Registers and Counters

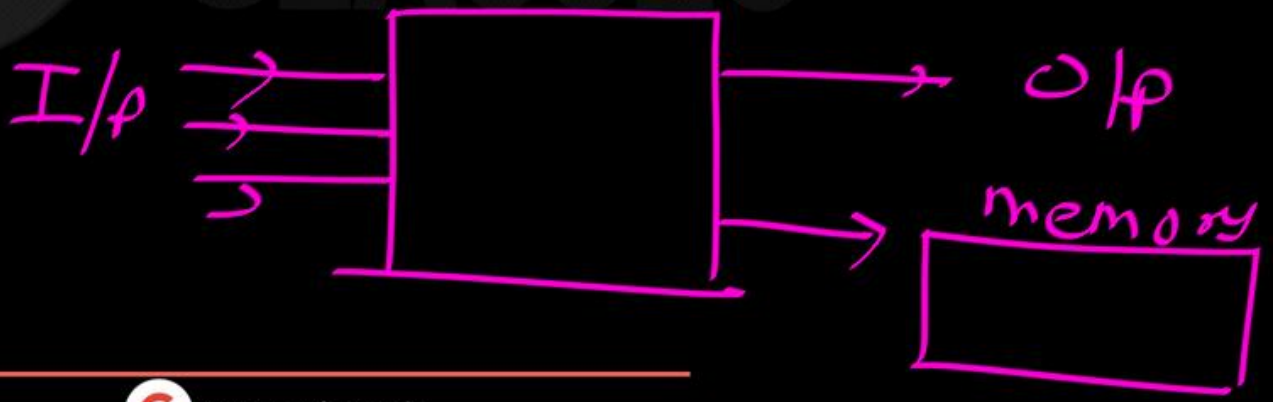
- Combinational Circuits :

The logic circuits in which the output depends only on the condition of the latest inputs.

Ex: logic gates, MUX, Demux, FA, HA, CLA, RCA -

- Sequential Circuits :

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.



Memory Elements :

- A memory stores data – usually one bit per element
- A snapshot of the memory is called the state
- A one bit memory is often called a bistable, i.e., it has 2 stable internal states i.e. 0,1.
- Flip-flops and latches are particular implementations of bistables

ff: 1-bit Storage Device

0 or 1

Sequential circuits

* The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are *combinatorial* in nature, i.e., the output(s) depends only on the *present* values of the inputs and *not* on their past values.
* In *sequential* circuits, the "state" of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values, depending on its previous state.
* In other words, a sequential circuit has a *memory* (of its past state) whereas a combinatorial circuit has no memory.
* Sequential circuits (together with combinatorial circuits) make it possible to build several useful applications, such as counters, registers, arithmetic/logic unit (ALU), all the way to microprocessors.

D-FF 1 bit

To store 4-bits

To store n-bits ⇒ n - FF

Sequential Circuit :

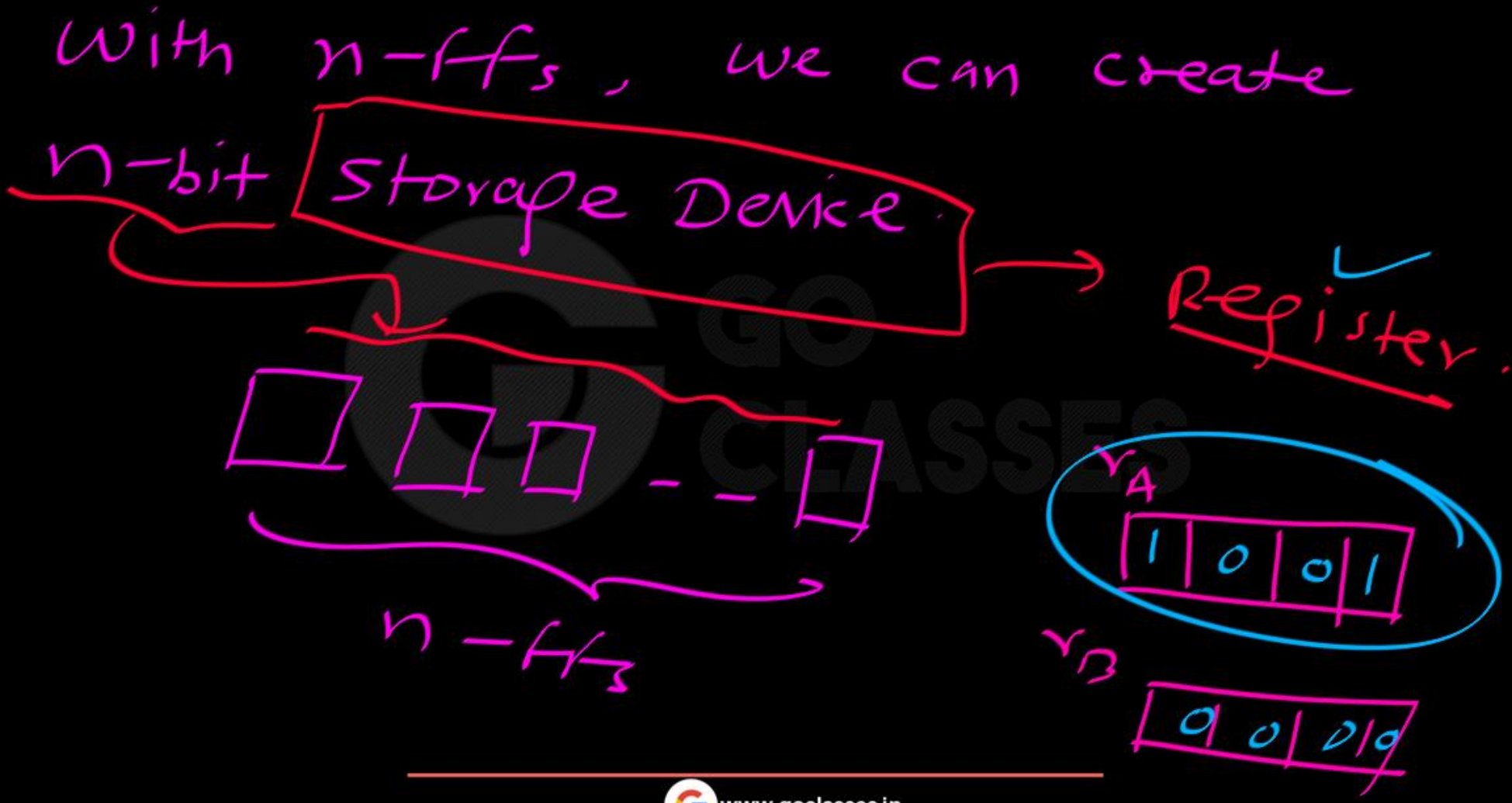
A clocked sequential circuit consists of a group of flip-flops and combinational gates. The flip-flops are essential because, in their absence, the circuit reduces to a purely combinational circuit (provided that there is no feedback among the gates).

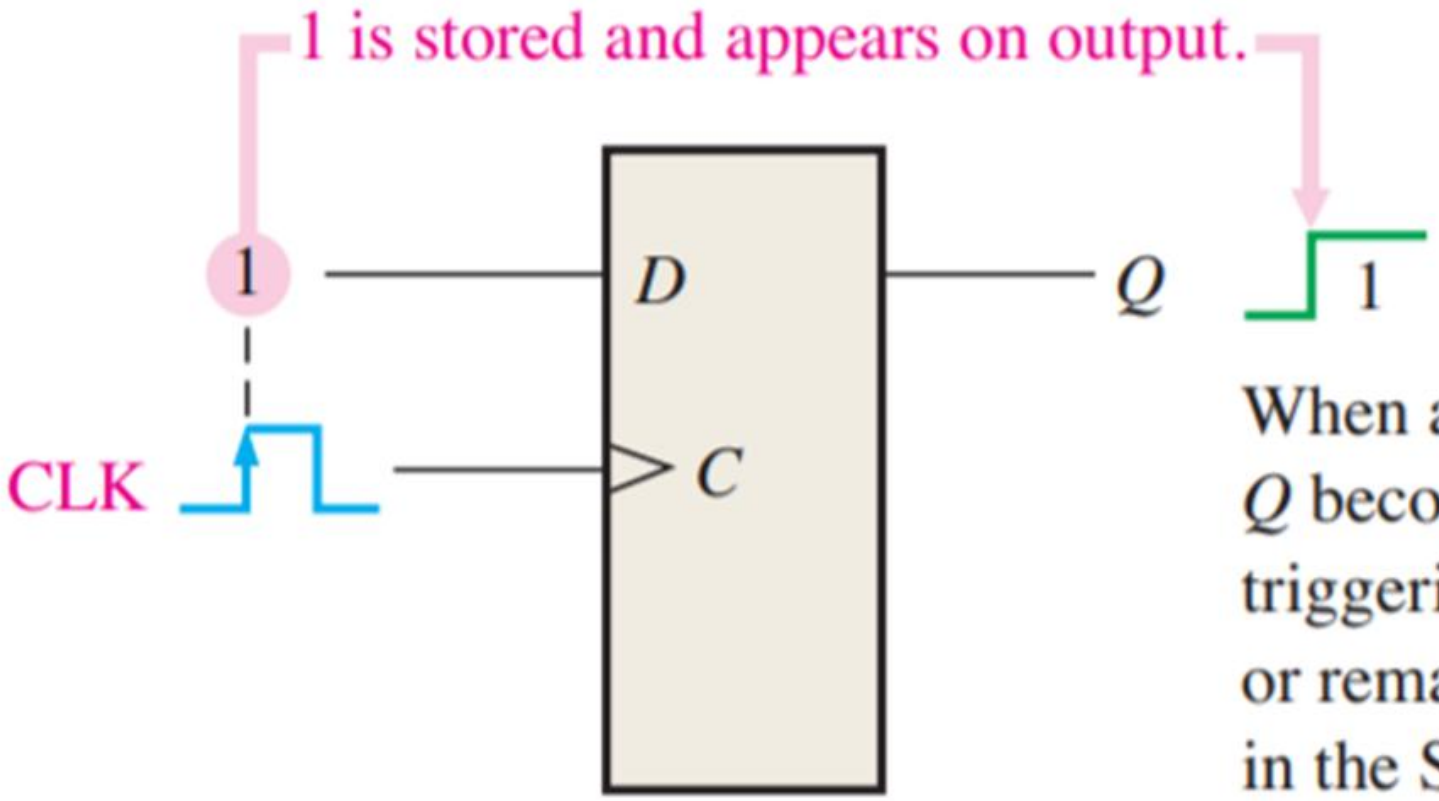
A circuit with flip-flops is considered a sequential circuit even in the absence of combinational gates.

Circuits that include flip-flops are usually classified by the function they perform rather than by the name of the sequential circuit. Two such circuits are registers and counters.

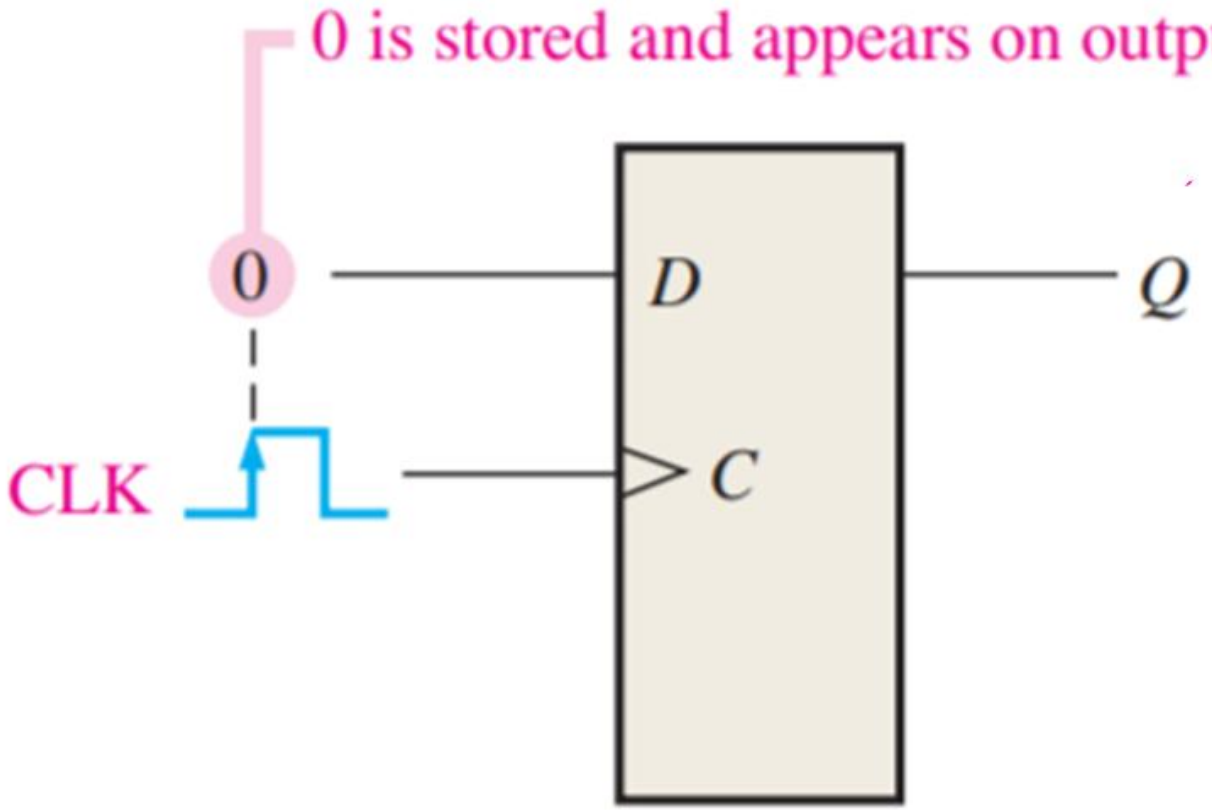
FlipFlop

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


With n-ffs, we can create
n-bit Storage Device
Register.
n-ffs
r_A
1 0 0 1
r_B
0 0 0 0



When a 1 is on D, Q becomes a 1 at the triggering edge of CLK or remains a 1 if already in the SET state.



When a 0 is on D, Q becomes a 0 at the triggering edge of CLK or remains a 0 if already in the RESET state.

Figure 8–1 illustrates the concept of storing a 1 or a 0 in a D flip-flop. A 1 is applied to the data input as shown, and a clock pulse is applied that stores the 1 by setting the flip-flop. When the 1 on the input is removed, the flip-flop remains in the SET state, thereby storing the 1. A similar procedure applies to the storage of a 0 by resetting the flip-flop, as also illustrated in Figure 8–1.

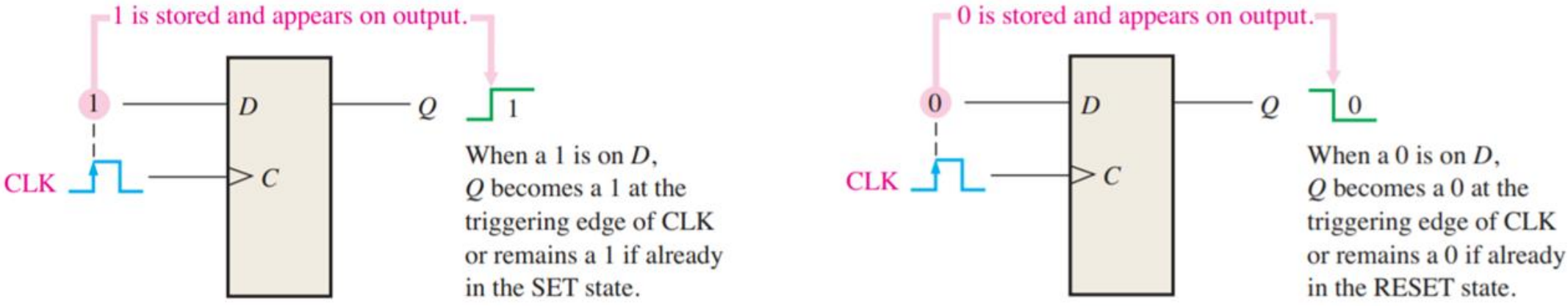


FIGURE 8–1 The flip-flop as a storage element.

Applications of Flip Flops :

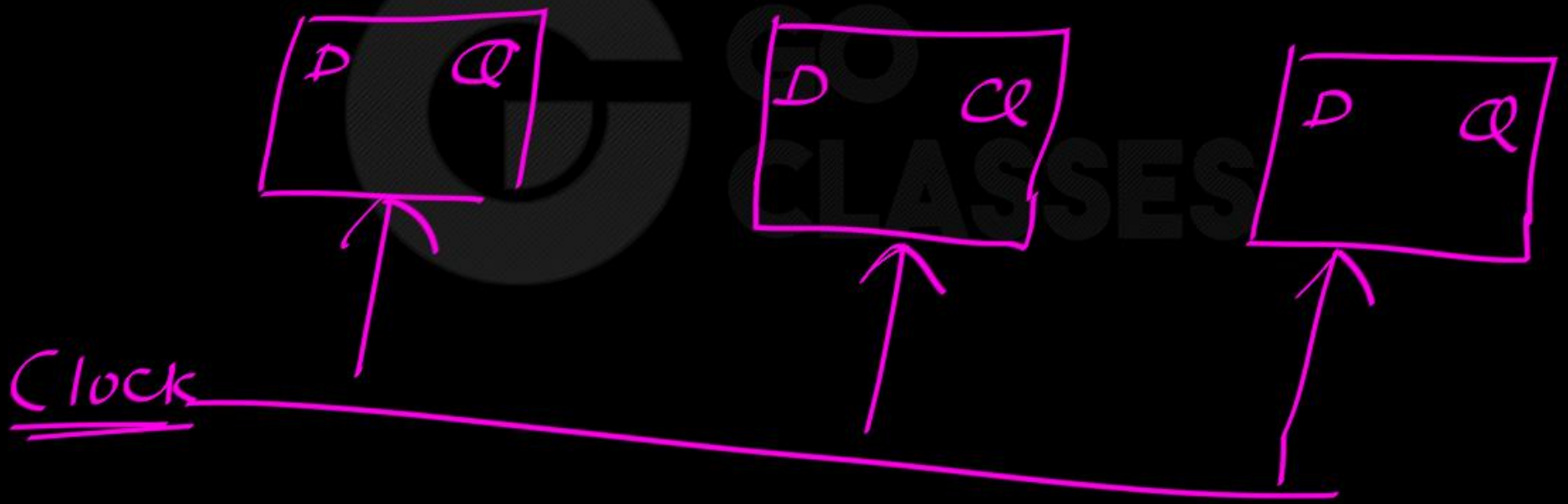
Flip Flops are mostly used as data storage devices, Hence : Registers.

We can also use flip flops to implement some other functionalities like frequency dividers etc.., Hence : Counters.

A register is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information. An n -bit register consists of a group of n flip-flops capable of storing n bits of binary information.

In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks. In its broadest definition, a register consists of a group of flip-flops together with gates that affect their operation. The flip-flops hold the binary information, and the gates determine how the information is transferred into the register.

Register: 3-bit Register



A counter is essentially a register that goes through a predetermined sequence of binary states. The gates in the counter are connected in such a way as to produce the prescribed sequence of states. Although counters are a special type of register, it is common to differentiate them by giving them a different name.

Counter

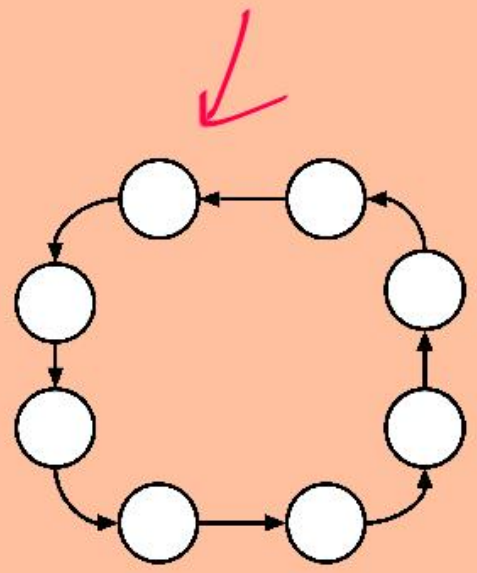
Eg: Traffic light :

A register consists of a group of flip-flops with a common clock input.

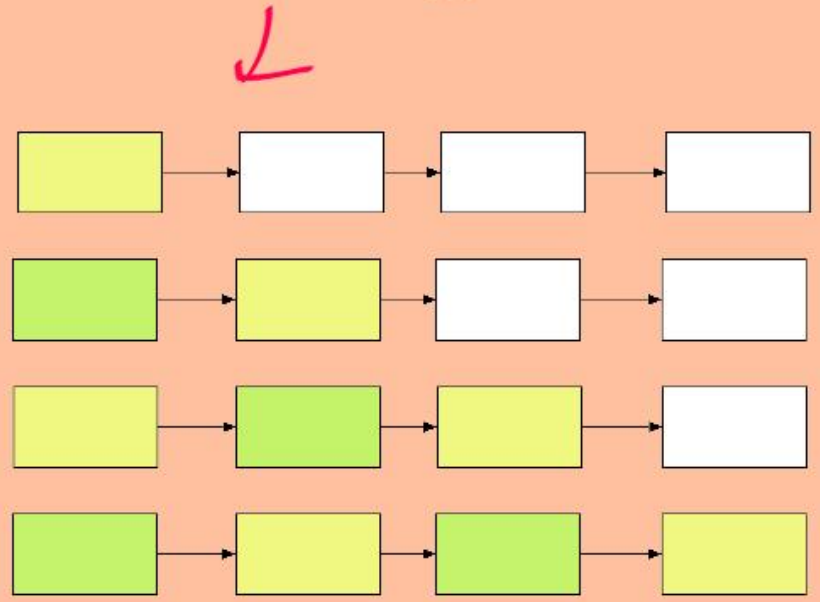
Registers are commonly used to store and shift binary data.

Counters are another simple type of sequential circuit. A counter is usually constructed from two or more flip-flops which change states in a prescribed sequence when input pulses are received.

Counter & Shift Register



VS



Registers and Counters

A circuit with flip-flops is considered a sequential circuit even in the absence of combinational logic. Circuits that include flip-flops are usually classified by the function they perform. Two such circuits are *registers* and *counters*:

- **Register** – is a group of flip-flops. Its basic function is to hold information within a digital system so as to make it available to the logic units during the computing process. However, a register may also have additional capabilities associated with it.
- **Counter** – is essentially a register that goes through a predetermined sequence of states. The gates in the counter are connected in such a way as to produce the prescribed sequence of binary states.

Applications of registers include serial addition, convolutional encoders for error-control coding, and pseudo-random binary sequence generators.

Counters are primarily used as pattern generators.

Registers

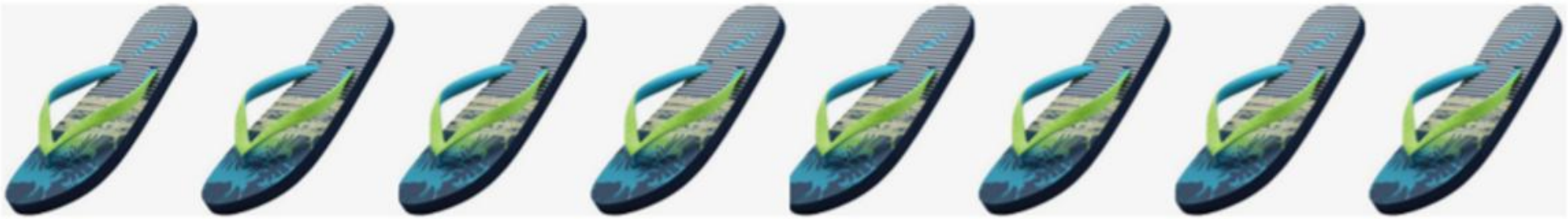
Register (Definition)

An n-bit structure consisting of flip-flops.

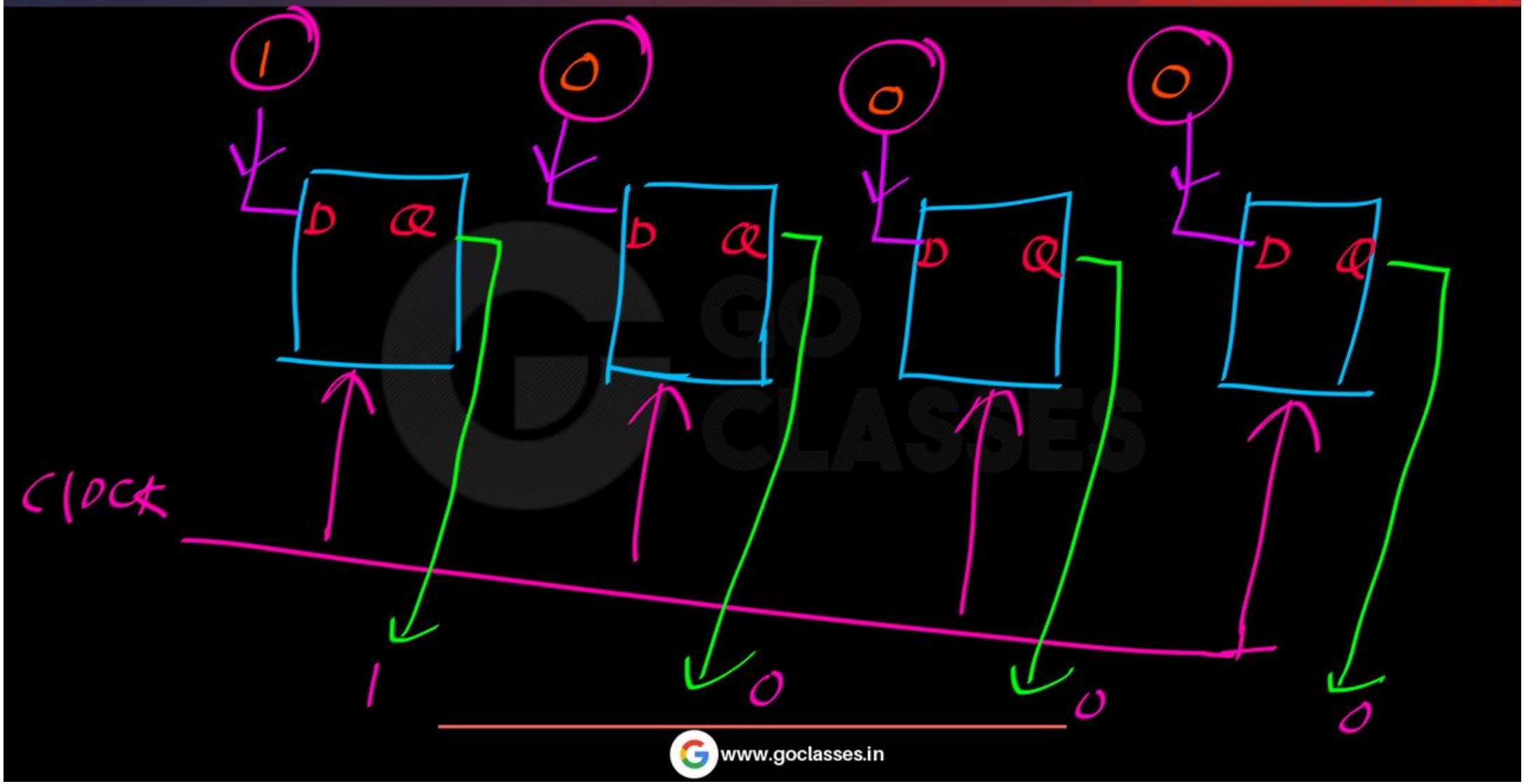
Flip-Flop

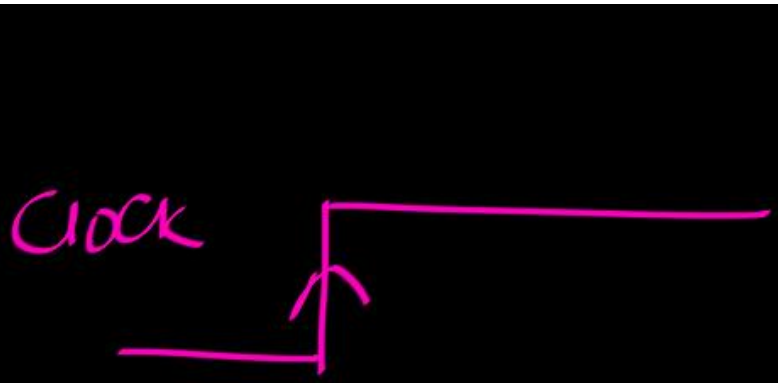
4-bit Register

8-bit Register




Objective:
1 0 0 0
R
1 0 0 0
LOAD
SIPO Register


1
0
0
0
D
Q
D
Q
D
Q
D
Q
CLOCK
1
0
0
0

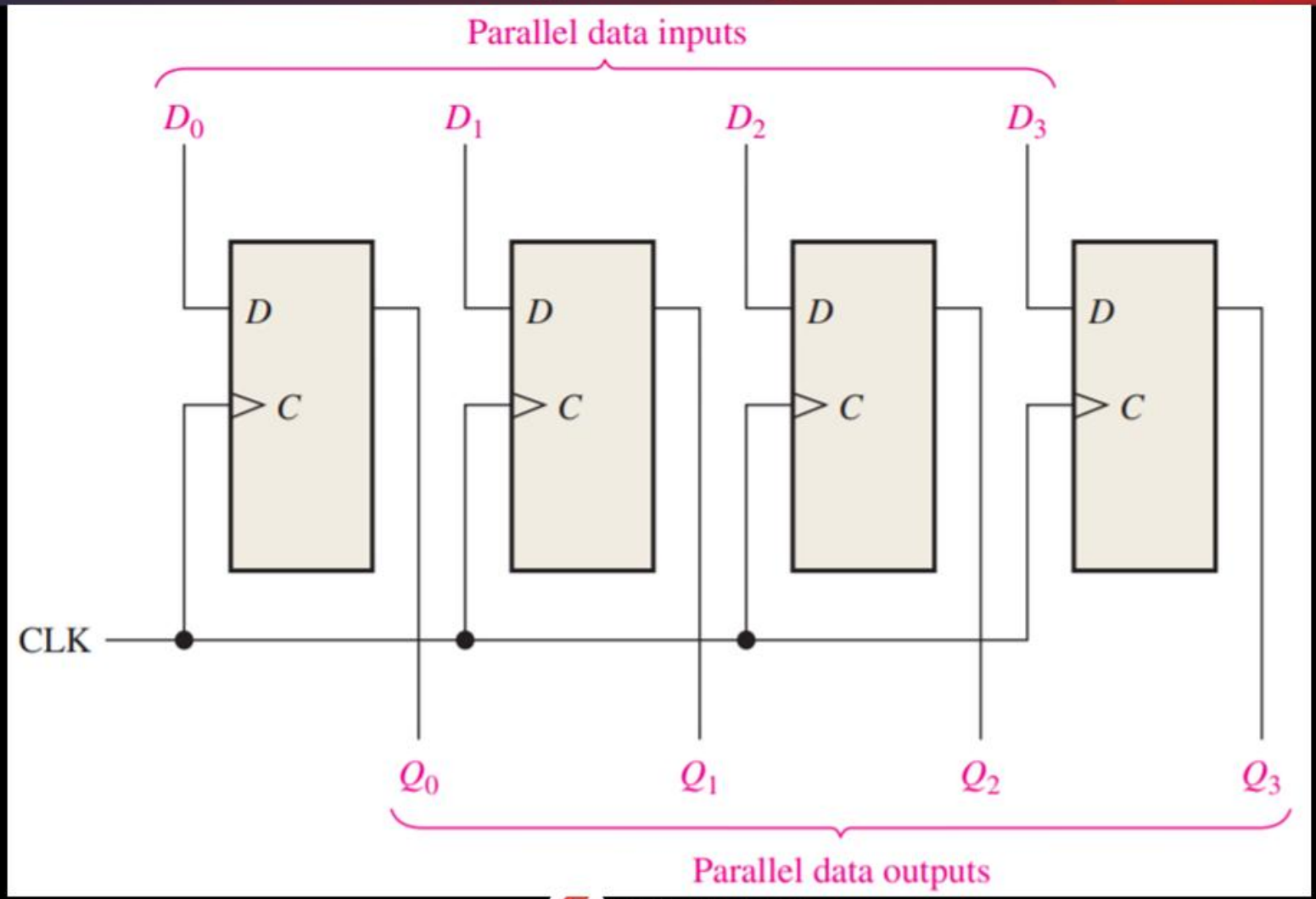


FF-output = the Data which is stored = State of FF

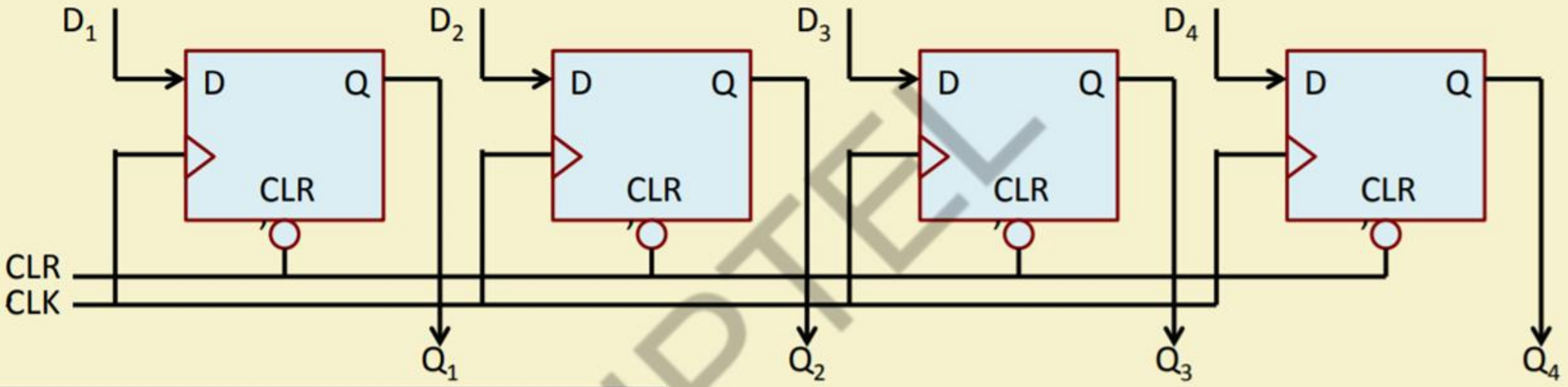
Storing Data in a ff:

that Data is Available on ff output

D-ff : (Data - ff) (Delay - ff)

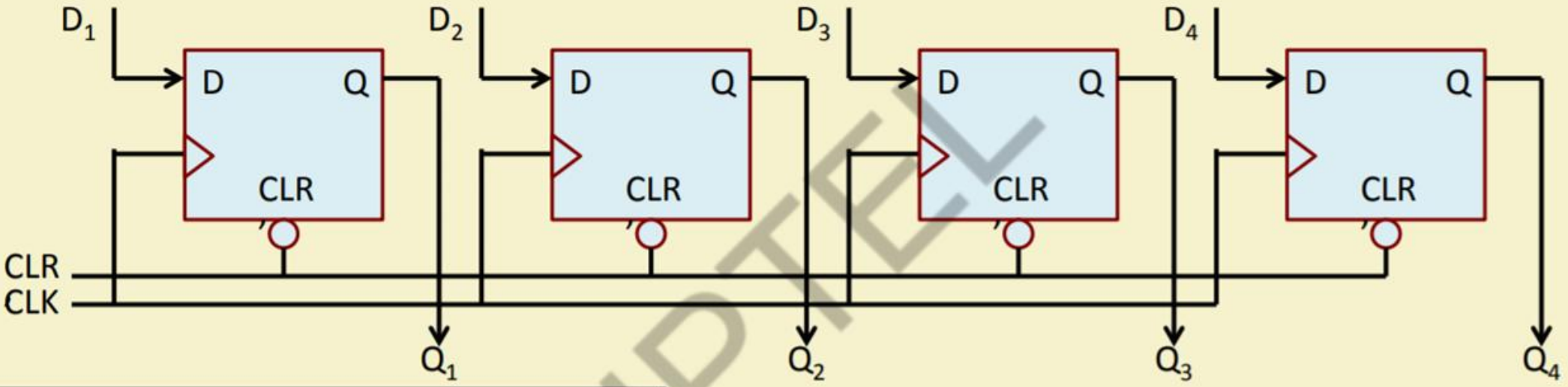


4-bit PIPO Register



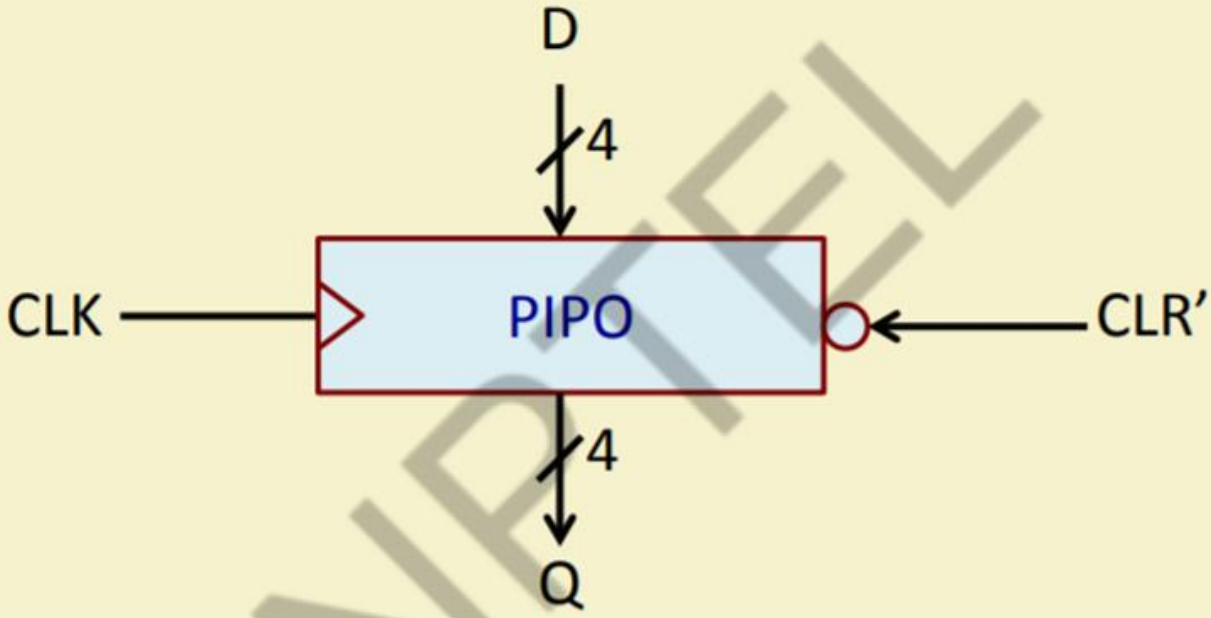
When the active clock edge arrives, input word $D_1D_2D_3D_4$ gets stored in the register, and available on the outputs $Q_1Q_2Q_3Q_4$.

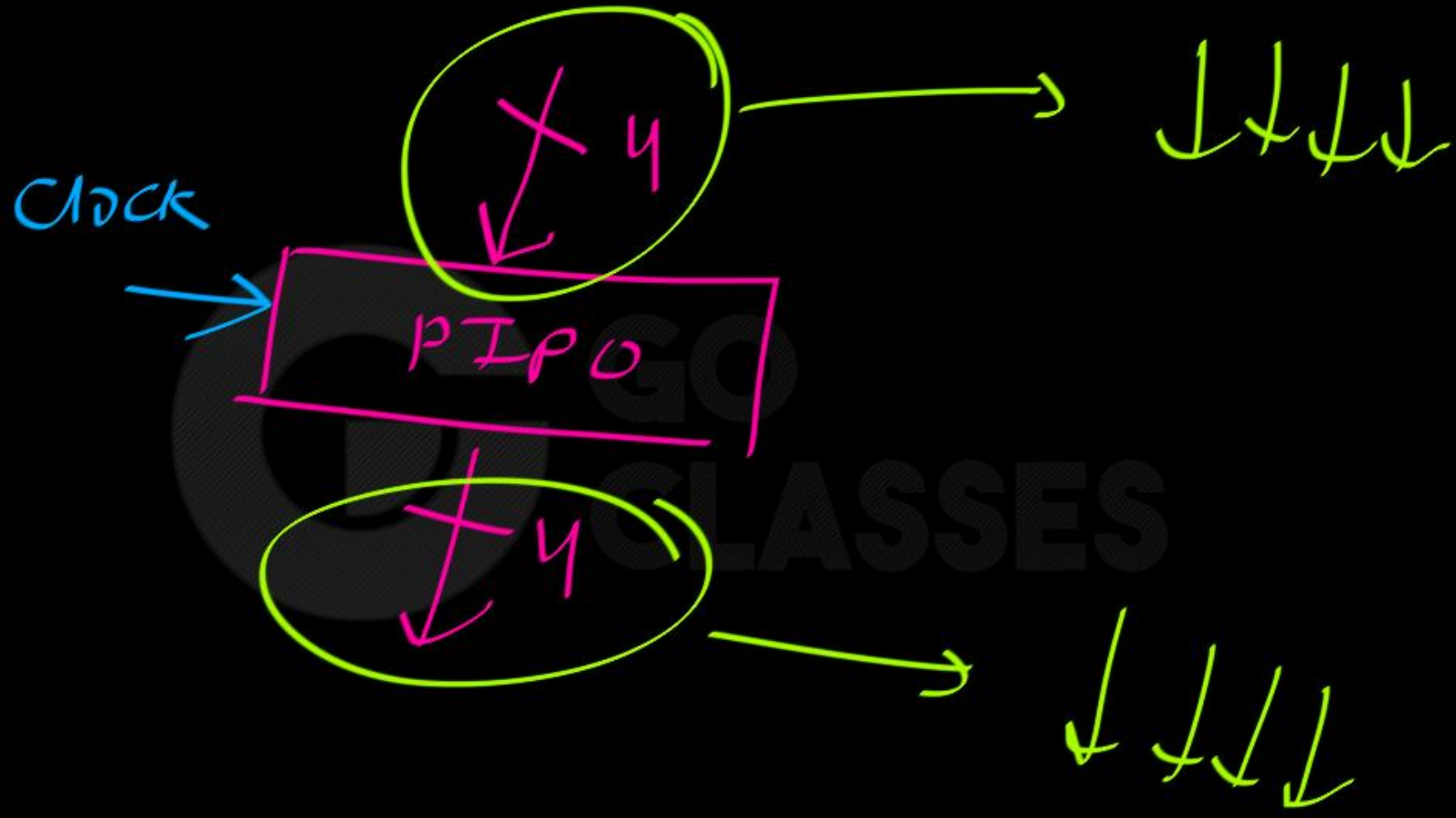
Basic PIPO Register



When the active clock edge arrives, input word $D_1D_2D_3D_4$ gets stored in the register, and available on the outputs $Q_1Q_2Q_3Q_4$.

Symbolic Representation (using vector notation)




Clock
PIPO

Vector Notation:

$\equiv$

A register is a digital circuit(Sequential Circuit) with two basic functions: data storage and data movement.

Registers consist of arrangements of flip-flops and are important in applications involving the storage and transfer of data in a digital system. A register has no specified sequence of states, except in certain very specialized applications. A register, in general, is used solely for storing and shifting/moving data (1s and 0s) entered into it from an external source and typically possesses no characteristic internal sequence of states.

Various types of registers are available commercially.

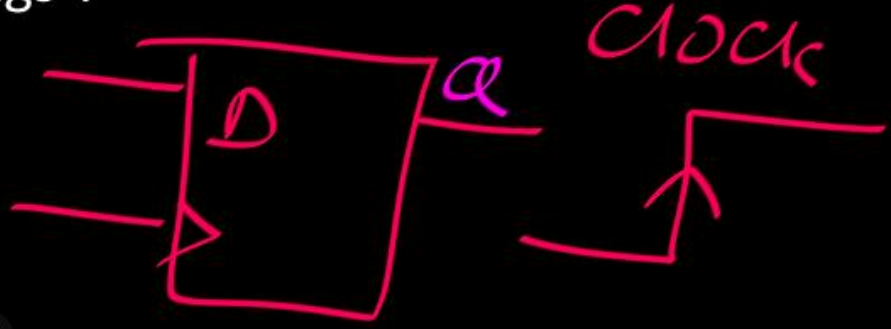
For Register, we need to understand three things :

How a flip-flop stores a data bit

The storage capacity of a register

The shift capability of a register

For Register, we need to understand three things :



How a flip-flop stores a data bit :
We have already seen.

