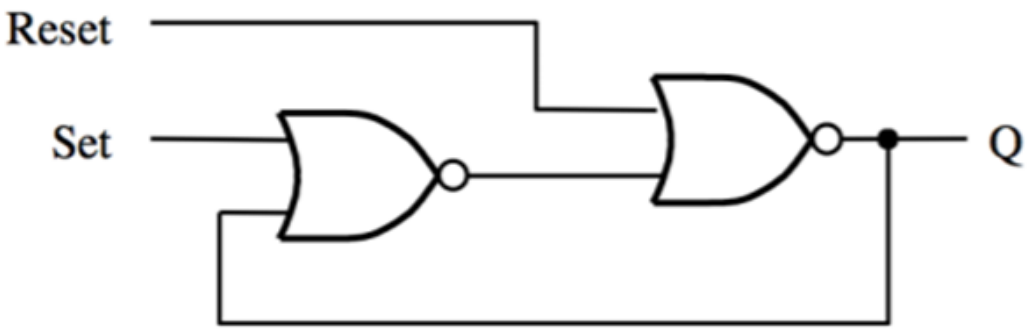
What is a latch?

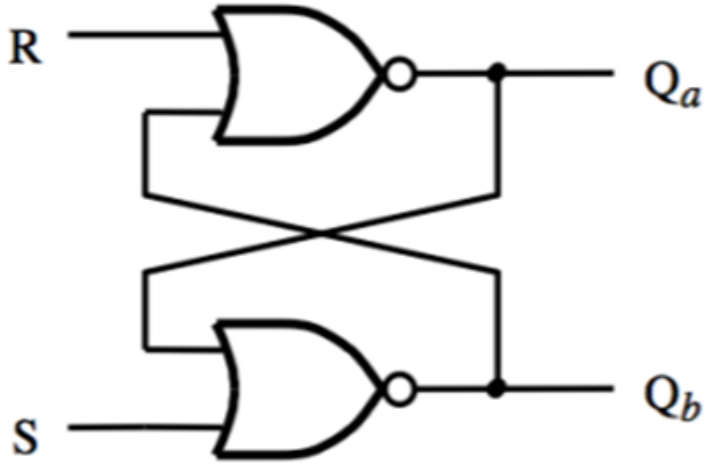
Back to the Basic Latch

The Basic Latch

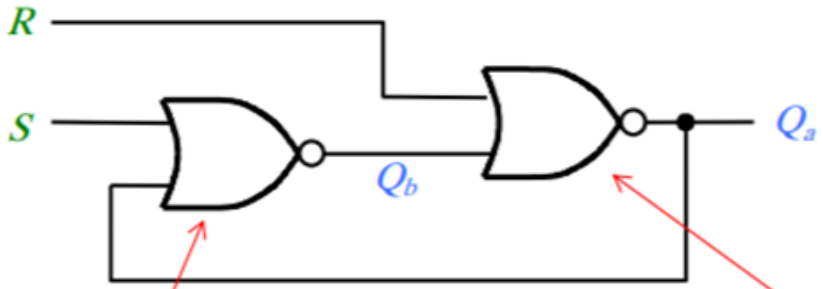


Two Different Ways to Draw the Same Circuit





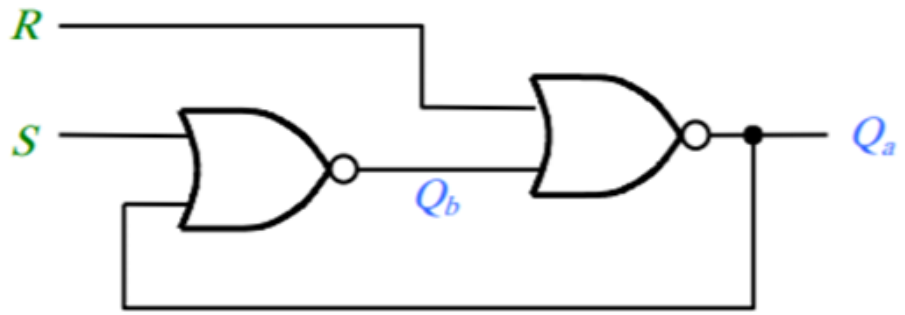
Analyzing The Basic Latch



S	Q_a	$Q_b = \text{NOR}(S, Q_a)$	
0	0	1	$\overline{Q_a}$
0	1	0	
1	0	0	0
1	1	0	

R	Q_b	$Q_a = \text{NOR}(R, Q_b)$	
0	0	1	$\overline{Q_b}$
0	1	0	
1	0	0	0
1	1	0	

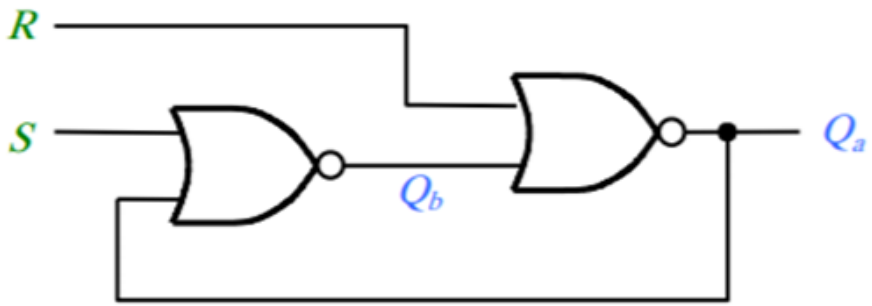
Analyzing The Basic Latch



S	Q_b
0	$\overline{Q}_a$
1	0

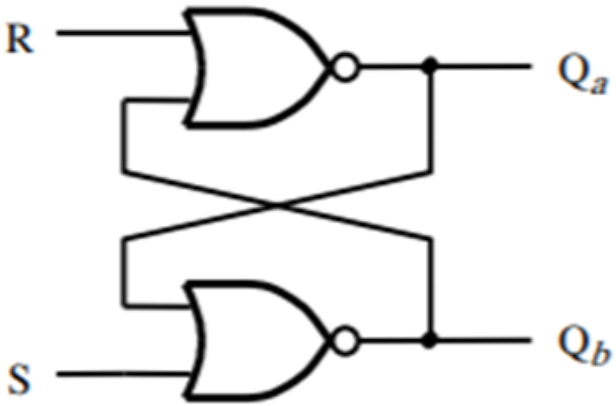
R	Q_a
0	$\overline{Q}_b$
1	0

Behavior of the Basic Latch



S	R	$Q_a(t+1)$	$Q_b(t+1)$	
0	0	$Q_a(t)$	$Q_b(t)$	Latch
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Undesirable

Circuit and Characteristic Table



(a) Circuit

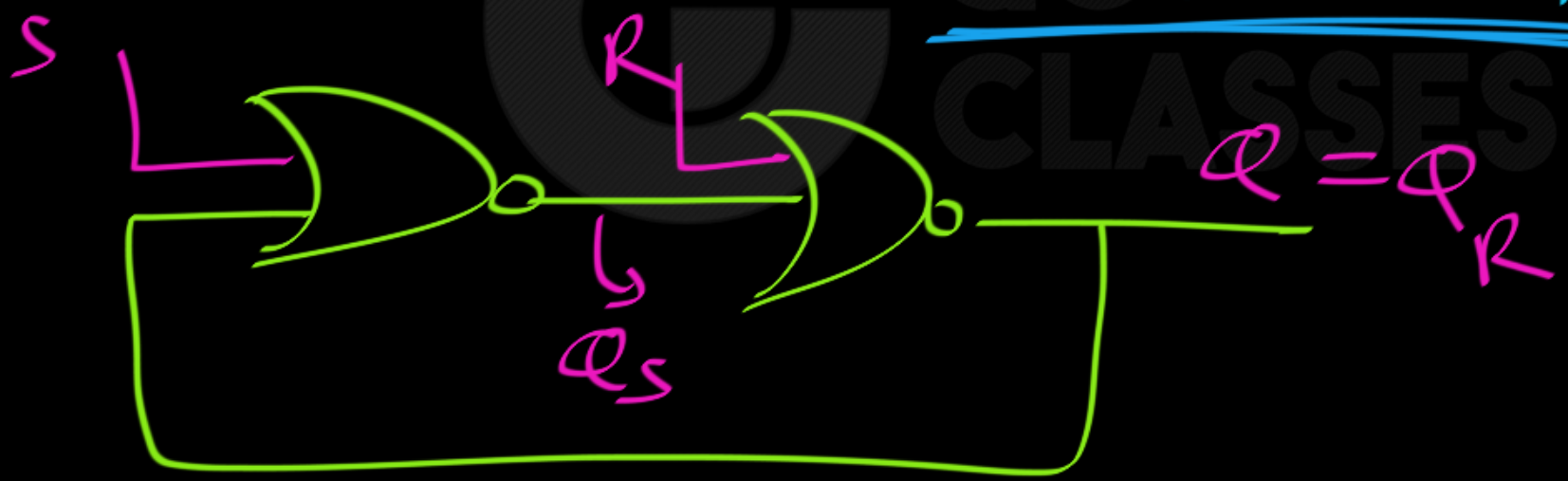
S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Characteristic table

Note that Q_a and Q_b are inverses of each other!