The storage capacity of a register

The storage capacity of a register is the total number of bits (1s and 0s) of digital data it can retain. Each stage (flip-flop) in a shift register represents one bit of storage capacity; therefore, the number of stages in a register determines its storage capacity.

The shift/move capability of a register

For Register, we need to understand three things :

How a flip-flop stores a data bit :

We have already seen.

The storage capacity of a register

The storage capacity of a register is the total number of bits (1s and 0s) of digital data it can retain. Each stage (flip-flop) in a shift register represents one bit of storage capacity; therefore, the number of stages in a register determines its storage capacity.

The shift capability of a register :

The shift capability of a register permits the movement of data from stage to stage within the register OR into or out of the register upon application of clock pulses.

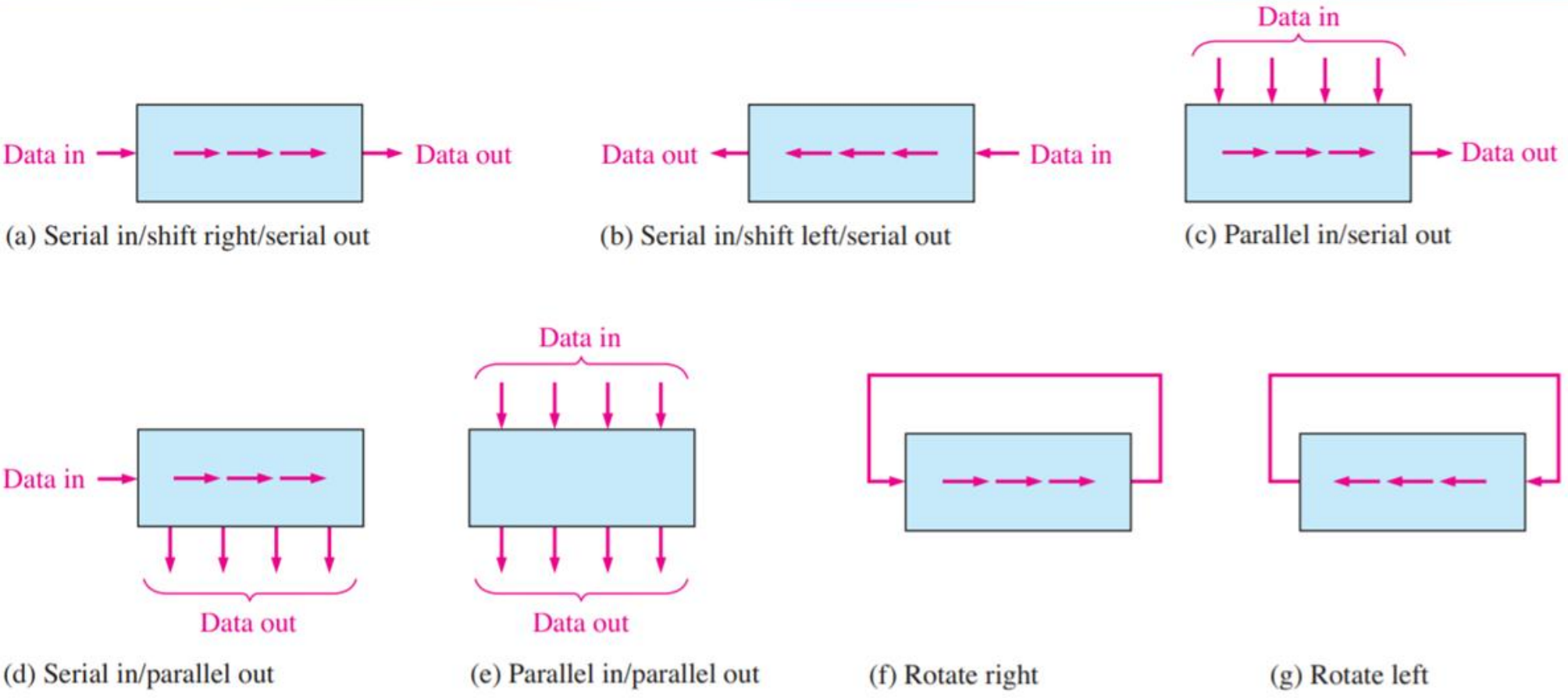


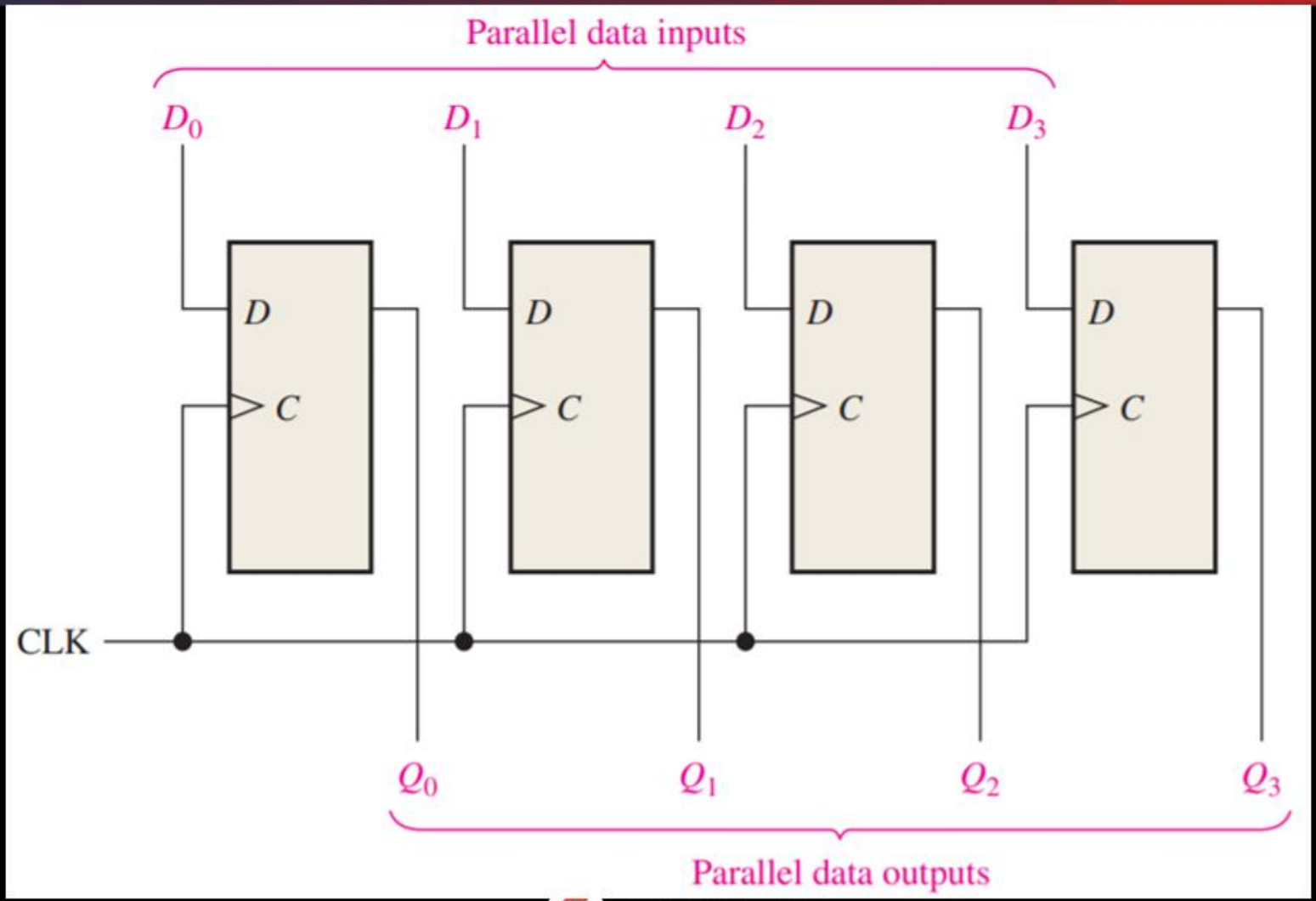
FIGURE 8–2 Basic data movement in registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

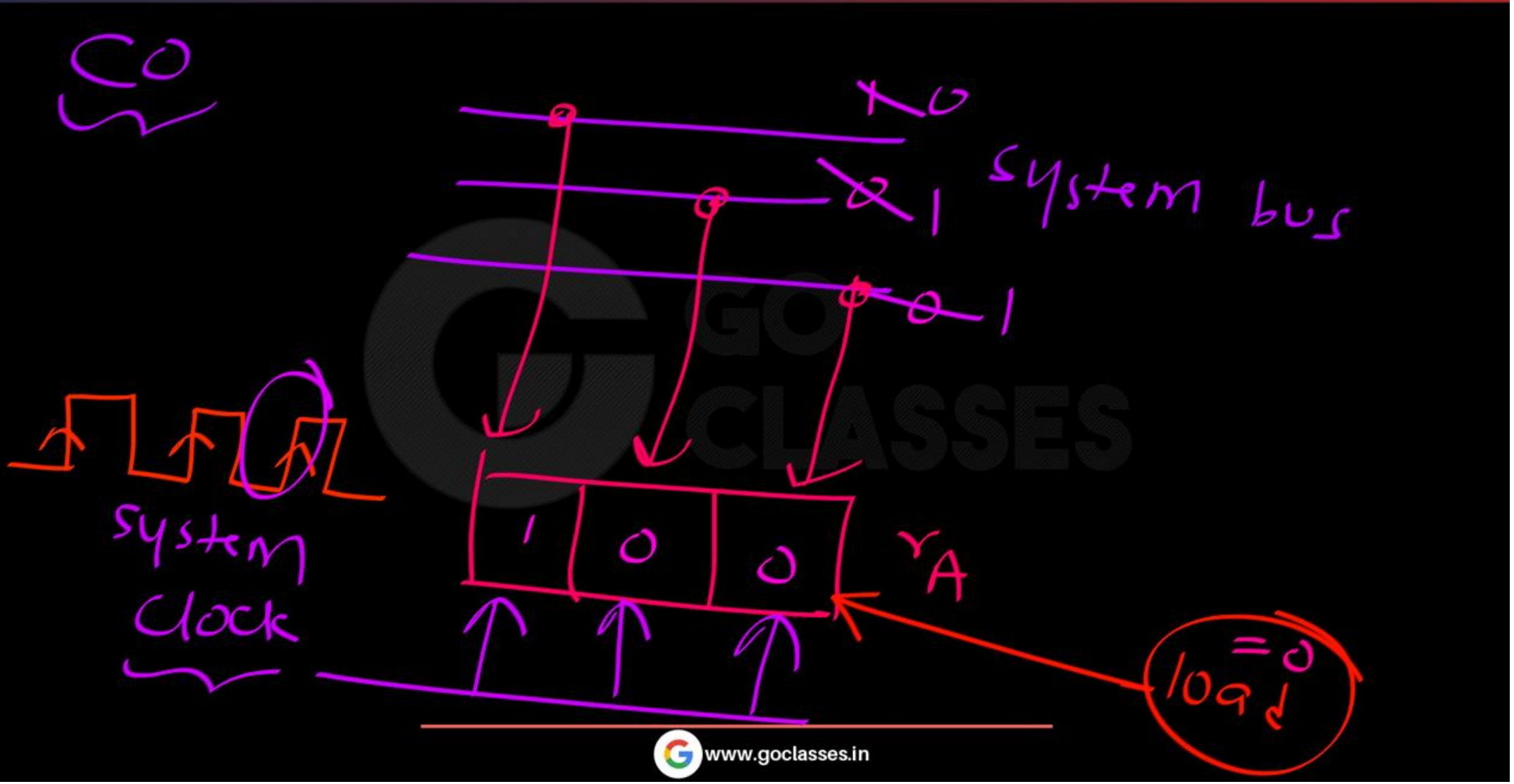
Register : n-bit Storage Device

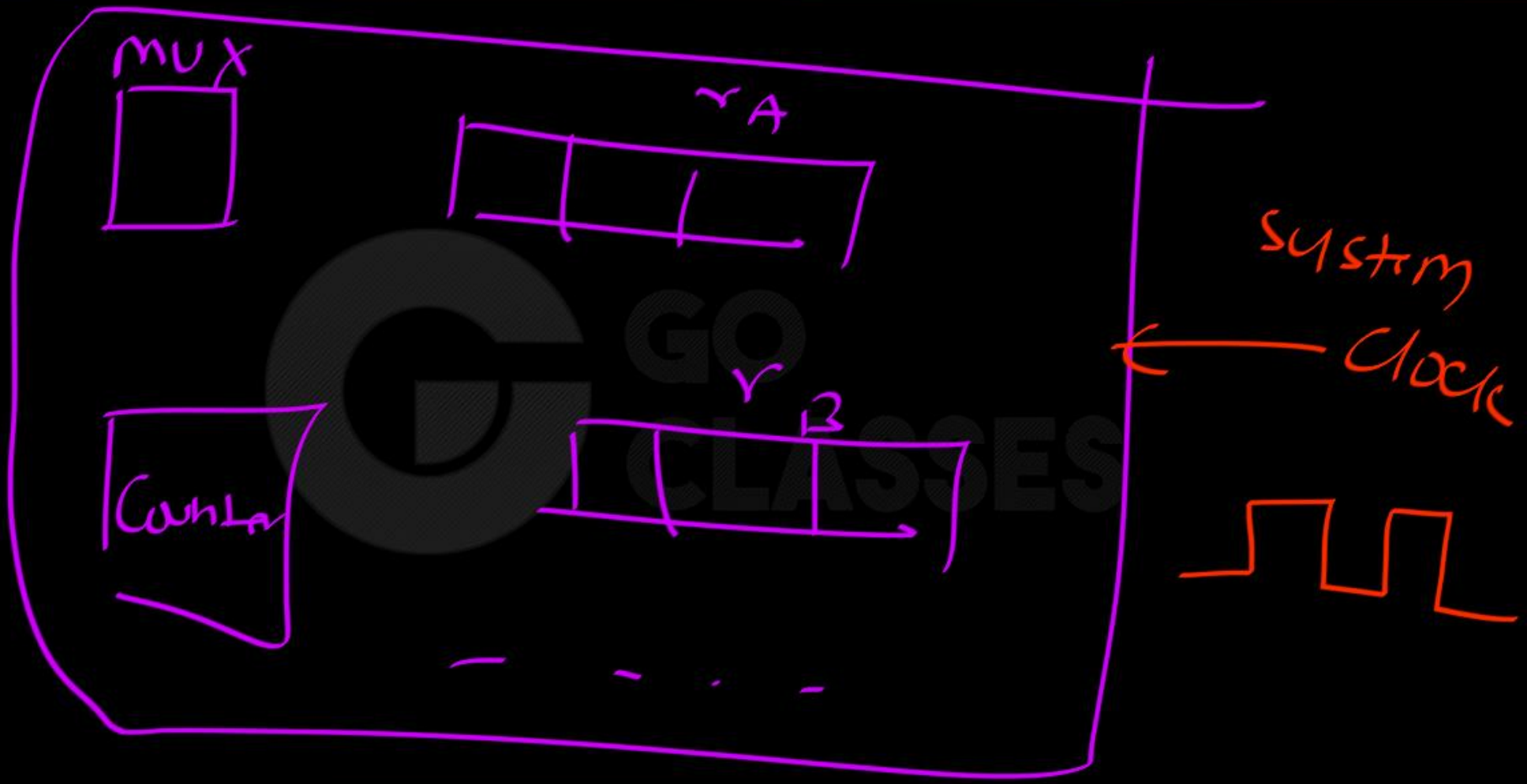
Based on How Data is moving within (or b/w) the register, we have many Different type of Registers.

Parallel-Access Register

(PIPO Register)


Parallel data inputs
D_0
D_1
D_2
D_3
D
C
D
C
D
C
D
C
CLK
Q_0
Q_1
Q_2
Q_3
Parallel data outputs


CO
x0
x1
system bus
1
system
clock
1
0
0
YA
load
=0


MUX
A
System
Clock
B
Counter

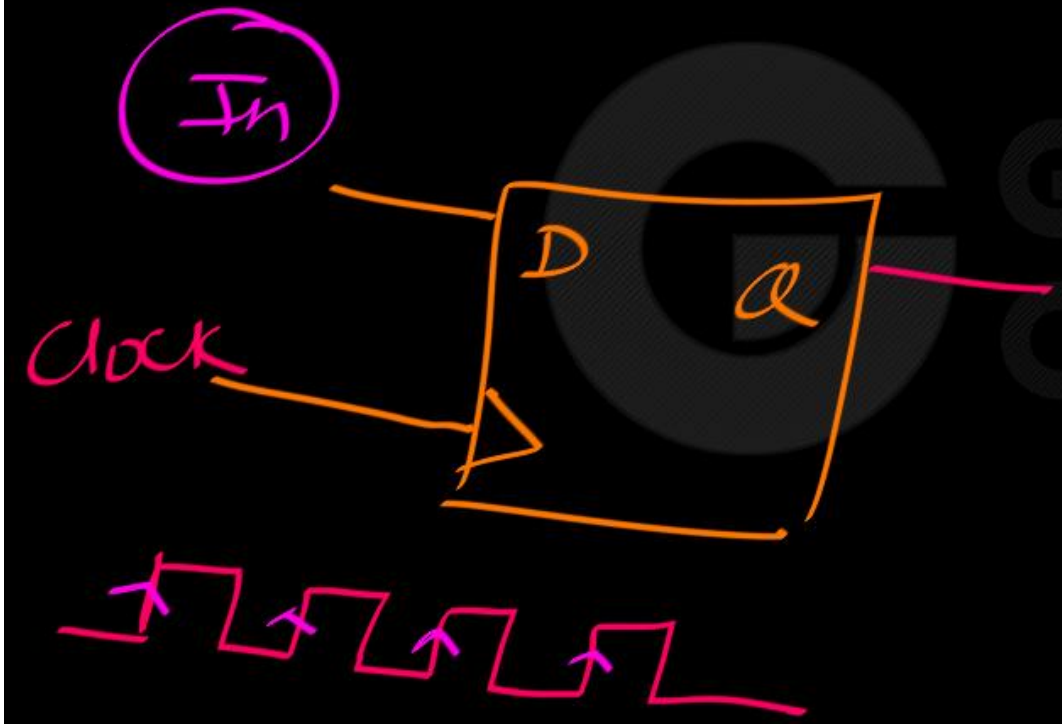
We want to choose/control When to store the Data :

We use "load" signal/ Input.

load = 1 ; Data must be stored

load = 0 ; Data Retained / unchanged.

1-bit Register:



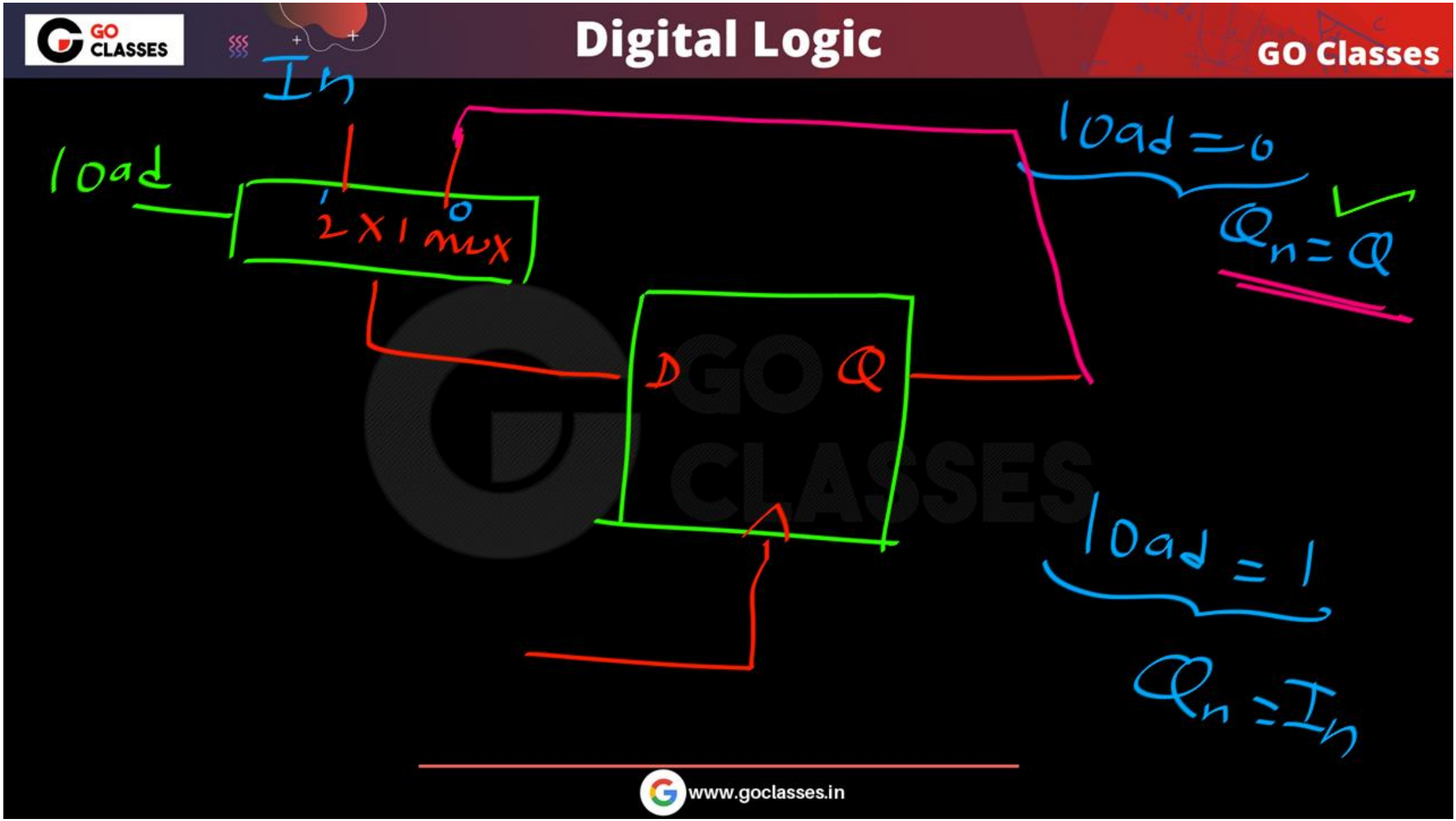
load Input

load = 1

$Q_n = I_n$

load = 0

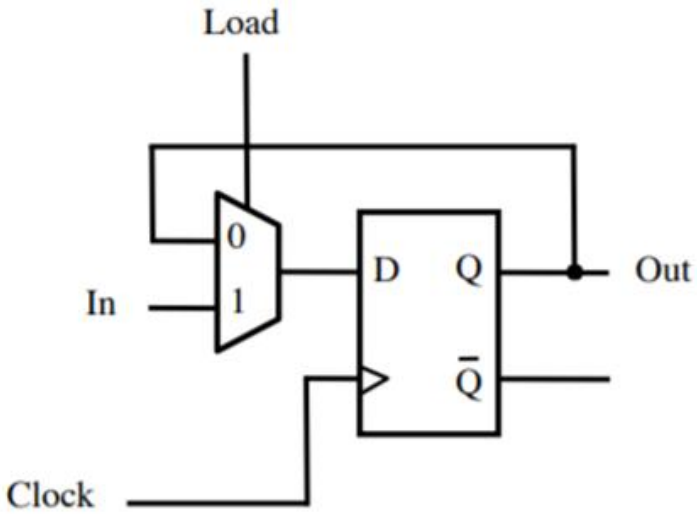
$Q_n = Q$


In
load
1
0
2 X 1 mux
load = 0
$Q_n = Q$
D
Q
load = 1
$Q_n = I_n$

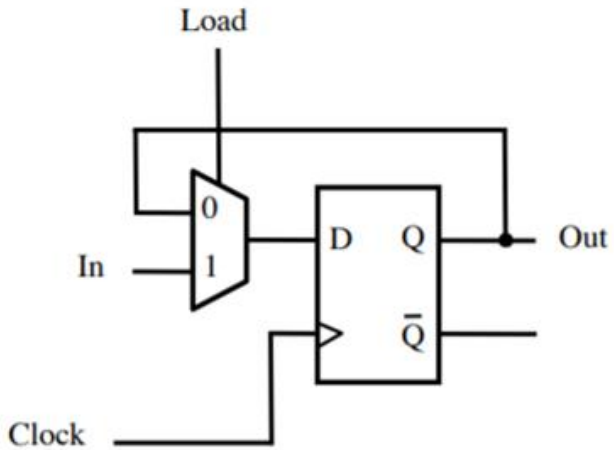
"load" input in Register:

To control when the Input Data should be stored in the Register.

1-Bit Parallel-Access Register



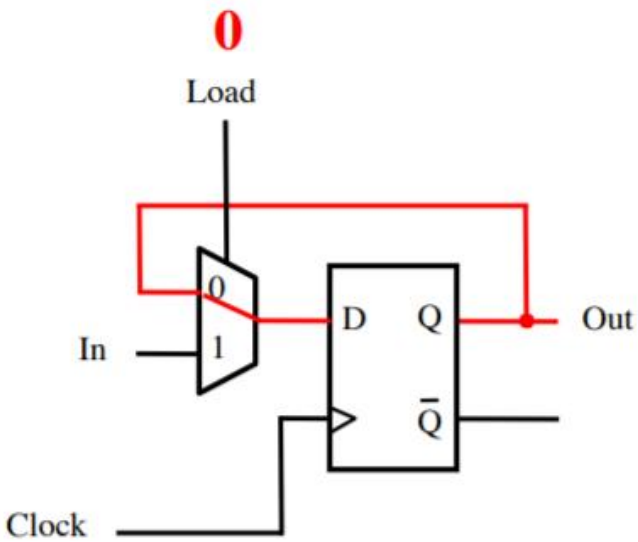
1-Bit Parallel-Access Register



The 2-to-1 multiplexer is used to select whether to load a new value into the D flip-flop or to retain the old value.

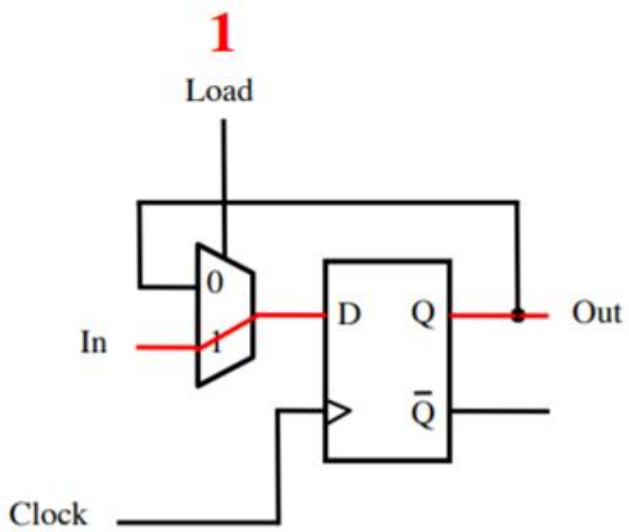
The output of this circuit is the Q output of the flip-flop.

1-Bit Parallel-Access Register



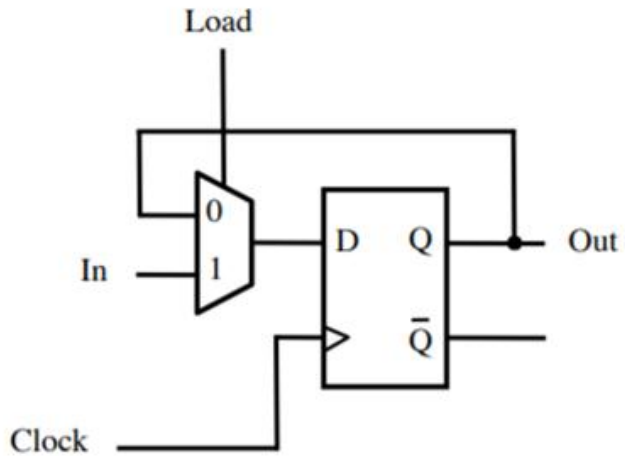
If Load = 0, then retain the old value.

1-Bit Parallel-Access Register



If Load = 1, then load the new value from In.

1-Bit Parallel-Access Register

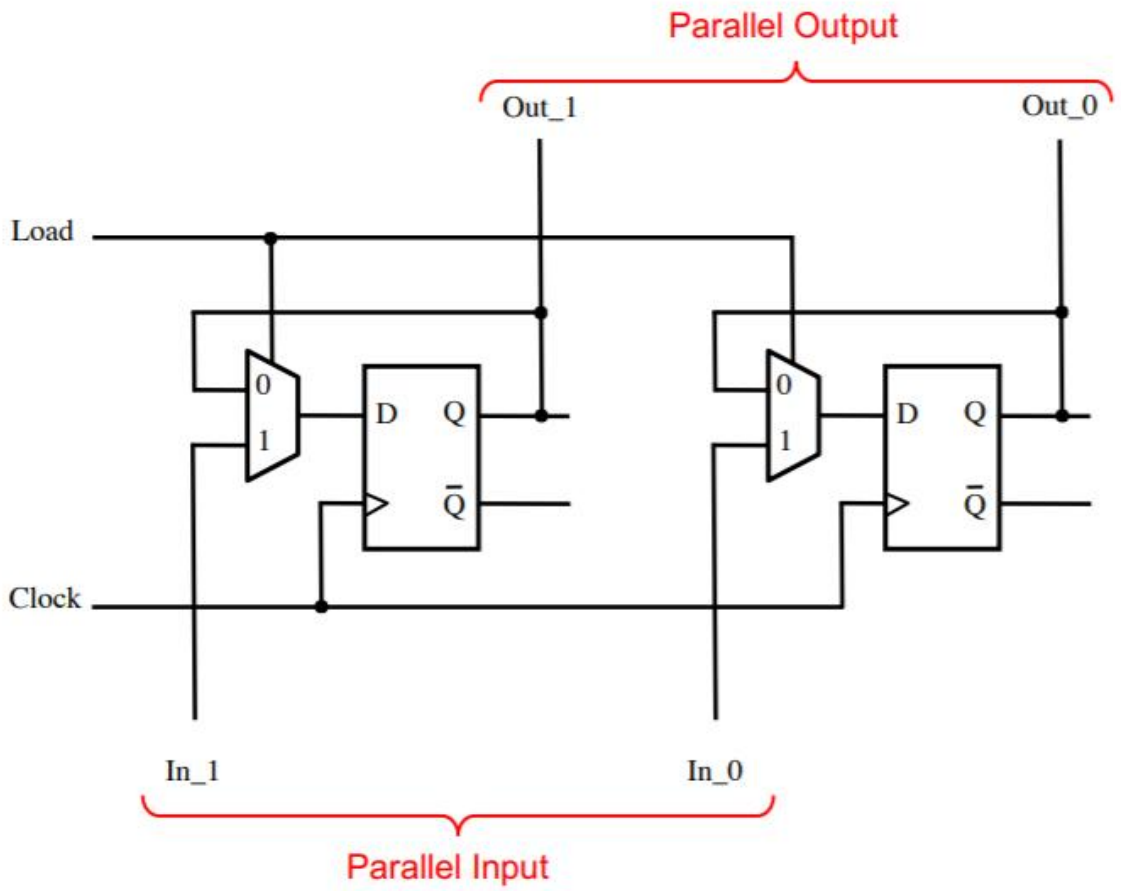


If Load = 0, then retain the old value.

If Load = 1, then load the new value from In.

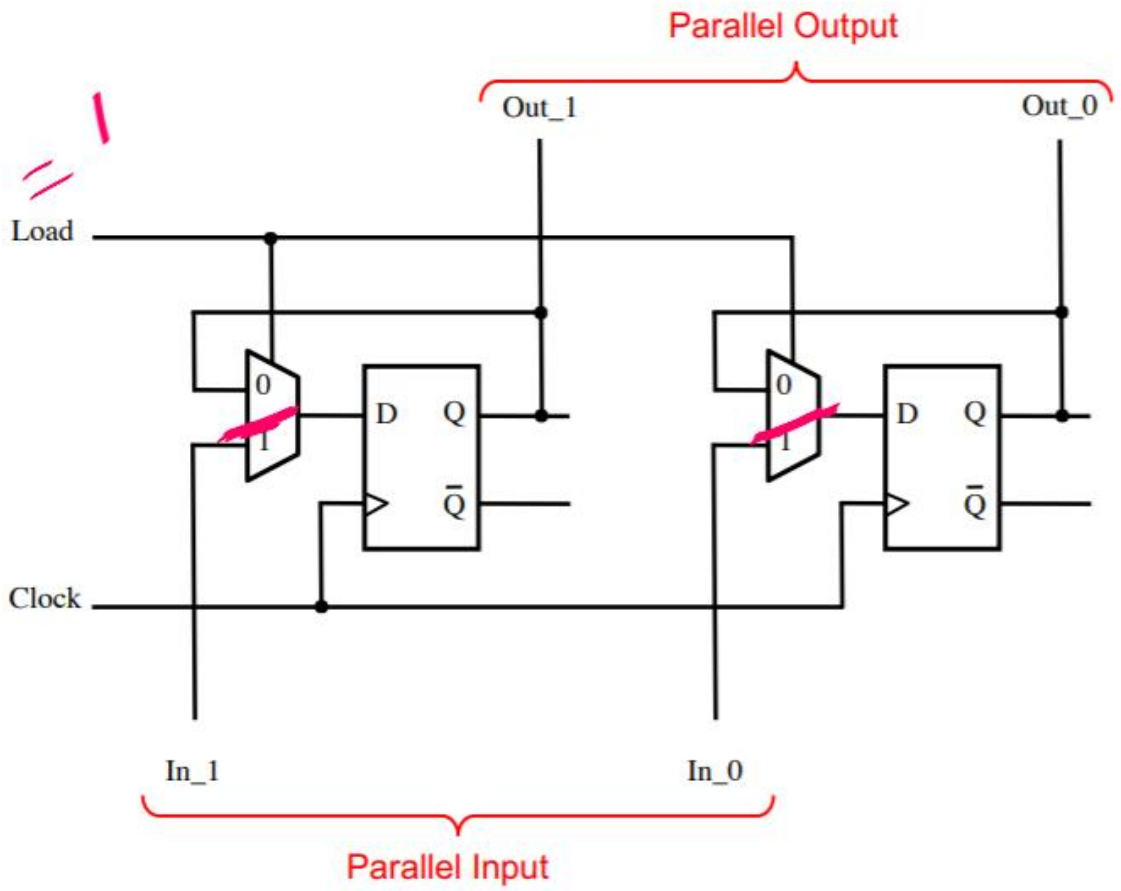
2-Bit Parallel-Access Register

2-bit PIPO Register with "load" control



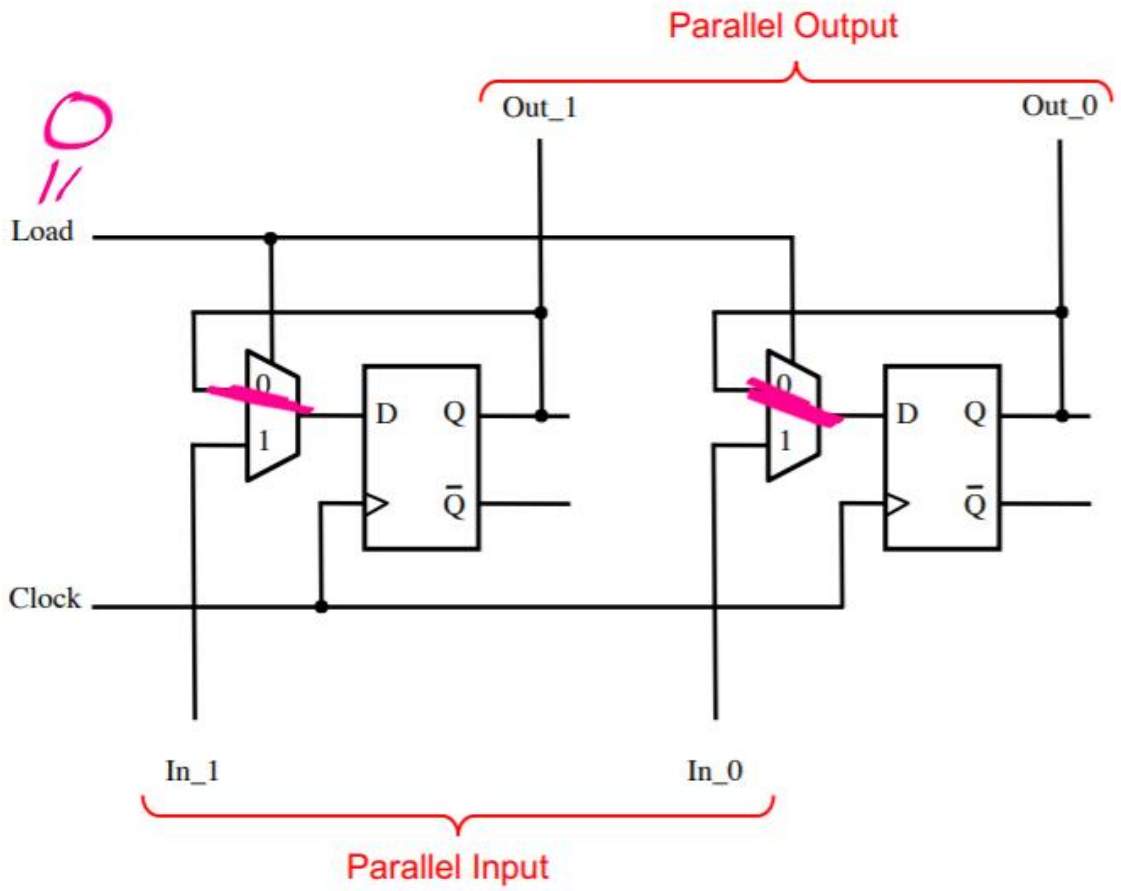
2-Bit Parallel-Access Register

2-bit PIPO Register with "load" control

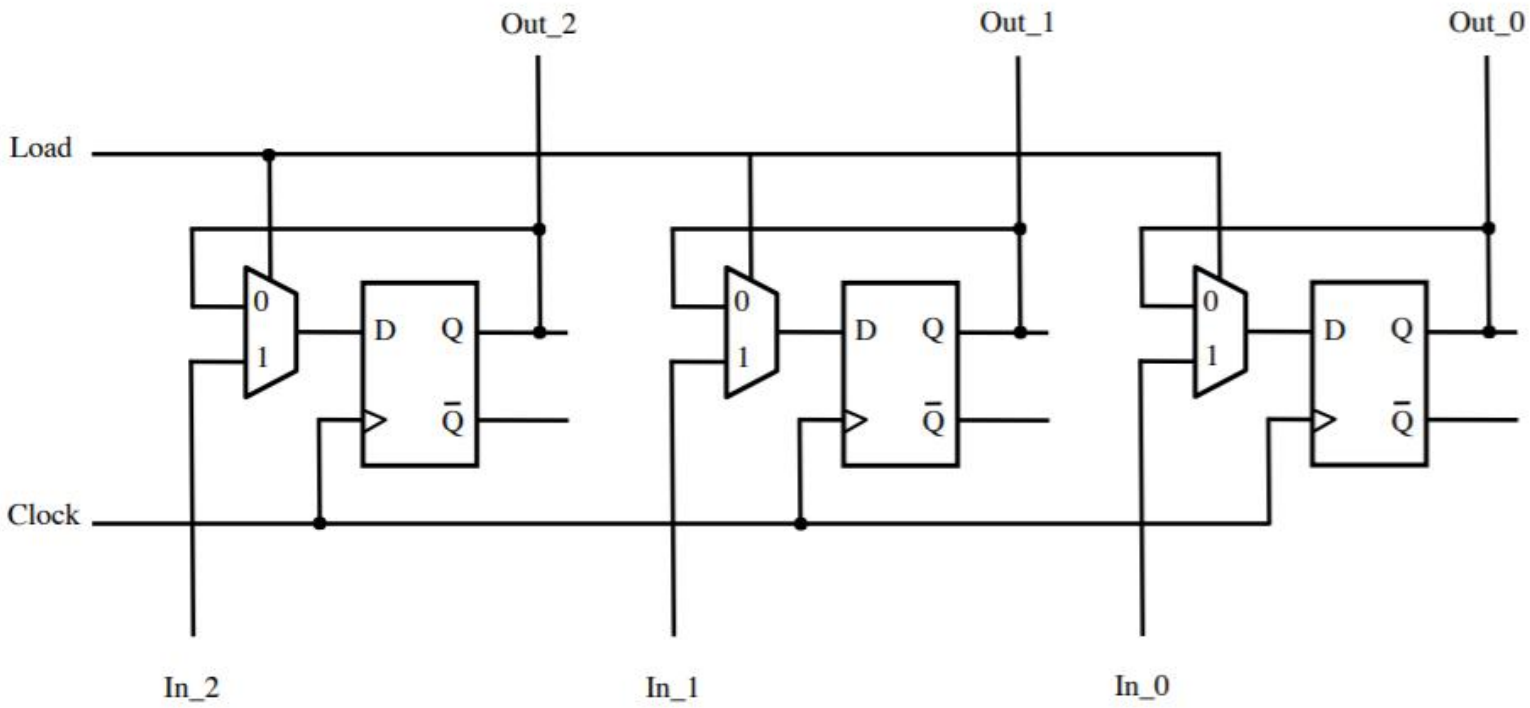


2-Bit Parallel-Access Register

2-bit PIPO Register with "load" control

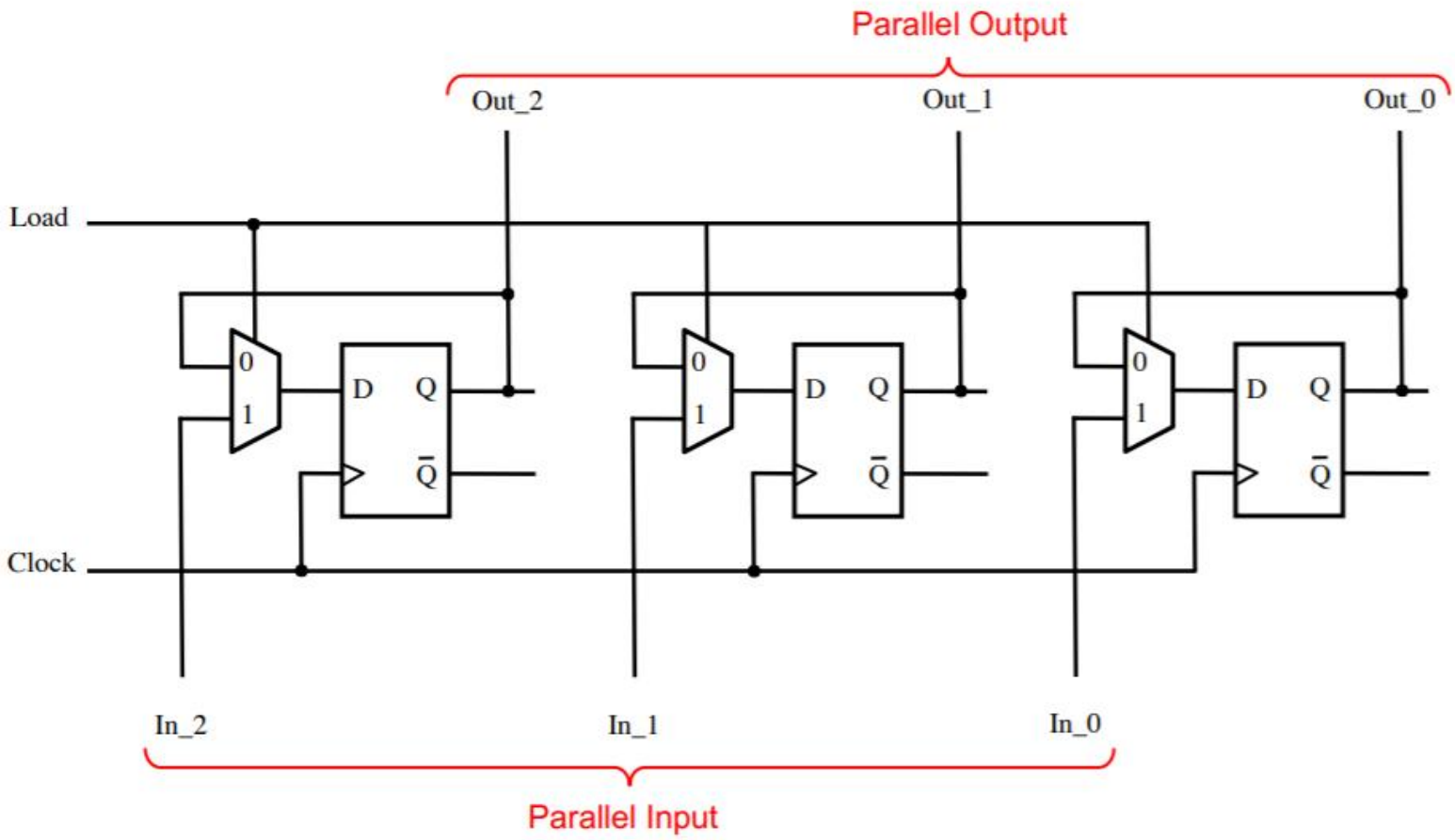


3-Bit Parallel-Access Register

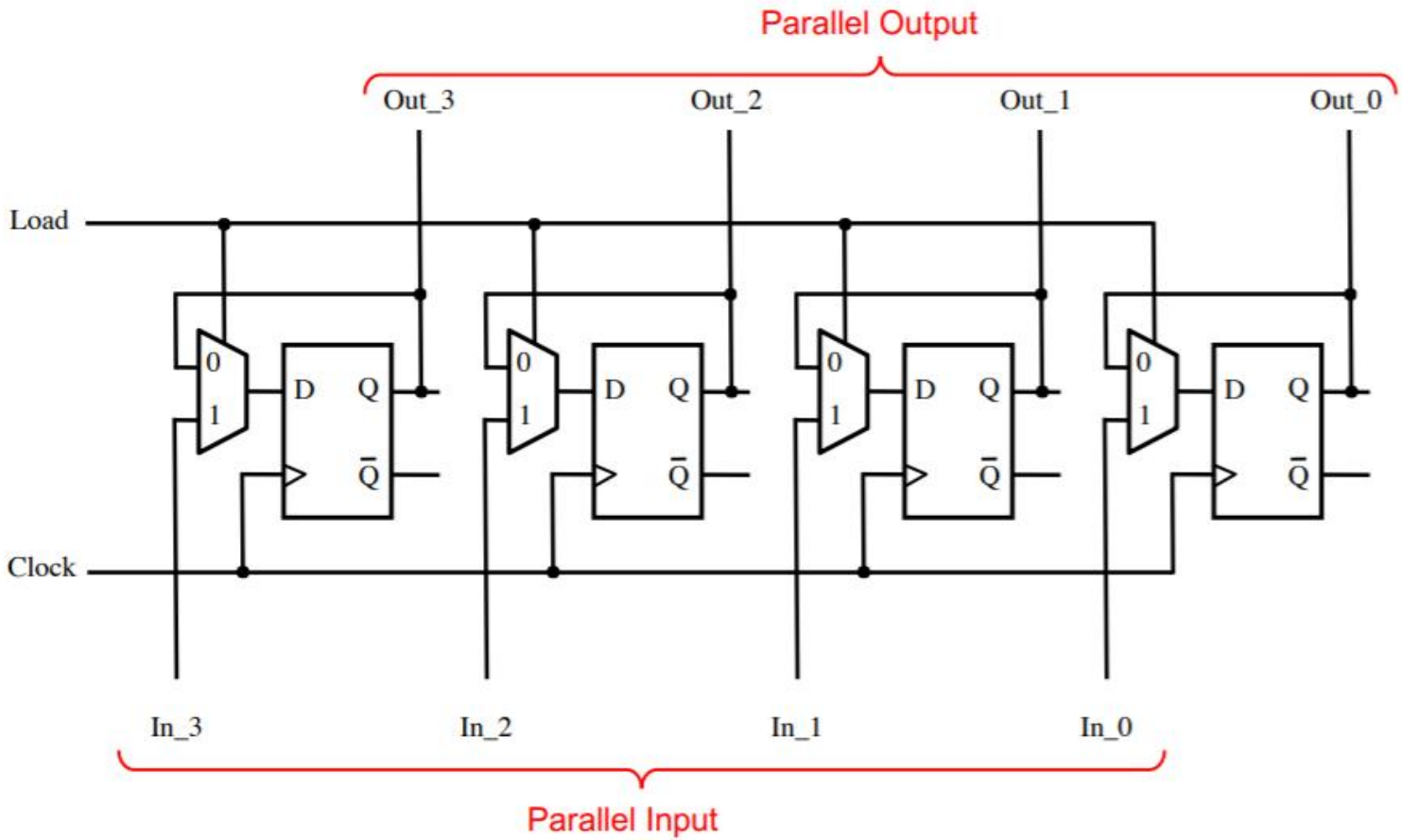


Notice that all flip-flops are on the same clock cycle.

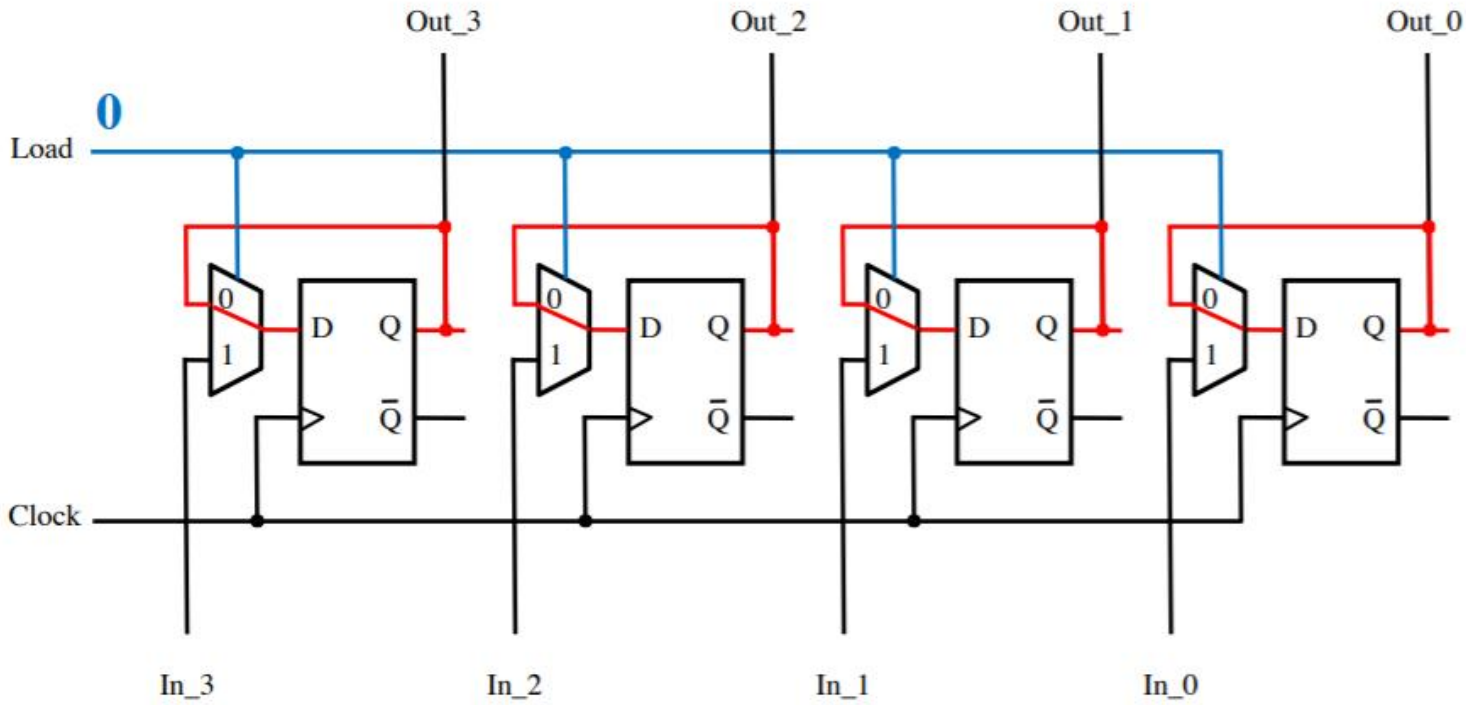
3-Bit Parallel-Access Register



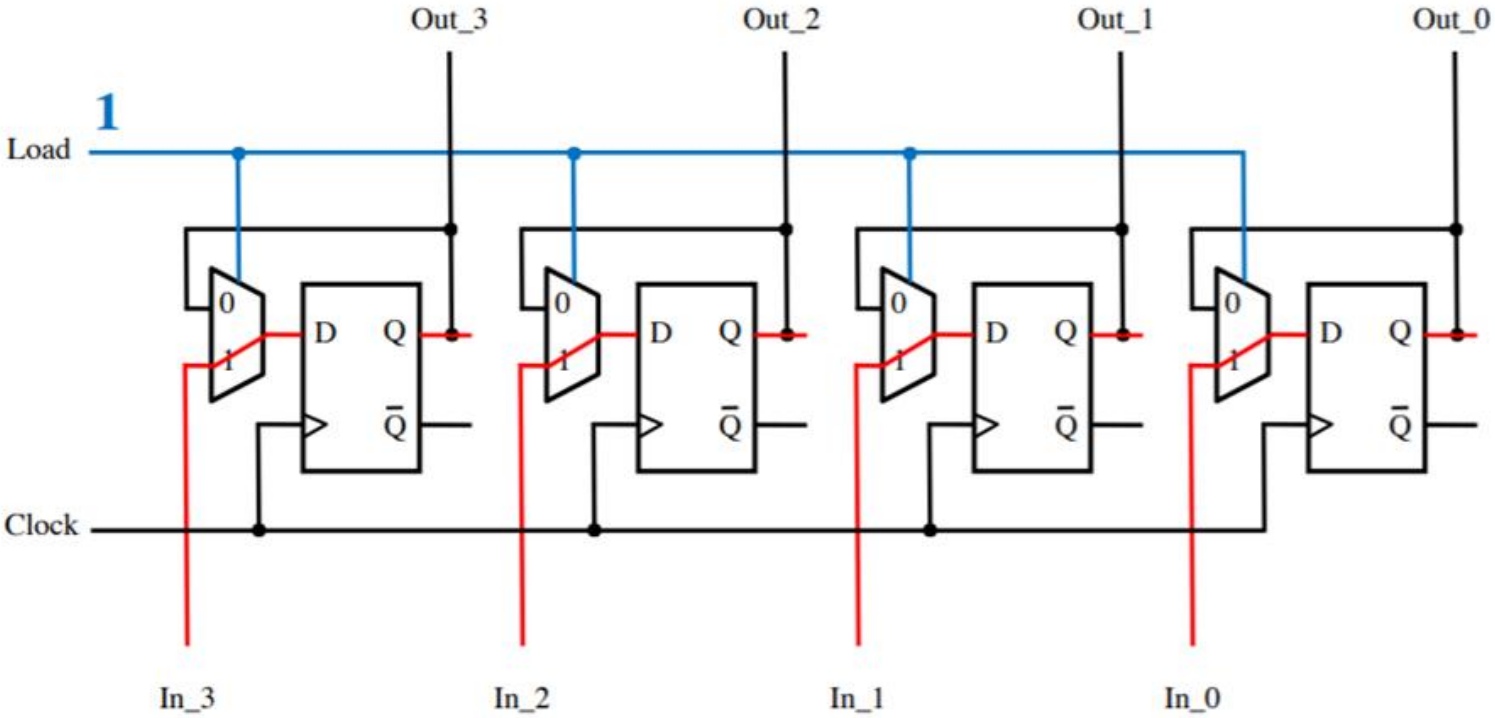
4-Bit Parallel-Access Register



4-Bit Parallel-Access Register

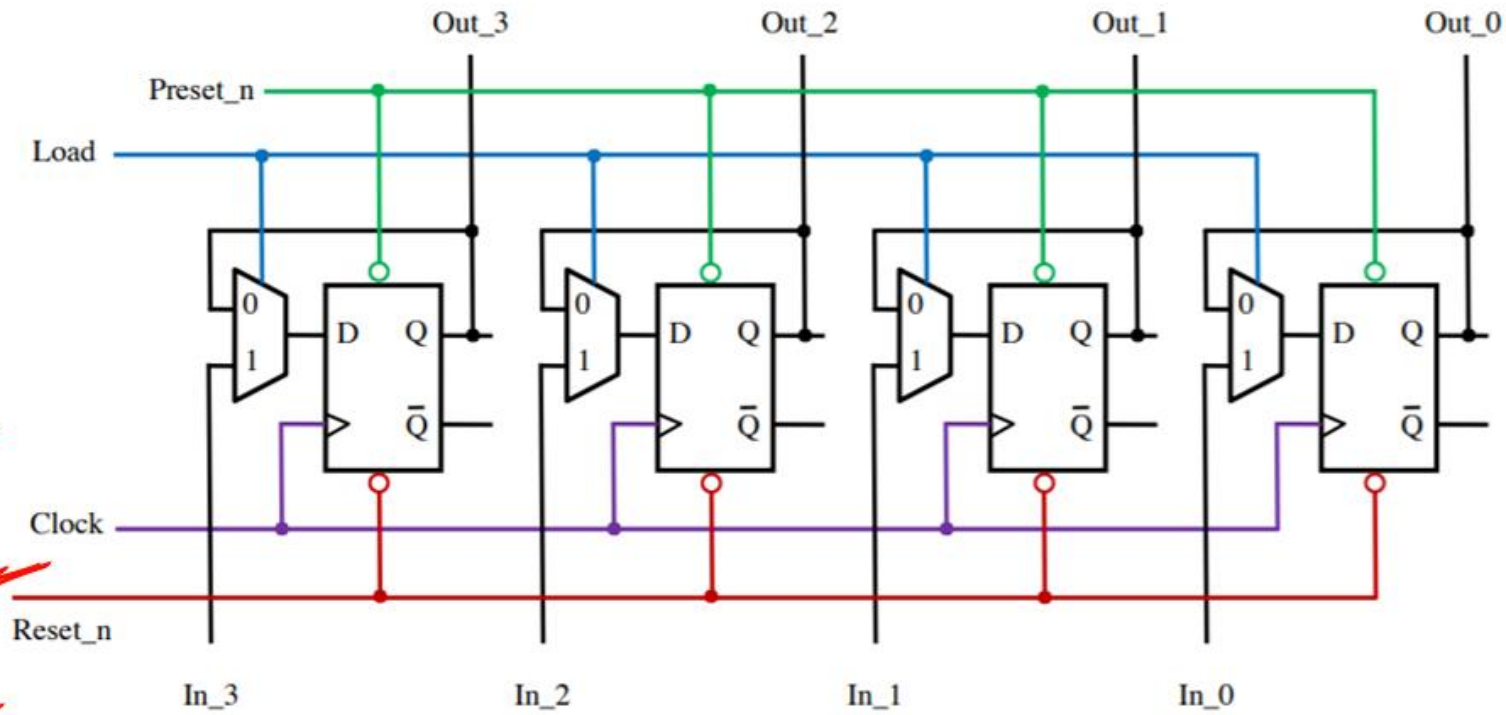


4-Bit Parallel-Access Register



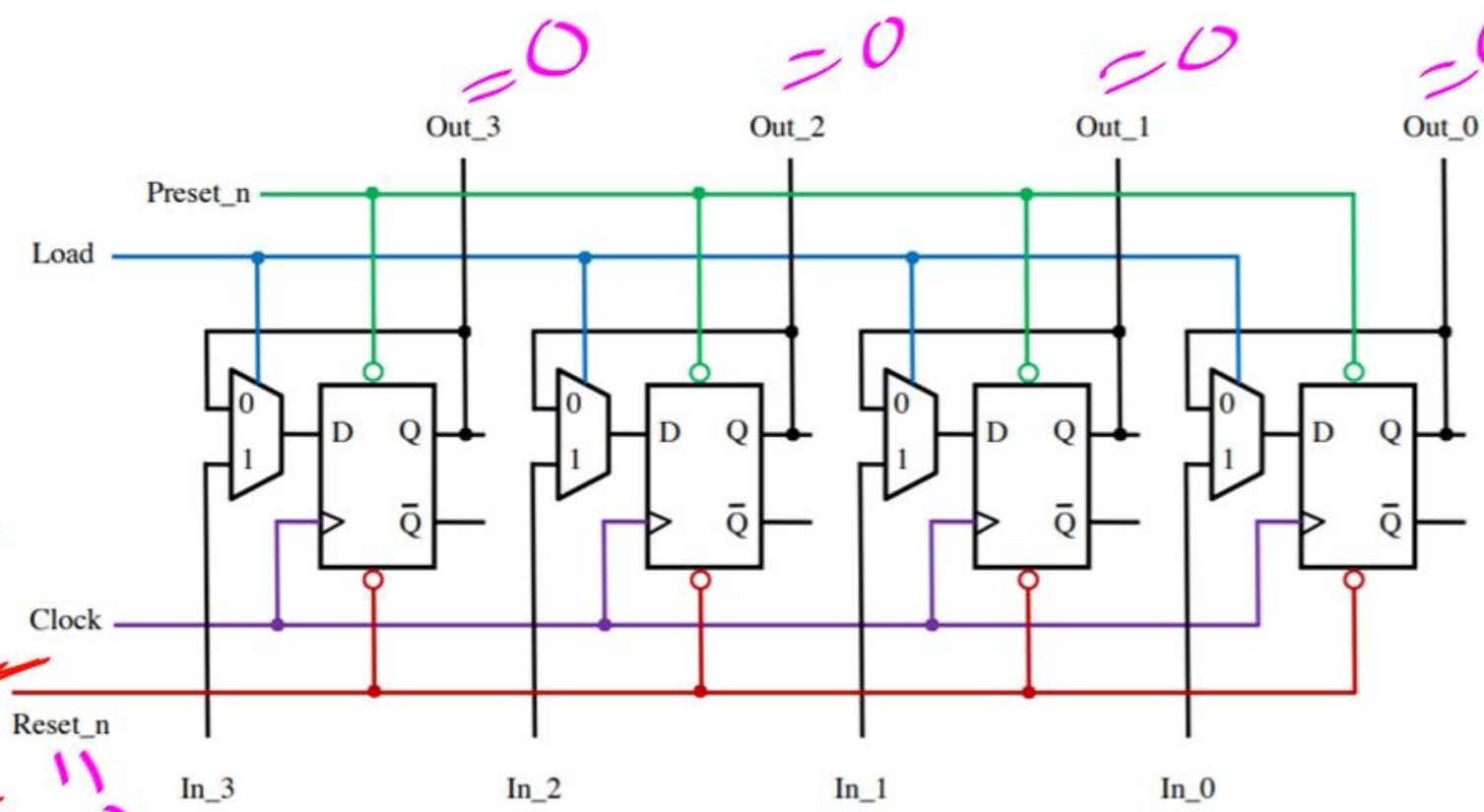
4 bit PIPO Register with load Reset Preset

4-Bit Parallel-Access Register



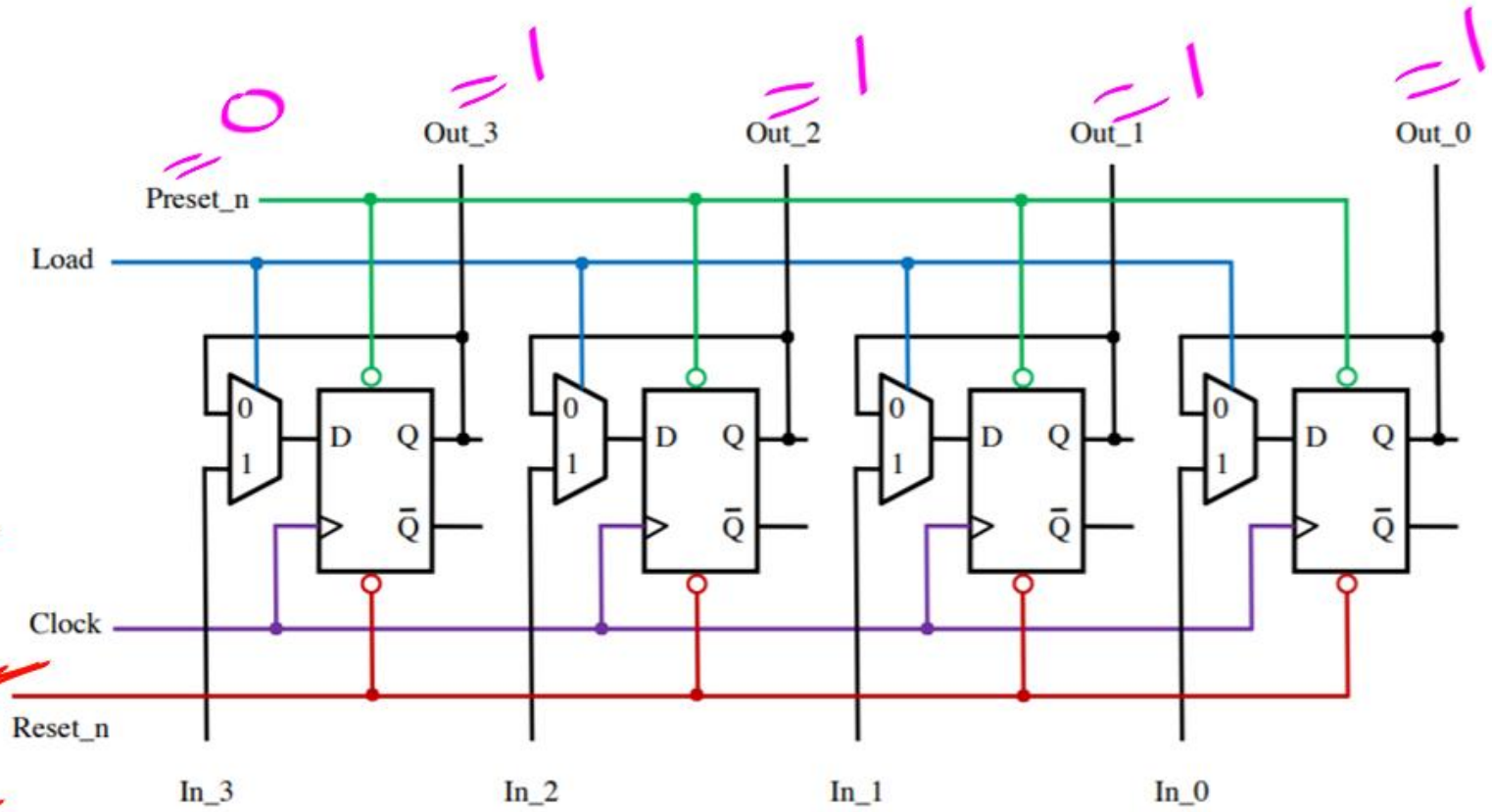
4 bit
PIPO
Register
with
load
Reset
Preset

4-Bit Parallel-Access Register



4 bit
PIPO
Register
with
load
Reset
Preset

4-Bit Parallel-Access Register



Register : (n-bit Register)

1. n - ffs
2. Clear input used to "Reset/Clear" the register content

Usually Active low

Making 0

Clear = 0 ⟶ Register Clears

Clear = 1 ⟶ Registr Normal Operation

(3) "load" Input:

load = 1 ; new Input Data stored

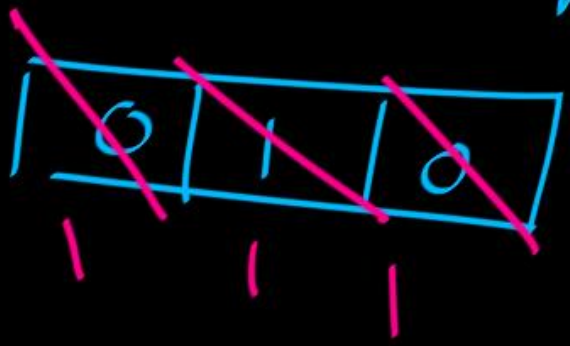
load = 0 ; Previous Data Retained

To Control when to store/Retain Data

(4) Preset :

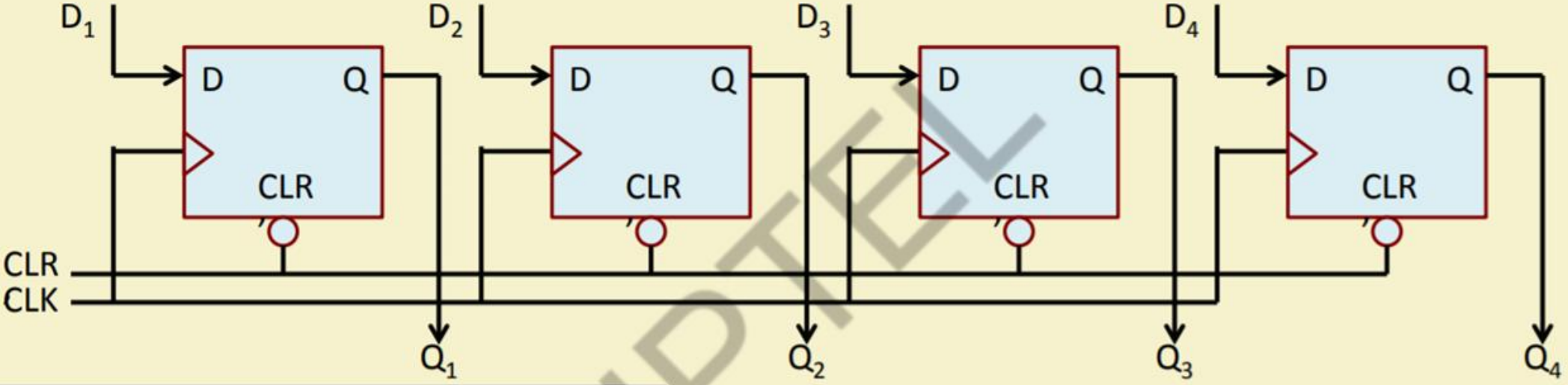
↳ making everything 1.

↳ usually Active low.



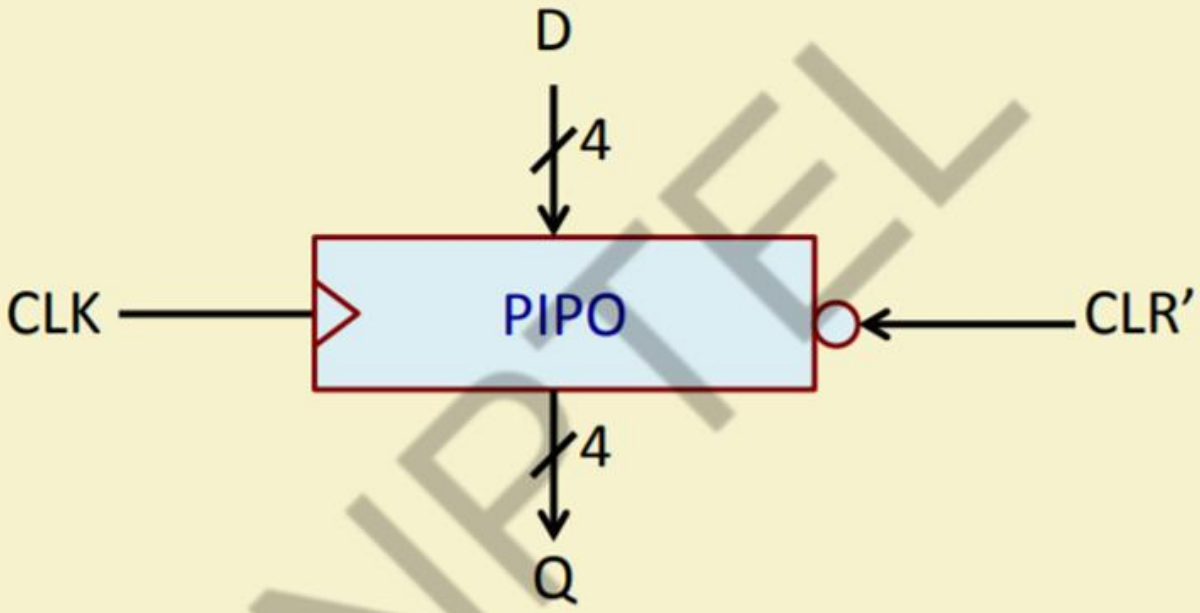
Symbol for Active low : ——○

Basic PIPO Register



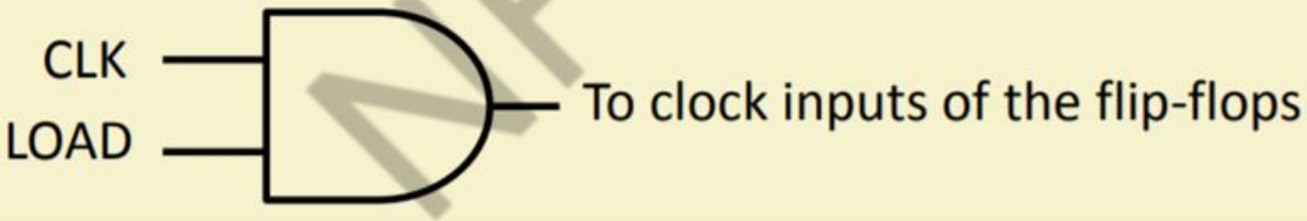
When the active clock edge arrives, input word $D_1D_2D_3D_4$ gets stored in the register, and available on the outputs $Q_1Q_2Q_3Q_4$.

Symbolic Representation (using vector notation)



Addition of LOAD signal

- In practice, the clock is coming continuously, and there is a separate signal LOAD that specifies when the register is to be loaded with new data.
- Two possible solutions:
  - *a) Use a gated clock:*
    - Not a good solution, as gating the clock with another signal can cause timing problems.



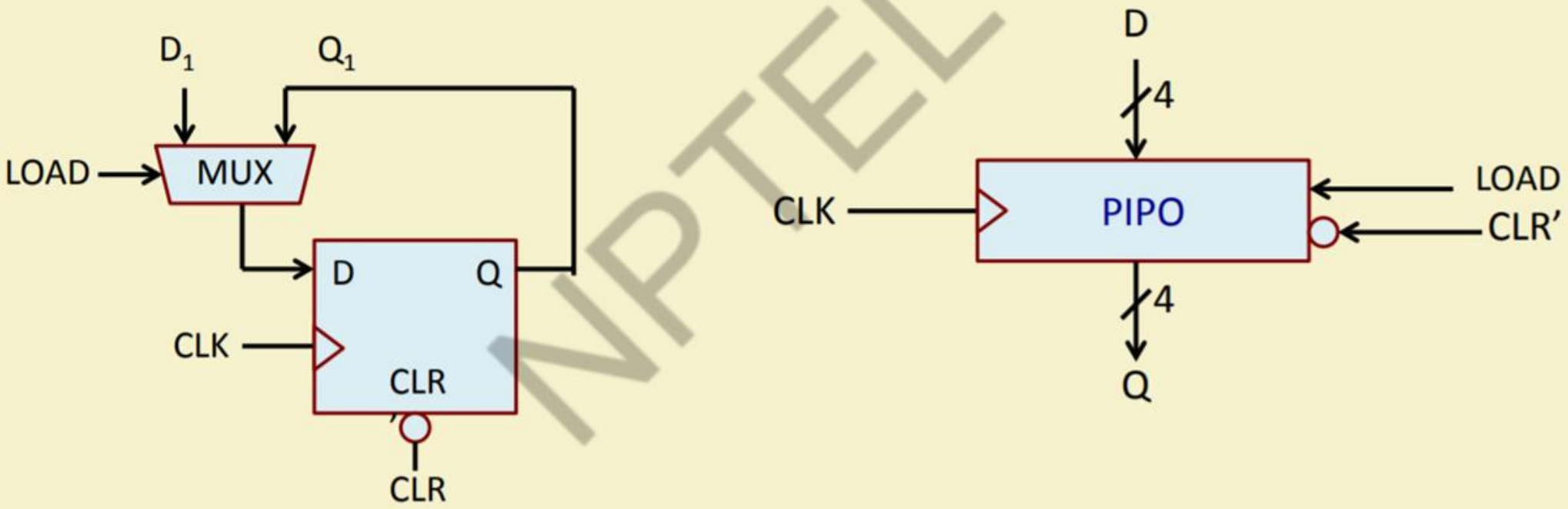
This register has a load signal that is ANDed with the clock. When Load = 0, the register is not clocked, and it holds its present value.

When it is time to load data into the register, Load is set to 1 for one clock period. When Load = 1, the clock signal (Clk) is transmitted to the flip-flop clock inputs and the data applied to the D inputs will be loaded into the flip-flops on the falling edge of the clock. For example, if the Q outputs are 0000 ($Q_3 = Q_2 = Q_1 = Q_0 = 0$) and the data inputs are 1101 ($D_3 = 1, D_2 = 1, D_1 = 0$ and $D_0 = 1$), after the falling edge of the clock Q will change from 0000 to 1101 as indicated. (The notation $0 \rightarrow 1$ at the flip-flop outputs indicates a change from 0 to 1.)

The flip-flops in the register have asynchronous clear inputs that are connected to a common clear signal, ClrN. The bubble at the clear inputs indicates that a logic 0 is required to clear the flip-flops. ClrN is normally 1, and if it is changed momentarily to 0, the Q outputs of all four flip-flops will become 0.

b) Separate out CLK and LOAD using a multiplexer circuit.

- Better and recommended solution.
- Each flip-flop in the register gets replaced by:



Shift Register

Shift Registers:

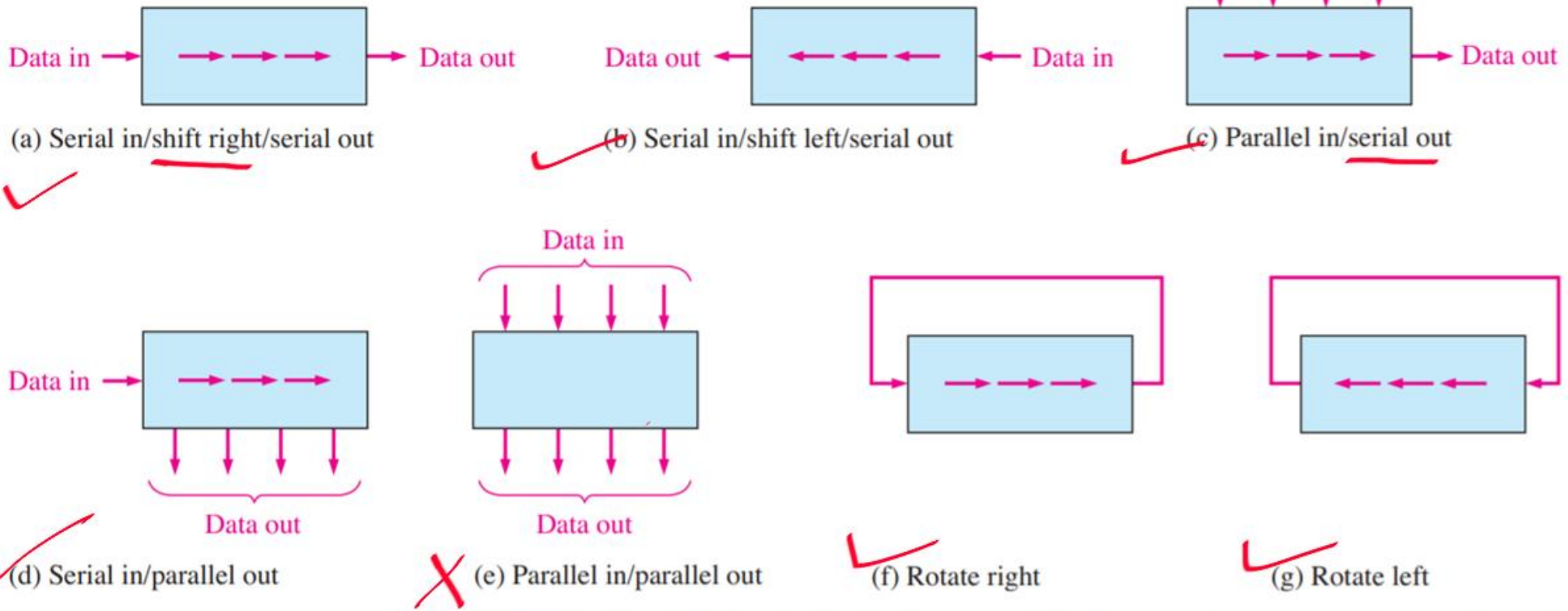


FIGURE 8–2 Basic data movement in registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

Shift Registers:

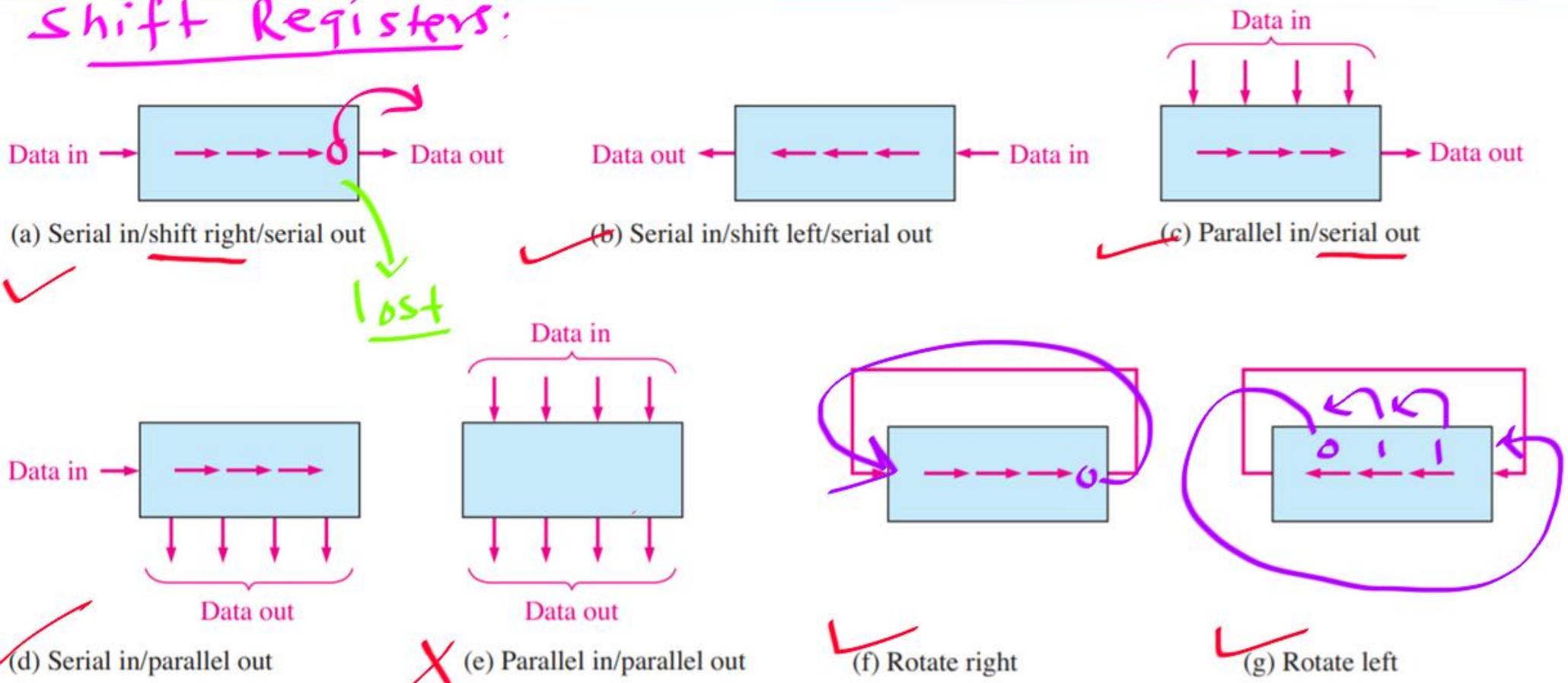
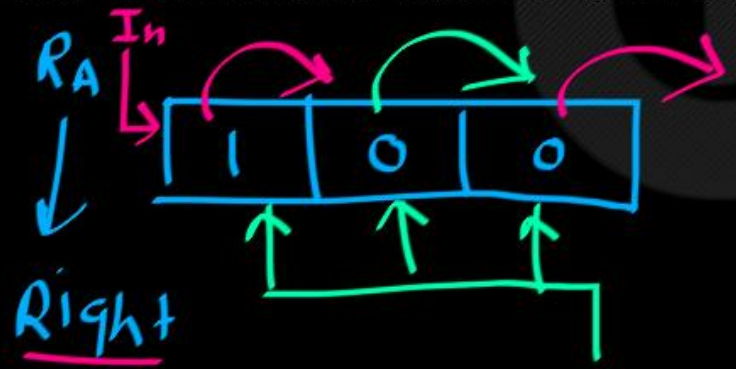
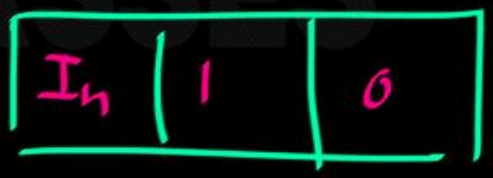


FIGURE 8–2 Basic data movement in registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

Shift Registers

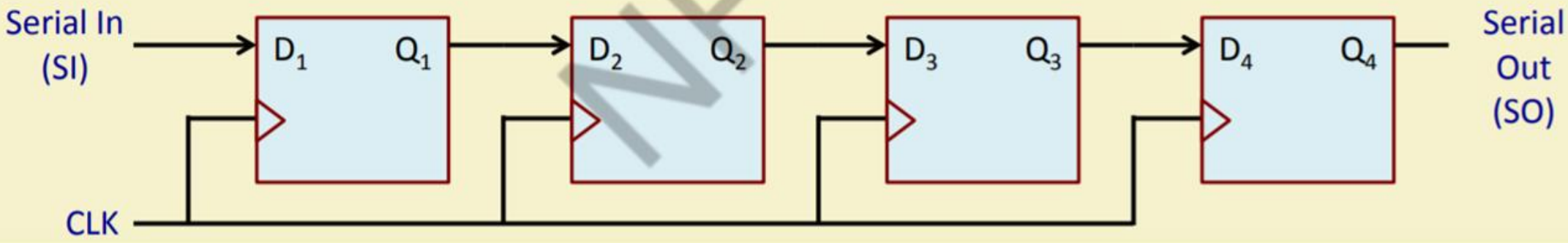
A shift register is a register in which binary data can be stored, and this data can be shifted to the left or right when a shift signal is applied. Bits shifted out one end of the register may be lost, or if the shift register is of cyclic type, bits shifted out one end are shifted back in the other end.