Notes: ① for all Allowed Input Comb-ination

$Q_S \neq Q_R$; $Q_S = \overline{Q_R}$



S	R	Q_S	$Q_R = \overline{Q}$
1	1	0	0

invalid i/p comb. (forbidden)

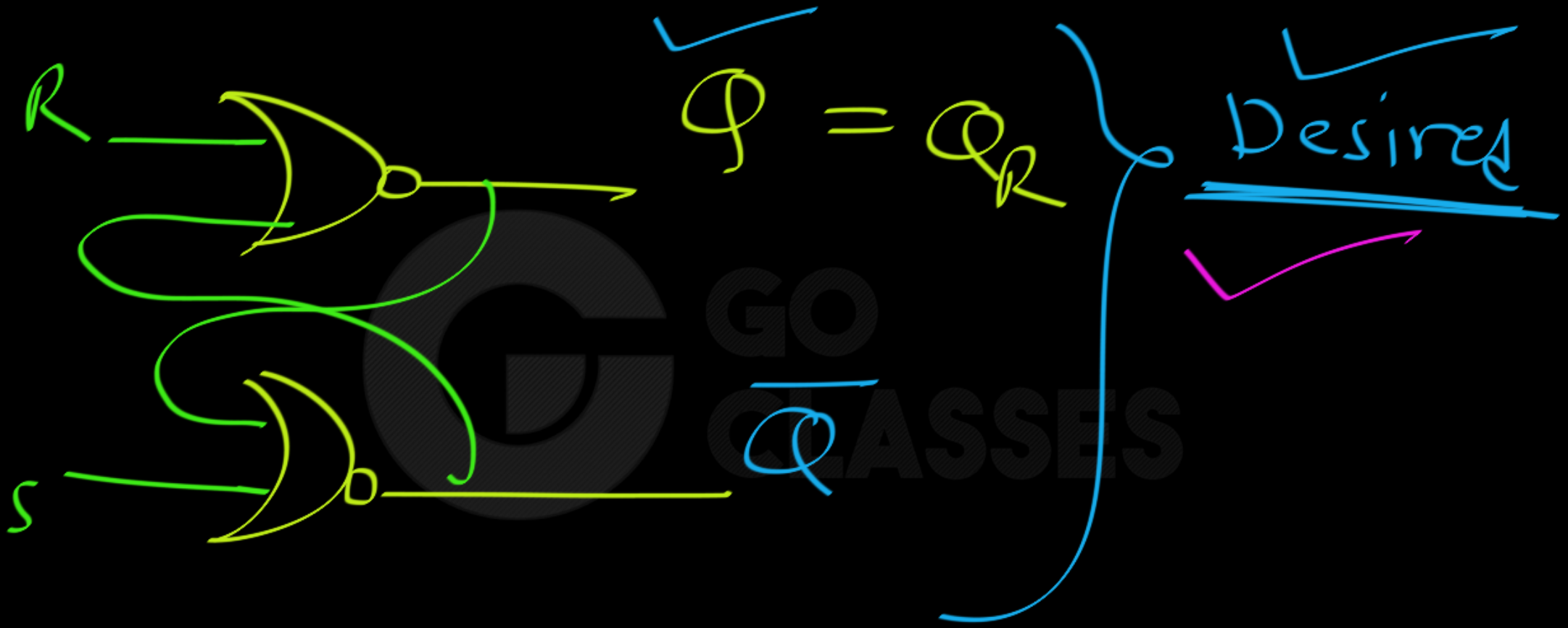
0	1	1	0
1	0	0	1
0	0	$\overline{Q_t}$	Q_t

$$Q_S = \overline{Q_R}$$

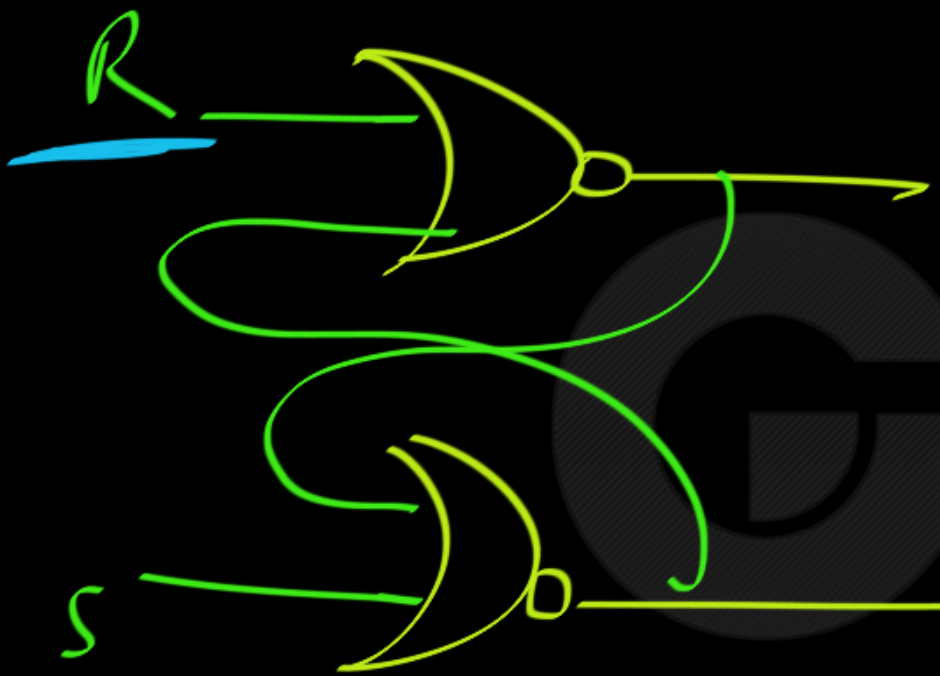
Note: If $Q_S = Q_R$ = 1

If we try to retain

Uncertain behaviour


R
S
Q = QR
Q
Desired

(2)



$Q = Q_R$

$= 0$

$\bar{Q} = 1$

(main) output

SR latch

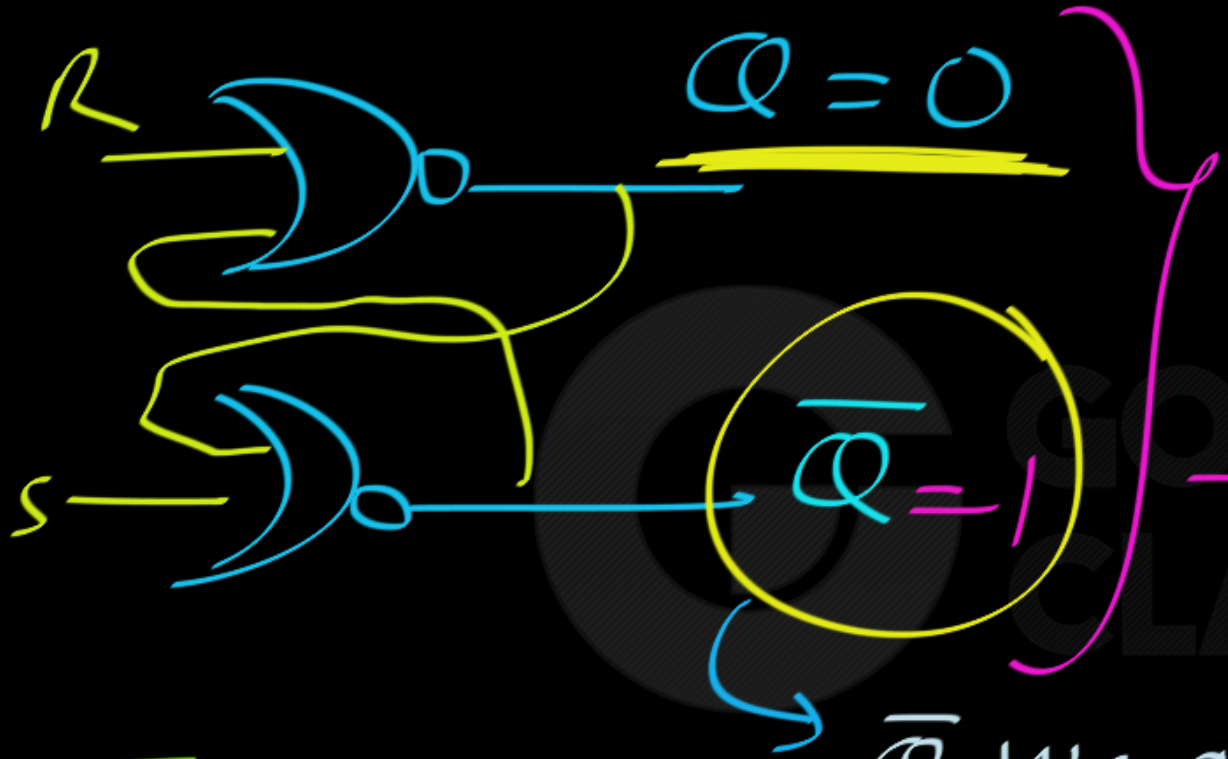
⇓

Storing 0

$Q = Q_R = 0$

State 0

Same



Storing 0 L

State = 0 L

o/p = 0

To get $\bar{Q}$; $\bar{Q}$ we are getting for free. we Don't need inverter.

R	S	$Q = Q_R$	Q_S
1	1	0	0
0	0	Uncertain / Undefined / Oscillating (when both NOR gate have same speed)	

If both NOR gate have same speed:

R	S	$Q_R = Q$	Q_S
1	1	0	0
0	0	~~1~~ ~~0~~	~~1~~ ~~0~~

oscillations

but in Reality ; Both NOR gate have Different speed :

$$\left\{ \begin{array}{l} Q_R = 0 \text{ , } Q_S = 1 \\ \text{OR} \\ Q_R = 1 \text{ , } Q_S = 0 \end{array} \right\} \text{ Uncertain}$$

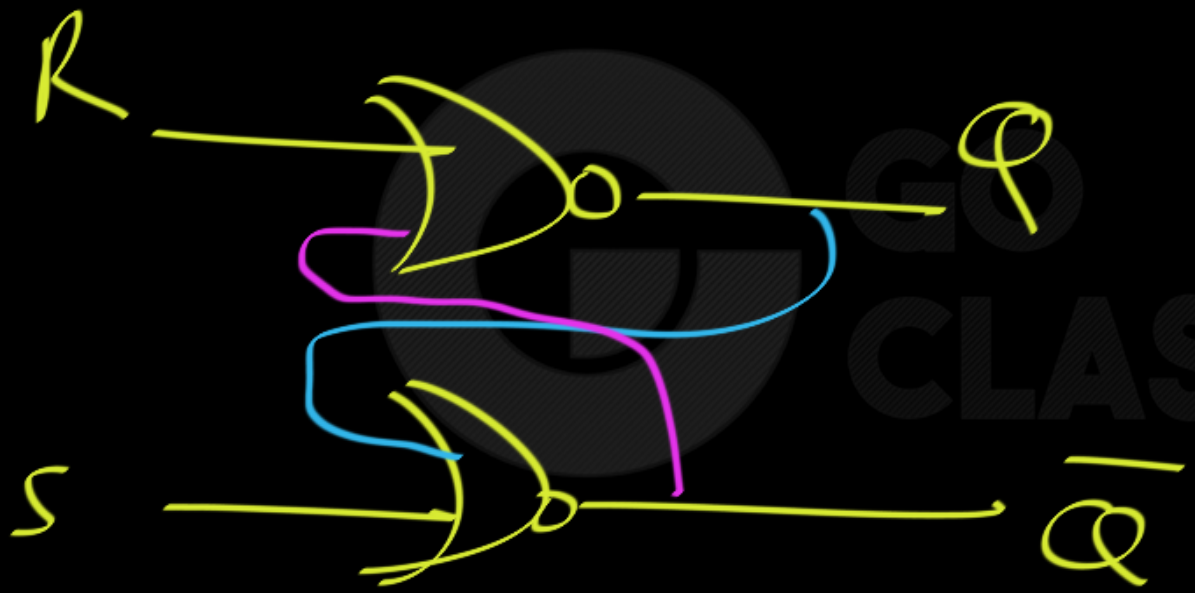
Oscillations and Undesirable States

- When S=1 and R=1 both outputs of the latch are equal to 0, i.e., Q_a=0 and Q_b=0.
- Thus, the two outputs are no longer complements of each other.
- This is undesirable as many of the circuits that we will build later with these latches rely on the assumption that the two outputs are always complements of each other.
- (This is obviously not the case for the basic latch, but we will patch it later to eliminate this problem).