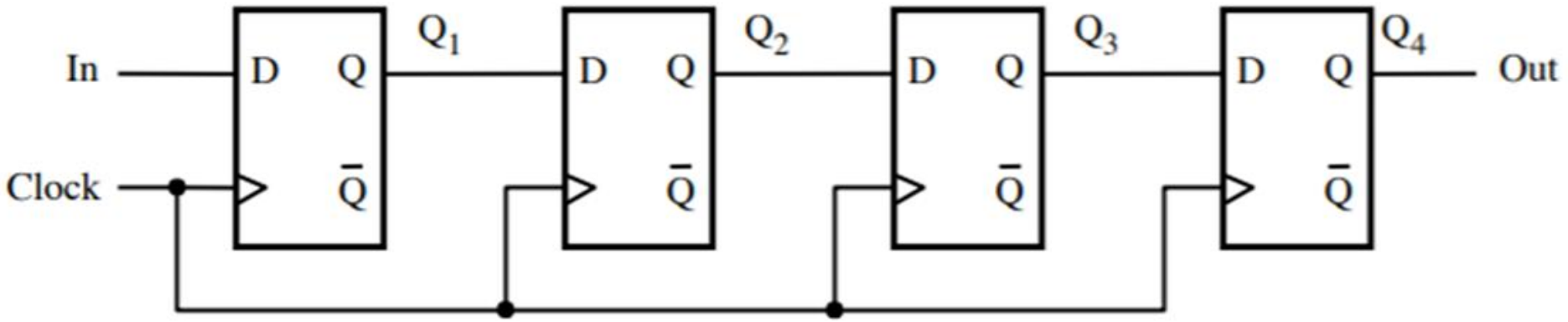


Shift Register

- A shift register is a register in which the binary data can be stored, and the data can be shifted to the left (or right) when a shift signal is applied.
  - New bit gets shifted in; bit shifted out typically get lost.
- Can be constructed simply by connecting D, SR or JK flip-flops in cascade.
- A 4-bit shift register is shown below:

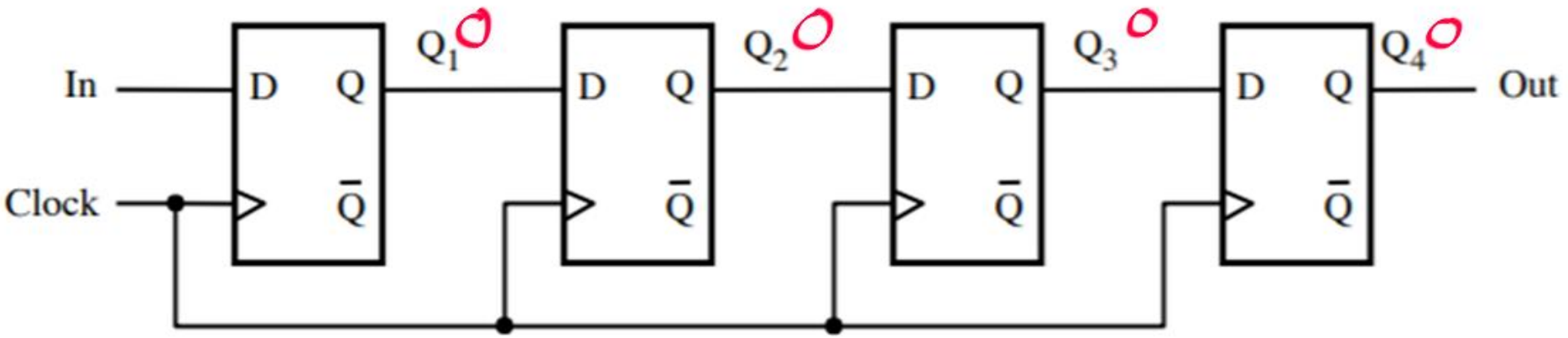


A simple shift register

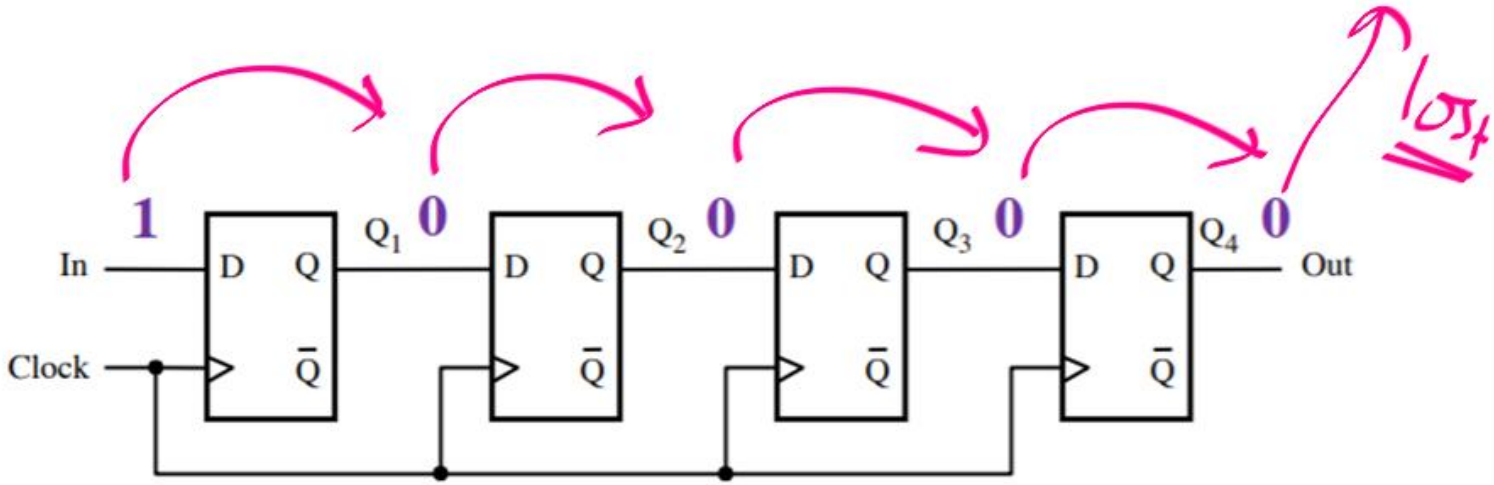


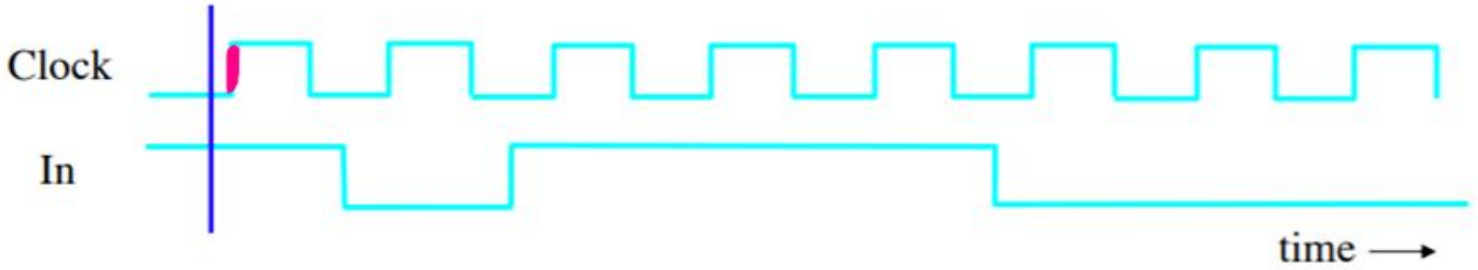
A simple shift register

Initially, register holds Data : 0000

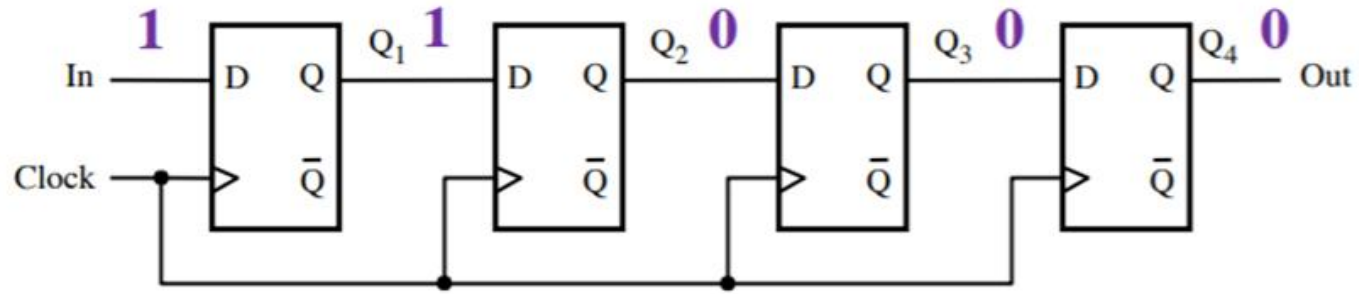


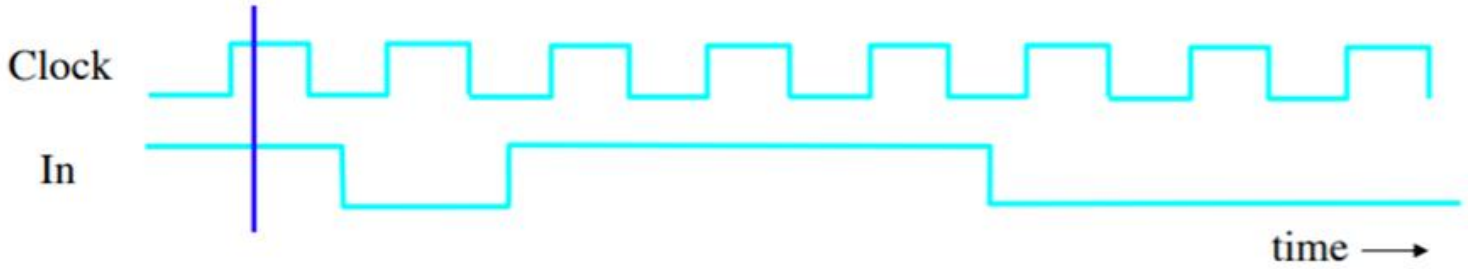
Shift Register Simulation



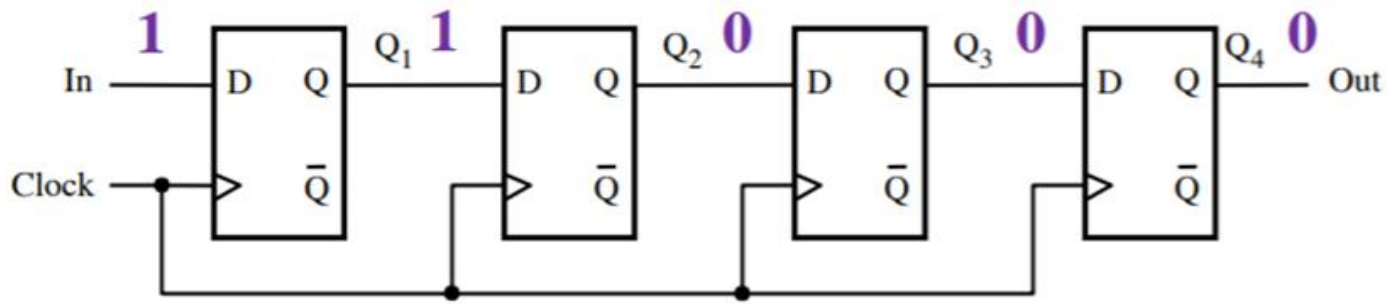


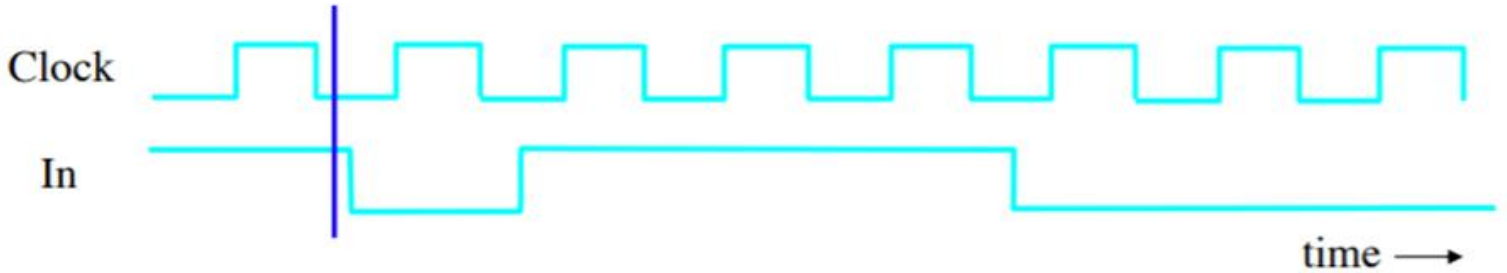
Shift Register Simulation





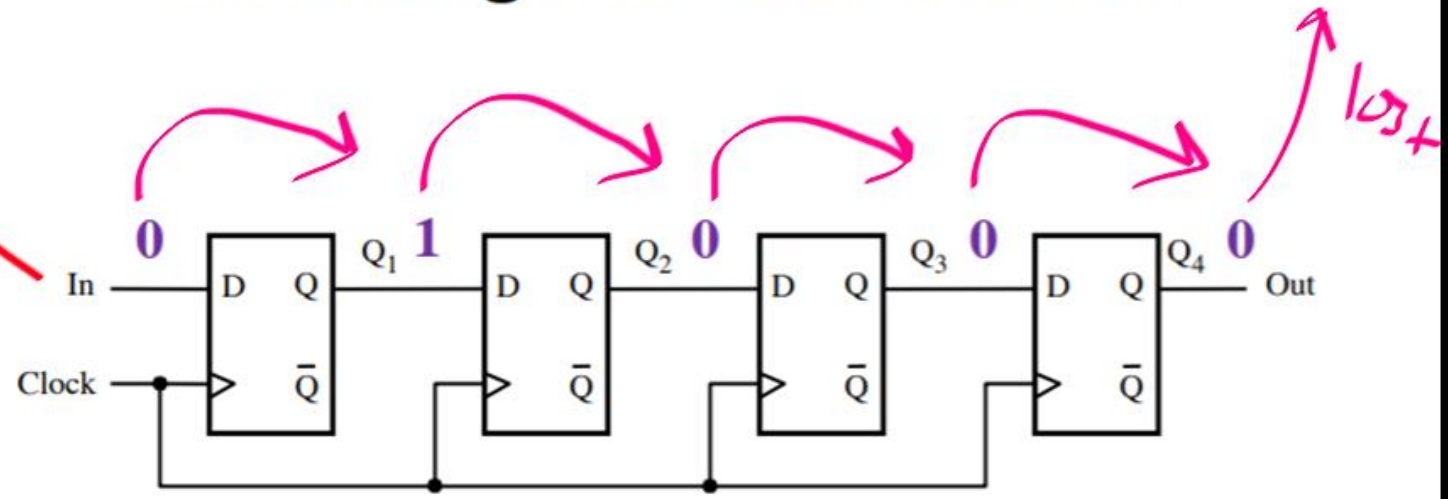
Shift Register Simulation

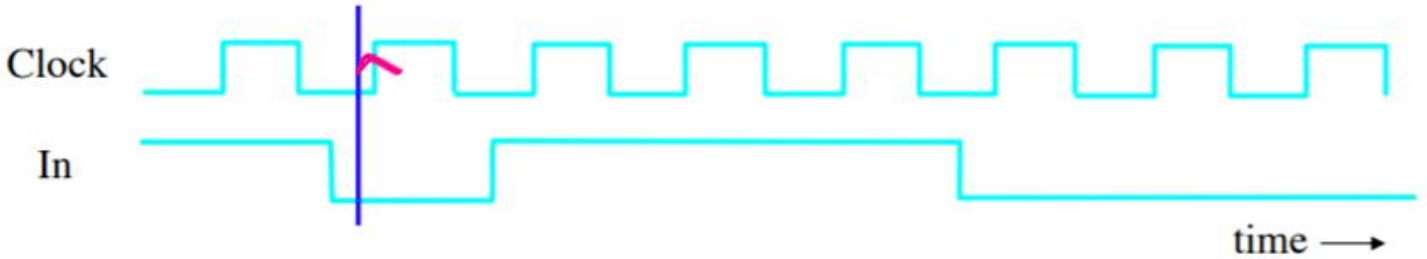




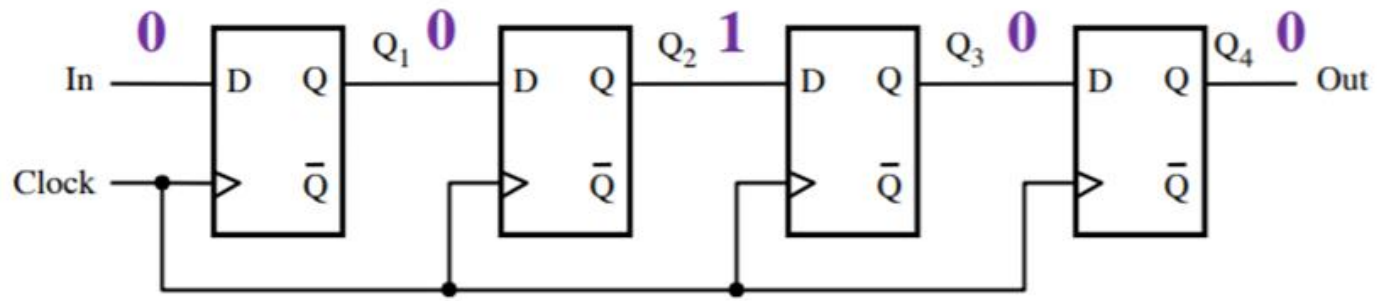
Shift Register Simulation

Supplied from External Source

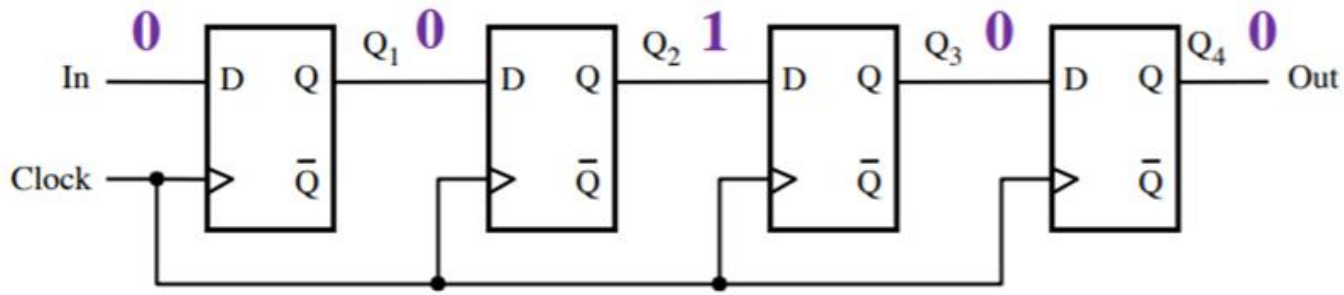


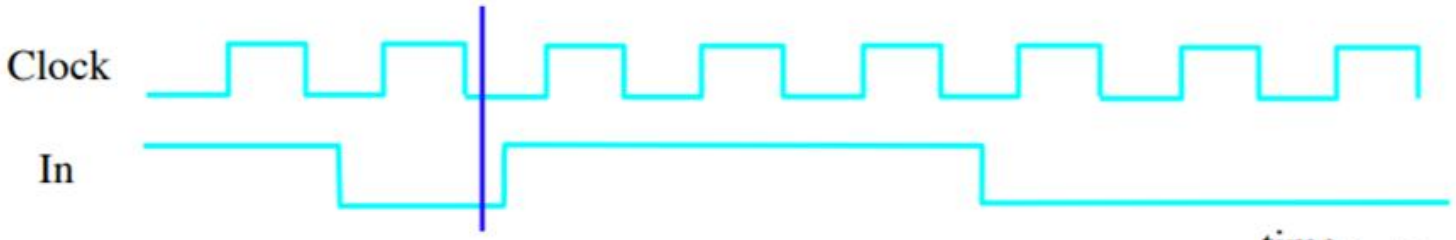


Shift Register Simulation

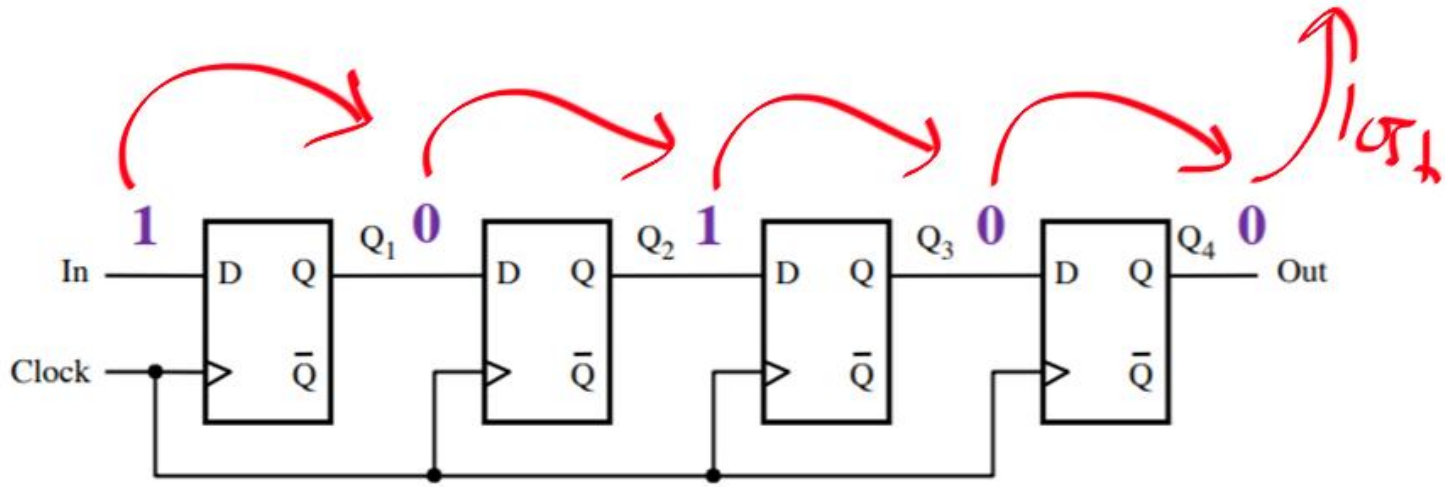


Shift Register Simulation

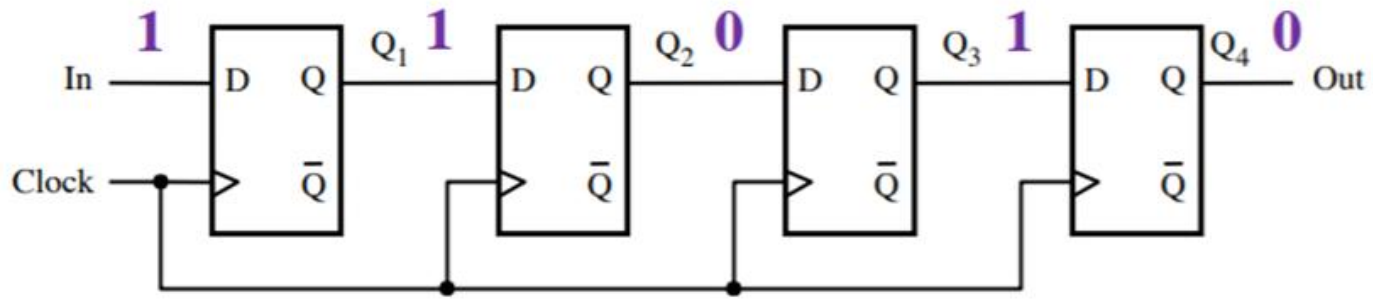


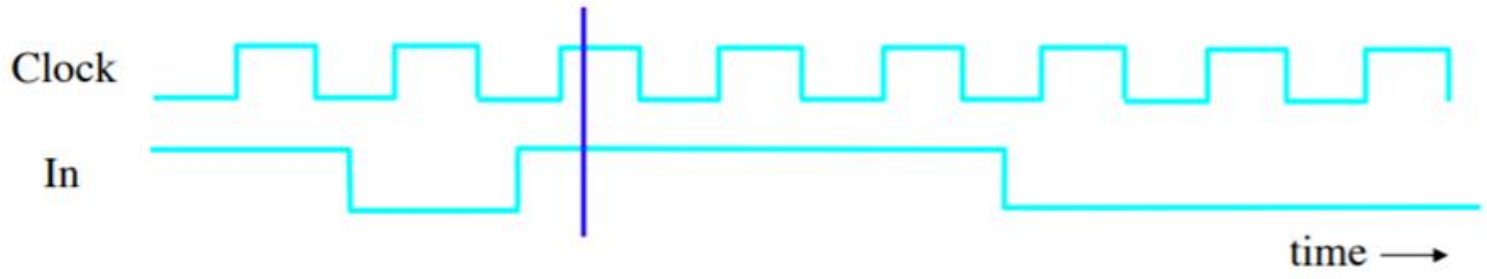


Shift Register Simulation

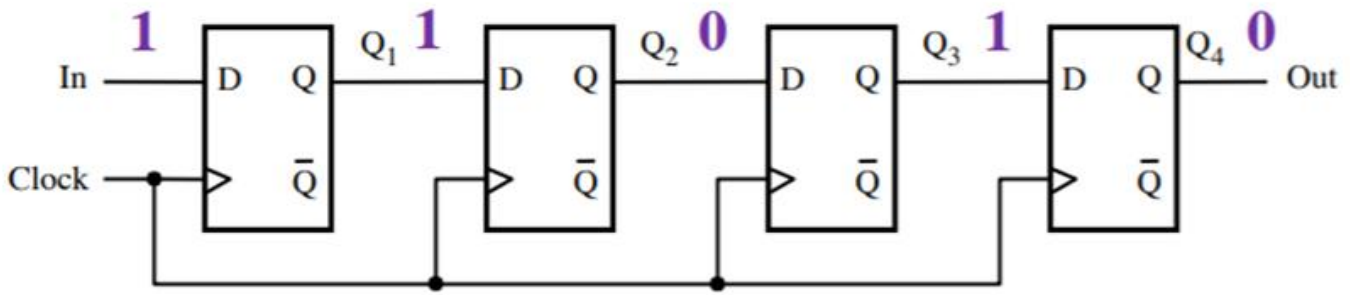


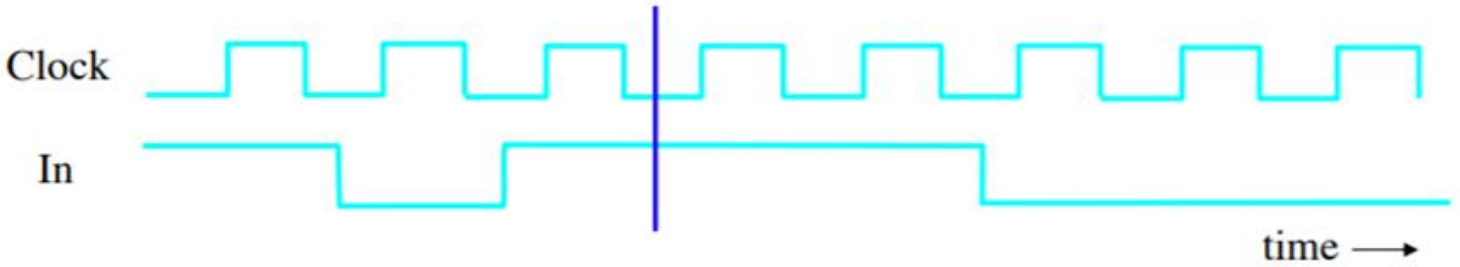
Shift Register Simulation



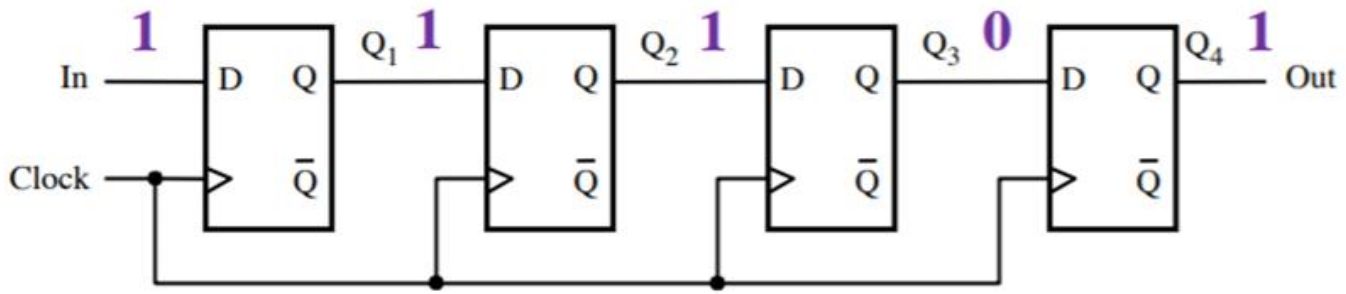


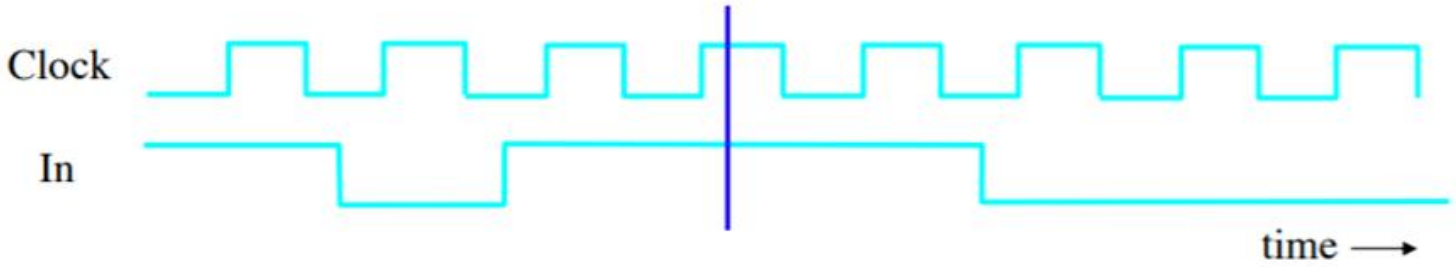
Shift Register Simulation



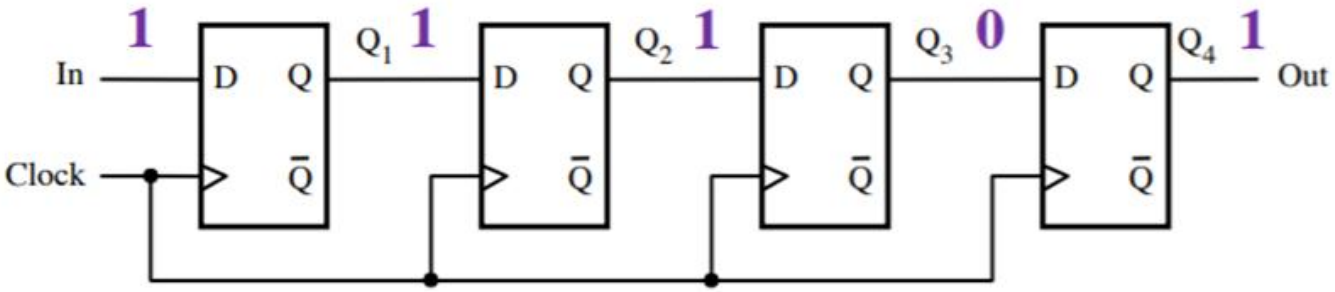


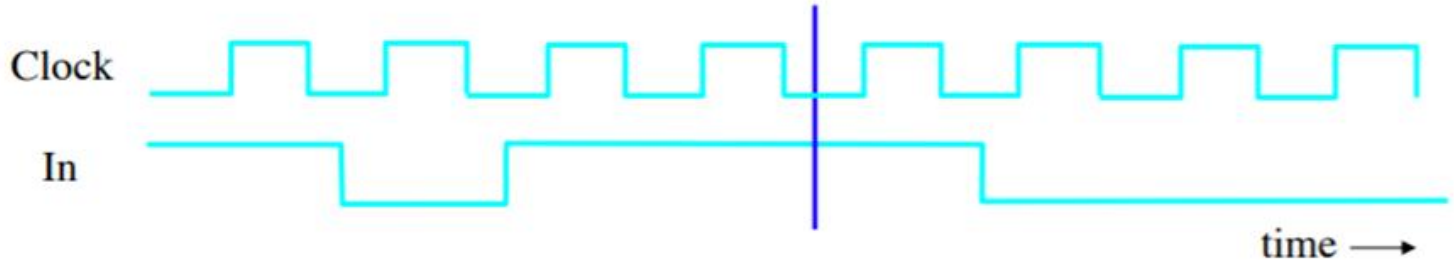
Shift Register Simulation



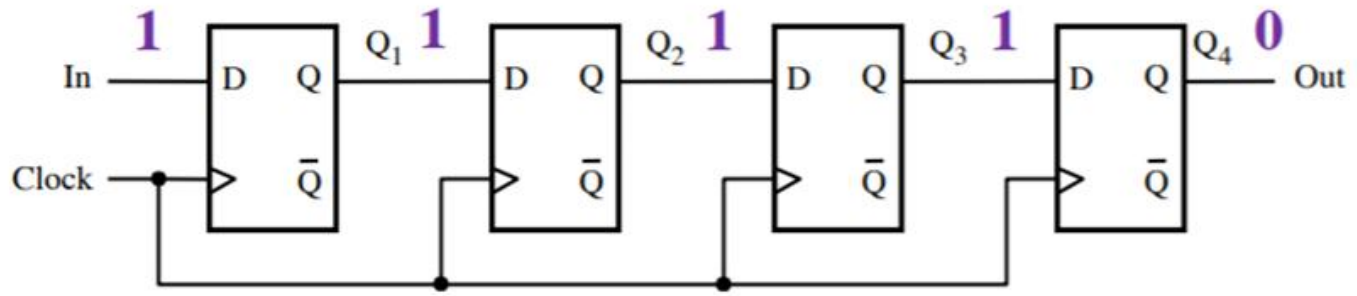


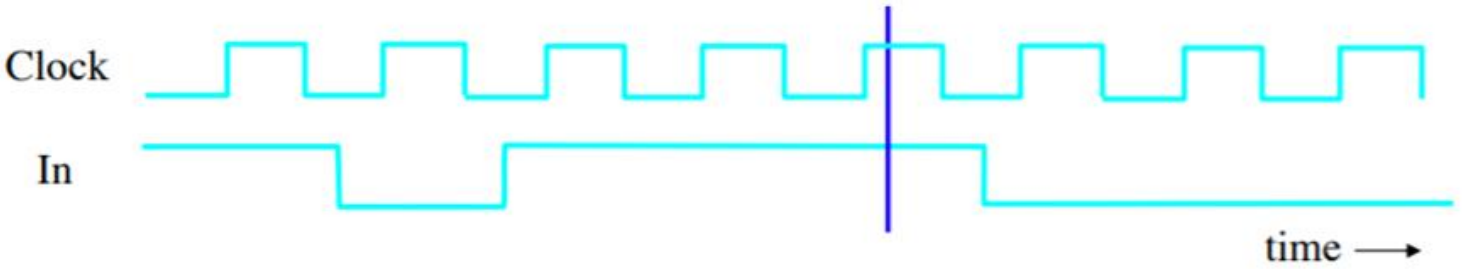
Shift Register Simulation



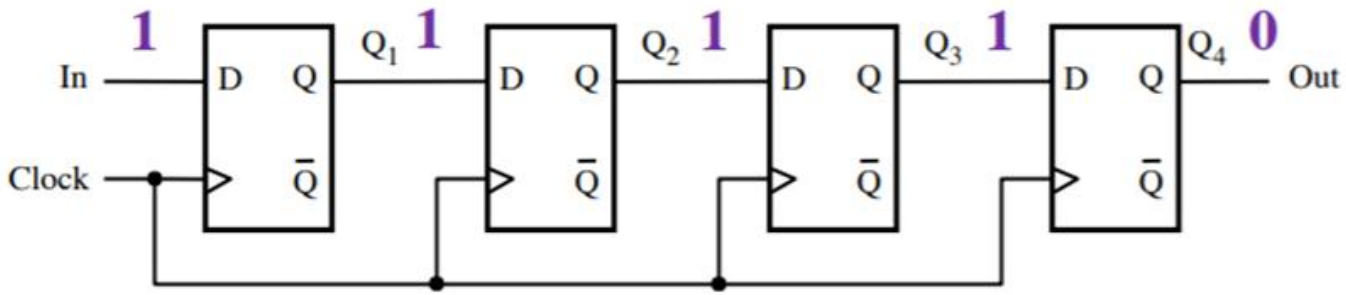


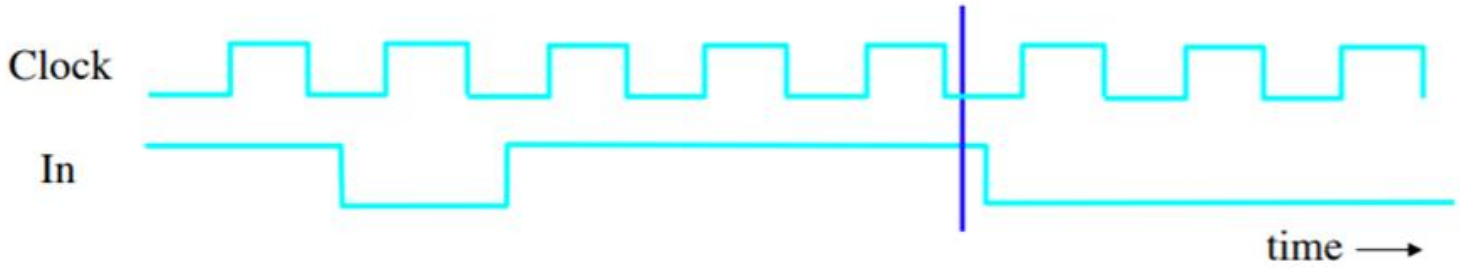
Shift Register Simulation



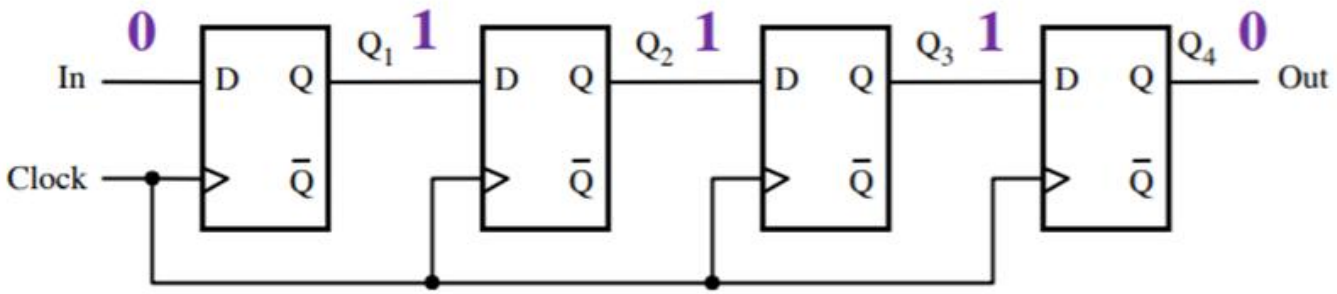


Shift Register Simulation

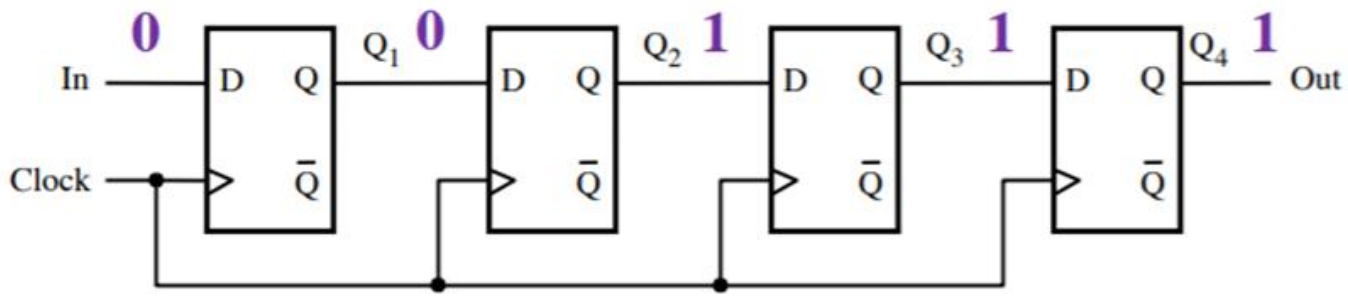




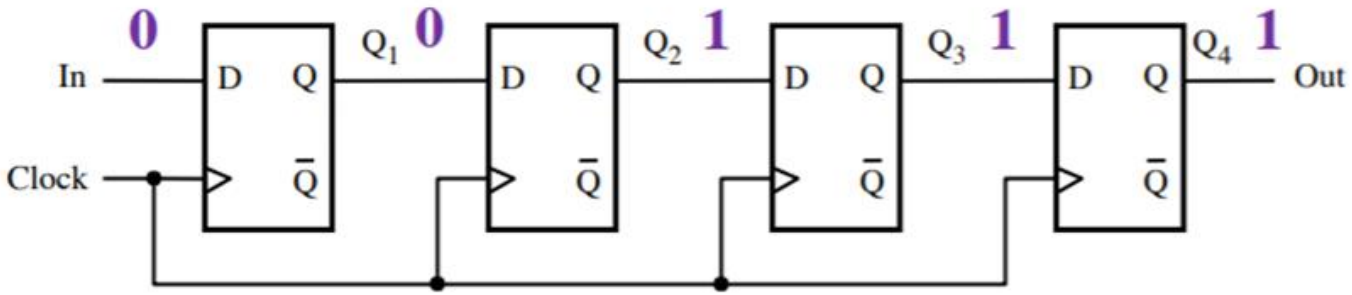
Shift Register Simulation



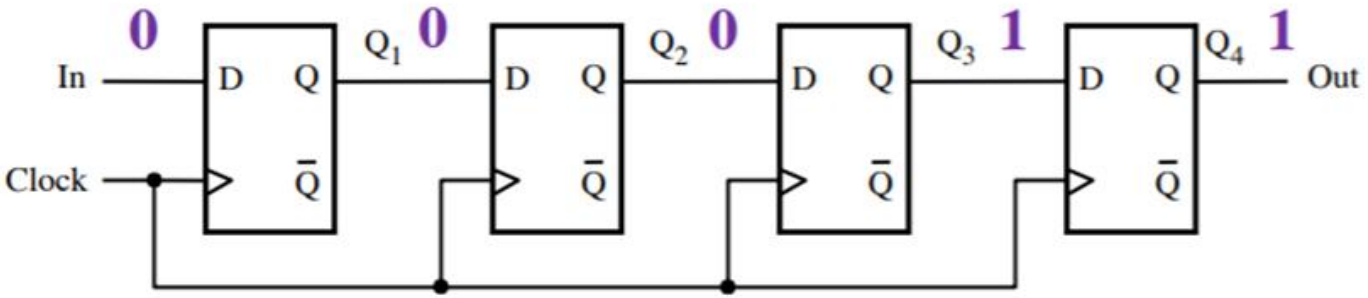
Shift Register Simulation



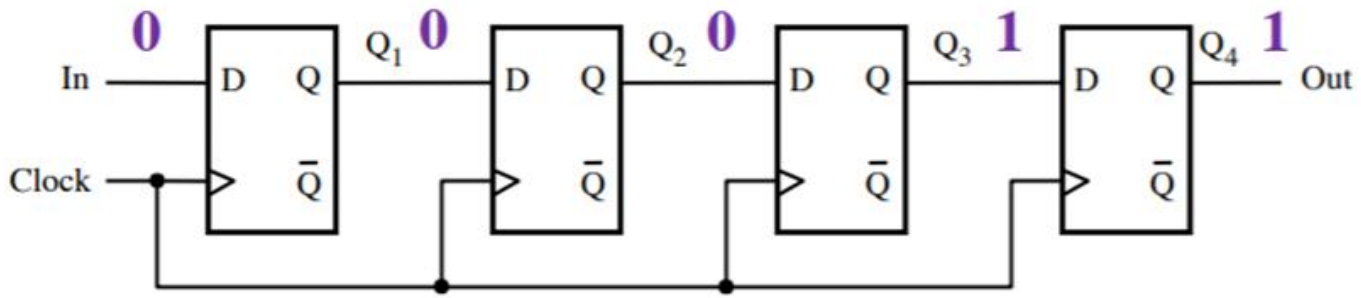
Shift Register Simulation

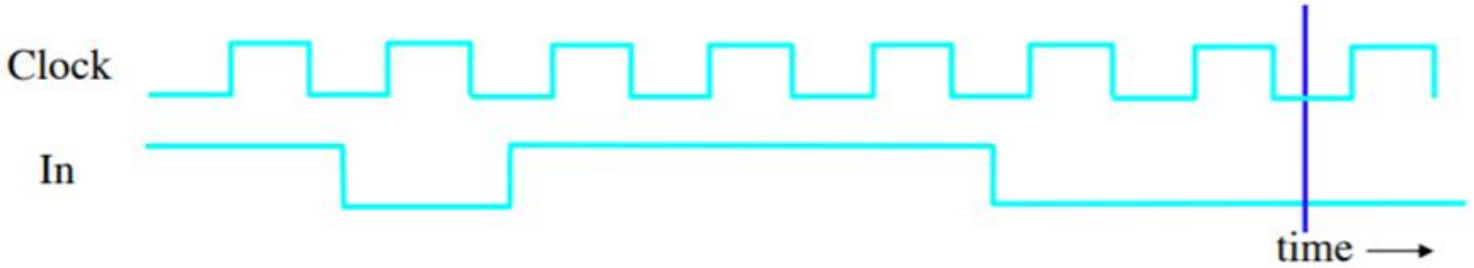


Shift Register Simulation

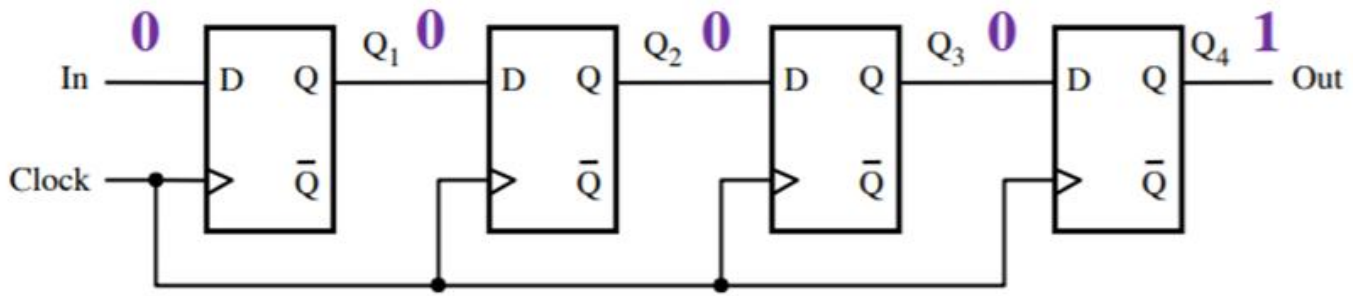


Shift Register Simulation

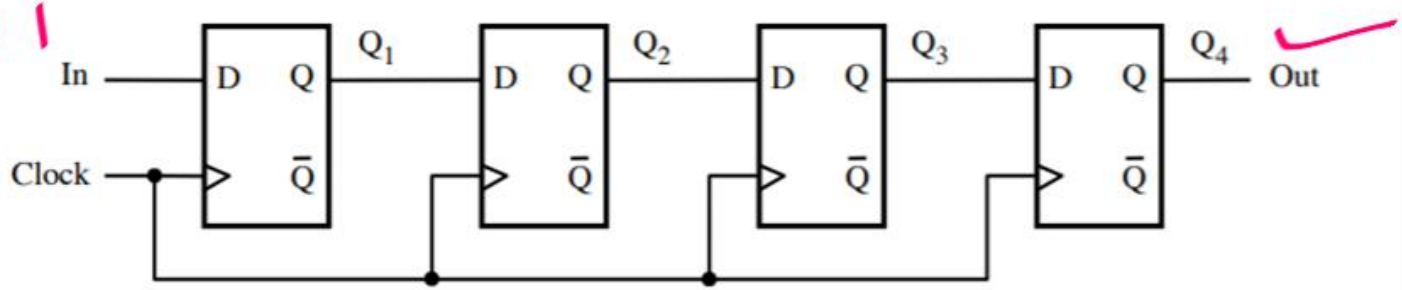




Shift Register Simulation



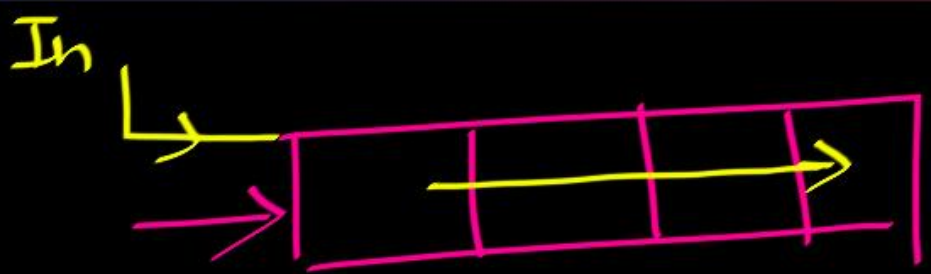
A simple shift register



(a) Circuit

time

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



	In	Q_1	Q_2	Q_3	Q_4	
t_0	1	0	0	0	0	$\downarrow$
t_1	0	1	0	0	0	$\downarrow$
t_2	1	0	1	0	0	
t_3	1	1	0	1	0	

Shift Right Register

- SISO (serial In, serial out) shift Right Register

Serial In/Serial Out Shift Registers

The serial in/serial out shift register accepts data serially—that is, one bit at a time on a single line. It produces the stored information on its output also in serial form. Let's first look at the serial entry of data into a typical shift register. Figure 8–3 shows a 4-bit device implemented with D flip-flops. With four stages, this register can store up to four bits of data.