Oscillations and Undesirable States

- An even bigger problem occurs when we transition from S=R=1 to S=R=0.
- When S=R=1 we have $Q_a=Q_b=0$. After the transition to S=R=0, however, we get $Q_a=Q_b=1$, which would immediately cause $Q_a=Q_b=0$, and so on.
- If the gate delays and the wire lengths are identical, then this oscillation will continue forever.
- In practice, the oscillation dies down and the output settles into either $Q_a=1$ and $Q_b=0$ or $Q_a=0$ and $Q_b=1$.
- The problem is that we can't predict which one of these two it will settle into.

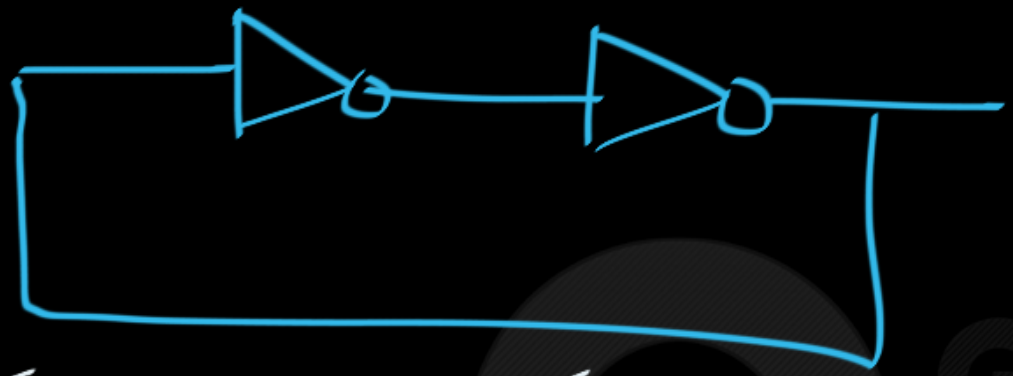
SR latch : ≡ SR latch using NOR



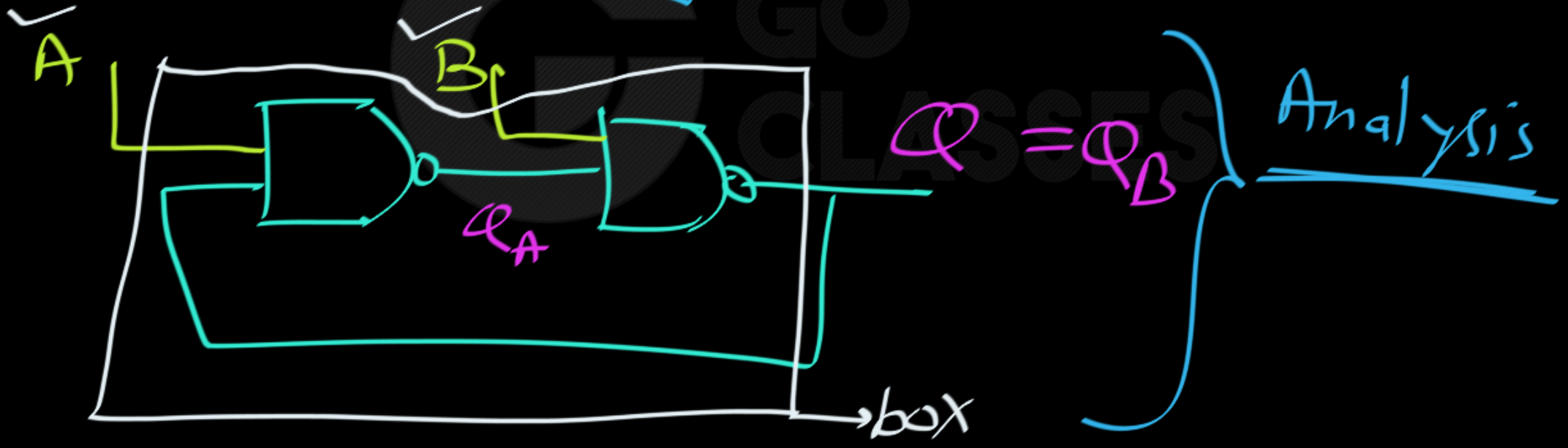
Next Topic:

SR Latch with NAND Gates

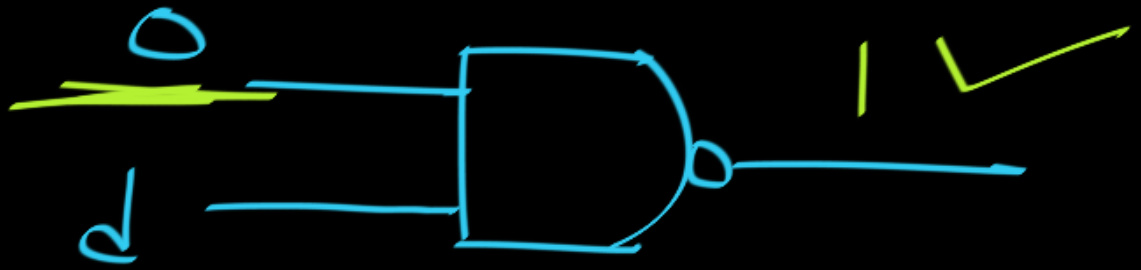
$\overline{S}\overline{R}$ latch

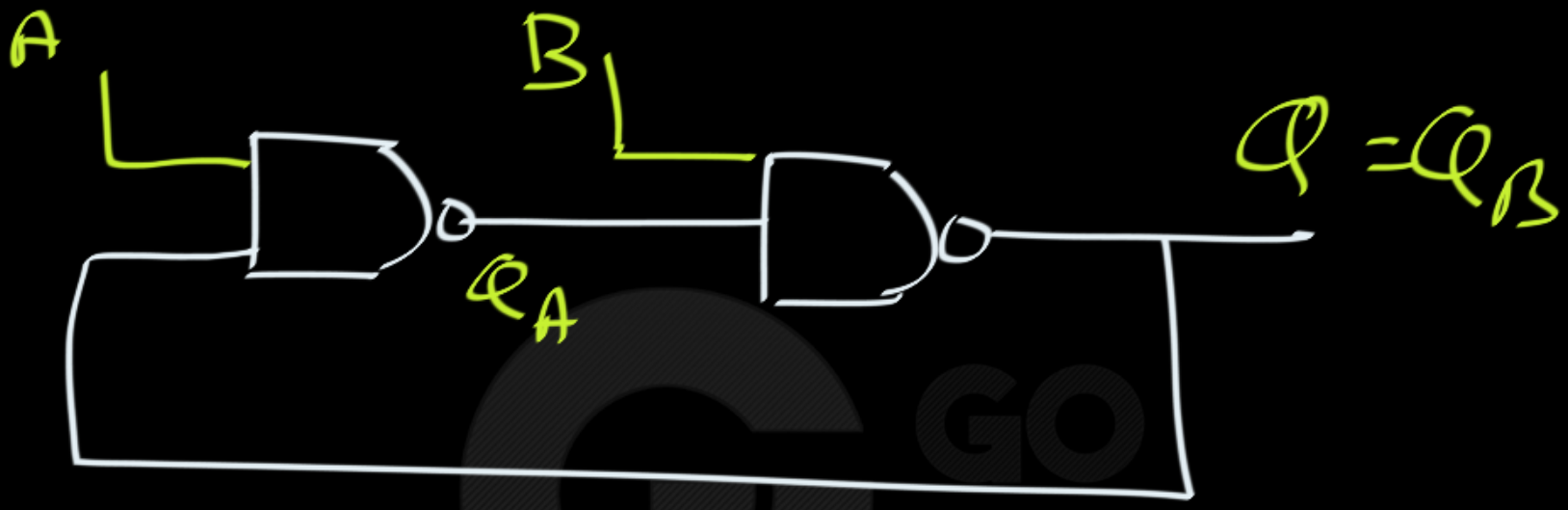


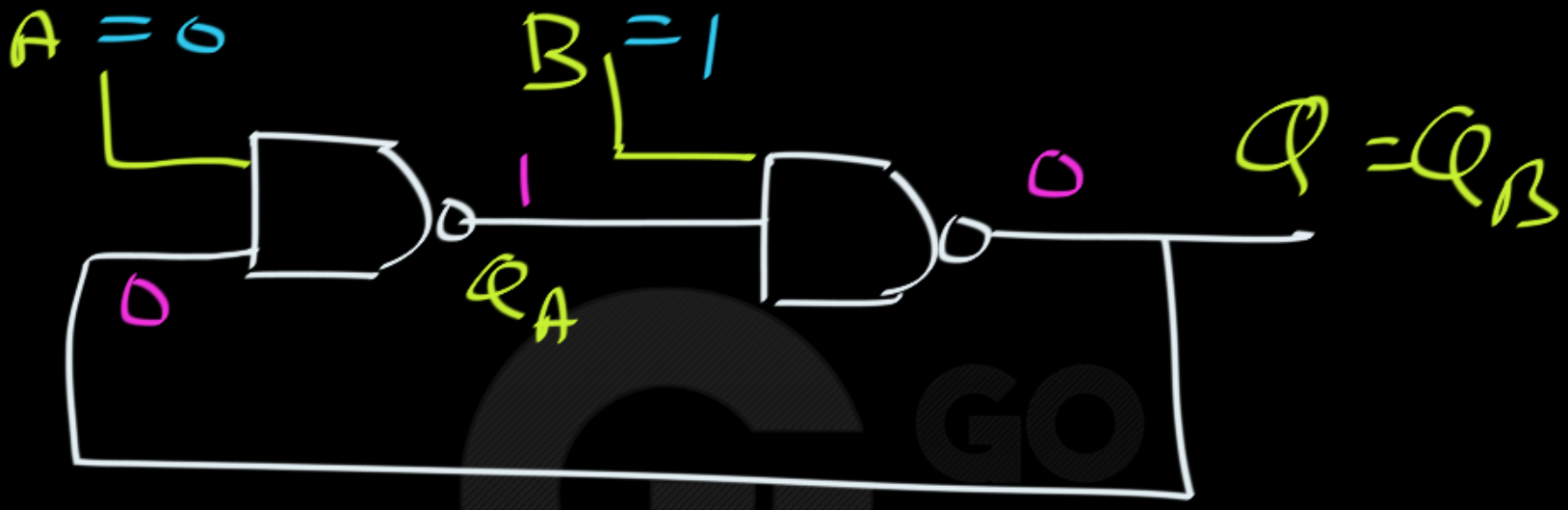
No user control



Analysis

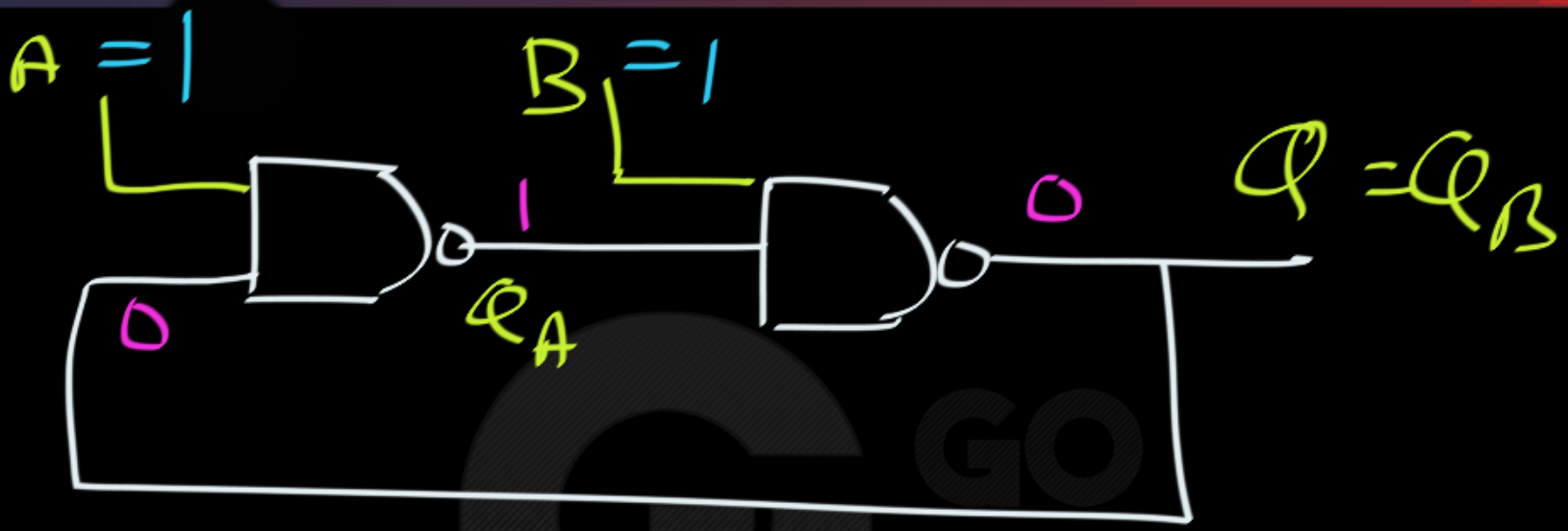

0
1


A
B
Q_A
$Q = Q_B$



Storing 0 (Reset)

A	B	Q_{t+1}
0	1	0

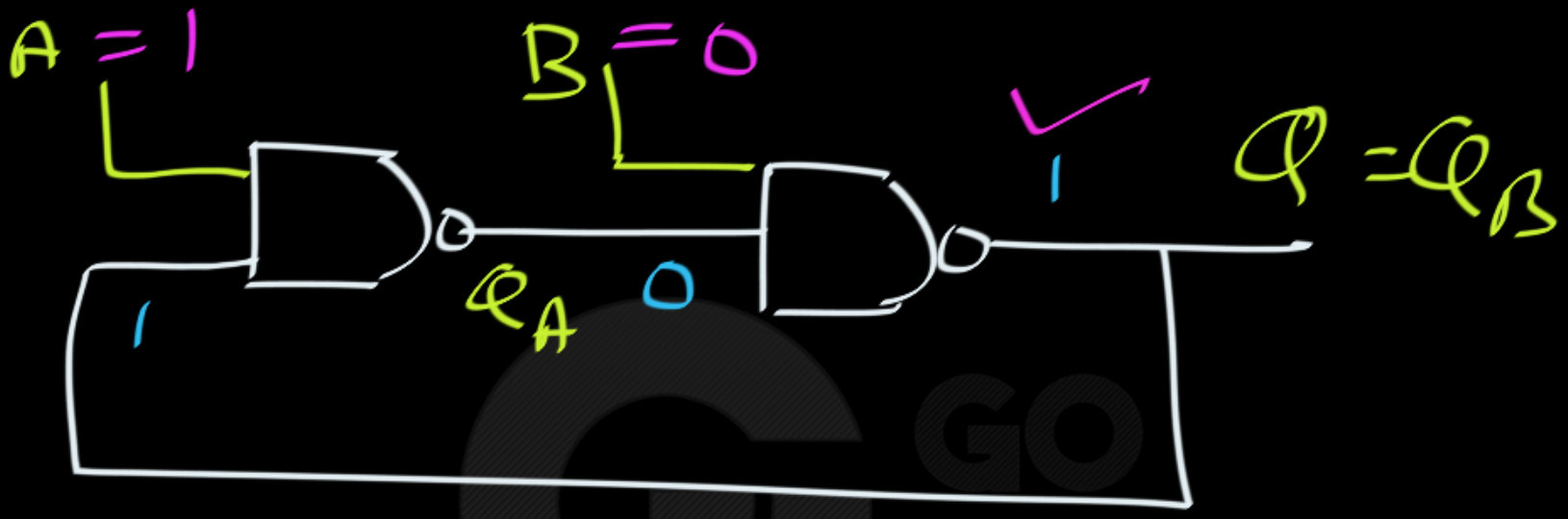


Retain

$A = 1, B = 1$

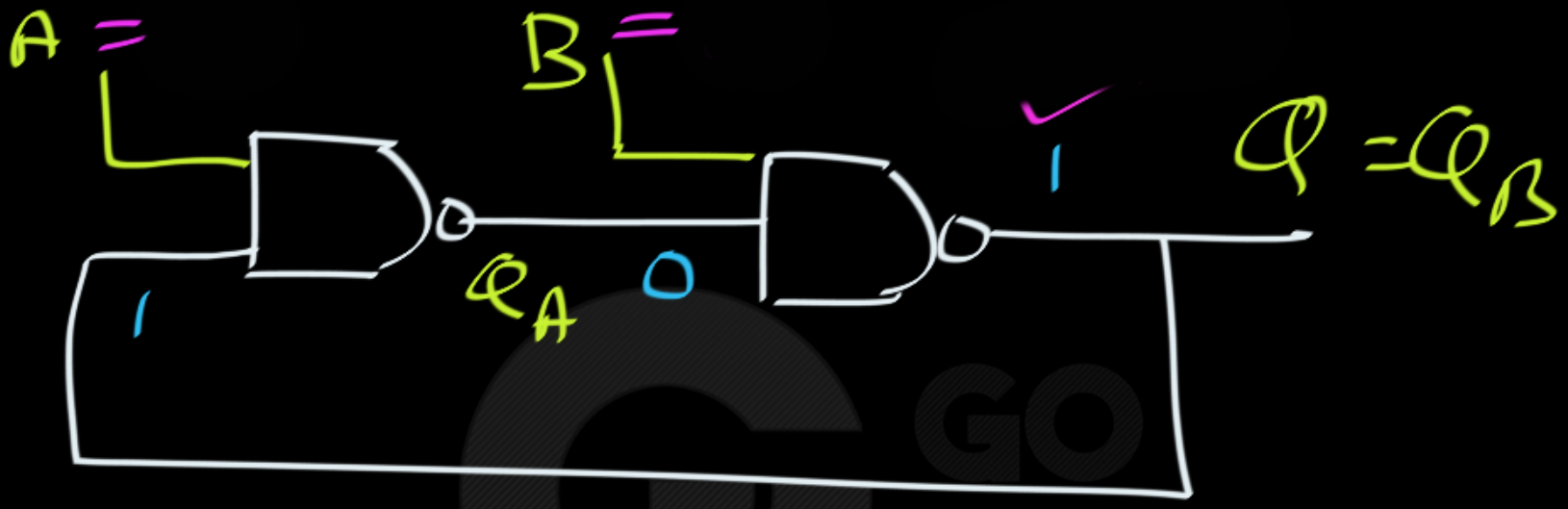
Retain

$Q_{t+1} = Q_t$



Storing 1 (Set)