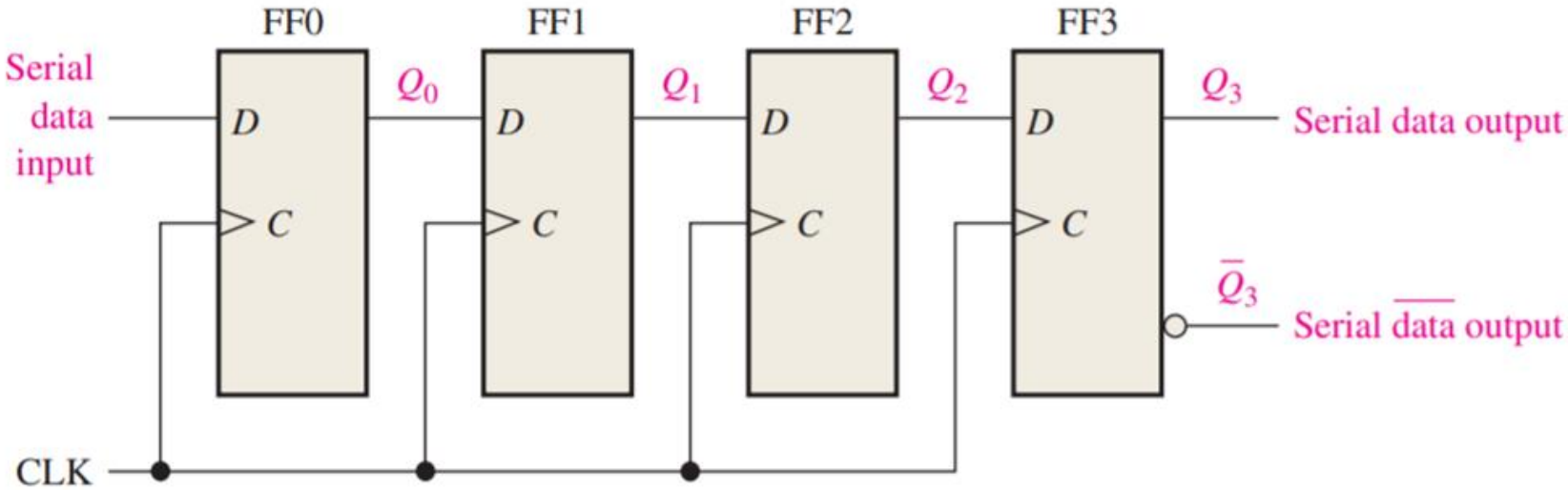
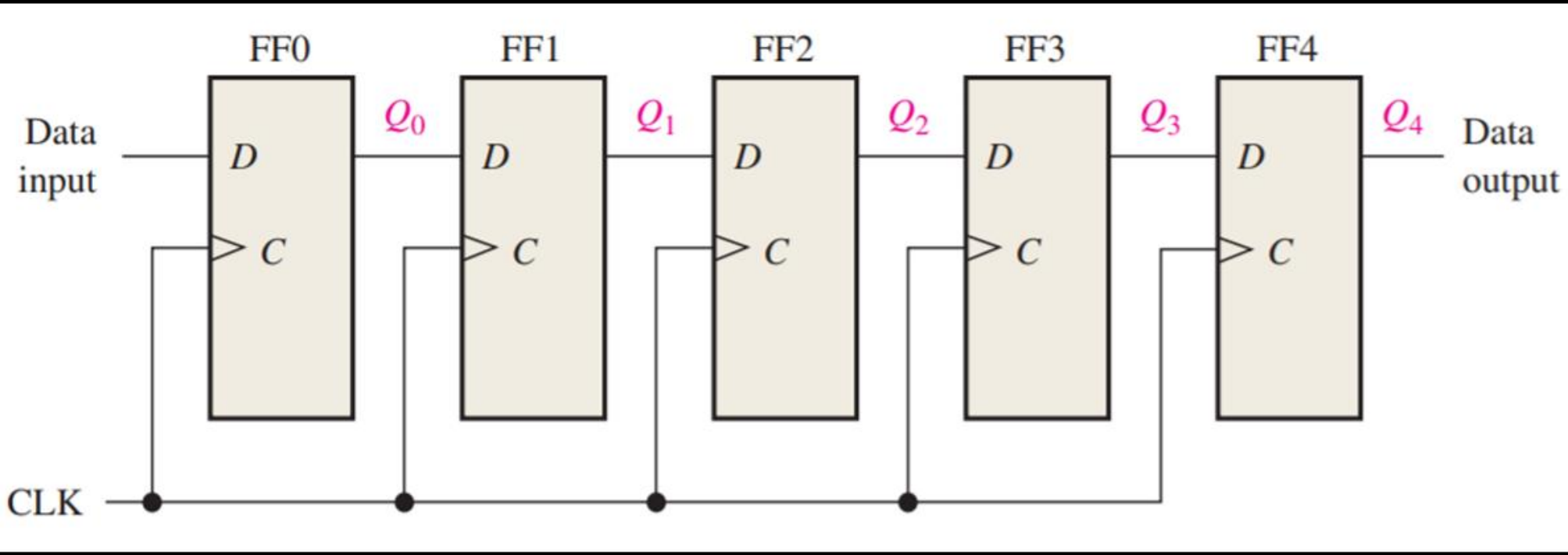
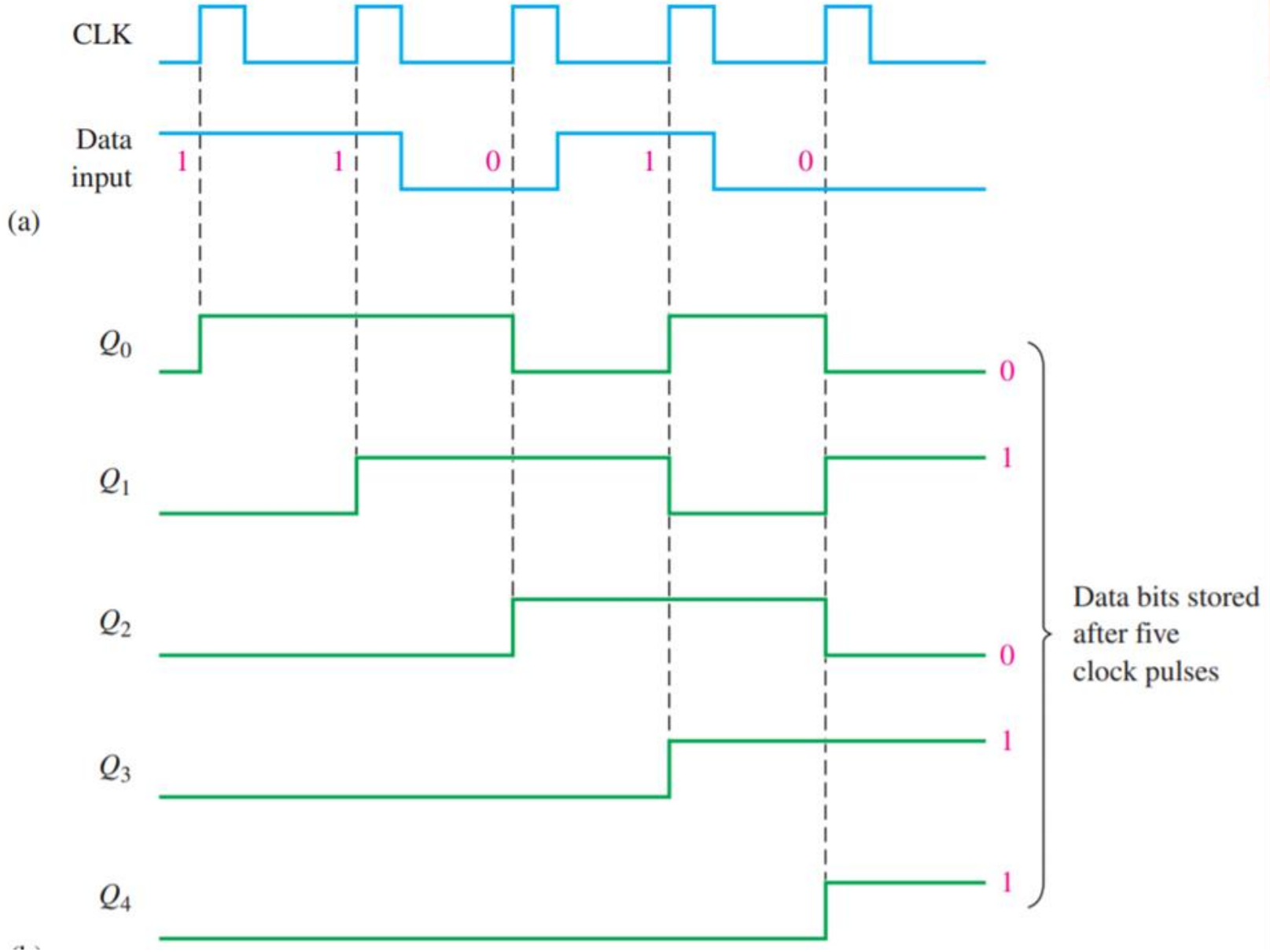


FIGURE 8–3 Serial in/serial out shift register.



5-bit SISO Shift Right Register


CLK
Data input
1
1
0
1
0
(a)
Q_0
Q_1
Q_2
Q_3
Q_4
0
1
0
1
1
Data bits stored after five clock pulses

SISO Shift-Right Register

Block Diagram:

A traditional logic block symbol for an 8-bit serial in/serial out shift register is shown in Figure 8–5. The "SRG 8" designation indicates a shift register (SRG) with an 8-bit capacity.

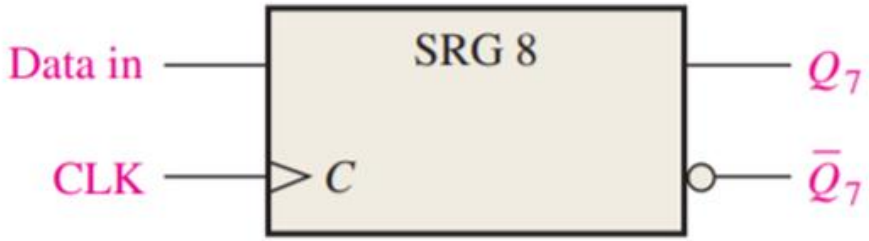


FIGURE 8–5 Logic symbol for an 8-bit serial in/serial out shift register.

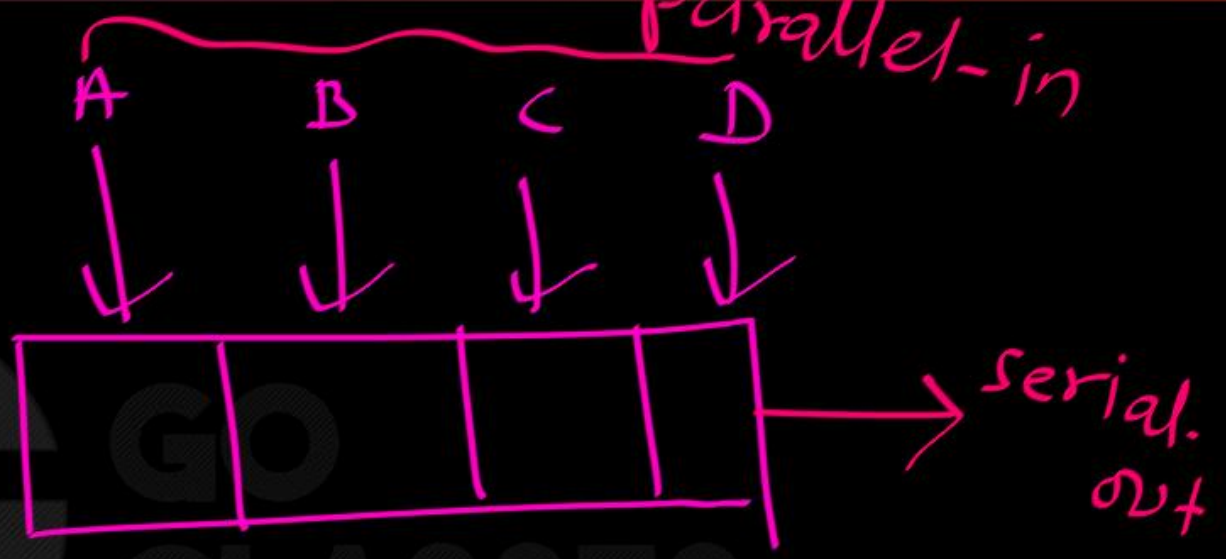
Shift Register :

(1) SISO

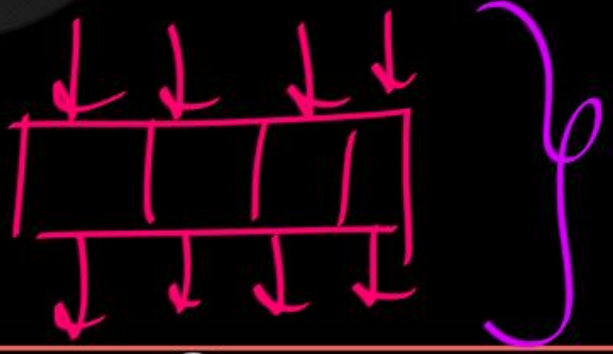
(2) SIPO

(3) PISO

Shift repister

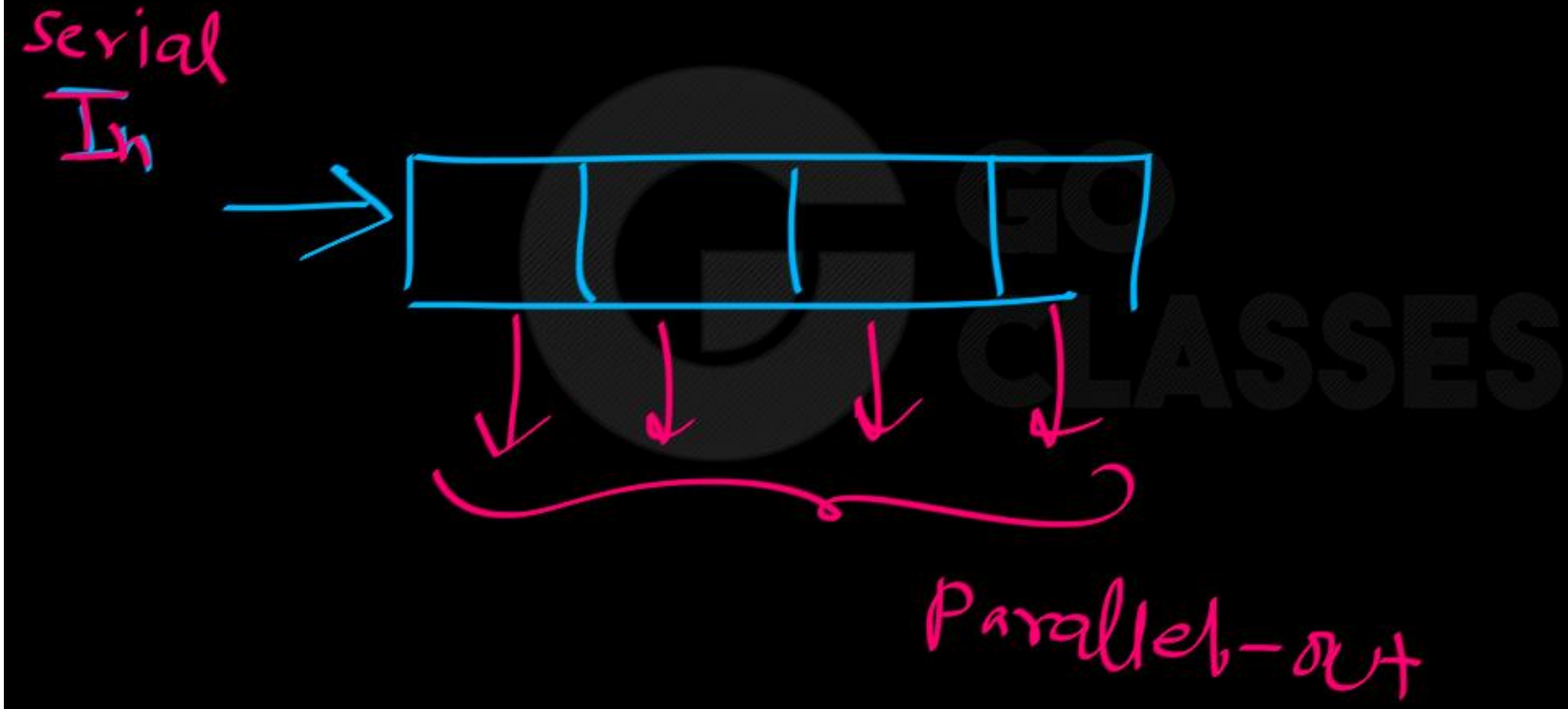


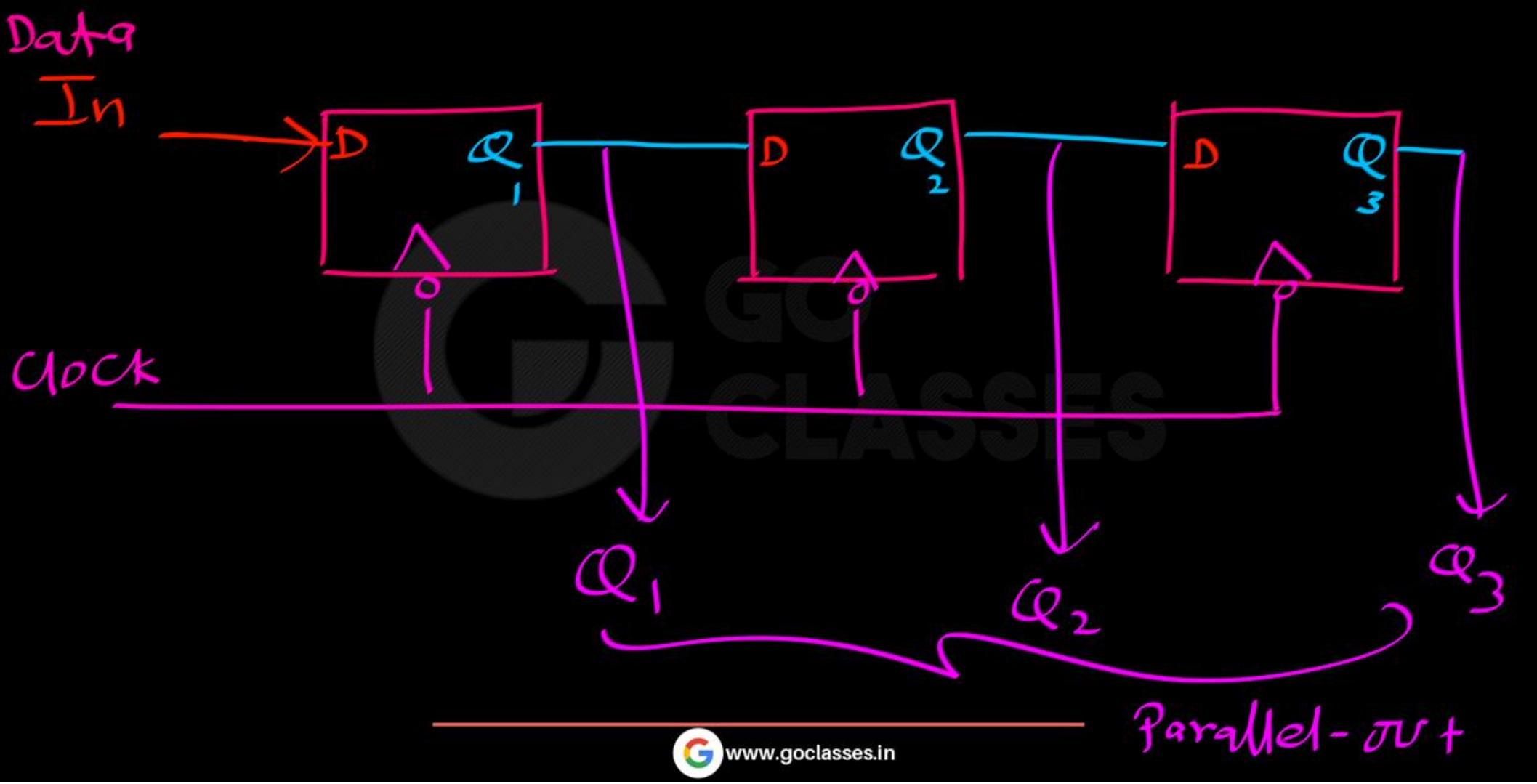
(4) PIPO



Not a shift repister

SIPO shift Reg:




Data In
D
Q1
D
Q2
D
Q3
Clock
Q1
Q2
Q3
Parallel - out

Serial In/Parallel Out Shift Registers

Data bits are entered serially (least-significant bit first) into a serial in/parallel out shift register in the same manner as in serial in/serial out registers. The difference is the way in which the data bits are taken out of the register; in the parallel output register, the output of each stage is available. Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously, rather than on a bit-by-bit basis as with the serial output. Figure 8–6 shows a 4-bit serial in/parallel out shift register and its logic block symbol.

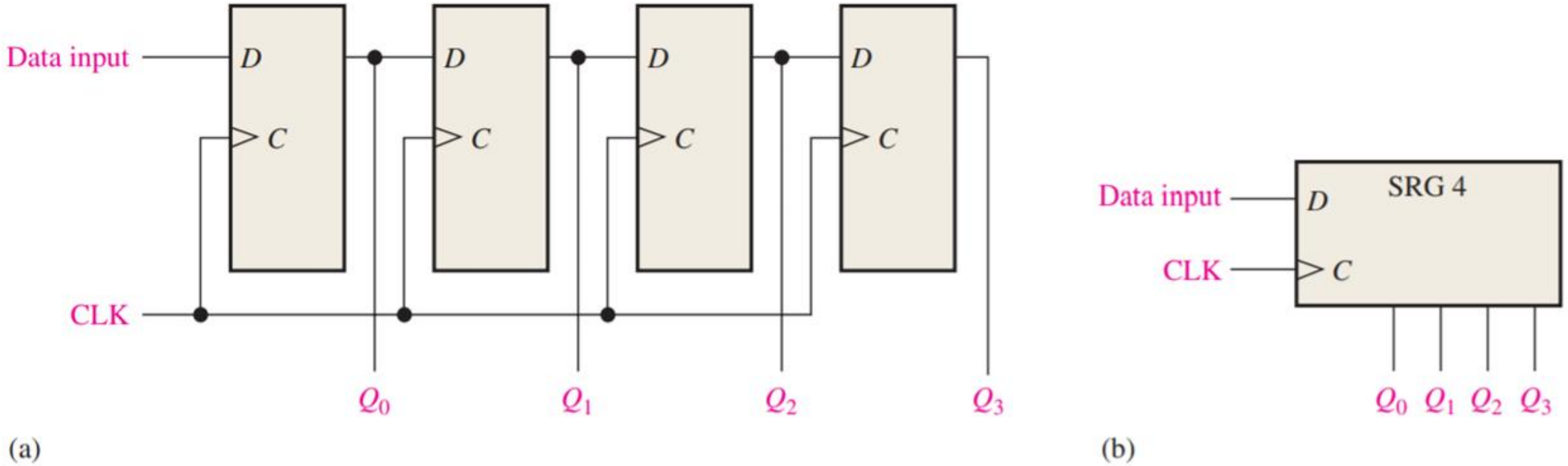


FIGURE 8–6 A serial in/parallel out shift register.

Show the states of the 4-bit register (SRG 4) for the data input and clock waveforms in Figure 8–7(a). The register initially contains all 1s.

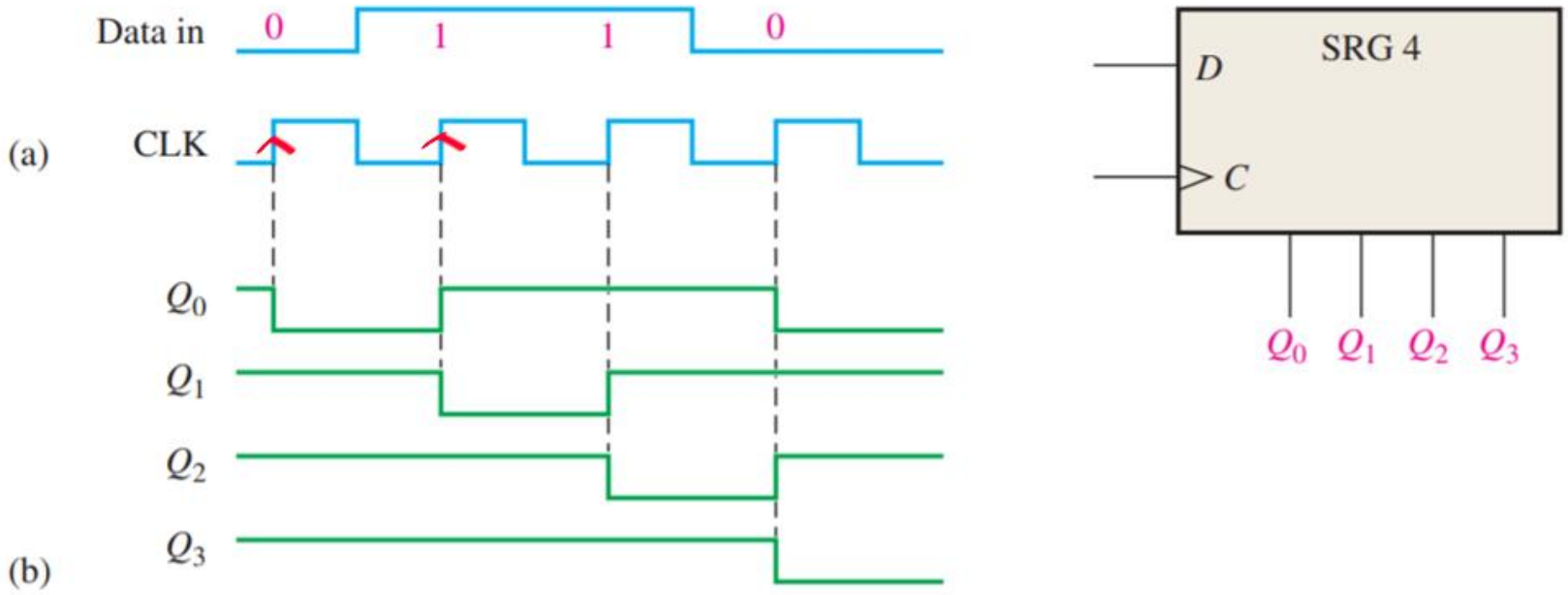
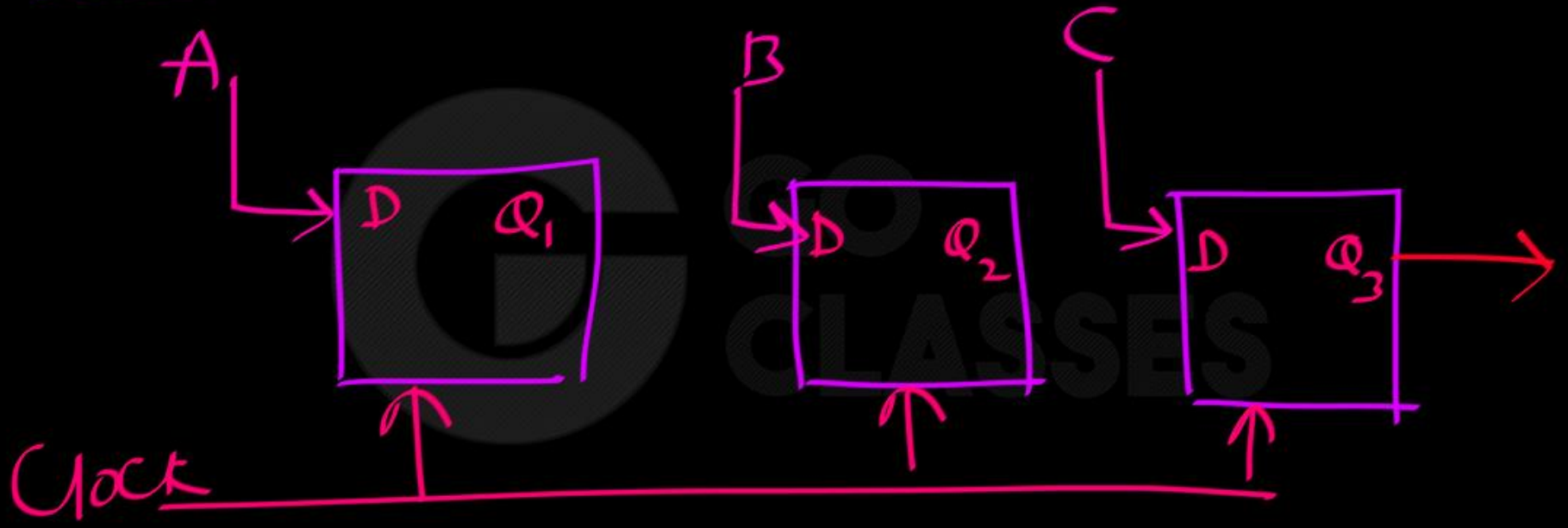


FIGURE 8–7

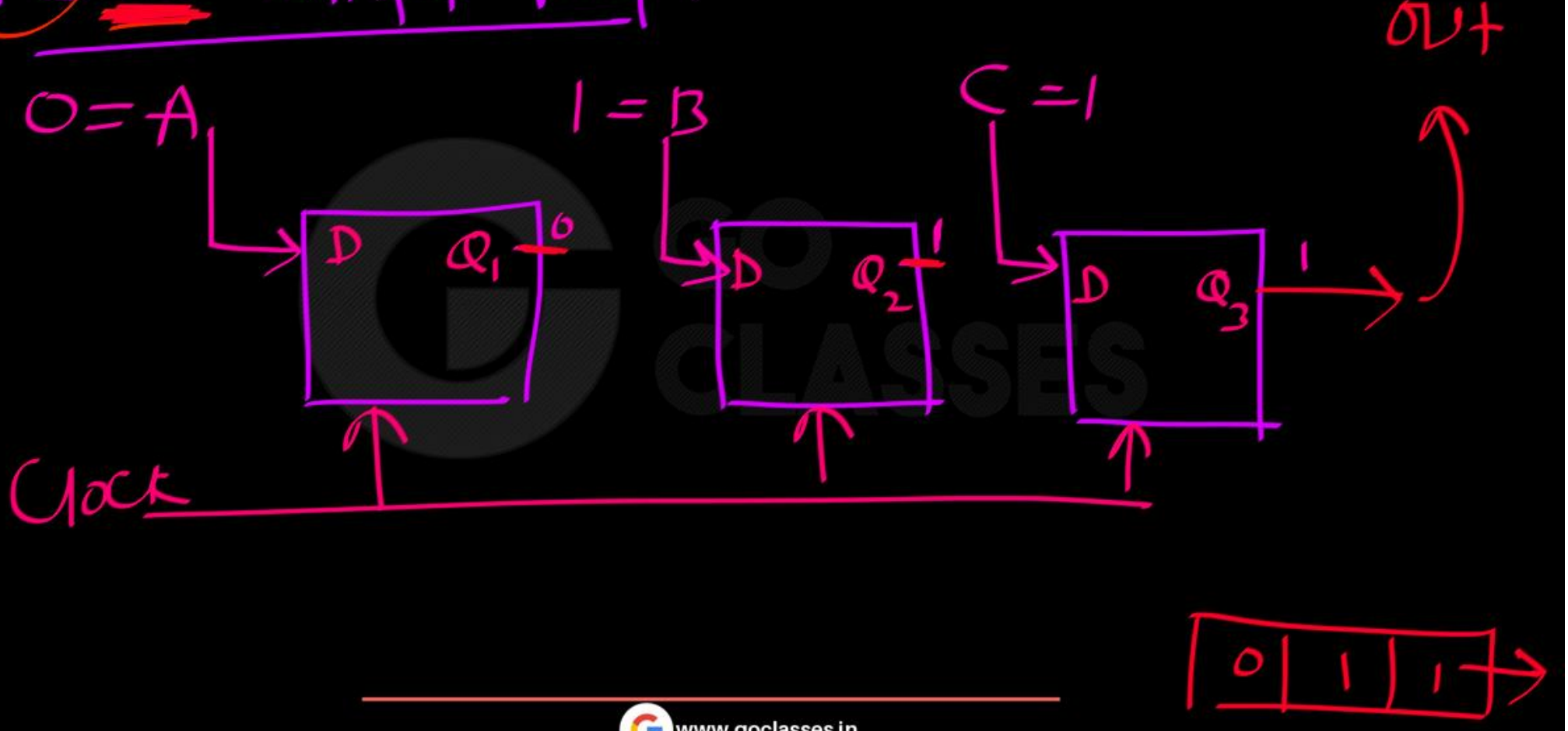
Solution

The register contains 0110 after four clock pulses. See Figure 8–7(b).

PISO shift Reg:



PISO shift Reg:



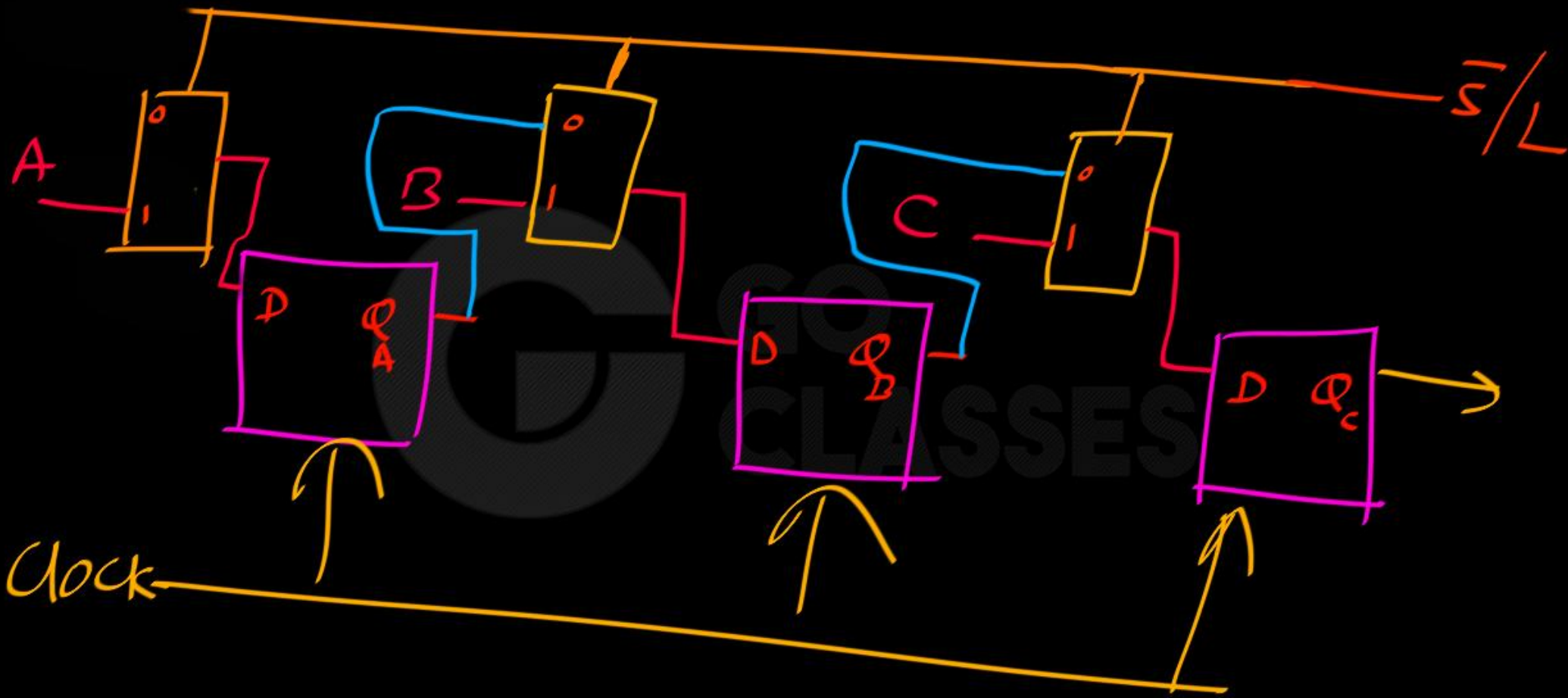
PISO Shift Register:

We use shift/load input.

S/L

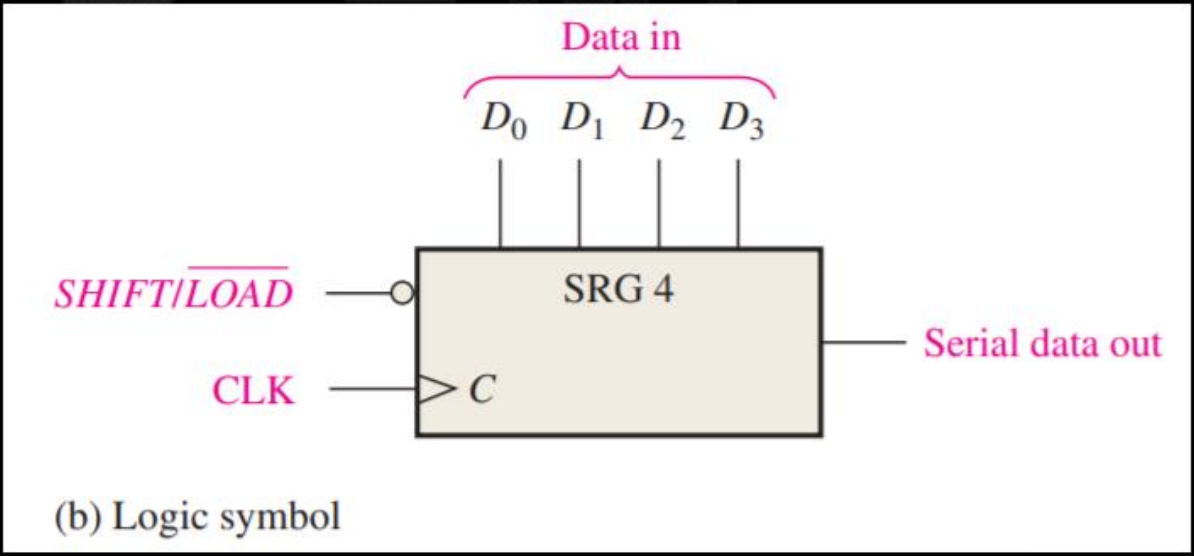
If $\overline{S}/L = 0$ then shift-right

If $\overline{S}/L = 1$ then load into register


A
B
C
S/L
D
Q_A
D
Q_B
D
Q_C
Clock

Parallel In/Serial Out Shift Registers

For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs. The serial output is the same as in serial in/serial out shift registers, once the data are completely stored in the register.