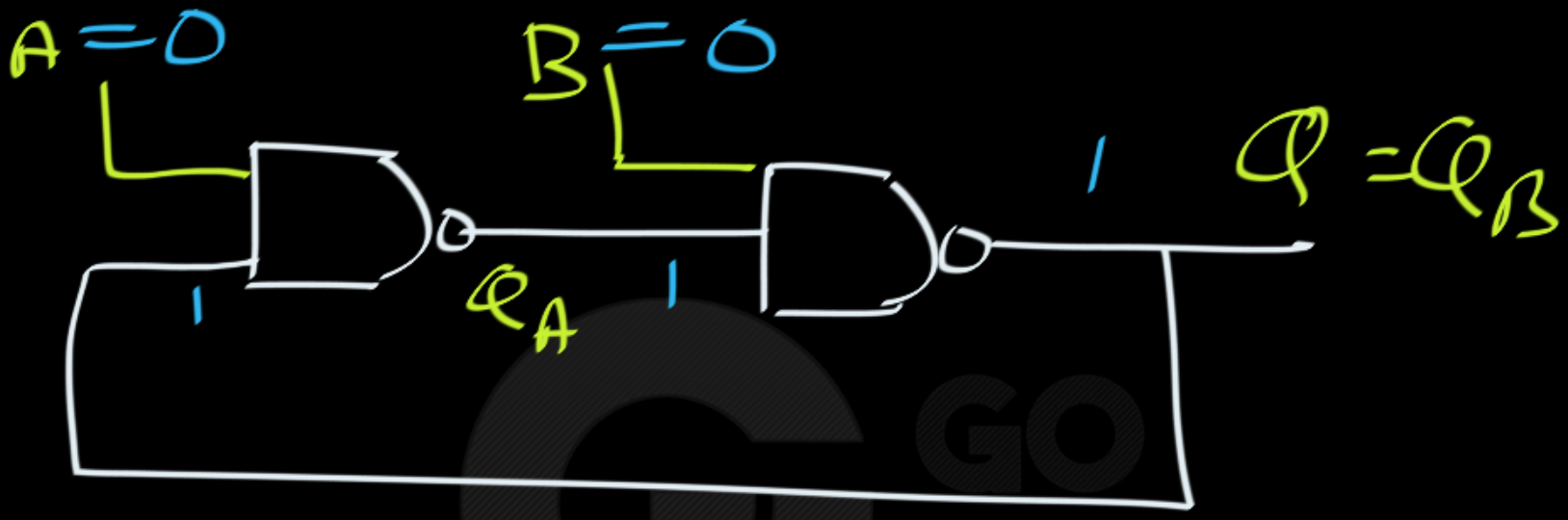
A	B	Q_{t+1}
1	0	1



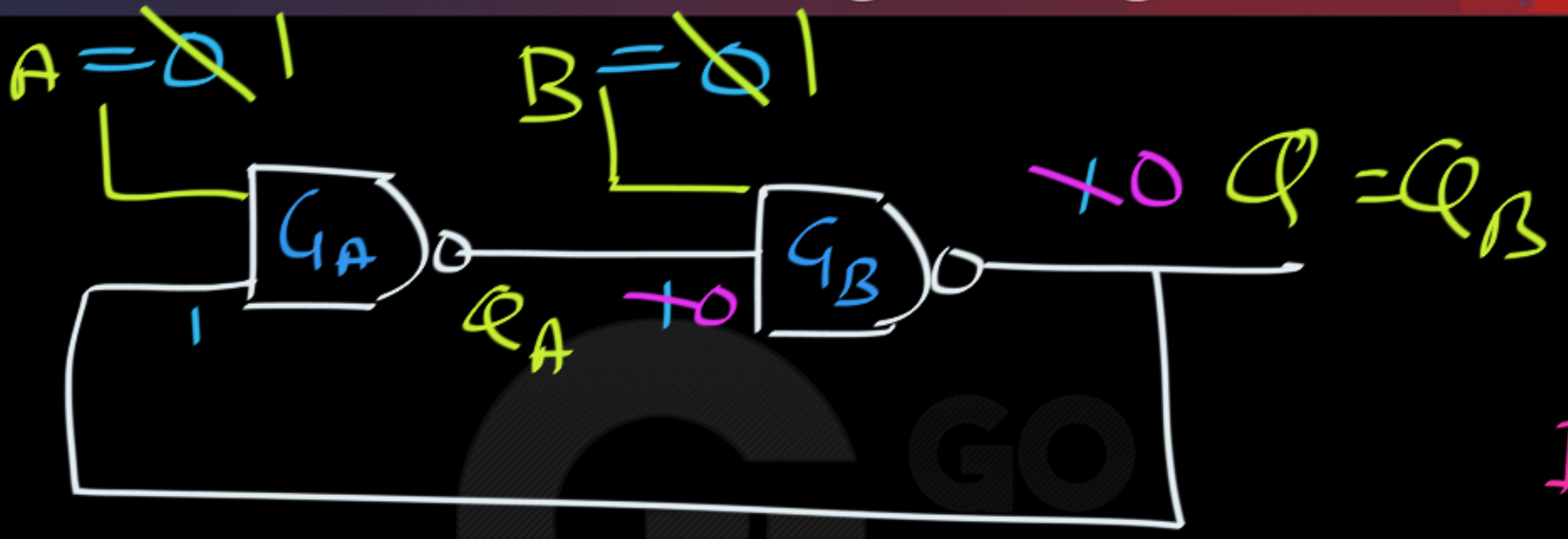
$A=1, B=1 \Rightarrow$ Retain

$Q_{t+1} = Q$

A	B	$Q = Q_B$	Q_A	
0	1	0	1	Reset
1	0	1	0	Set
1	1	Retain	Retain	Retain
0	0	?	?	



A	B	Q_A	Q_B
0	0	1	1



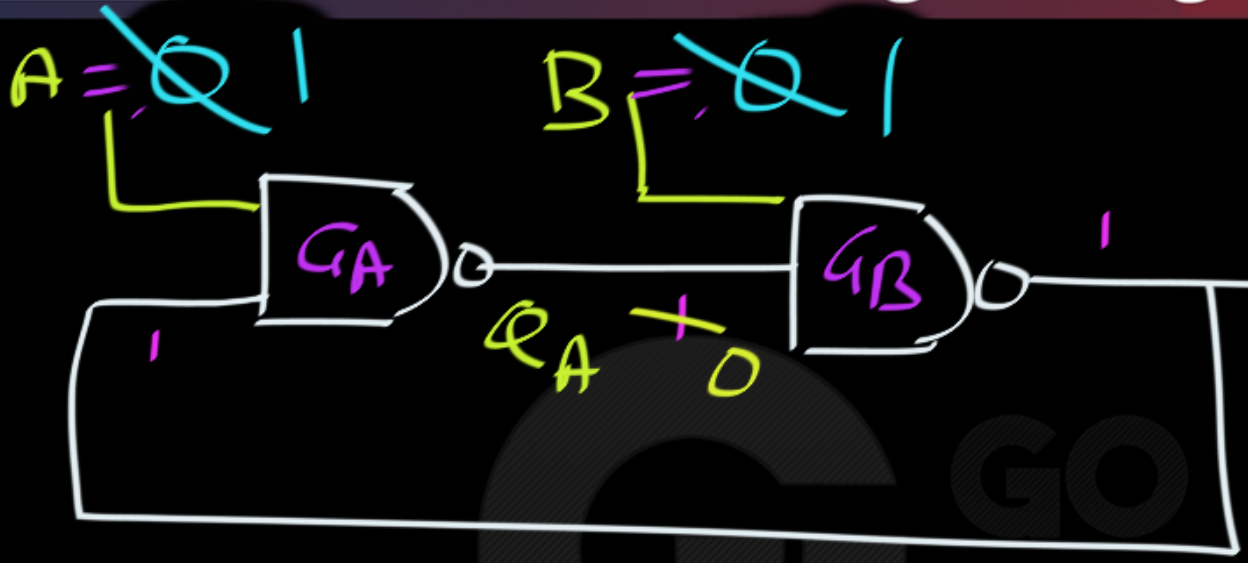
A	B	Q_A	Q_B
0	0	1	1
1	1	$\bar{Q}_A$ → 0	$\bar{Q}_B$ → 0

If $G_A = G_B$

Speed

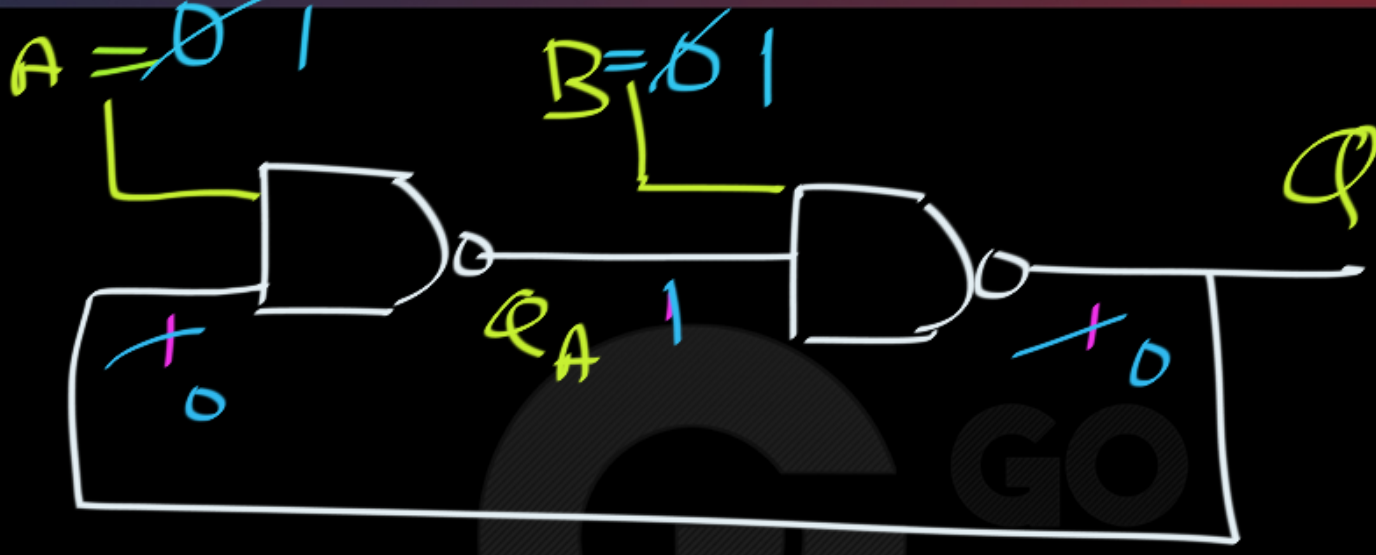
↓

oscillation



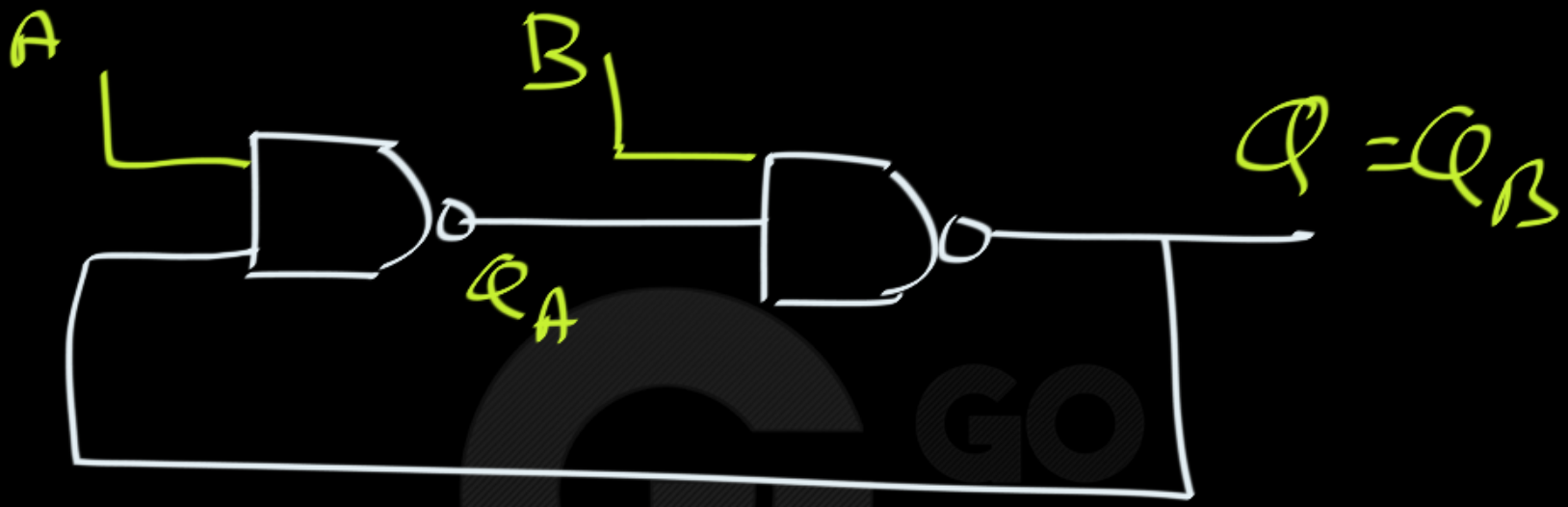
If $Q_A > Q_B$ Speed

A	B	Q_A	Q_B
0	0	1	1
1	1	0	1



A	B	Q_A	Q_B
0	0	1	1
1	1	1	0

If $Q_B > Q_A$
Speed



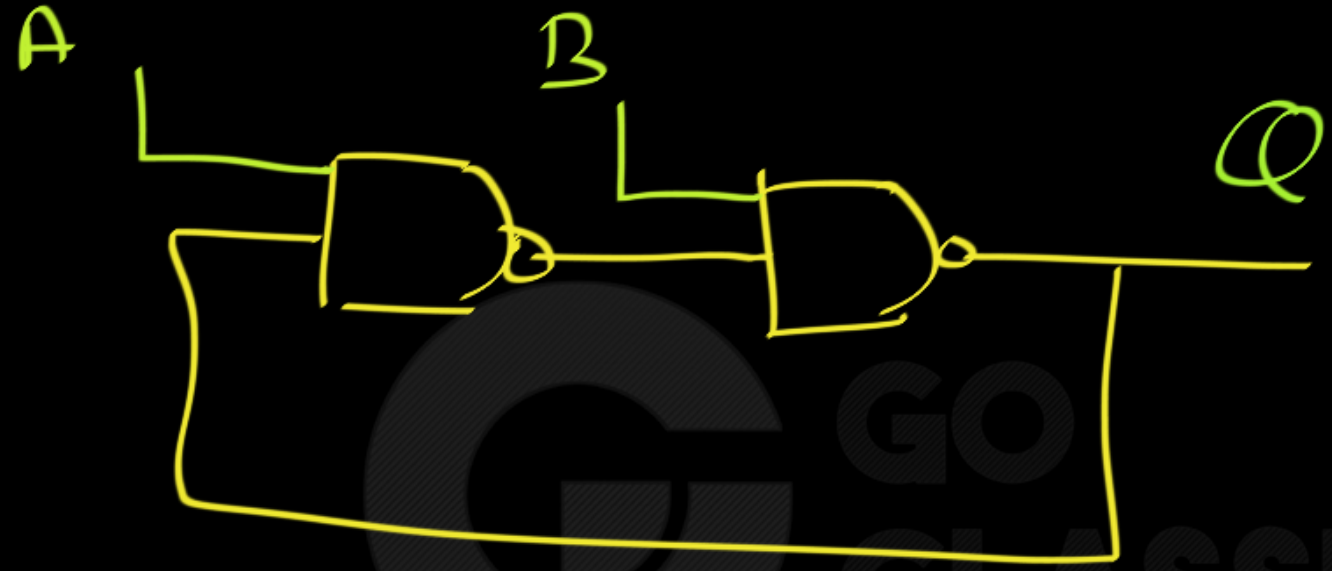
Problem:

A	B
0	0

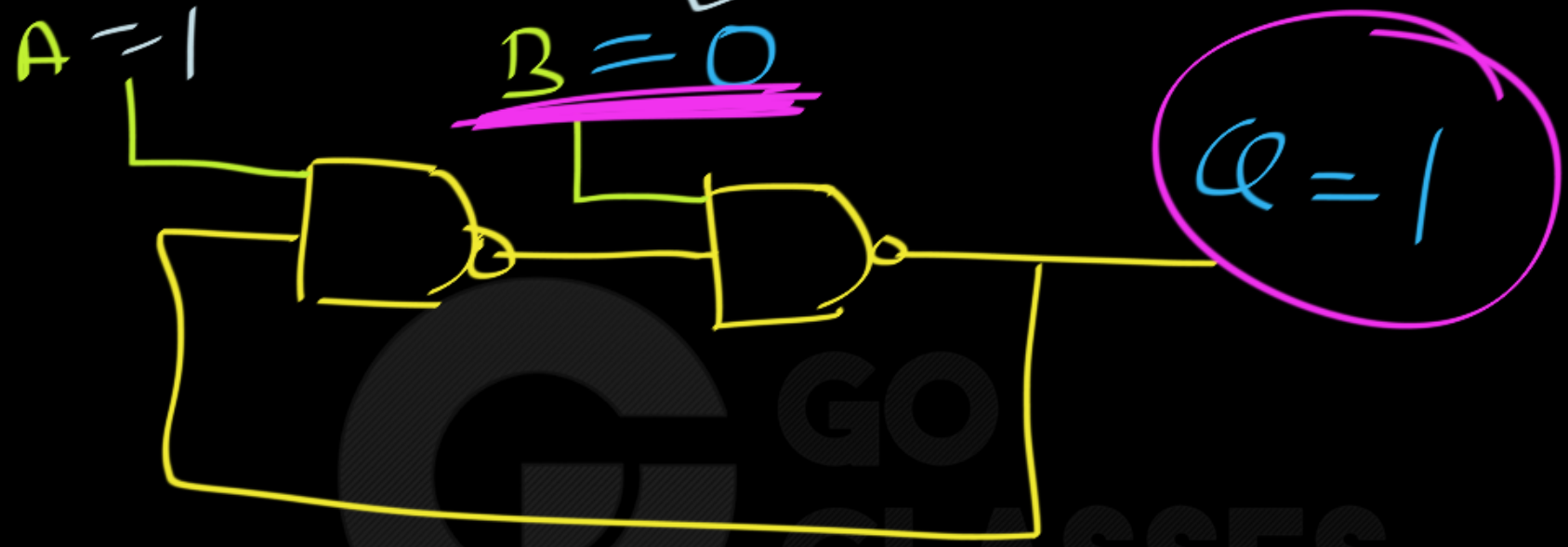
⟹

A	B
1	1

then Uncertain behaviour



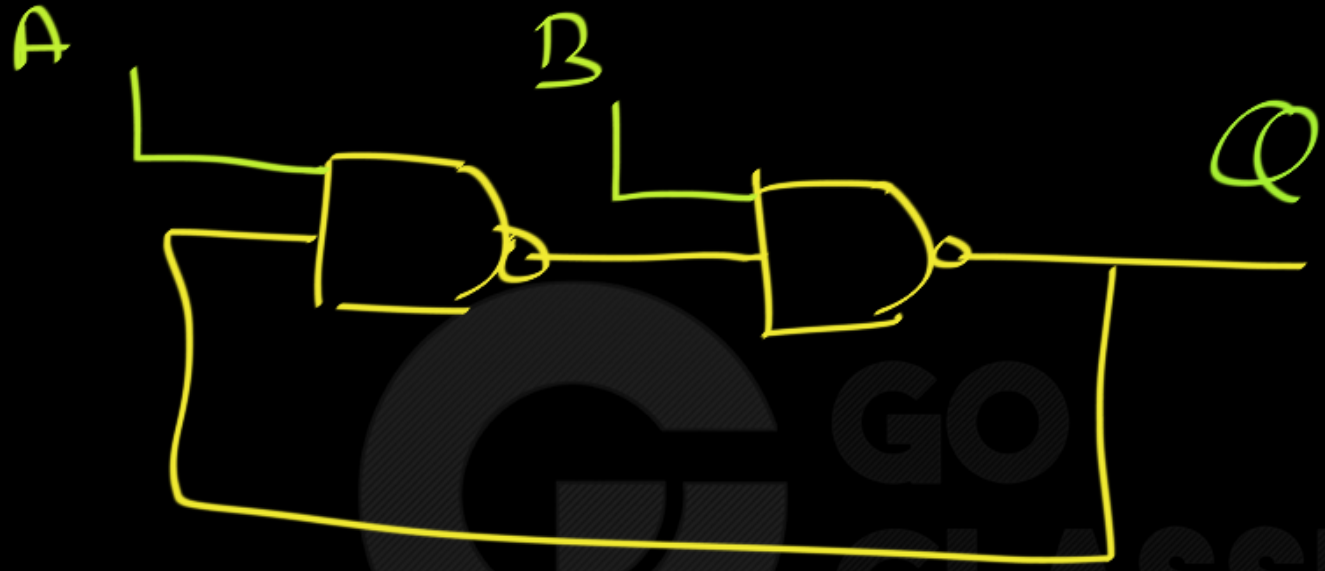
Q: Set it ? (Q=1)