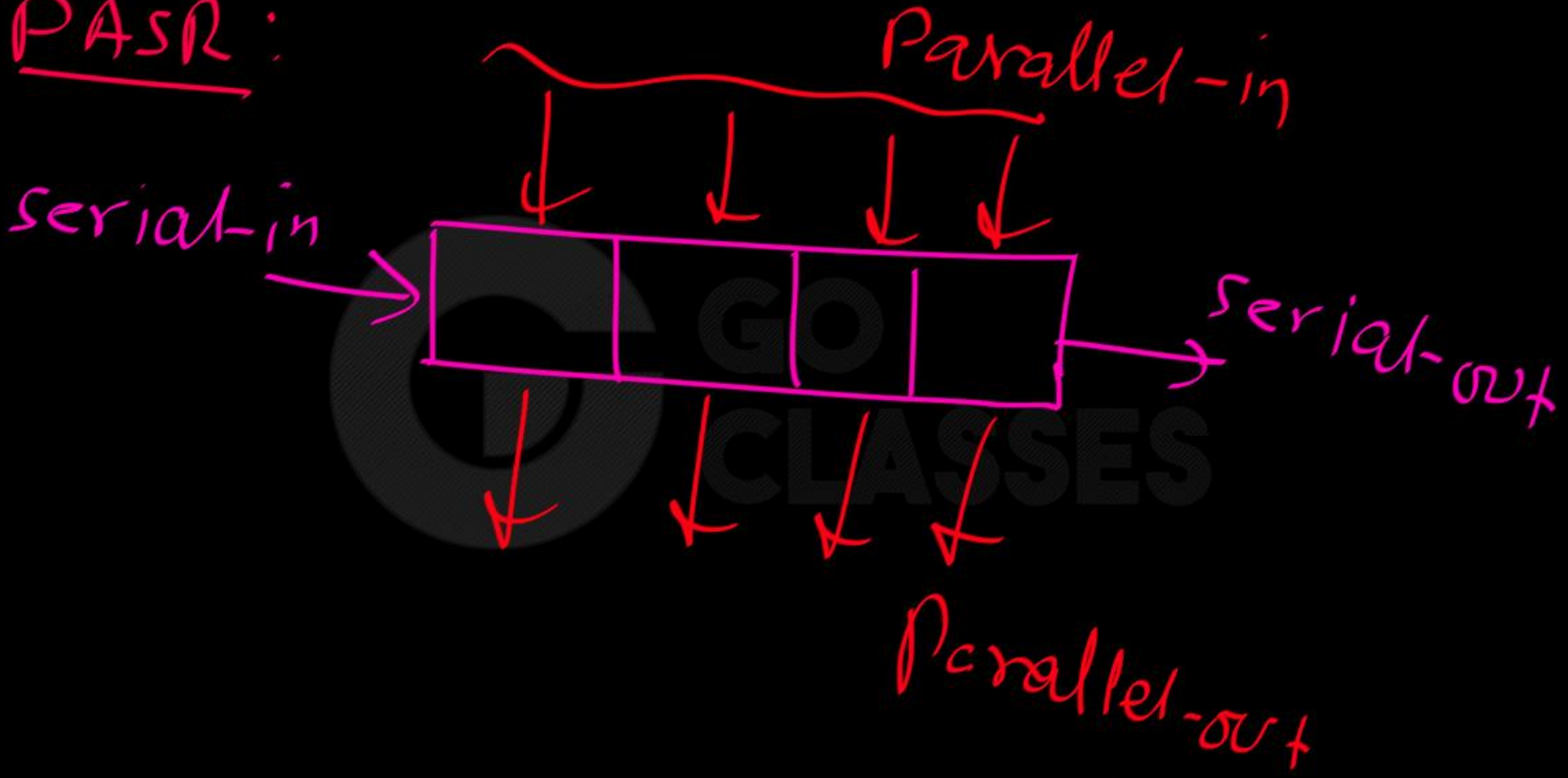


(b) Logic symbol

Parallel-Access Shift Register

(PASR)

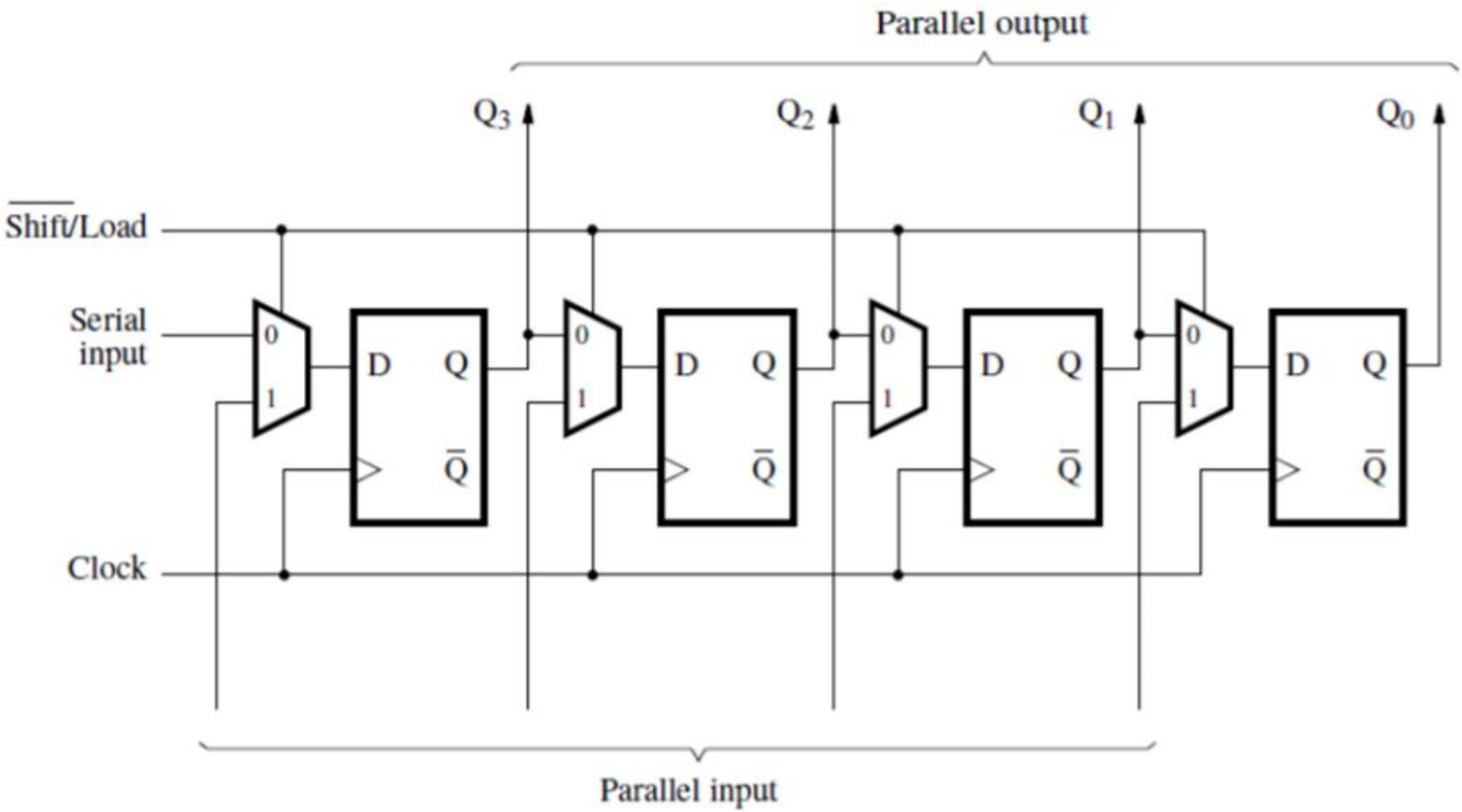
PASR :



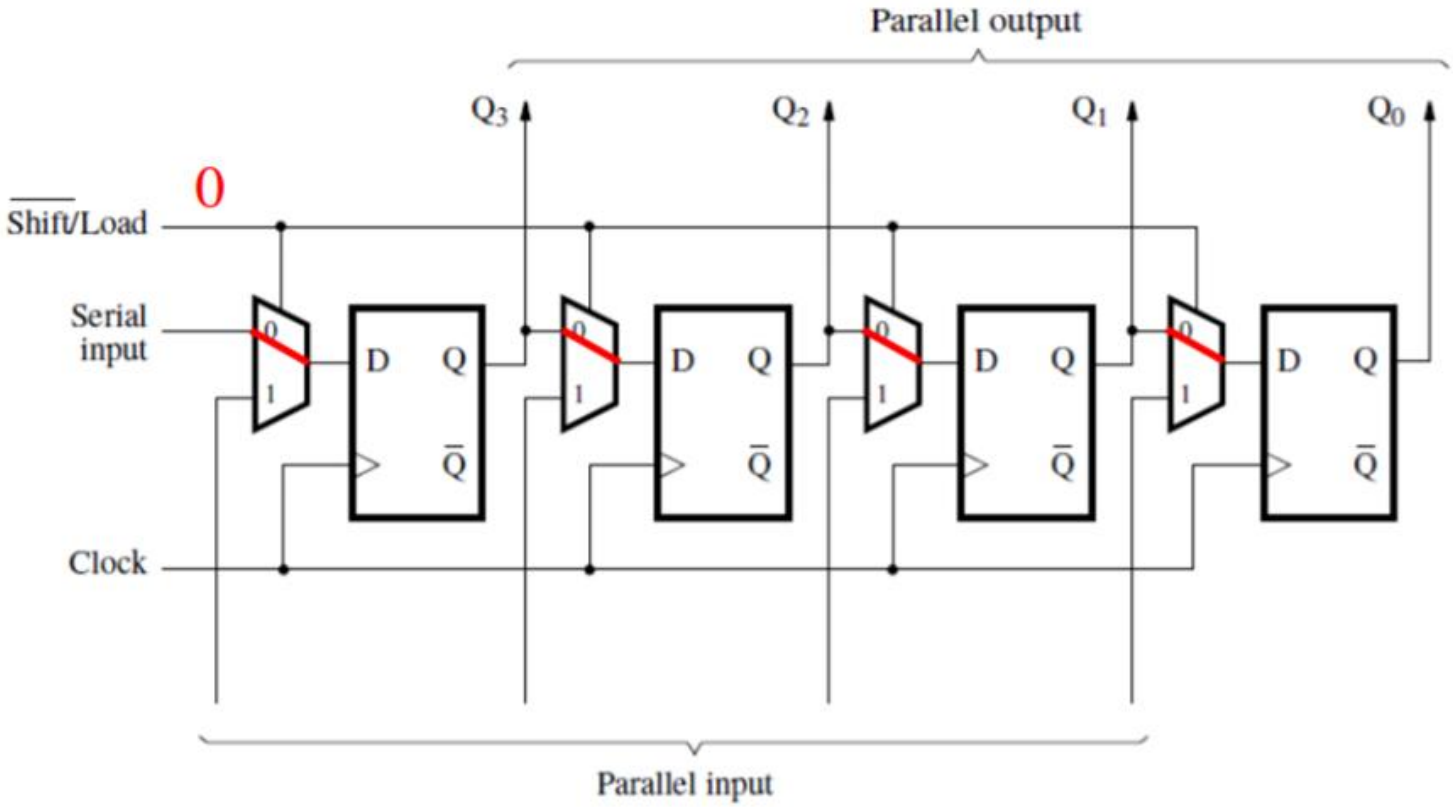
We use "shift/load"

$\overline{S}/L = 1$ then Parallel load (PIPO)

$\overline{S}/L = 0$ then SISO

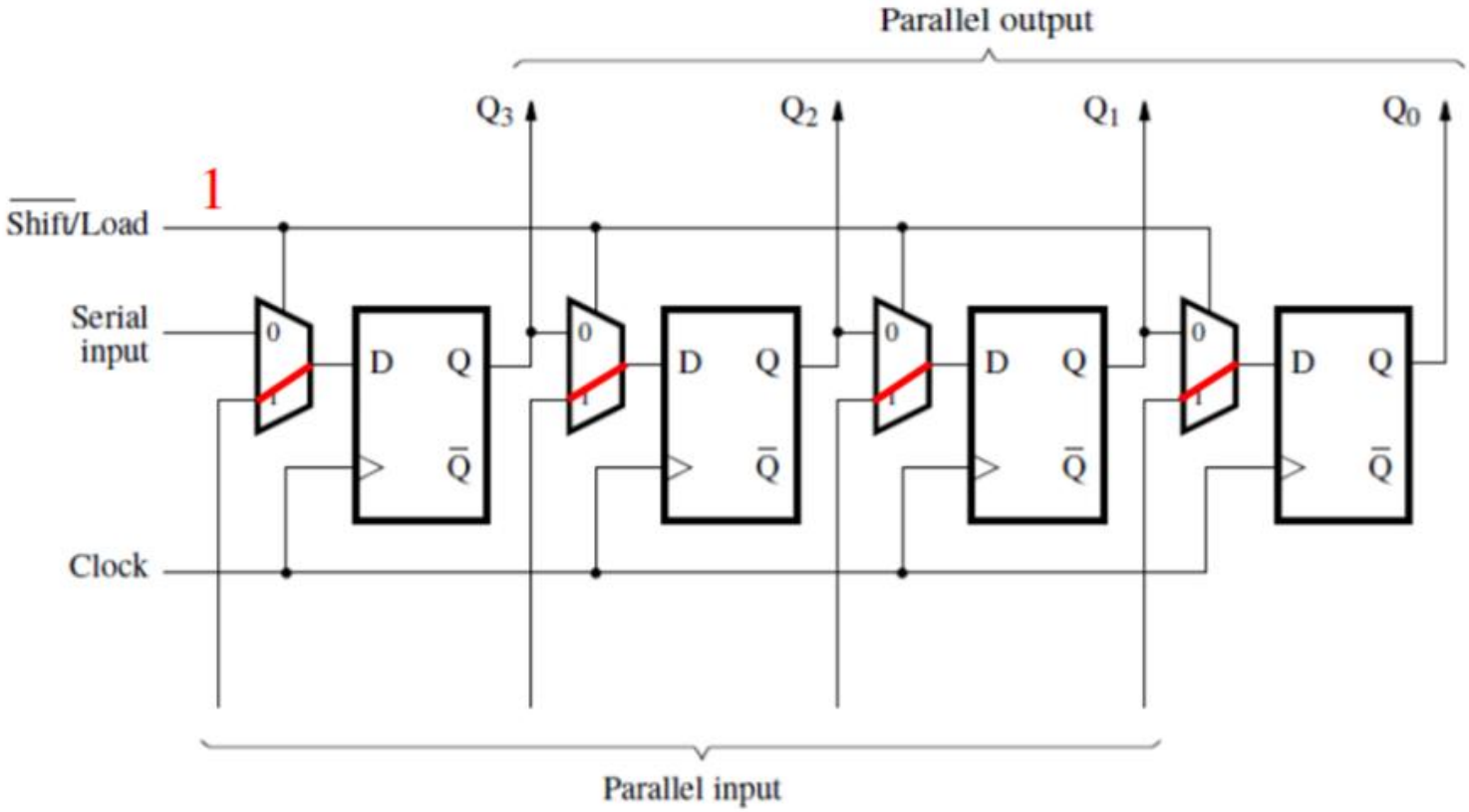

Parallel output
Q_3
Q_2
Q_1
Q_0
$\overline{\text{Shift/Load}}$
Serial input
Clock
0
1
D
Q
$\overline{Q}$
Parallel input

Parallel-access shift register

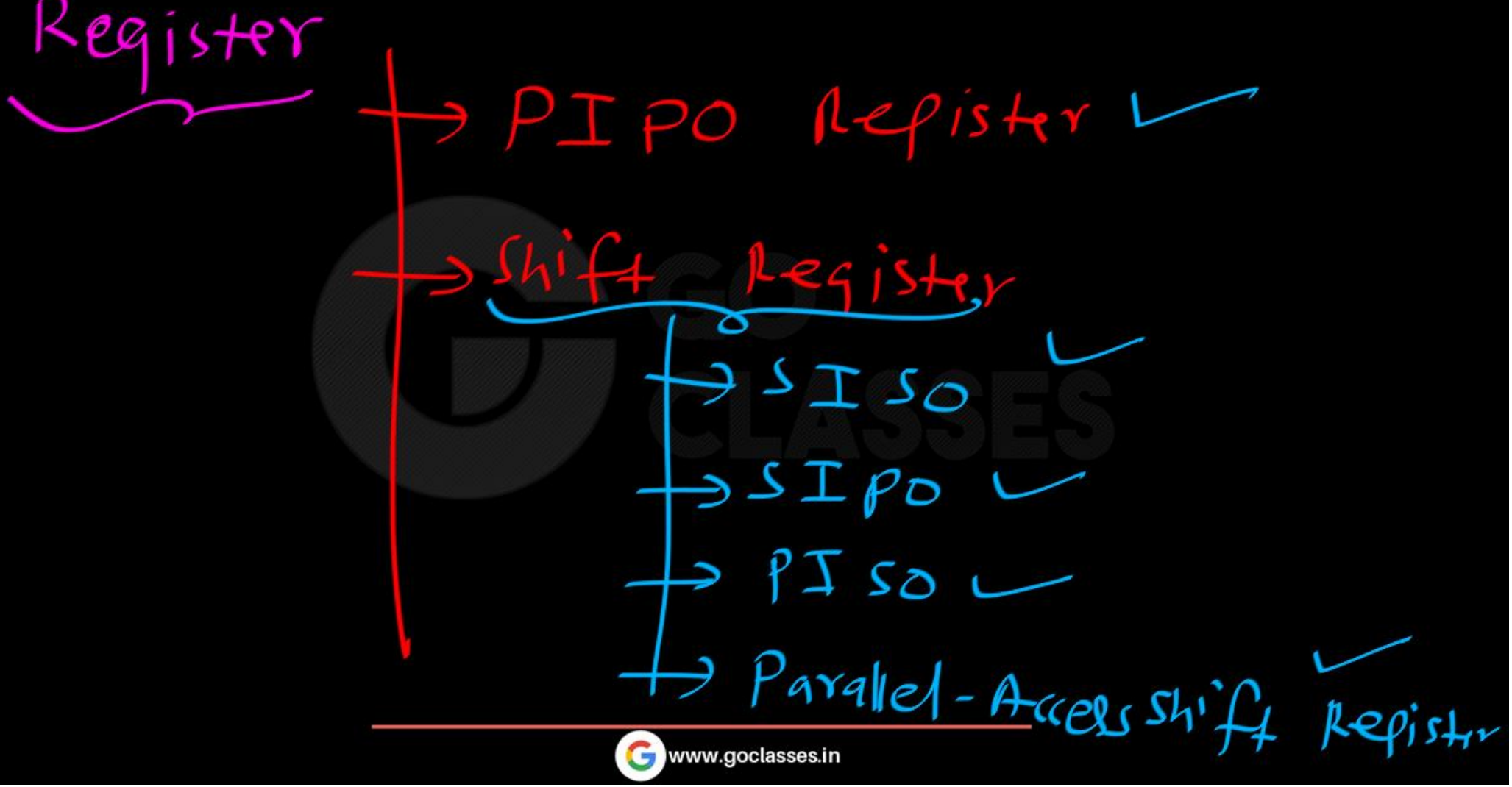


When Load=0, this behaves like a shift register.

Parallel-access shift register



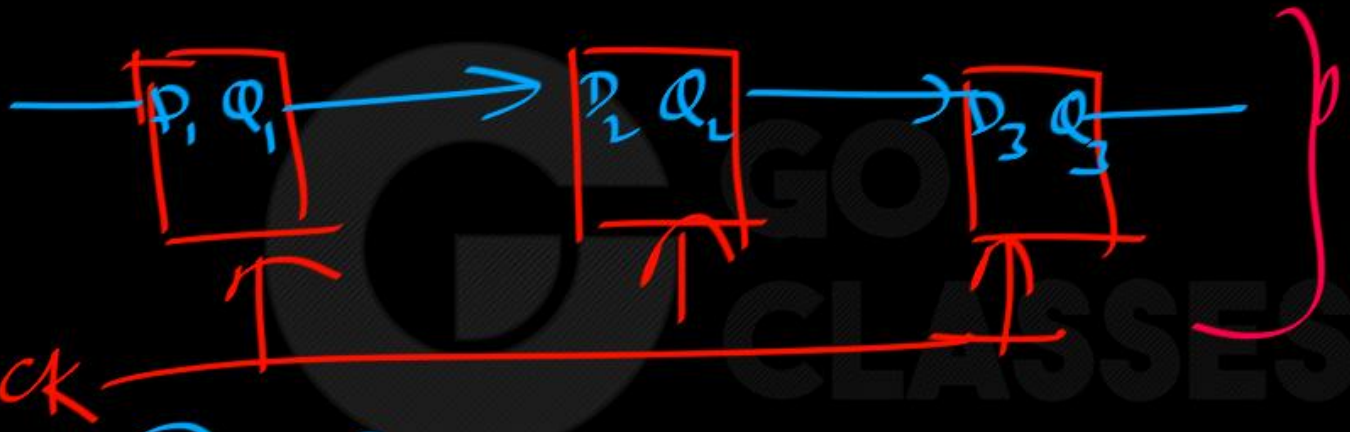
When Load=1, this behaves like a parallel-access register. (PIPO)

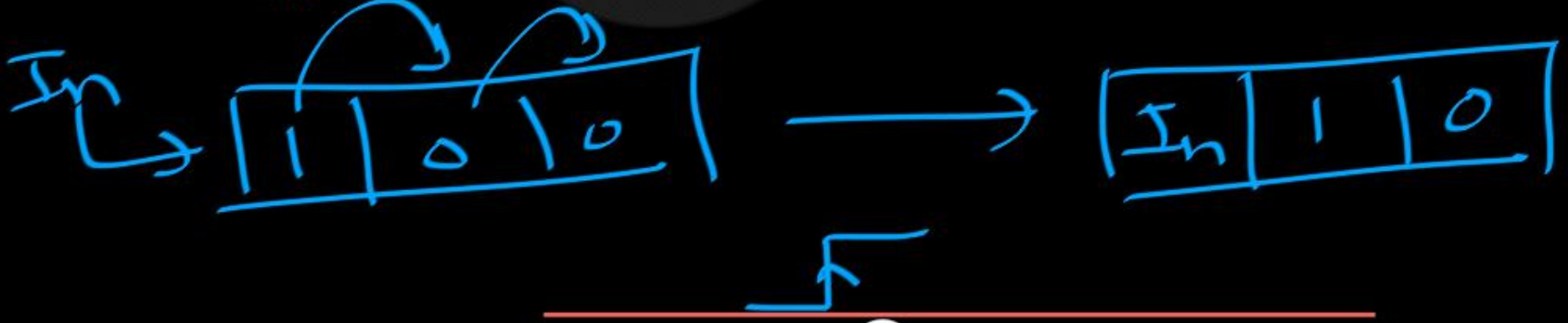

Register
PIPO Register
Shift Register
SISO
SIPO
PISO
Parallel-Access Shift Register

Variations of Shift Registers

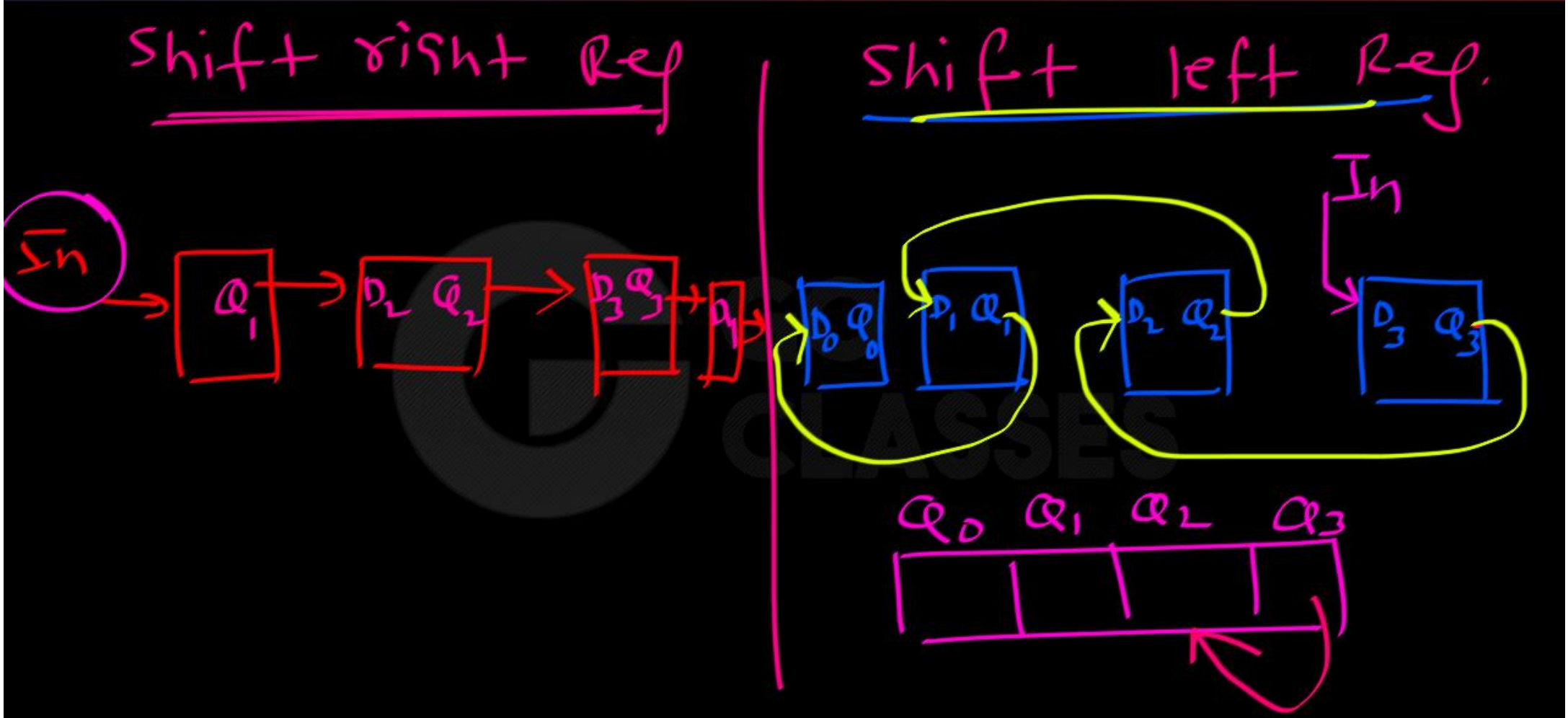
- Depending upon the way the various stages are connected, there can be various types of shift registers:
  a) Ring counter
  b) Twisted ring or Johnson counter
  c) Bidirectional shift register
  d) Universal shift register
  e) Linear feedback shift register

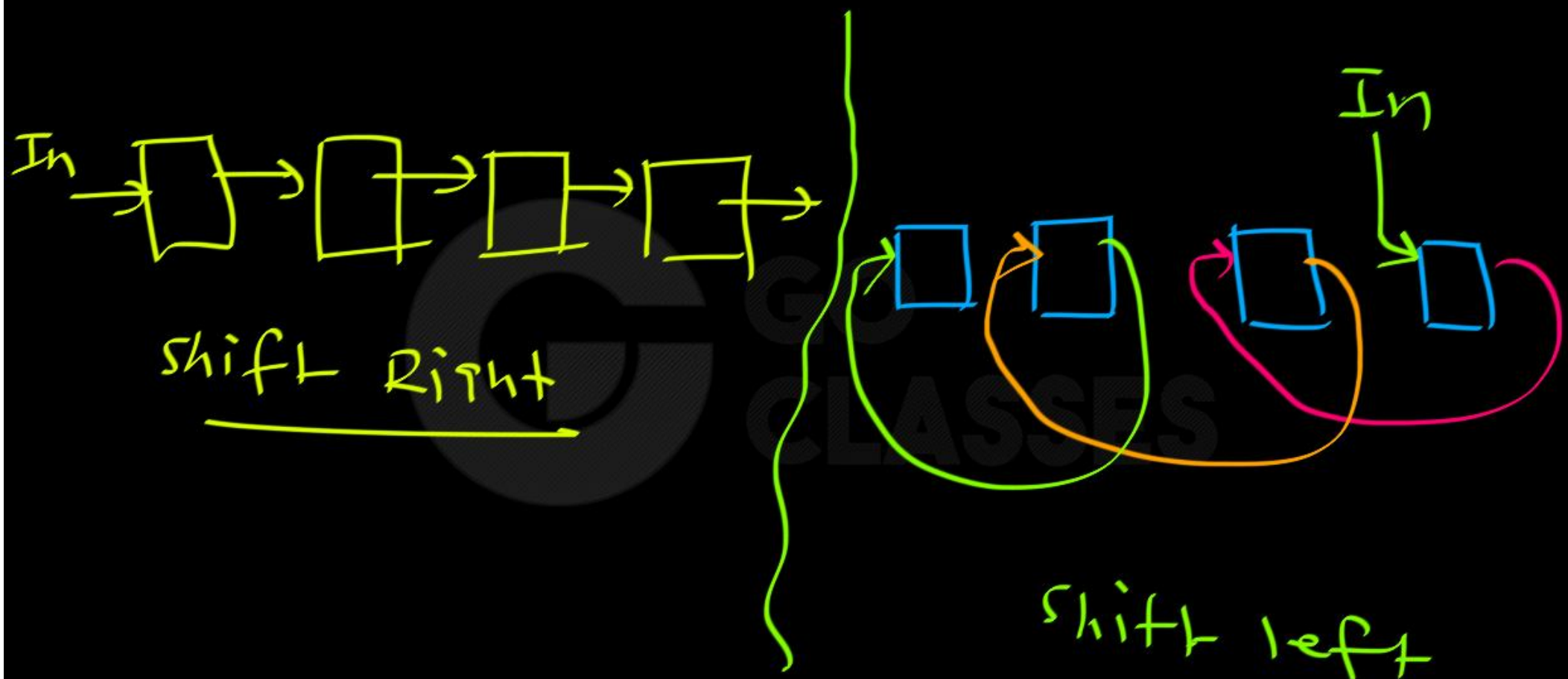
Unidirectional shift right register





Unidirectional shift left register


Shift right Reg
Shift left Reg.
In
In
Q_1
D_2 Q_2
D_3 Q_3
D_0 Q_0
D_1 Q_1
D_2 Q_2
D_3 Q_3
Q_0 Q_1 Q_2 Q_3


In
Shift Right
In
Shift left

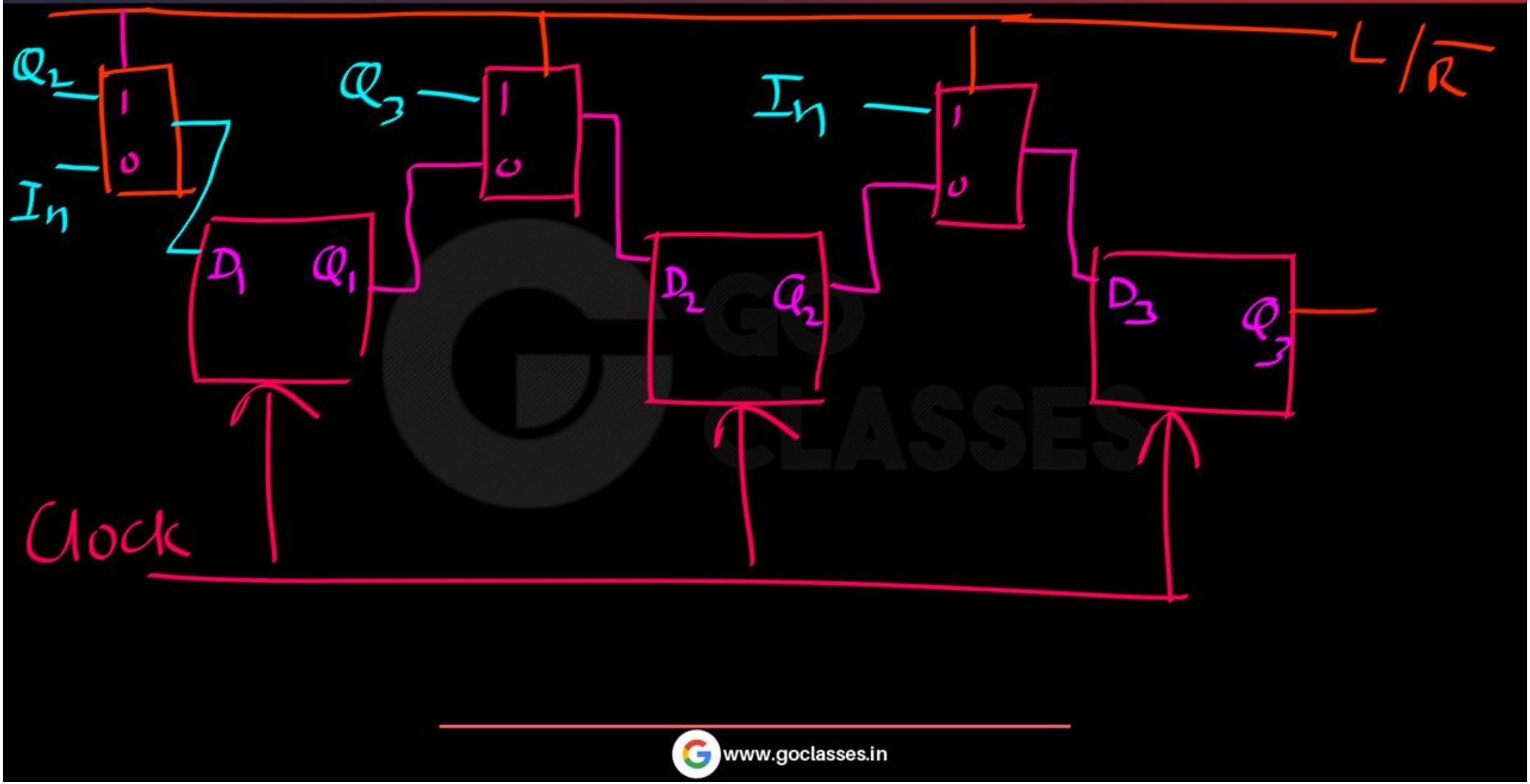
Bidirectional Shift Registers

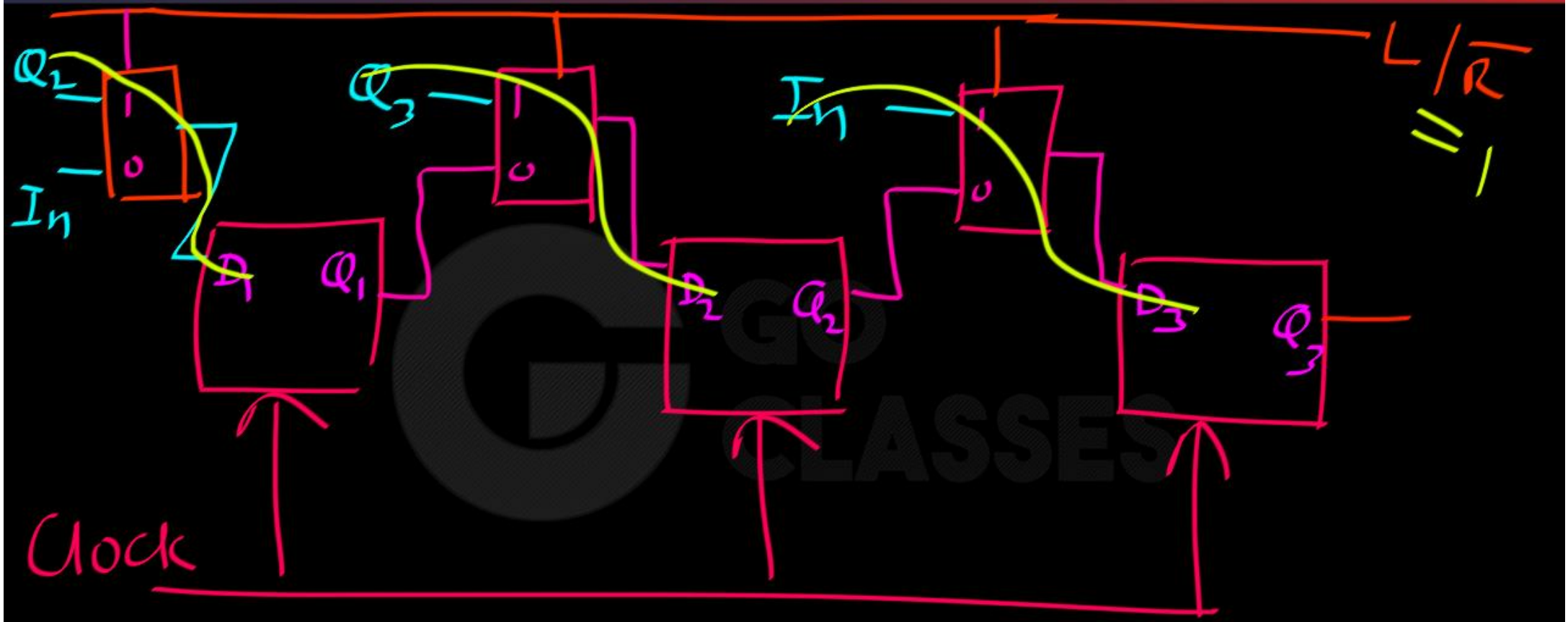
- Based on a control input, the shift register works in either the shift-right or the shift-left modes.
- Multiplexers are used to feed the appropriate signals to the D inputs of the flip-flops.
  - A control input *L/R'* selects the multiplexer inputs.
  - Determines whether to shift left or to shift right.

Bidirectional shift reg:

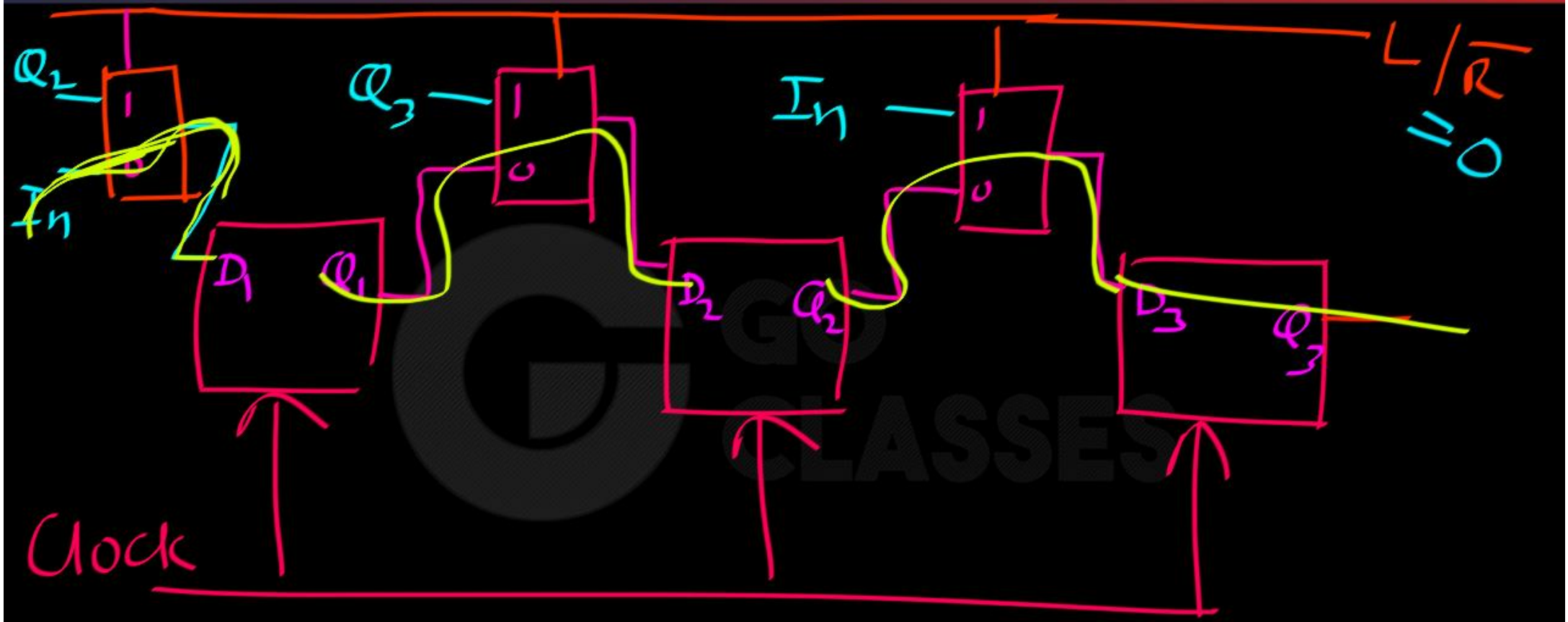
If $L/R' = 0$ then Right shift.

If $L/R' = 1$ then left ".

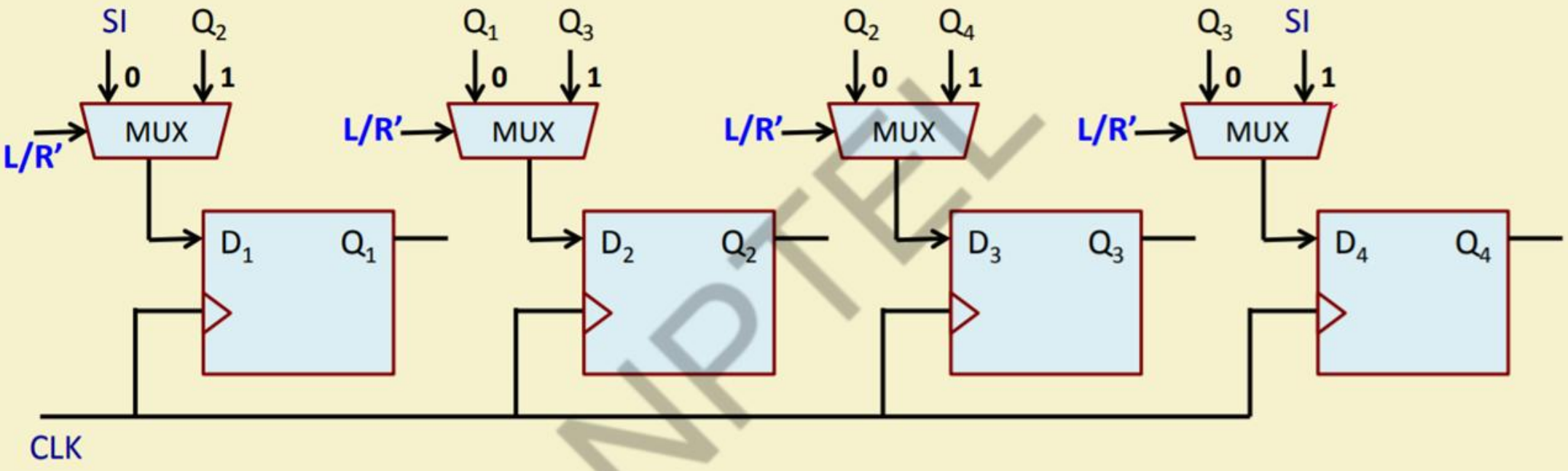

Q2
I/O
In
Q3
I/O
In
I/O
L/R
D1
Q1
D2
Q2
D3
Q3
Clock



If $L/\overline{R} = 1 \Rightarrow$ shift left.



So $L/\bar{R} = 0 \Longrightarrow$ shift Right



A 4-bit bidirectional shift register

Universal Shift Register

If the flip-flop outputs of a shift register are accessible, then information entered serially by shifting can be taken out in parallel from the outputs of the flip-flops. If a parallel load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.