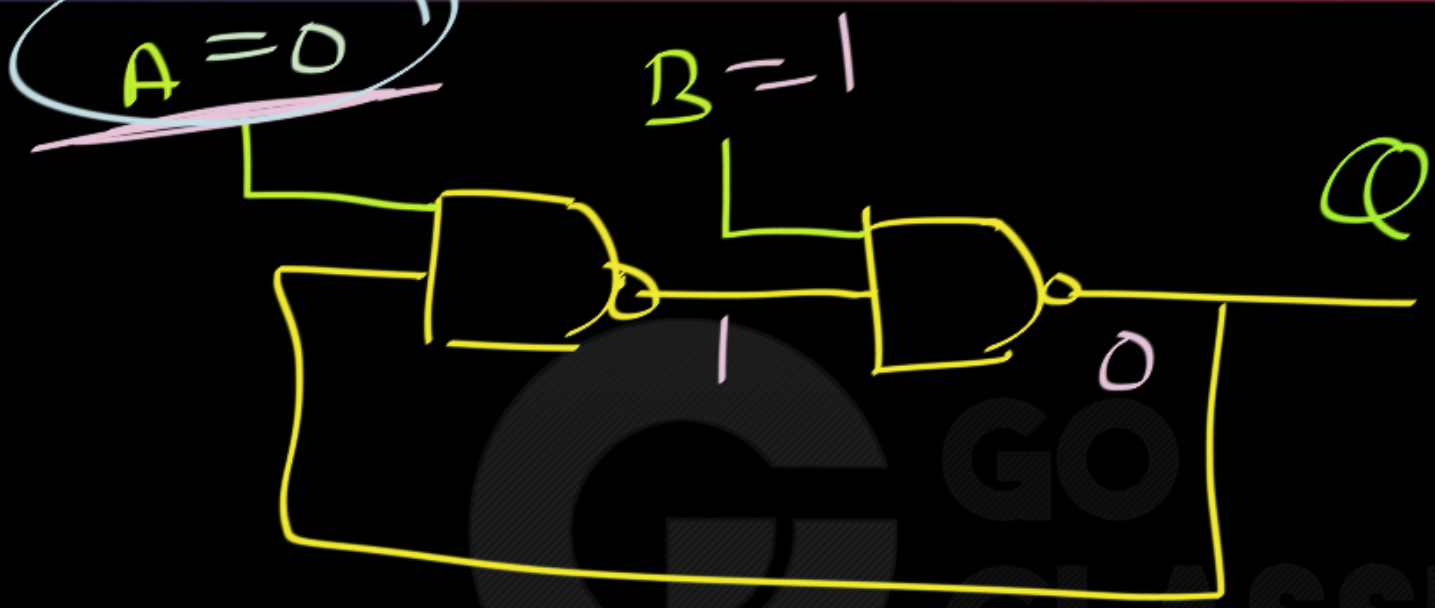
Q: Set it ?

B ⇒ working for Set (time)

But Active low wise

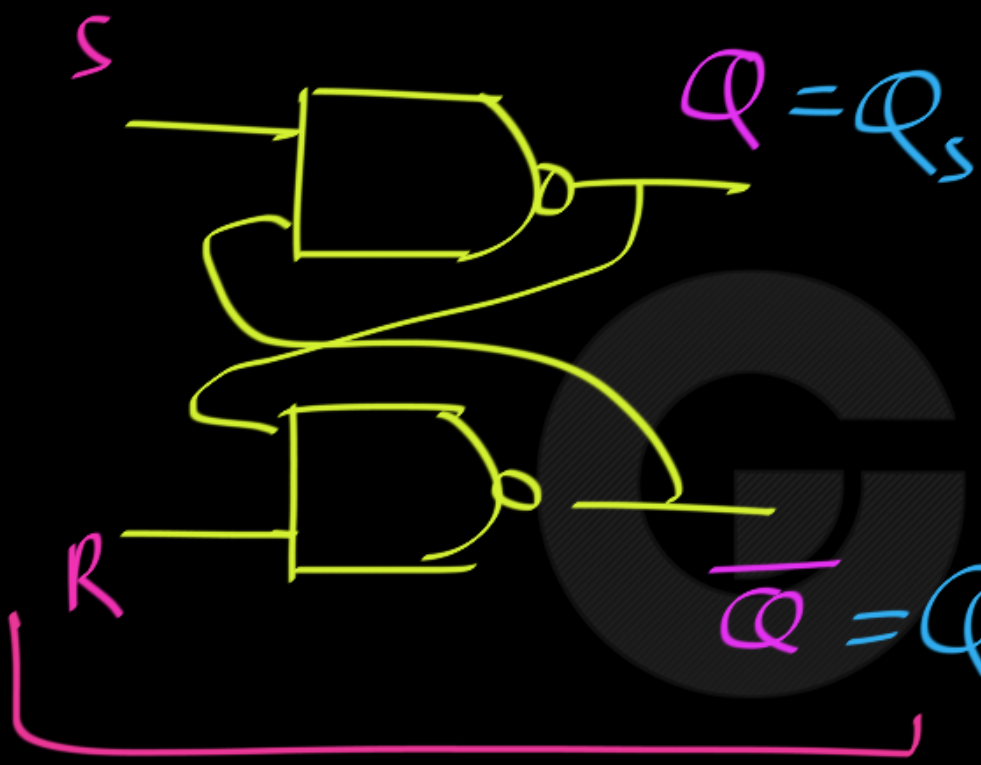


Q: Reset it ? (Q=0)



Q: Reset it ? (Q = 0)

(A is working as Reset but Active low wise)



$\bar{S}\bar{R}$ latch

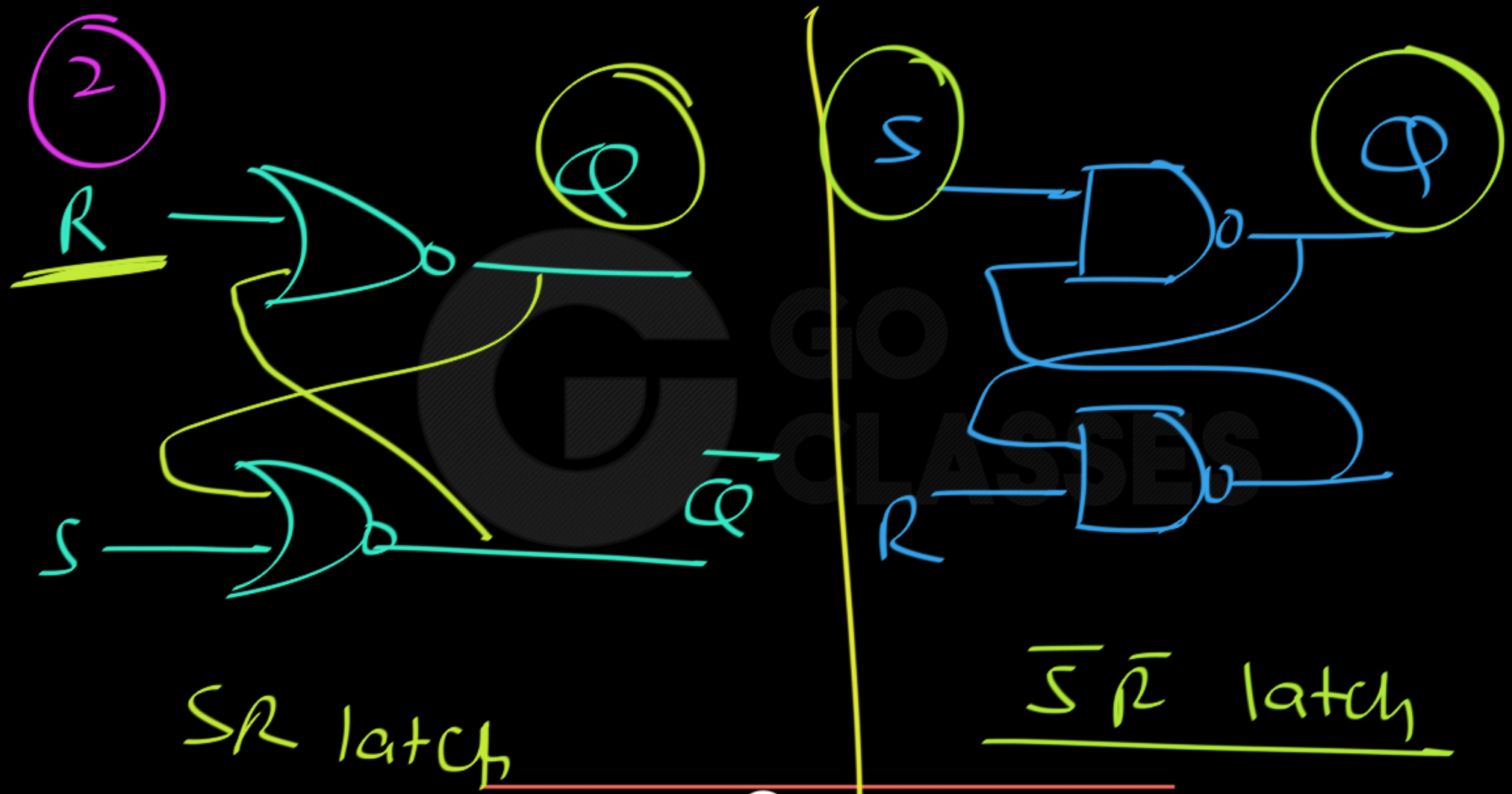
S	R	Q_{t+1}
0	1	1 (set)
1	0	0 (reset)
1	1	retain = Q_t
0	0	forbidden

Comparison of SR latch and $\overline{S}\,\overline{R}$ latch:

(1) SR latch = SR latch with NOR

$\overline{S}\,\overline{R}$ latch = " " " NAND

→ Some Authors


2
R
Q
S
$\bar{Q}$
SR latch
S
Q
R
$\bar{S}\bar{R}$ latch

(3) SR latch is Active High.

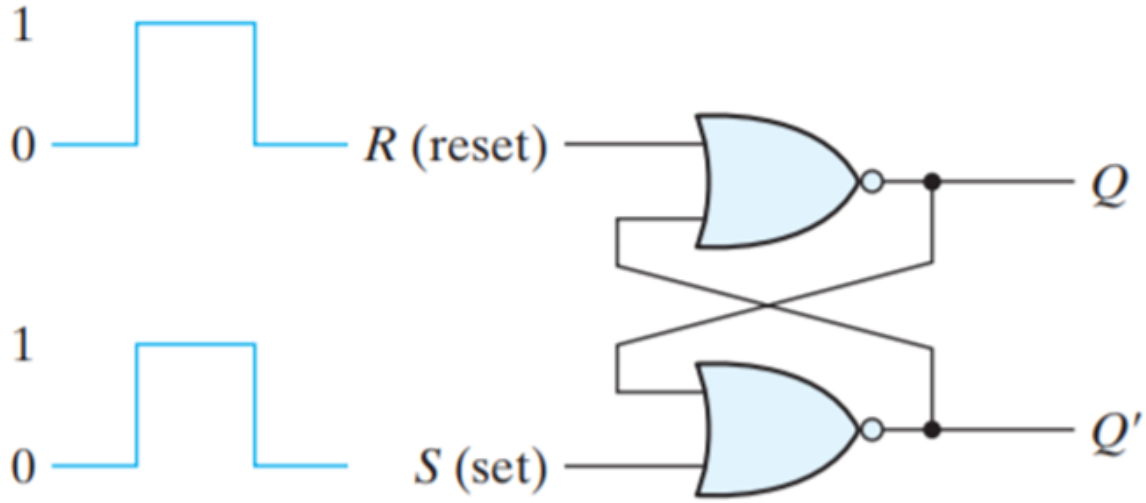
(SR latch with NOR)

$\overline{S}\overline{R}$ latch is Active low.

S	R	SR latch $Q = Q_R$	$\bar{Q}_S$	SR latch $Q = Q_S$	$\bar{Q}_R$
1	0	1 (Set)	0	0 (Reset)	1
0	1	0 (Reset)	1	1 (Set)	0
0	0	Retain	Retain	1	1 (forbidden)
1	1	0	0 (forbidden)	Retain	Retain

SR Latch

The *SR* latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled *S* for set and *R* for reset. The *SR* latch constructed with two cross-coupled NOR gates is shown in Fig. 5.3. The latch has two useful states. When output $Q = 1$ and $Q' = 0$, the latch is said to be in the *set state*. When $Q = 0$ and $Q' = 1$, it is in the *reset state*. Outputs Q and Q' are normally the complement of each other. However, when both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 (rather than be mutually complementary) occurs. If both inputs are then switched to 0 simultaneously, the device will enter an unpredictable or undefined state or a meta-stable state. Consequently, in practical applications, setting both inputs to 1 is forbidden.

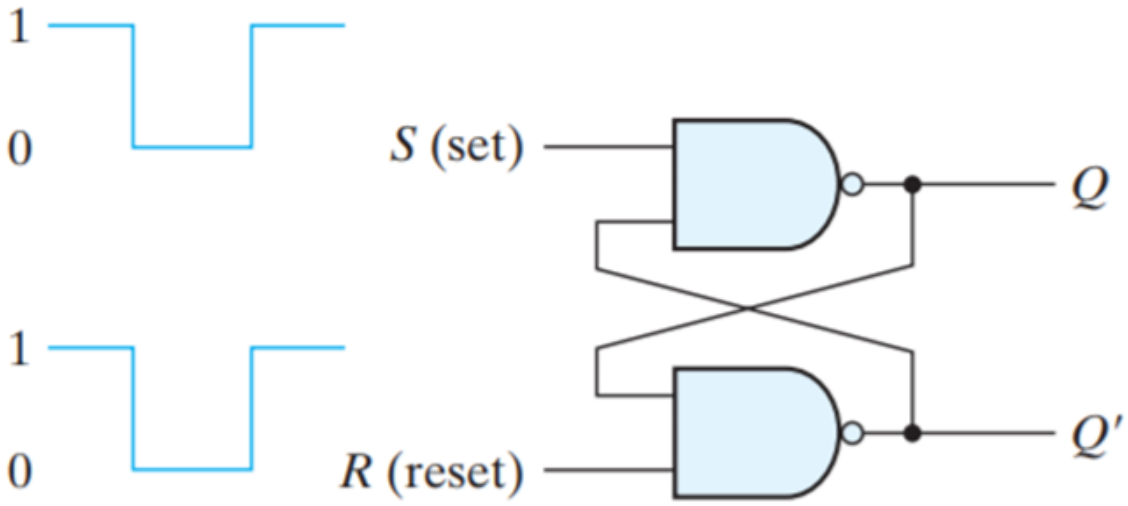


(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

FIGURE 5.3
SR latch with NOR gates



(a) Logic diagram

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

FIGURE 5.4
SR **latch with NAND gates**

In comparing the NAND with the NOR latch, note that the input signals for the NAND require the complement of those values used for the NOR latch. Because the NAND latch requires a 0 signal to change its state, it is sometimes referred to as an $S'R'$ latch. The primes (or, sometimes, bars over the letters) designate the fact that the inputs must be in their complement form to activate the circuit.

Oscillations and Undesirable States

- The basic latch with NAND gates also suffers form oscillation problems, similar to the basic latch implemented with NOR gates.

Note:

in SR latch :

Problem ?

Not the problem

(forbidden Combination)

This Transition is the problem

S R
0 0

Retain Combination

Note:

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

Problem?

This Transition is the problem

Not the problem

(forbidden Combination)

Retain Combination

Note:

in SR latch, $\bar{S}\bar{R}$ latch

Problem?

GATE CSE 2004 | Question: 18, ISRO2007-31

In an SR latch made by cross-coupling two NAND gates, if both S and R inputs are set to 0, then it will result in

A. $Q = 0, Q' = 1$
B. $Q = 1, Q' = 0$
C. $Q = 1, Q' = 1$
D. Indeterminate states

GATE CSE 2004 | Question: 18, ISRO2007-31

In an SR latch made by cross-coupling two NAND gates, if both S and R inputs are set to 0, then it will result in

A. $Q = 0, Q' = 1$
B. $Q = 1, Q' = 0$
C. $Q = 1, Q' = 1$
D. Indeterminate states

$\bar{S}\,\bar{R}$ latch

Not the Answer.

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

Majority will Answer this


D = S
Q = 1
D = R
$\bar{Q}$ = 1

GATE IT 2007 | Question: 7

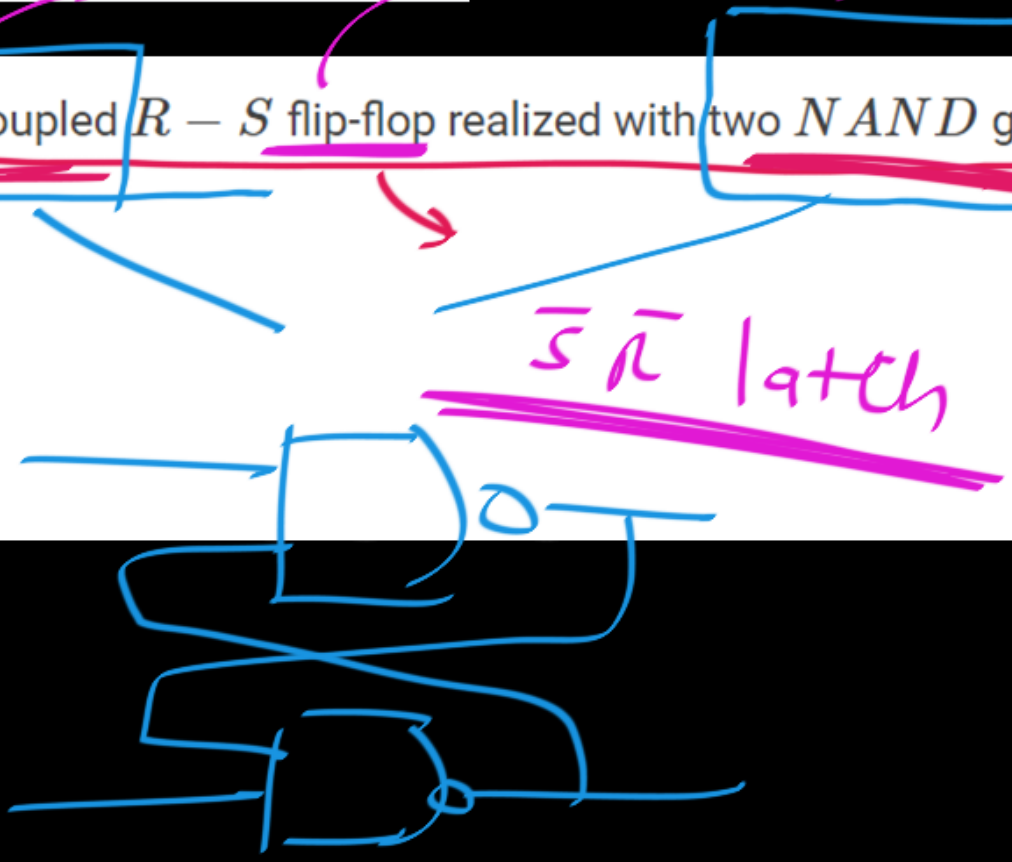
Which of the following input sequences for a cross-coupled $R - S$ flip-flop realized with two $NAND$ gates may lead to an oscillation?

A. $11, 00$
B. $01, 10$
C. $10, 01$
D. $00, 11$

GATE IT 2007 | Question: 7

Which of the following input sequences for a cross-coupled $R - S$ flip-flop realized with two $NAND$ gates may lead to an oscillation?

A. $11, 00$
B. $01, 10$
C. $10, 01$
D. $00, 11$



In an SR latch made by cross-coupling two NAND gates, if both S and R inputs are set to 0, then it will result in

A. $Q = 0, Q' = 1$
B. $Q = 1, Q' = 0$
C. $Q = 1, Q' = 1$
D. Indeterminate states

GATE 2004

Which of the following input sequences for a cross-coupled $R - S$ flip-flop realized with two $NAND$ gates may lead to an oscillation?

A. $11, 00$
B. $01, 10$
C. $10, 01$
D. $00, 11$

GATE 2007