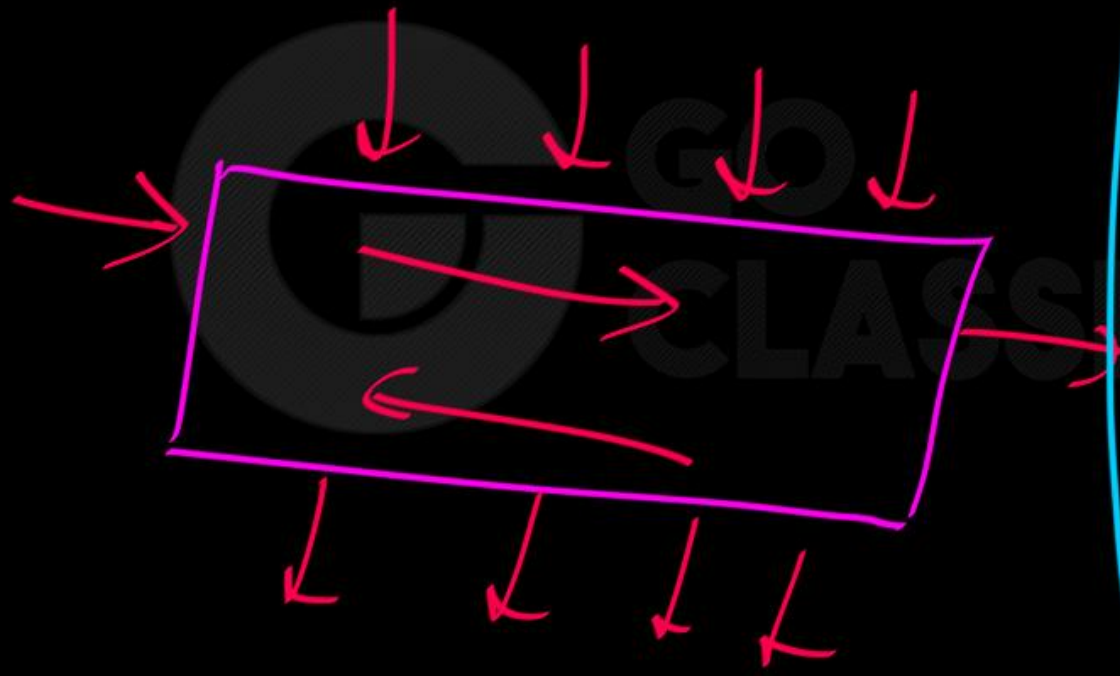
Some shift registers provide the necessary input and output terminals for parallel transfer. They may also have both shift-right and shift-left capabilities. The most general shift register has the following capabilities:

1. A *clear* control to clear the register to 0.
2. A *clock* input to synchronize the operations.

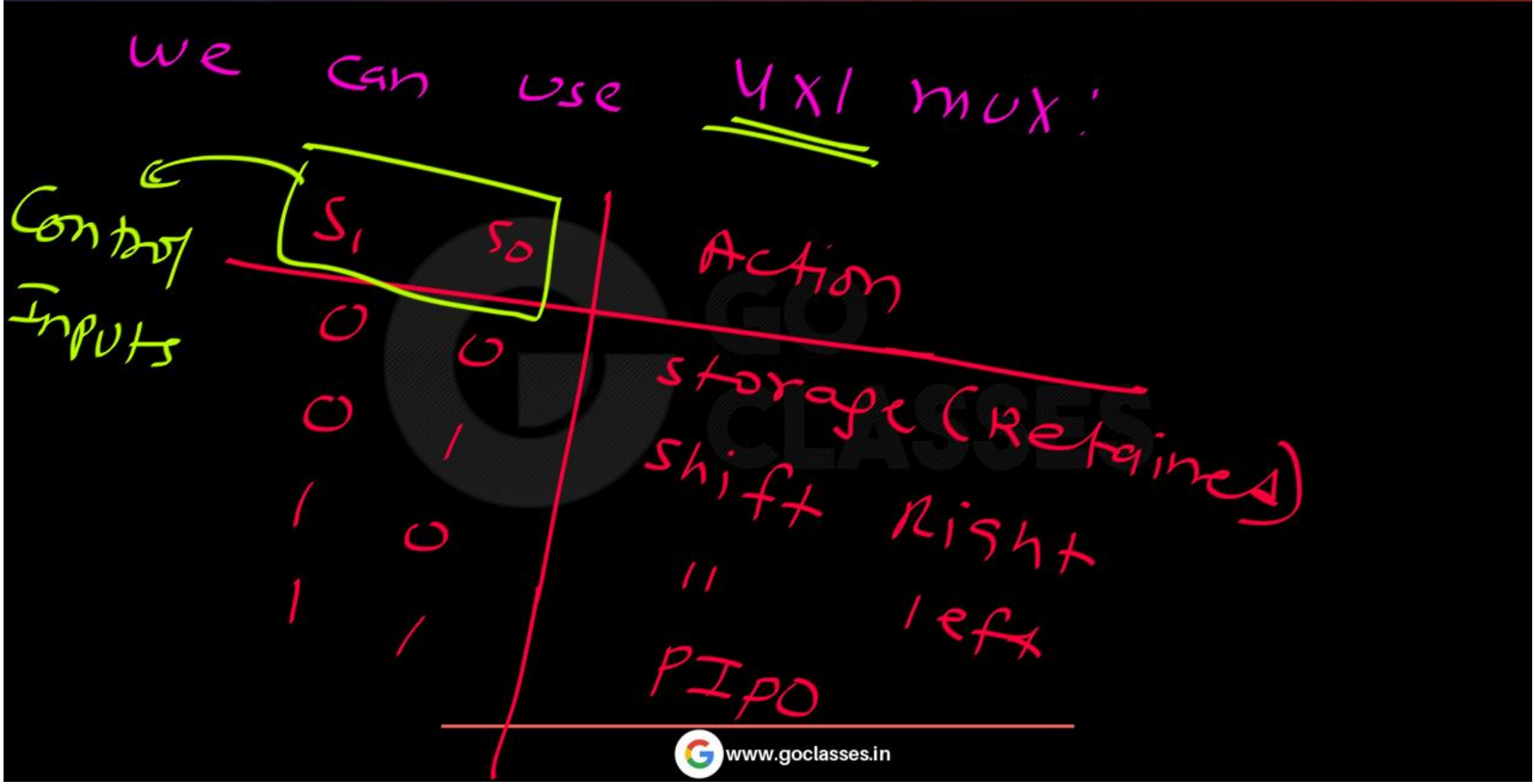
3. A *shift-right* control to enable the shift-right operation and the *serial input* and *output* lines associated with the shift right.
4. A *shift-left* control to enable the shift-left operation and the *serial input* and *output* lines associated with the shift left.
5. A *parallel-load* control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. n parallel output lines.
7. A control state that leaves the information in the register unchanged in response to the clock. Other shift registers may have only some of the preceding functions, with at least one shift operation.

A register capable of shifting in one direction only is a *unidirectional* shift register. One that can shift in both directions is a *bidirectional* shift register. If the register has both shifts and parallel-load capabilities, it is referred to as a *universal shift register.*

Universal shift Register

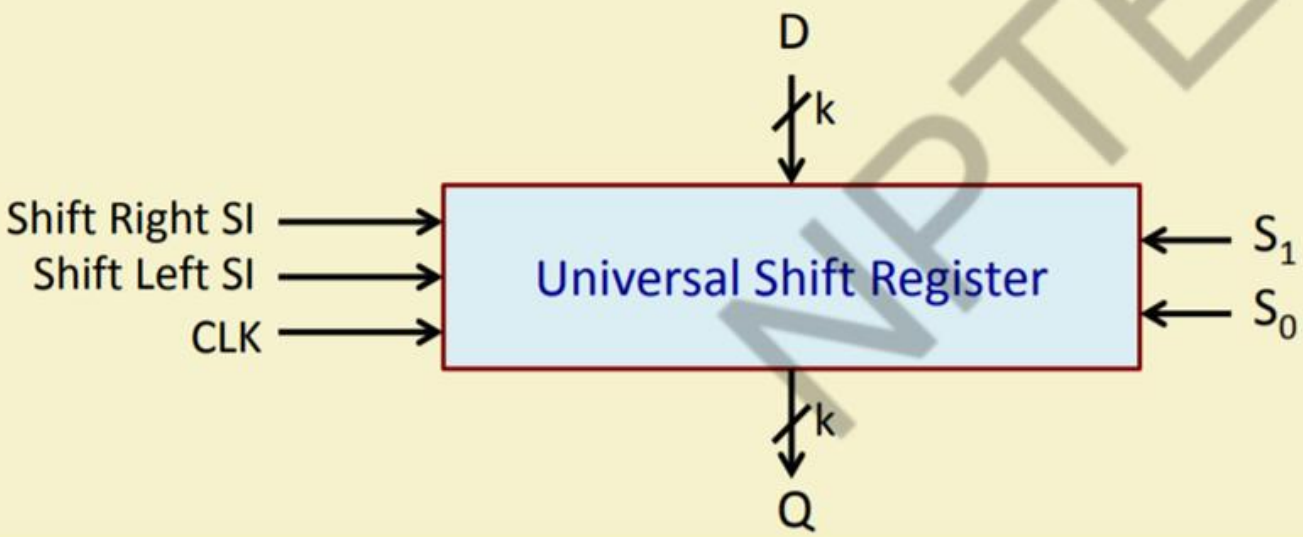


PIPO
SISO
bidirectional
storage
(retain)

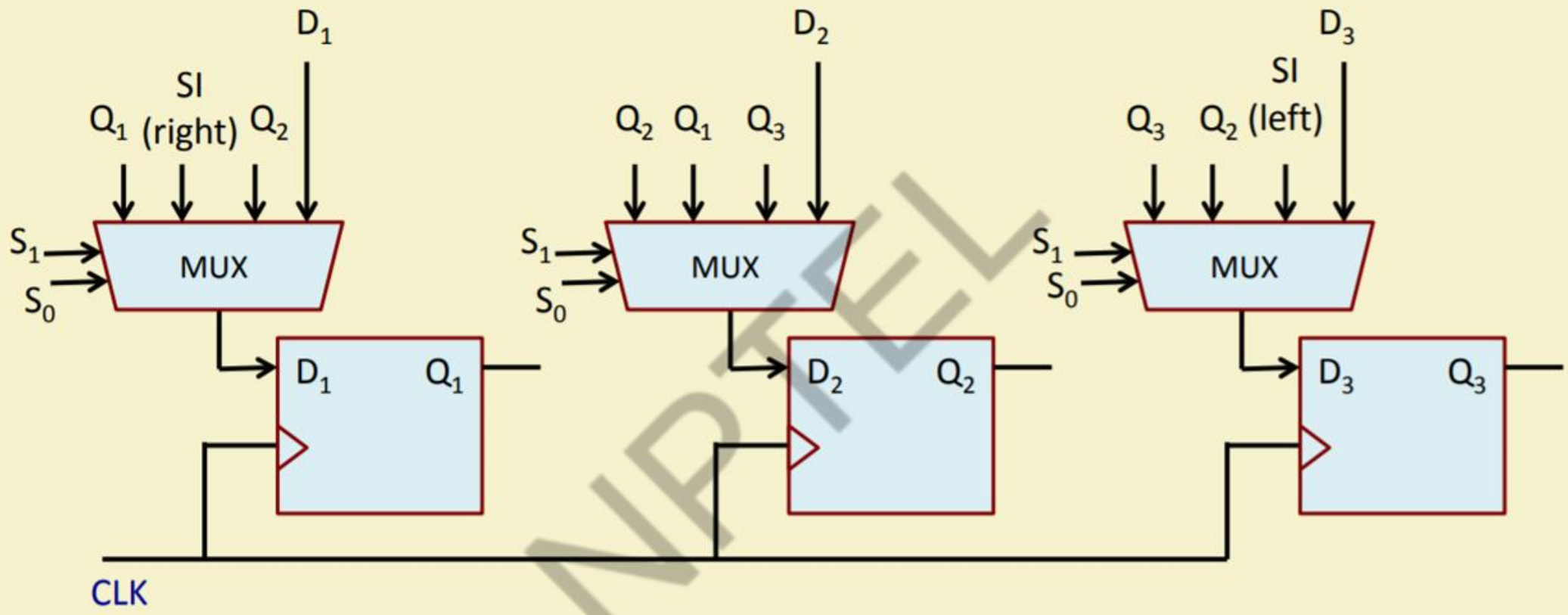

We can use 4x1 mux:
Control Inputs
S1
S0
Action
0 0
storage (Retained)
0 1
shift Right
1 0
"
1 1
left
PIPO

(d) Universal Shift Register

- A universal shift register is a bidirectional shift register, whose input can be either in serial form or in parallel form, and whose output can also be either in serial form or in parallel form.

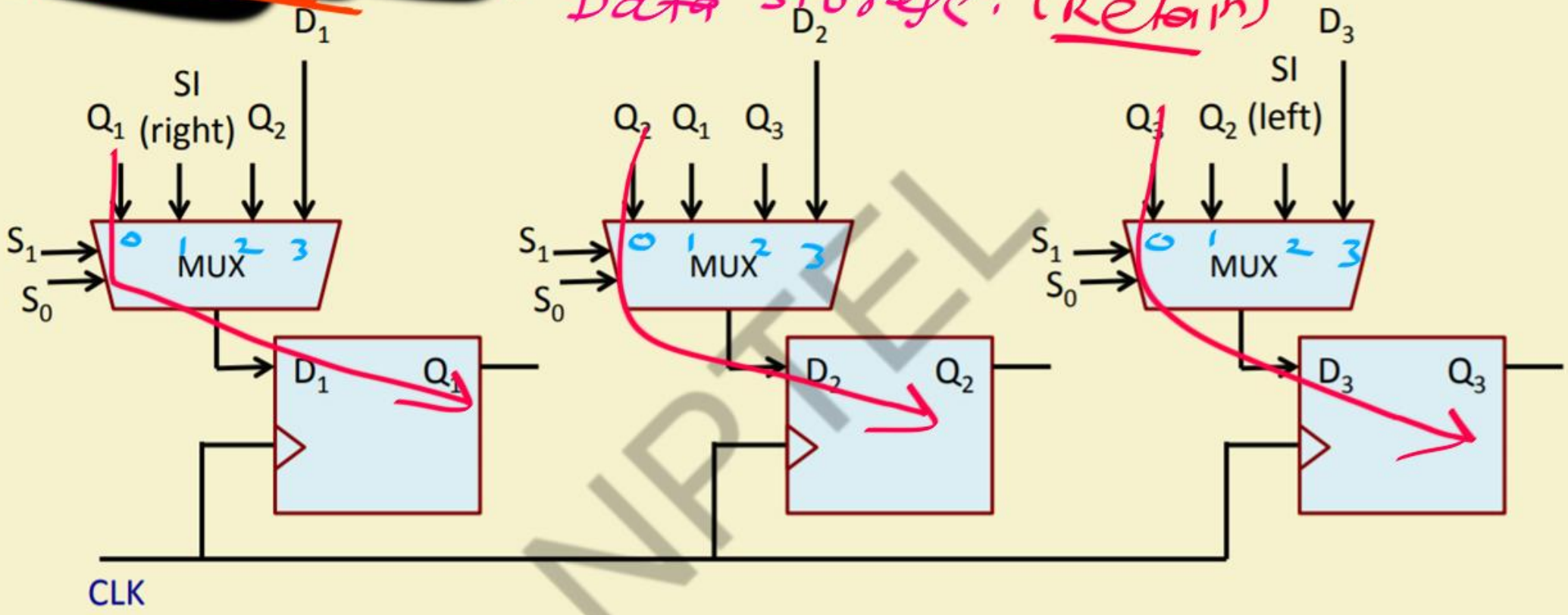


Control Inputs		Action
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load



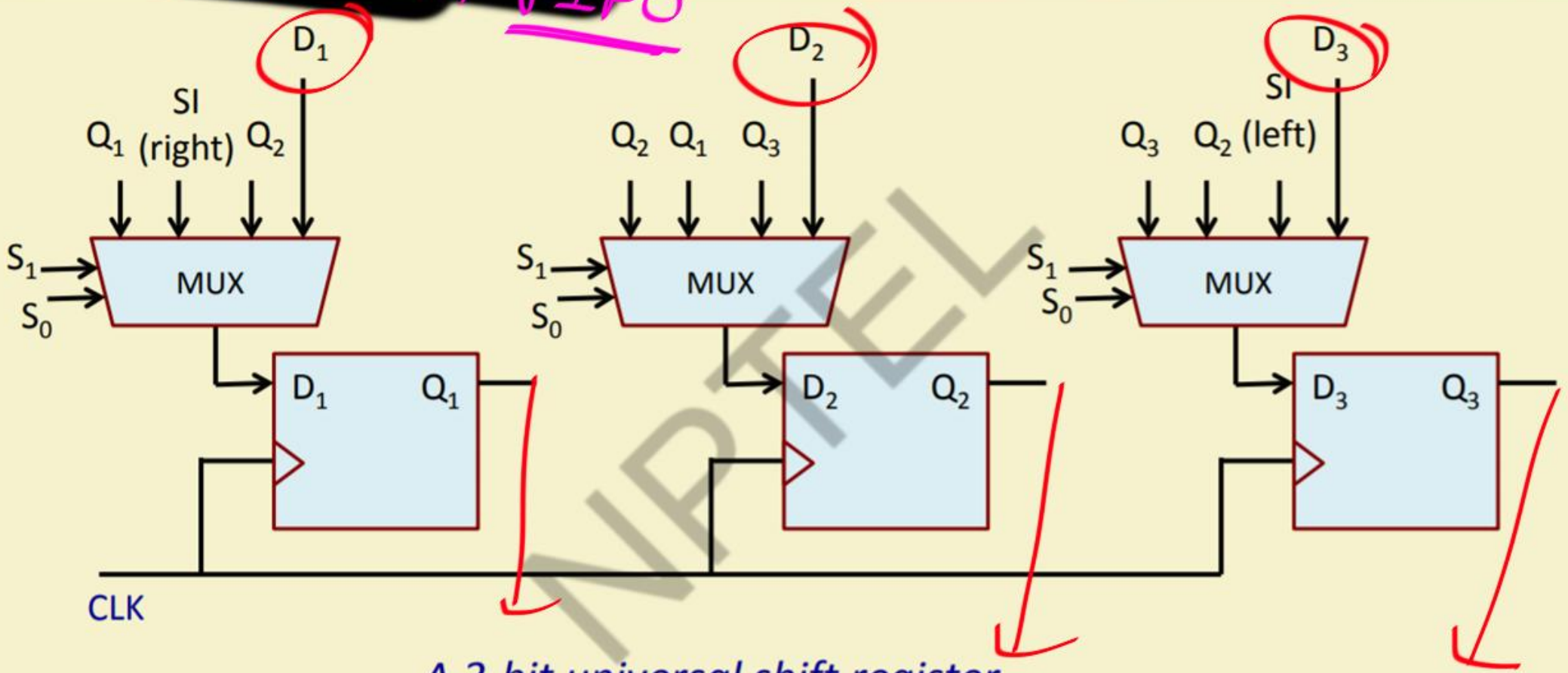
A 3-bit universal shift register

If $S_1=0=S_0$ then Data storage. (Retain)



A 3-bit universal shift register

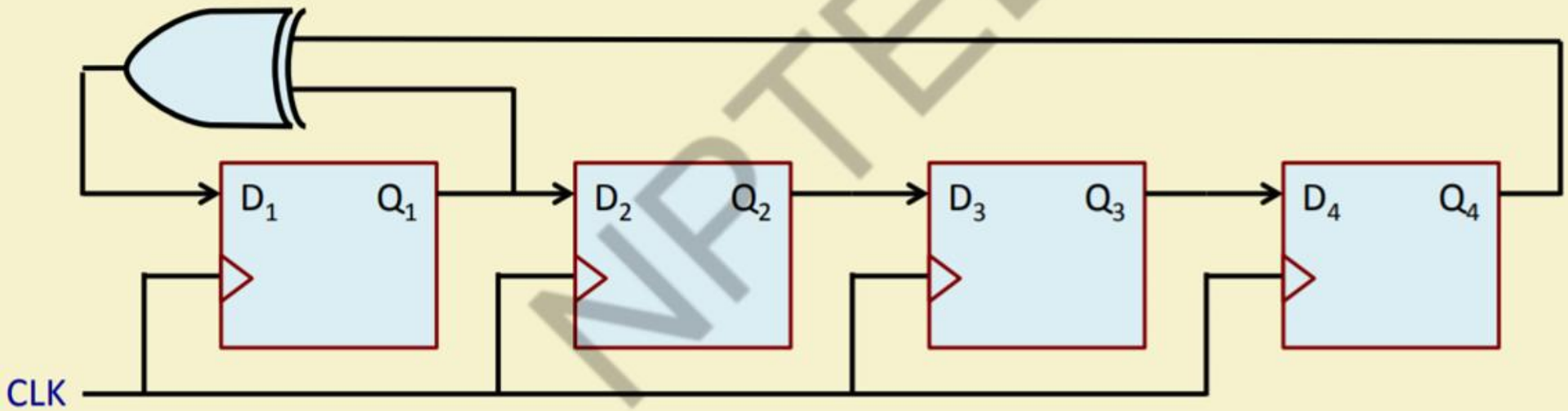
$S_1 = S_0 = 1$ then PIPO



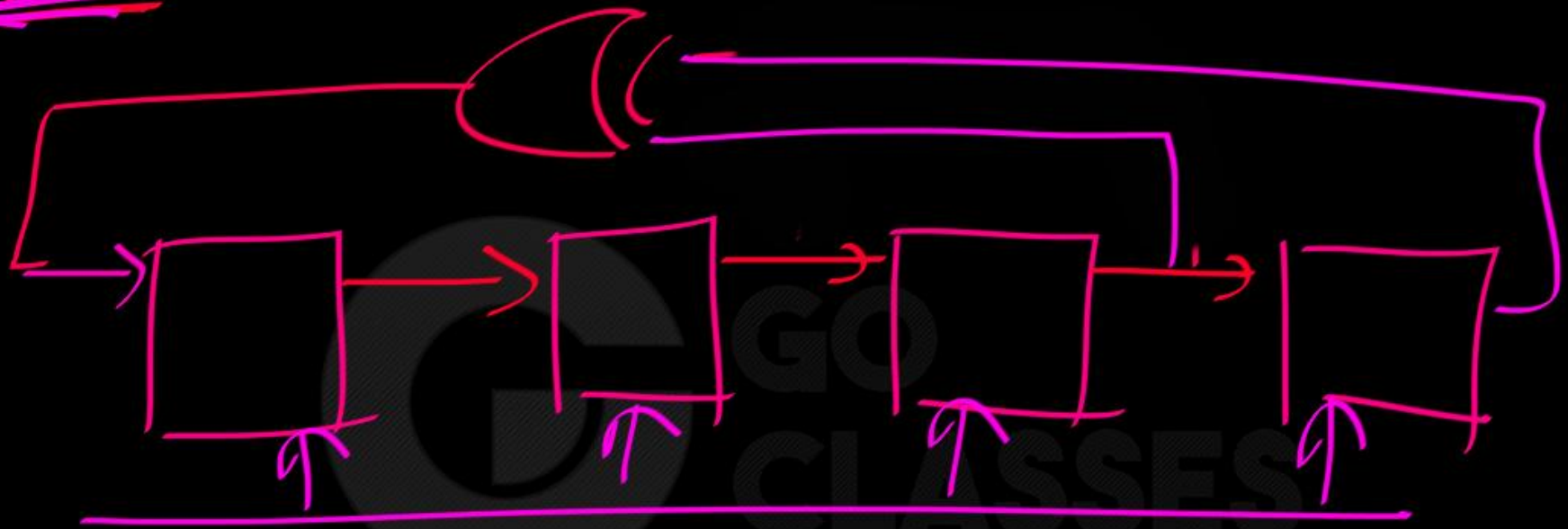
A 3-bit universal shift register

(e) Linear Feedback Shift Register (LFSR)

- It is basically a SISO right shift register, where the serial input is generated as a linear combination (exclusive-OR) of some of the flip-flop outputs.

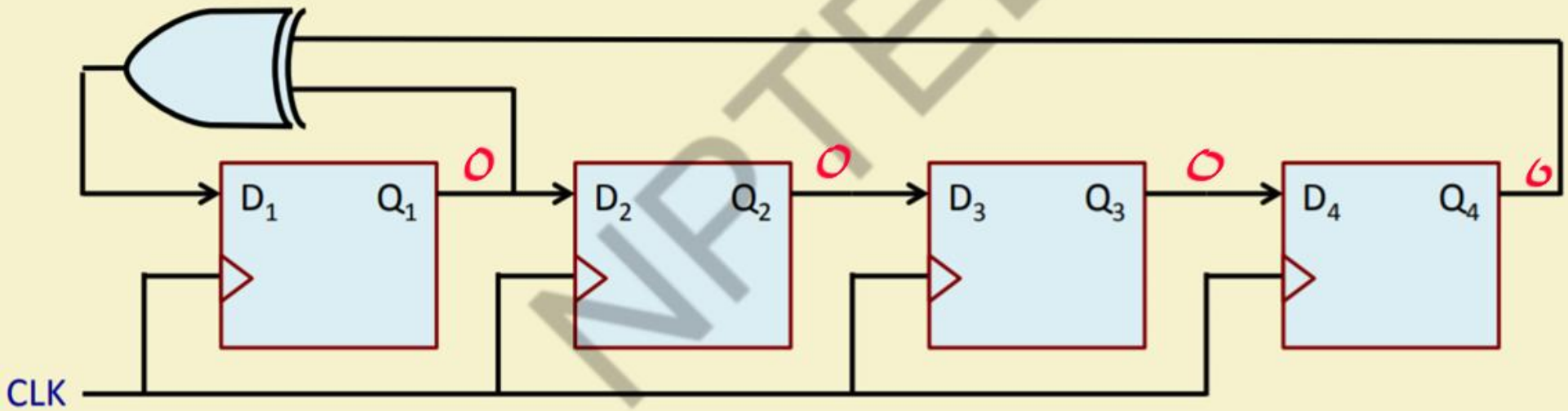


LFSR



(e) Linear Feedback Shift Register (LFSR)

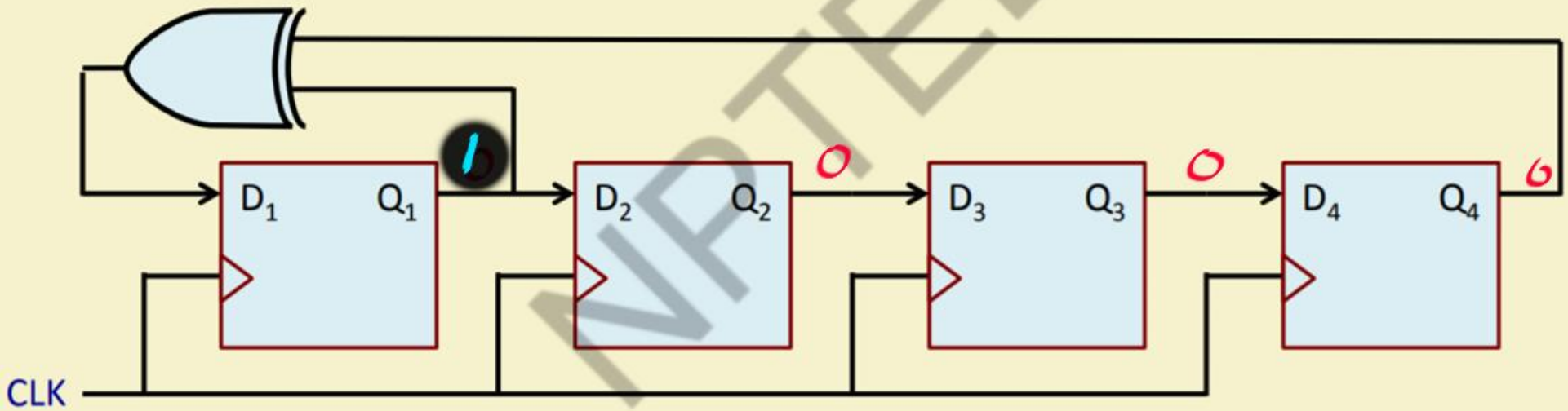
- It is basically a SISO right shift register, where the serial input is generated as a linear combination (exclusive-OR) of some of the flip-flop outputs.

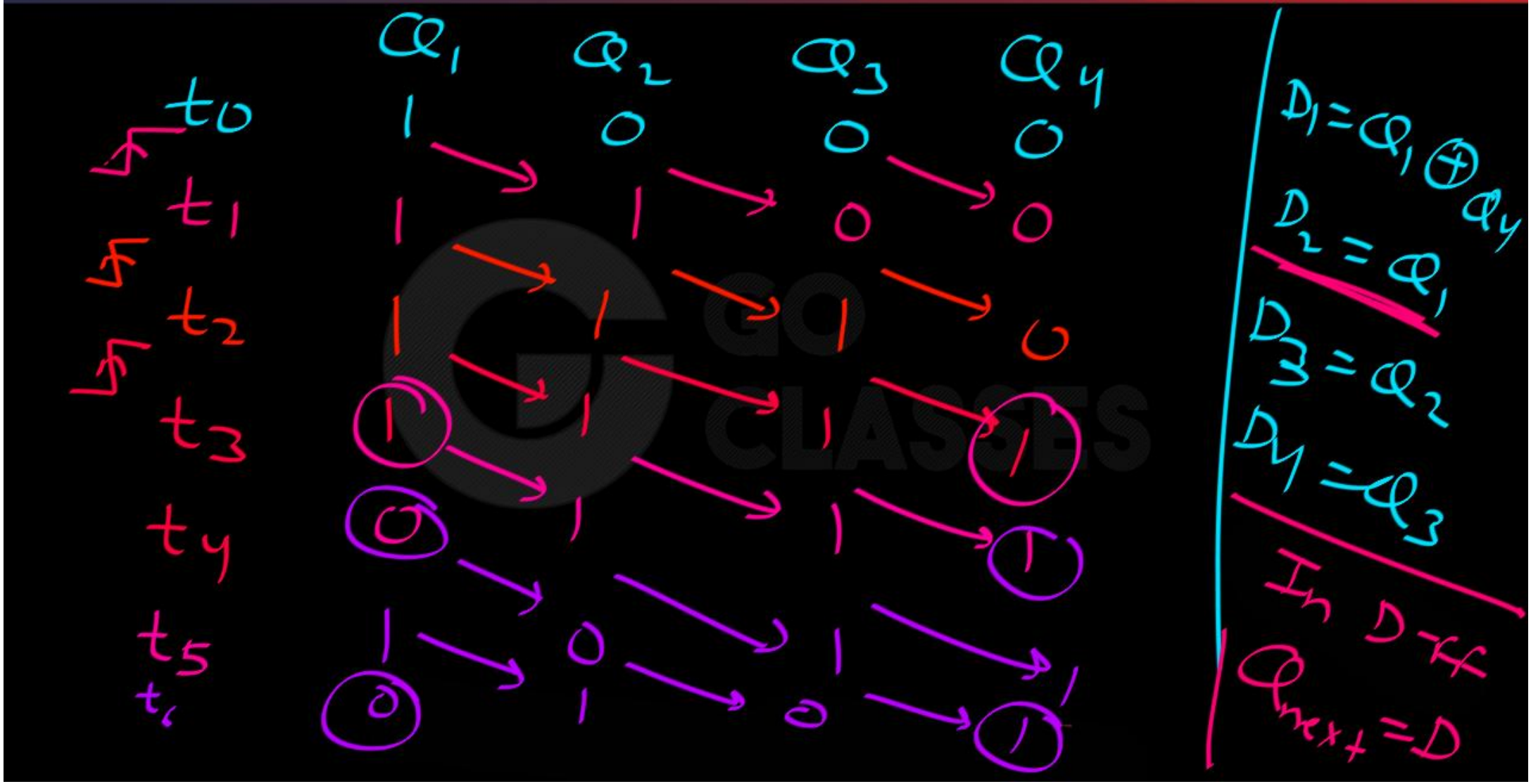



Q_1 Q_2 Q_3 Q_4
t_0 0 0 0 0
t_1 0 0 0 0
0000
$D_1 = Q_1 \oplus Q_4$
$D_2 = Q_1$
$D_3 = Q_2$
$D_4 = Q_3$
So Don't initialize as 0,0,0,0

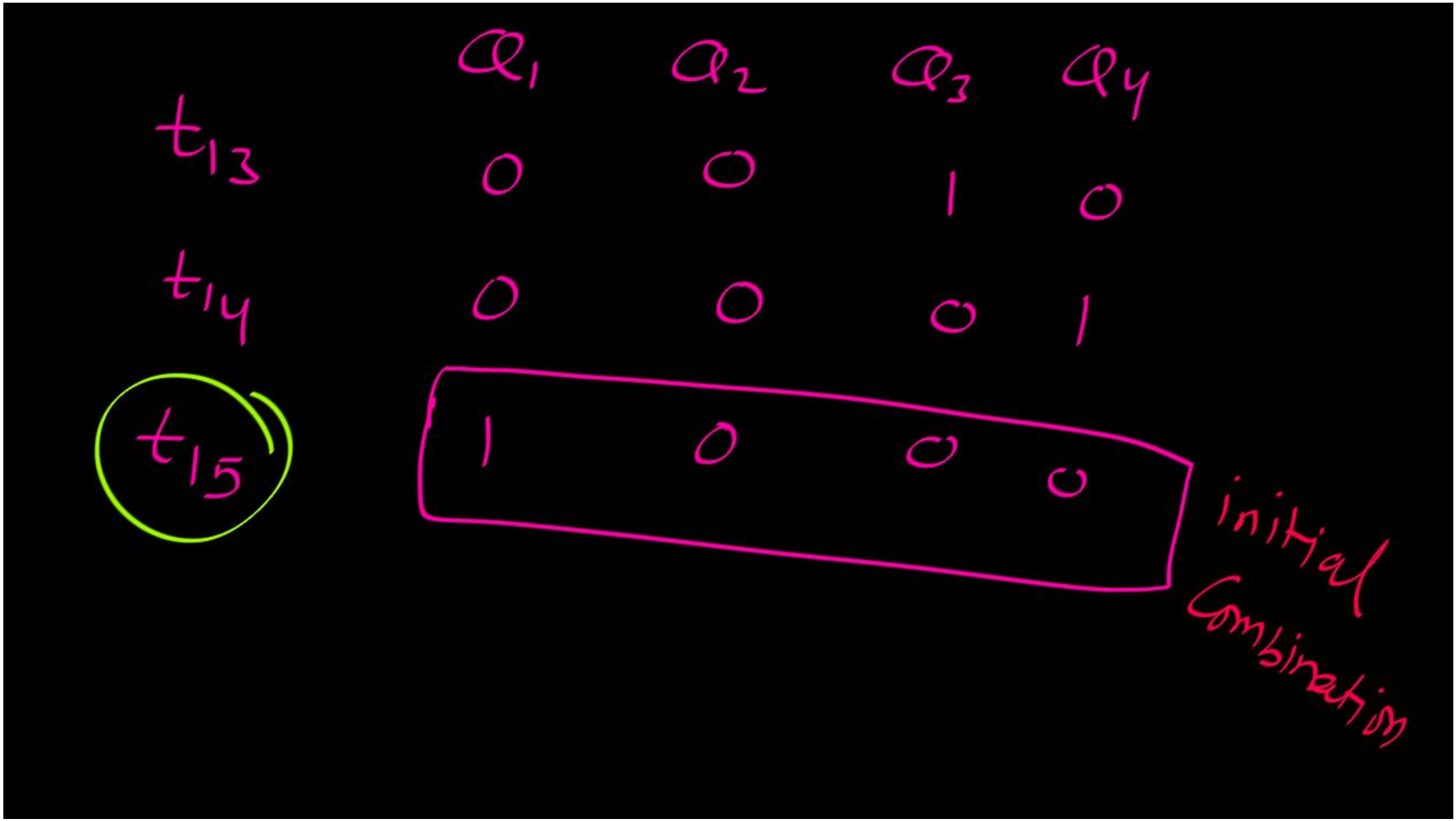
(e) Linear Feedback Shift Register (LFSR)

- It is basically a SISO right shift register, where the serial input is generated as a linear combination (exclusive-OR) of some of the flip-flop outputs.

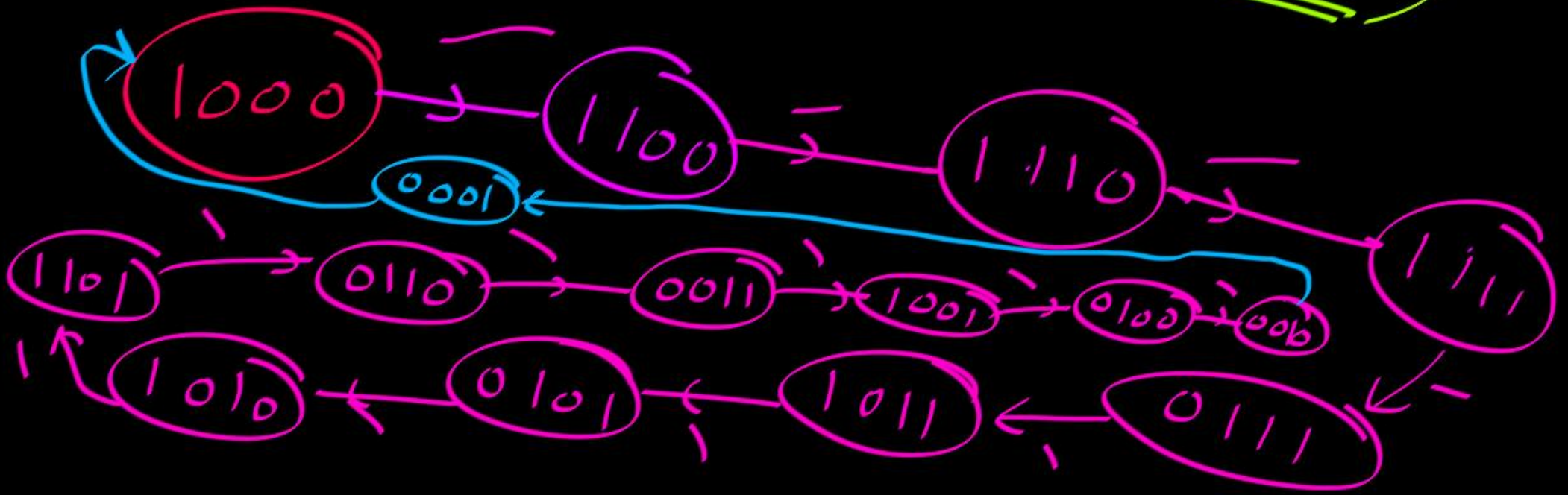


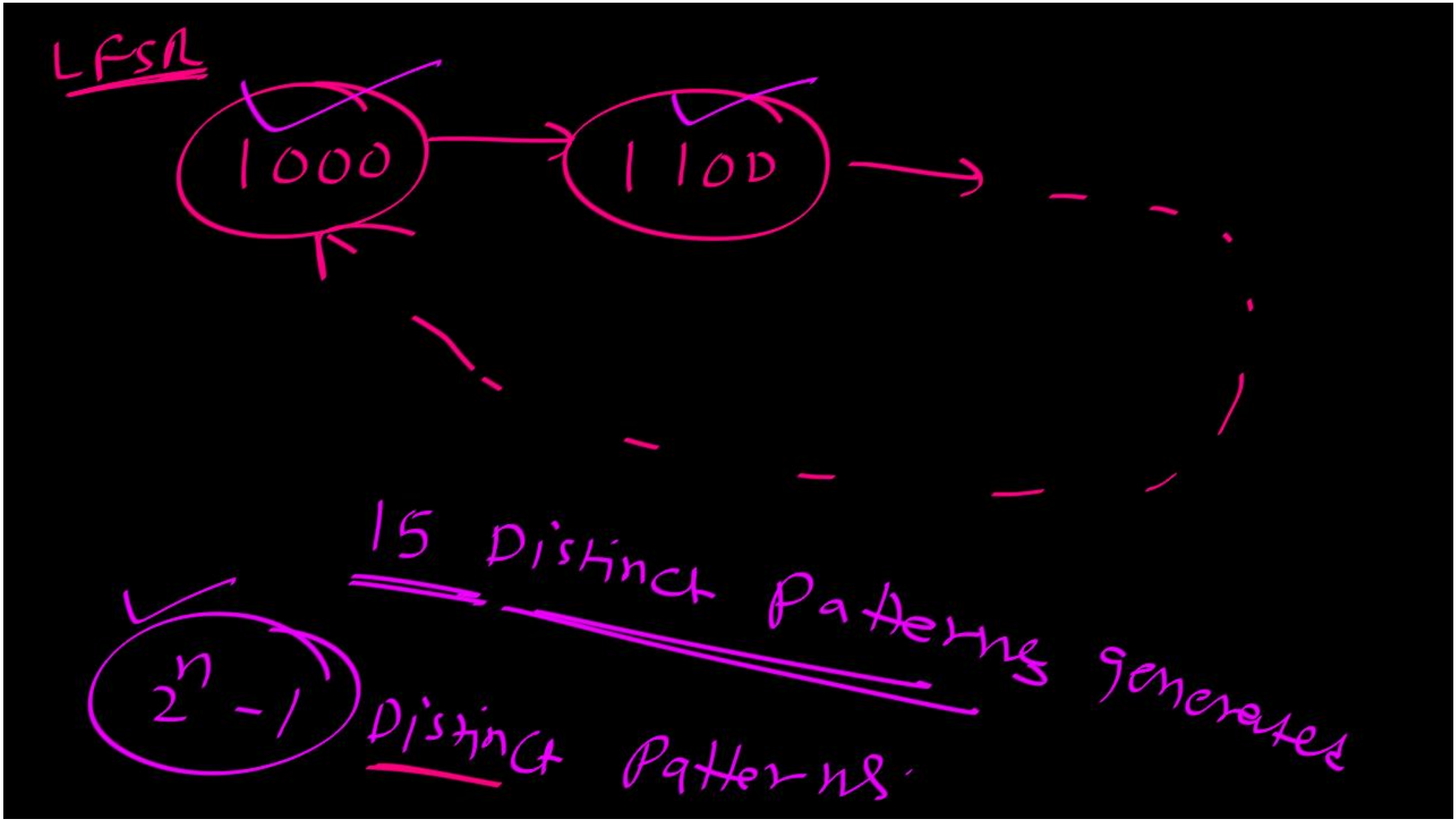

Q_1 Q_2 Q_3 Q_4
t_0 1 0 0 0
t_1 1 1 0 0
t_2 1 1 1 0
t_3 1 1 1 1
t_4 0 1 1 1
t_5 1 0 1 1
t_6 0 1 0 1
$D_1 = Q_1 \oplus Q_4$
$D_2 = Q_1$
$D_3 = Q_2$
$D_4 = Q_3$
In D-FF
$Q_{next} = D$

	Q_1	Q_2	Q_3	Q_4
t_7	1	0	1	0
t_8	1	1	0	1
t_9	0	1	1	0
t_{10}	0	0	1	1
t_{11}	1	0	0	1
t_{12}	0	1	0	0


a_1 a_2 a_3 a_4
t_{13} 0 0 1 0
t_{14} 0 0 0 1
t_{15}
1 0 0 0
initial combination

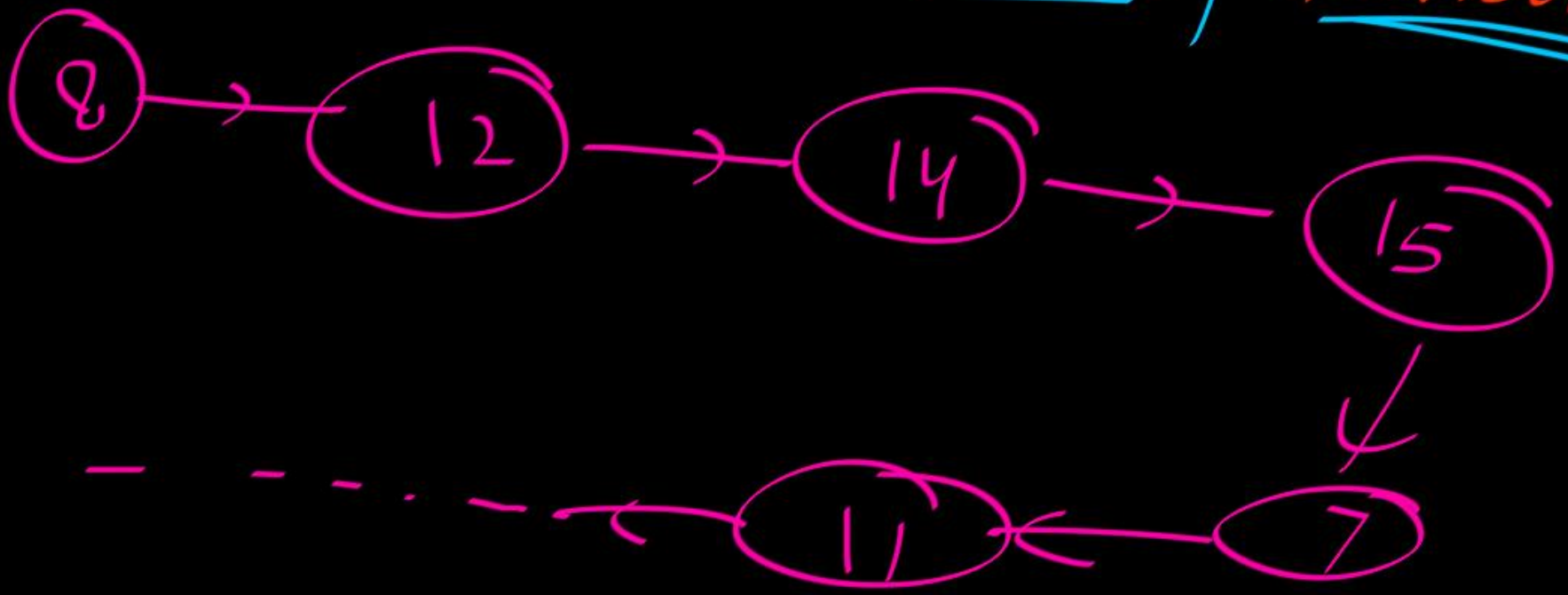
So After 15 Clock pulse, we get same initial Data (1000)

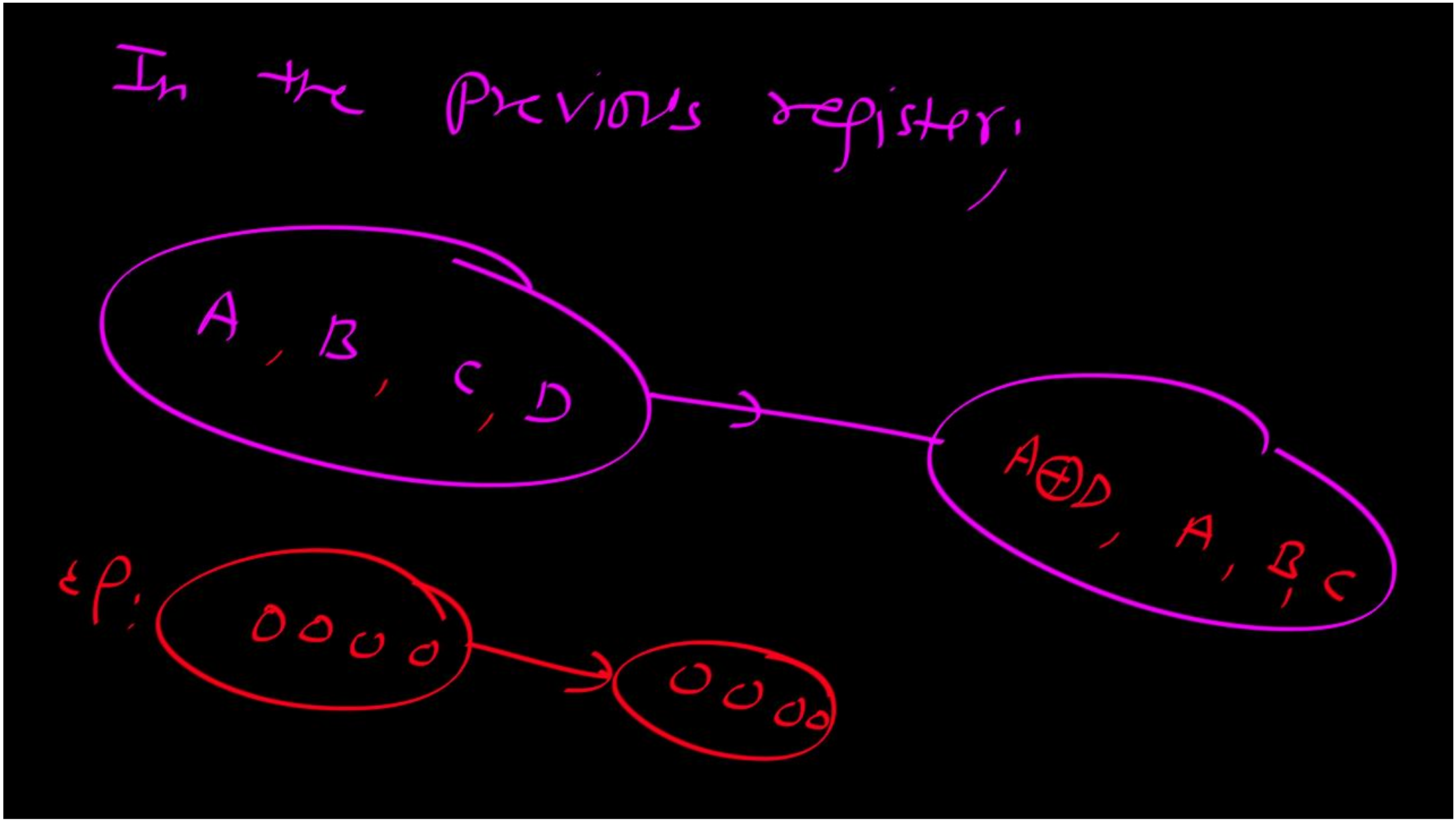



LFSR
1000
1101
15 Distinct Patterns generated
$2^n - 1$
Distinct Patterns.

Application of LFSR:

Random-number / sequence Generation



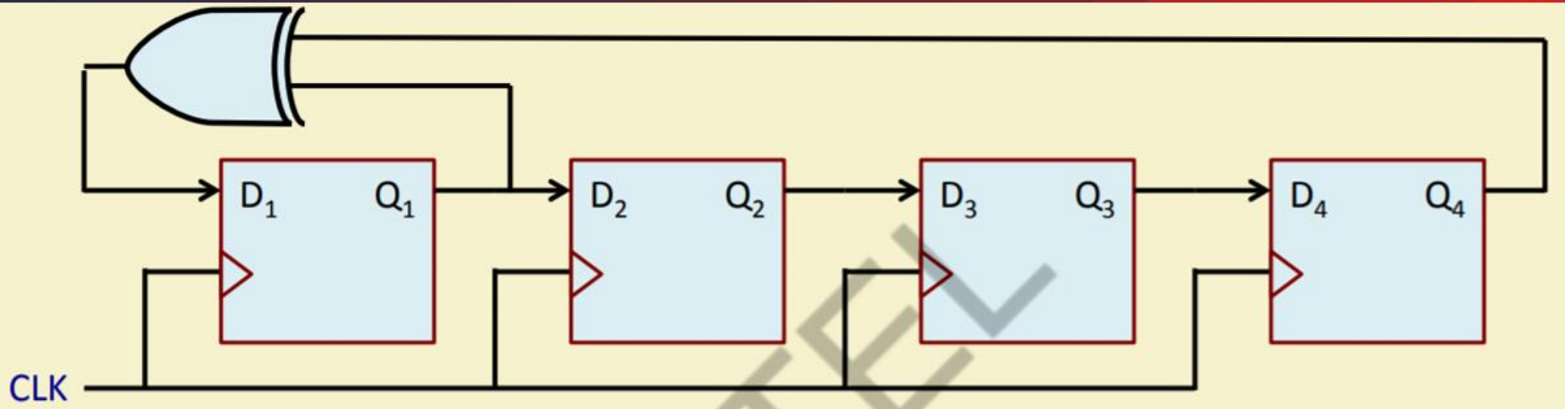

In the previous register,
A, B, C, D
A⊕D, A, B, C
EP:
0 0 0 0
0 0 0 0

flip-flop: → state = output

state = 1

sequential circuit :

state : 011



CLK pulses

	0	1	2	3	4	5	6	7	8	9	10	11	
Q_1	0	1	1	1	1	0	1	0	1	0	0	0	...
Q_2	0	0	1	1	1	1	0	0	0	1	0	0	...
Q_3	0	0	0	1	1	1	1	1	0	0	1	0	...
Q_4	1	0	0	0	1	1	1	1	1	0	0	1	...

LFSR

- The LFSR is typically initialized to *1 0 0 … 0*.
- If the taps are properly chosen, an *n*-bit LFSR can generate 2^n-1 distinct patterns without repeating.
  - The all-0 pattern cannot be generated.
- The LFSR generated patterns has very good randomness.
  - Useful in many applications.

Register → n-bit Data storage Device

- PIPO
- shift reg
  - SISO
  - SIPO
  - PISO
  - bidirectional, universal, LFSR

Specification of Register type :

2

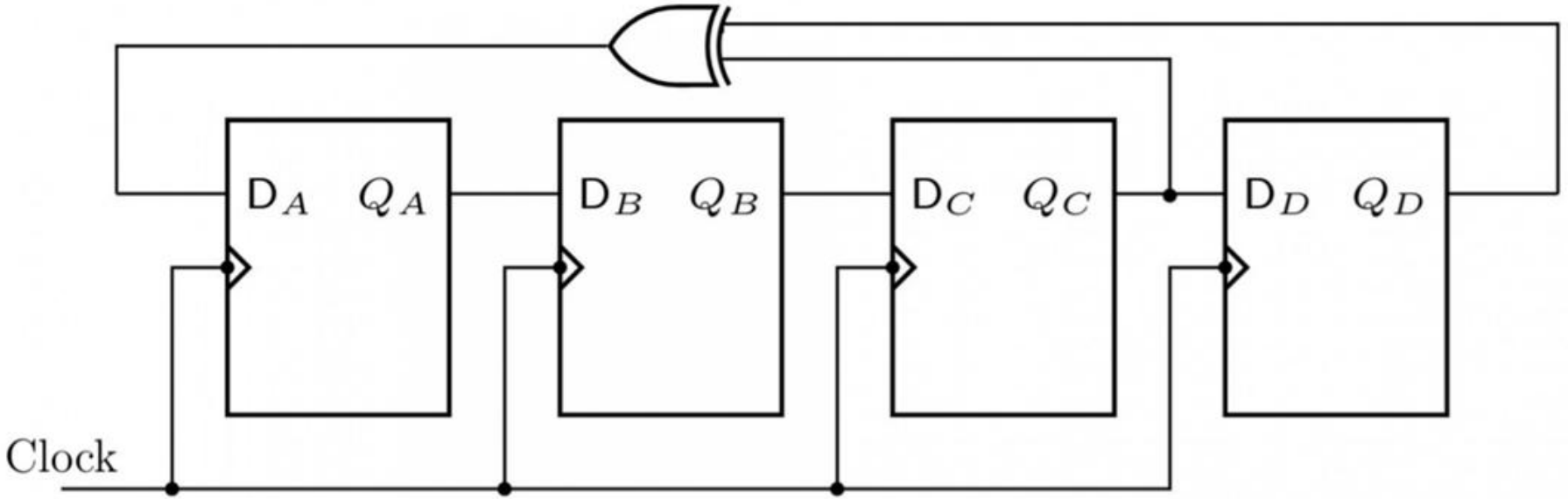
implement

4.31.1 Shift Registers: GATE CSE 1987 | Question: 13-a

top

https://gateoverflow.in/82607

The below figure shows four D-type flip-flops connected as a shift register using a XOR gate. The initial state and three subsequent states for three clock pulses are also given.



State	Q_A	Q_B	Q_C	Q_D
Initial	1	1	1	1
After the first clock	0	1	1	1
After the second clock	0	0	1	1
After the third clock	0	0	0	1

The state $Q_AQ_BQ_CQ_D$ after the fourth clock pulse is

A. 0000

B. 1111
C. 1001
D. 1000

$D_A = Q_C \oplus Q_D$; In D-ff

$Q_{next} = D$

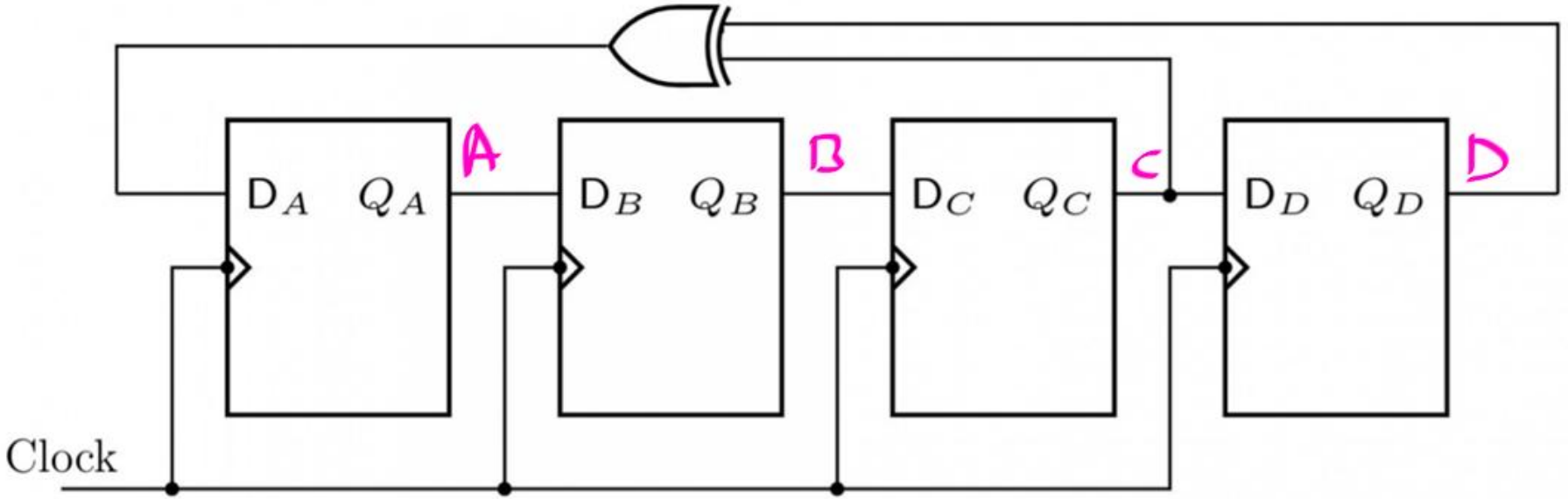
next state ↓ present input ↓

Q_A, Q_B, Q_C, Q_D → $Q_C \oplus Q_D, Q_A, Q_B, Q_C$

4.31.1 Shift Registers: GATE CSE 1987 | Question: 13-a top

https://gateoverflow.in/82607

The below figure shows four D-type flip-flops connected as a shift register using a XOR gate. The initial state and three subsequent states for three clock pulses are also given.

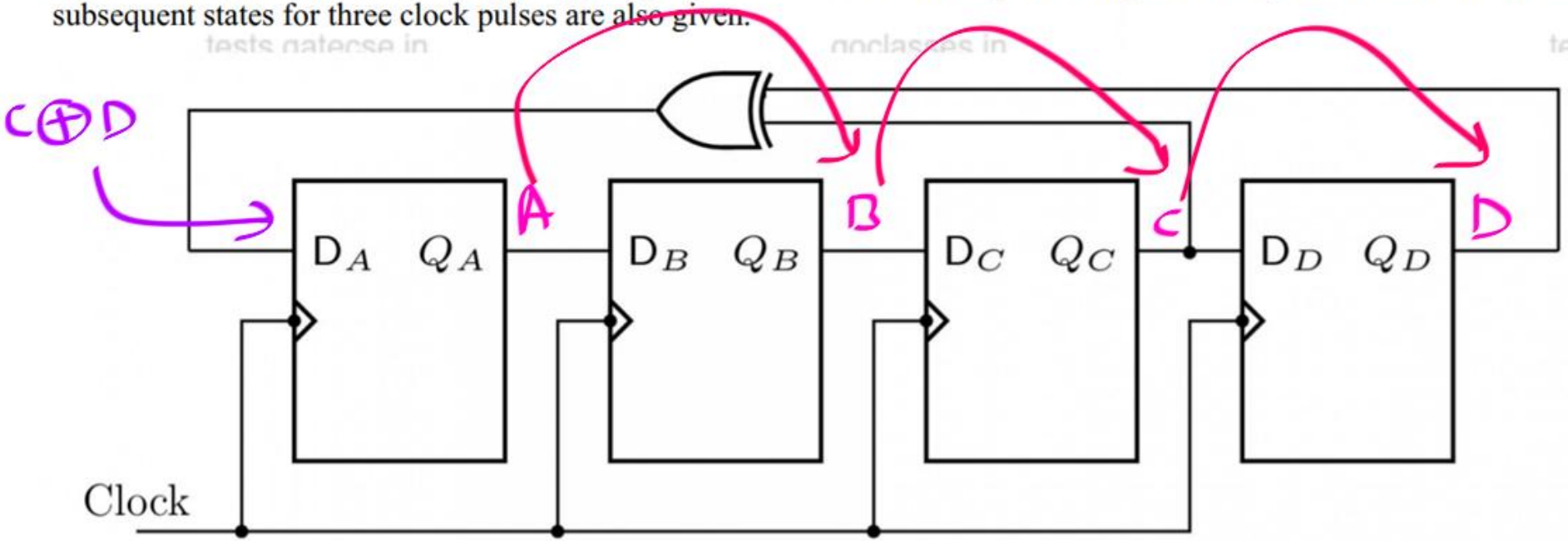


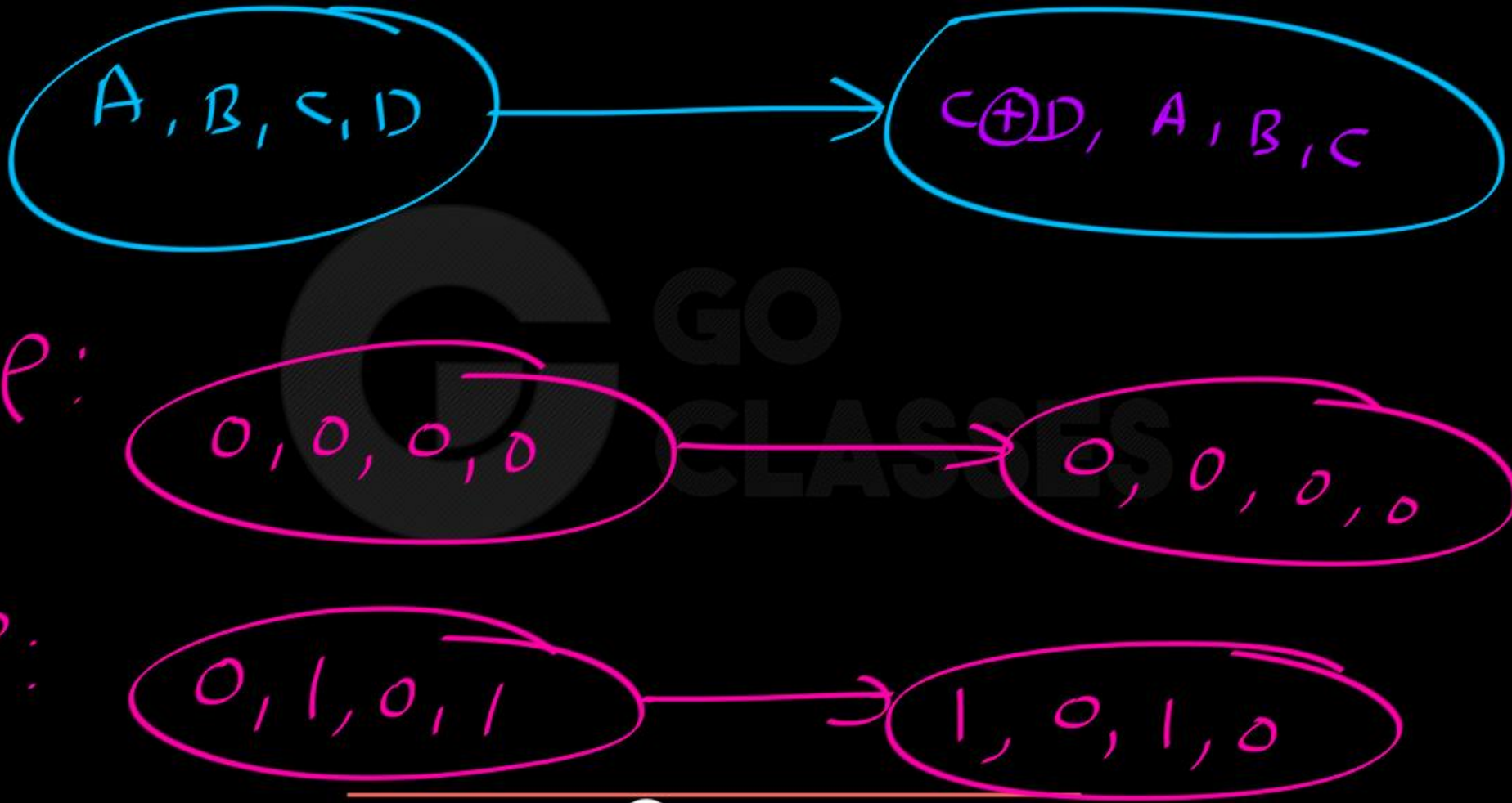
LFSR (linear feedback shift register)

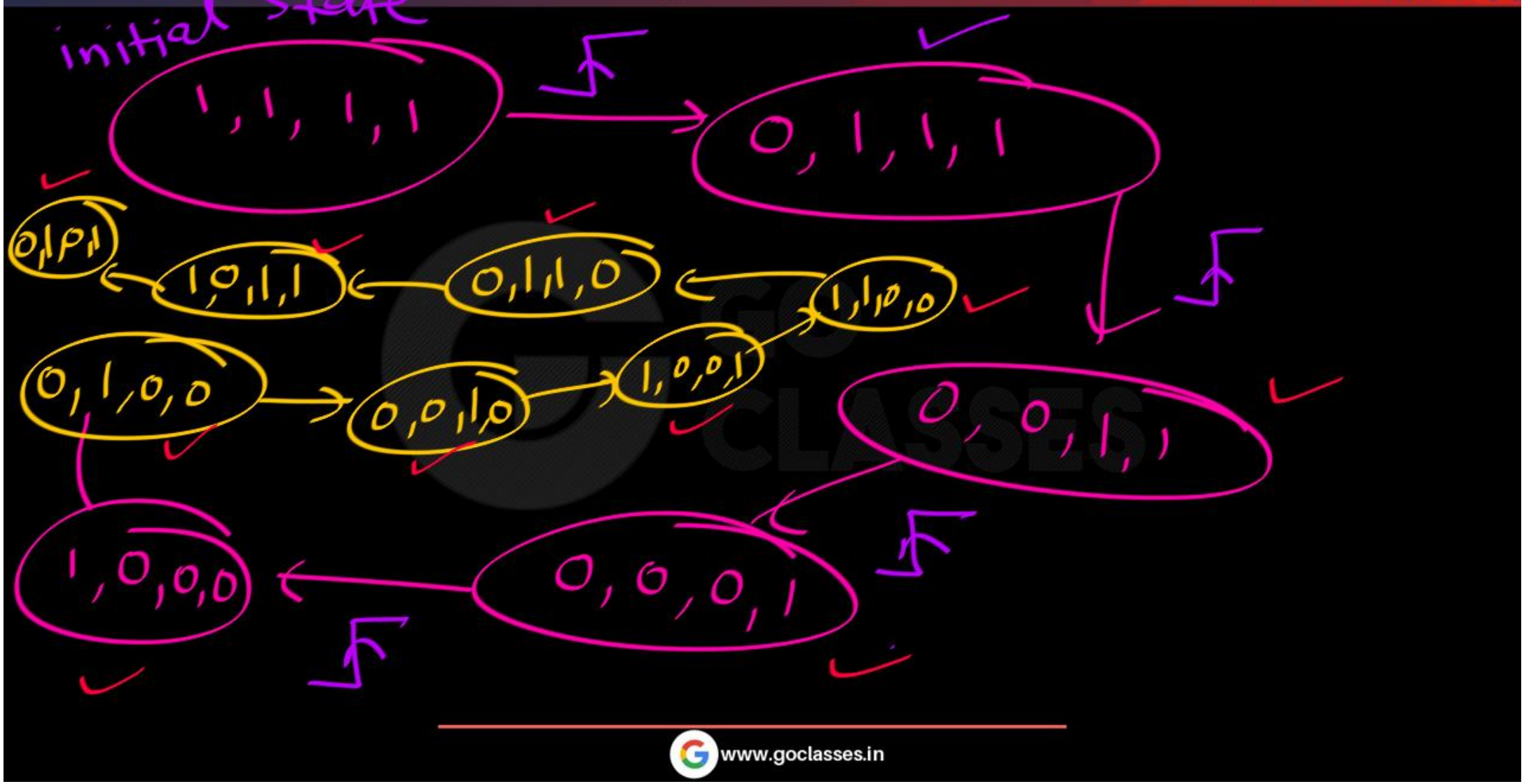
4.31.1 Shift Registers: GATE CSE 1987 | Question: 13-a top

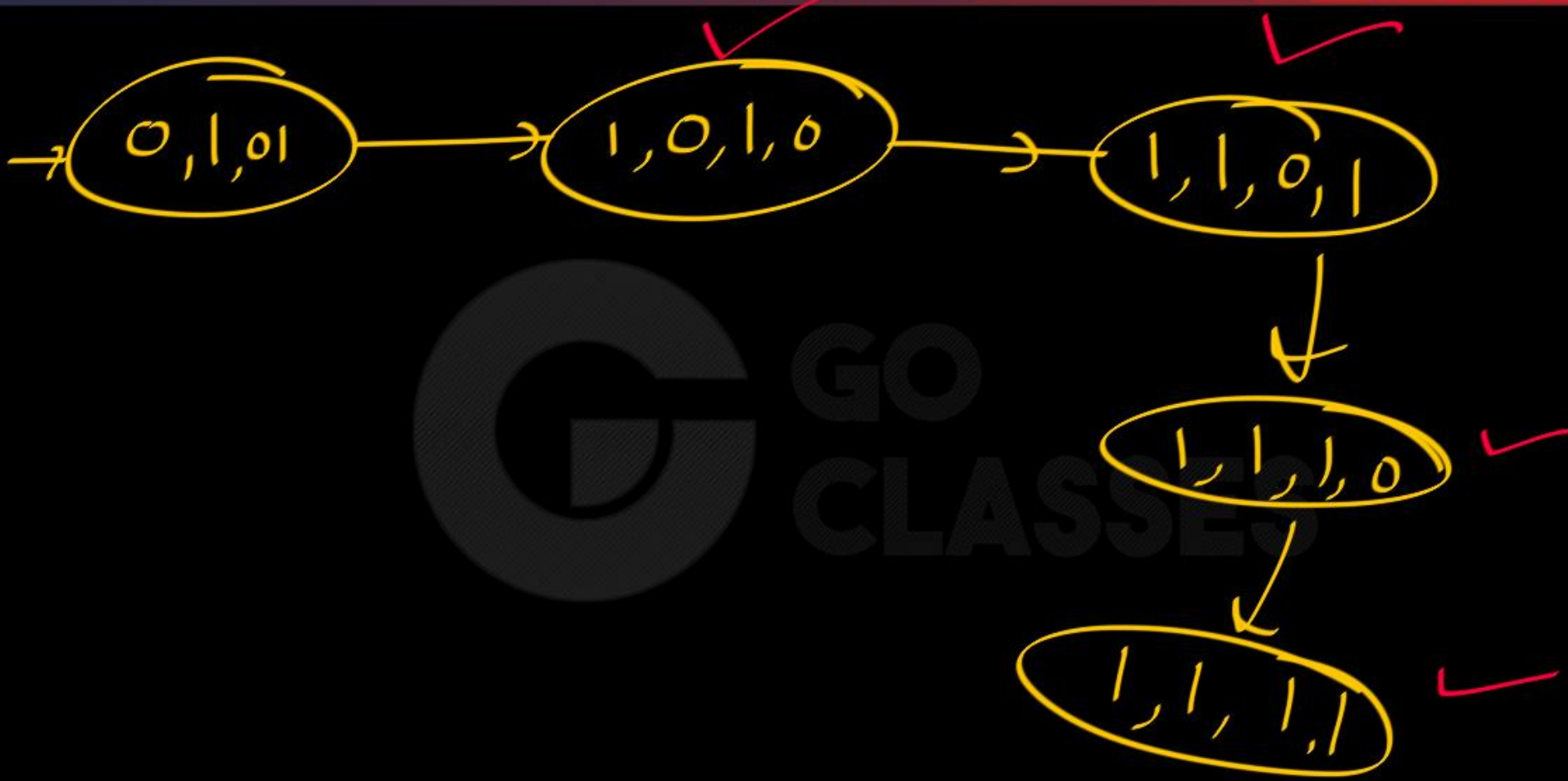
https://gateoverflow.in/82607

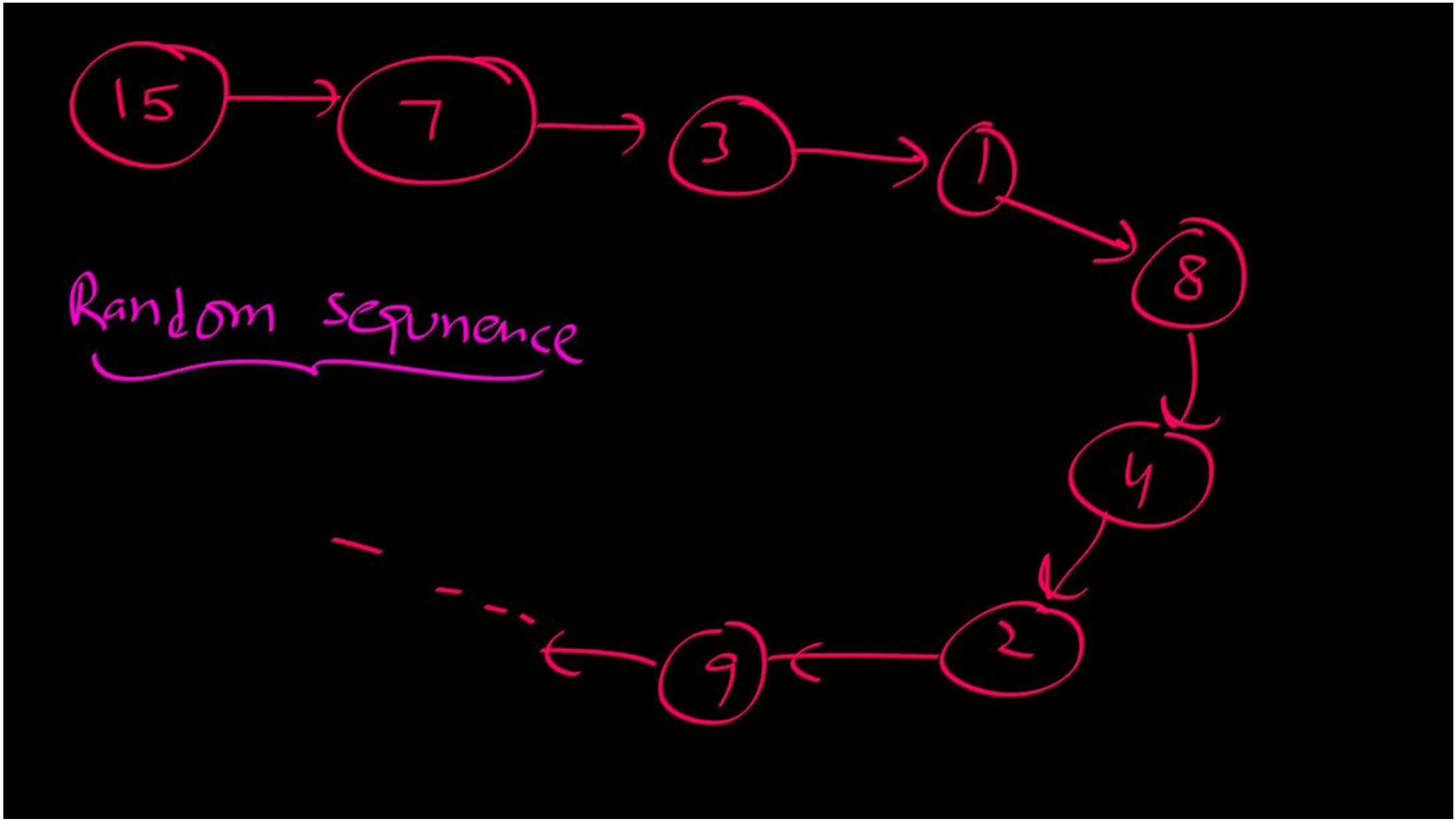
The below figure shows four D-type flip-flops connected as a shift register using a XOR gate. The initial state and three subsequent states for three clock pulses are also given.




A, B, C, D
C⊕D, A, B, C
Ex:
0, 0, 0, 0
0, 0, 0, 0
Ex:
0, 1, 0, 1
1, 0, 1, 0


initial state
1, 1, 1, 1
0, 1, 1, 1
0, 1, 1, 0
0, 1, 0, 0
0, 0, 0, 1
1, 0, 0, 0


0,1,01
1,0,1,0
1,1,0,1
1,1,1,0
1,1,1,1


15
7
3
1
8
4
2
9
Random sequence

After 15 Clock Pulses

So, 15 Distinct patterns are generated.

Application of LFSR:

Random sequence Generator.
" Numbers "

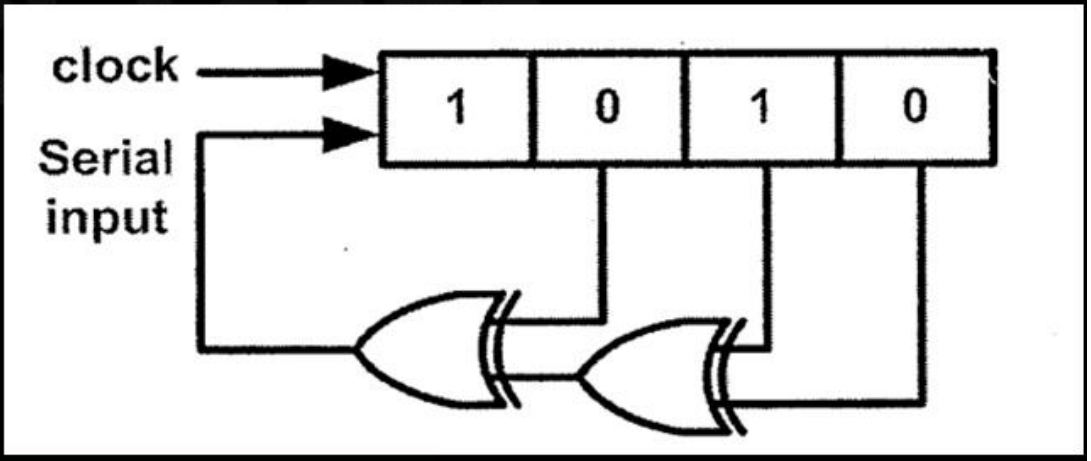
ISRO 2010-ECE Shift register

ISRO 2007- ECE Shift Register

The shift register shown in the given figure is initially loaded with the bit pattern 1010. Subsequently the shift register is clocked, and with each clock pulse the pattern gets shifted by one bit position to the right. With each shift, the bit at the serial input is pushed to the left most position (MSB). After how many clock pulses will the content of the shift register become 1010 again?

a) 3
b) 7
c) 11
d) 15

https://gateoverflow.in/120382/Isro-2007-ece-shift-register
https://gateoverflow.in/120043/Isro-2010-ece-shift-register



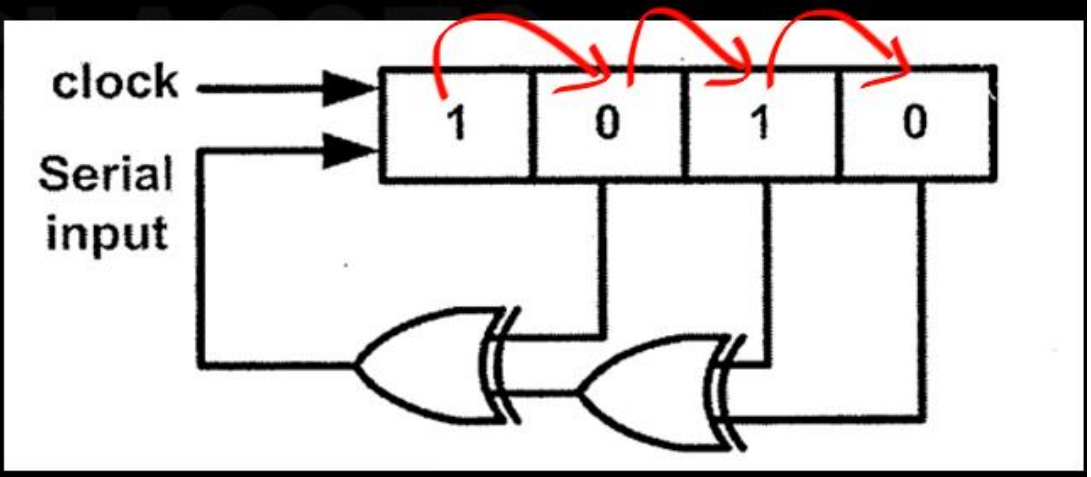
ISRO 2010-ECE Shift register

ISRO 2007- ECE Shift Register

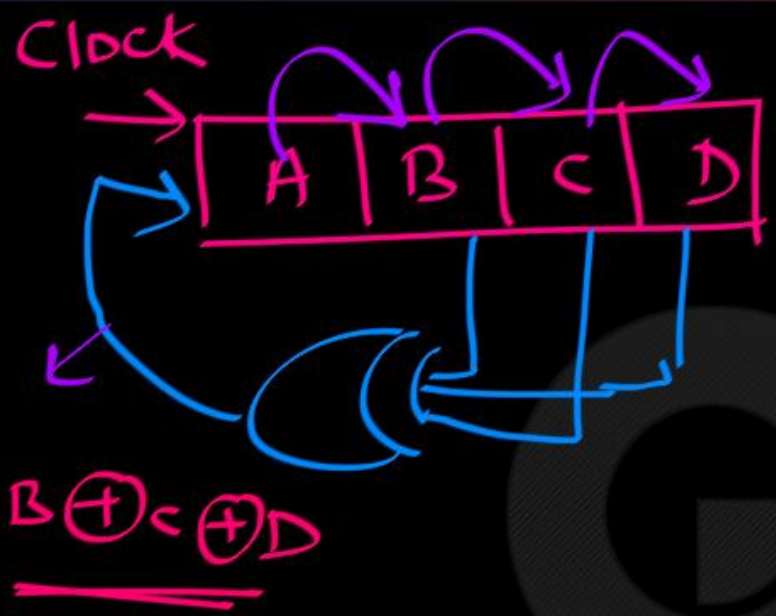
The shift register shown in the given figure is initially loaded with the bit pattern 1010. Subsequently the shift register is clocked, and with each clock pulse the pattern gets shifted by one bit position to the right. With each shift, the bit at the serial input is pushed to the left most position (MSB). After how many clock pulses will the content of the shift register become 1010 again?

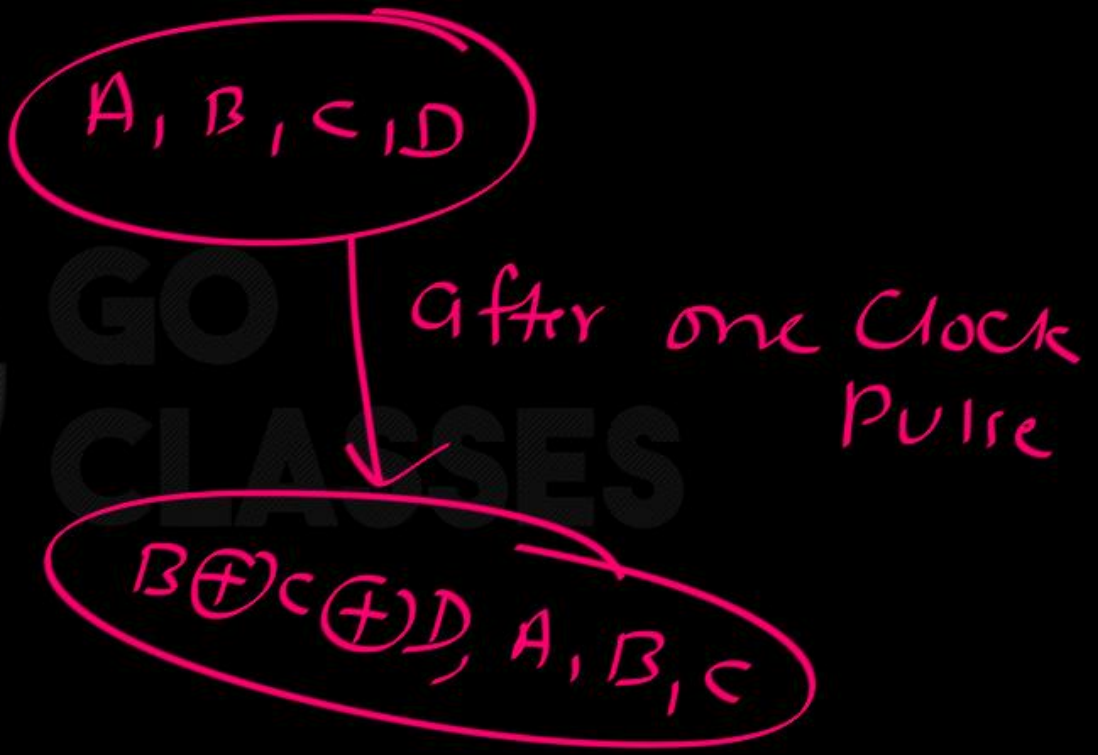
a) 3
b) 7
c) 11
d) 15

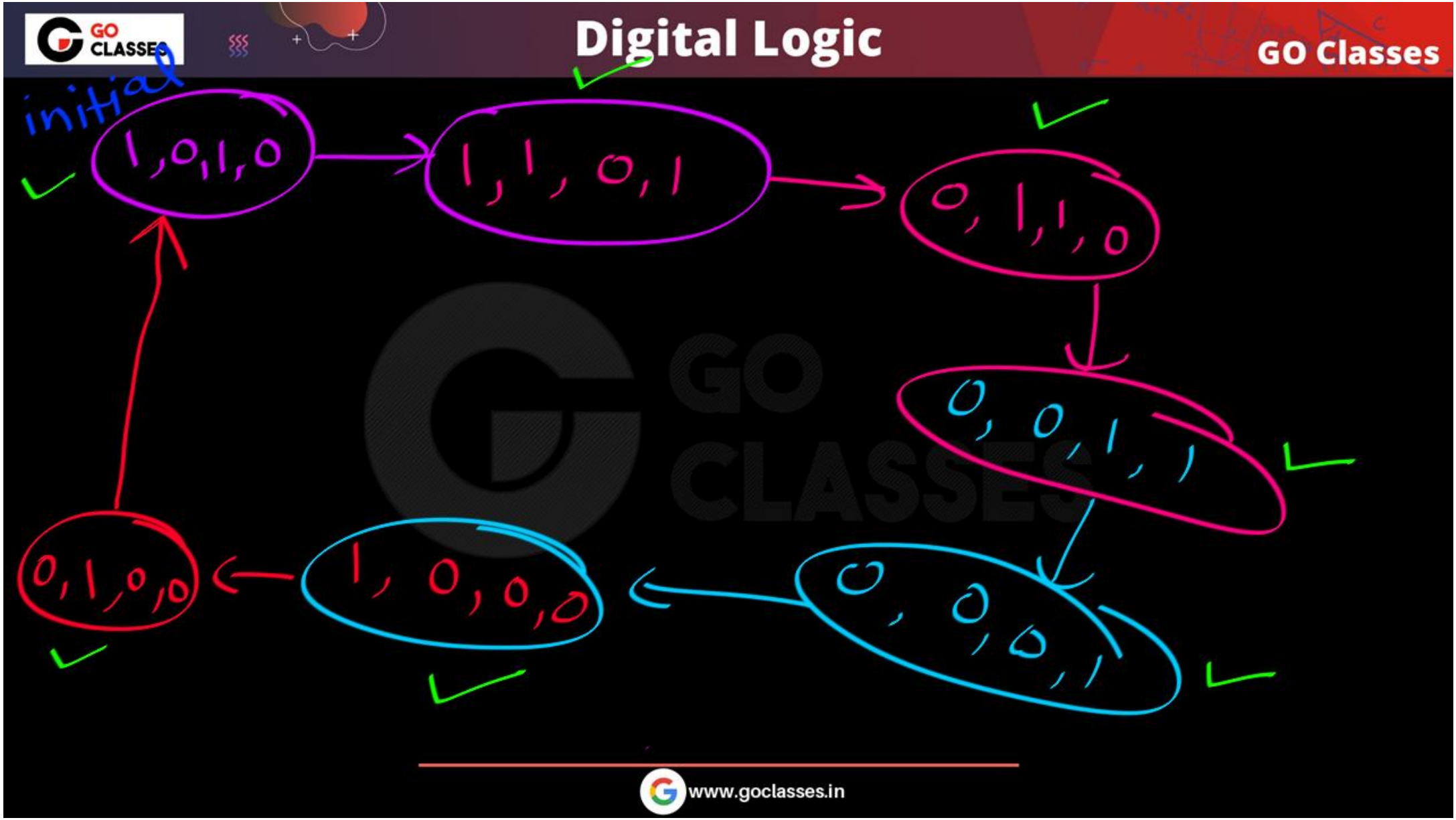
LFSR
linear feedback shift register

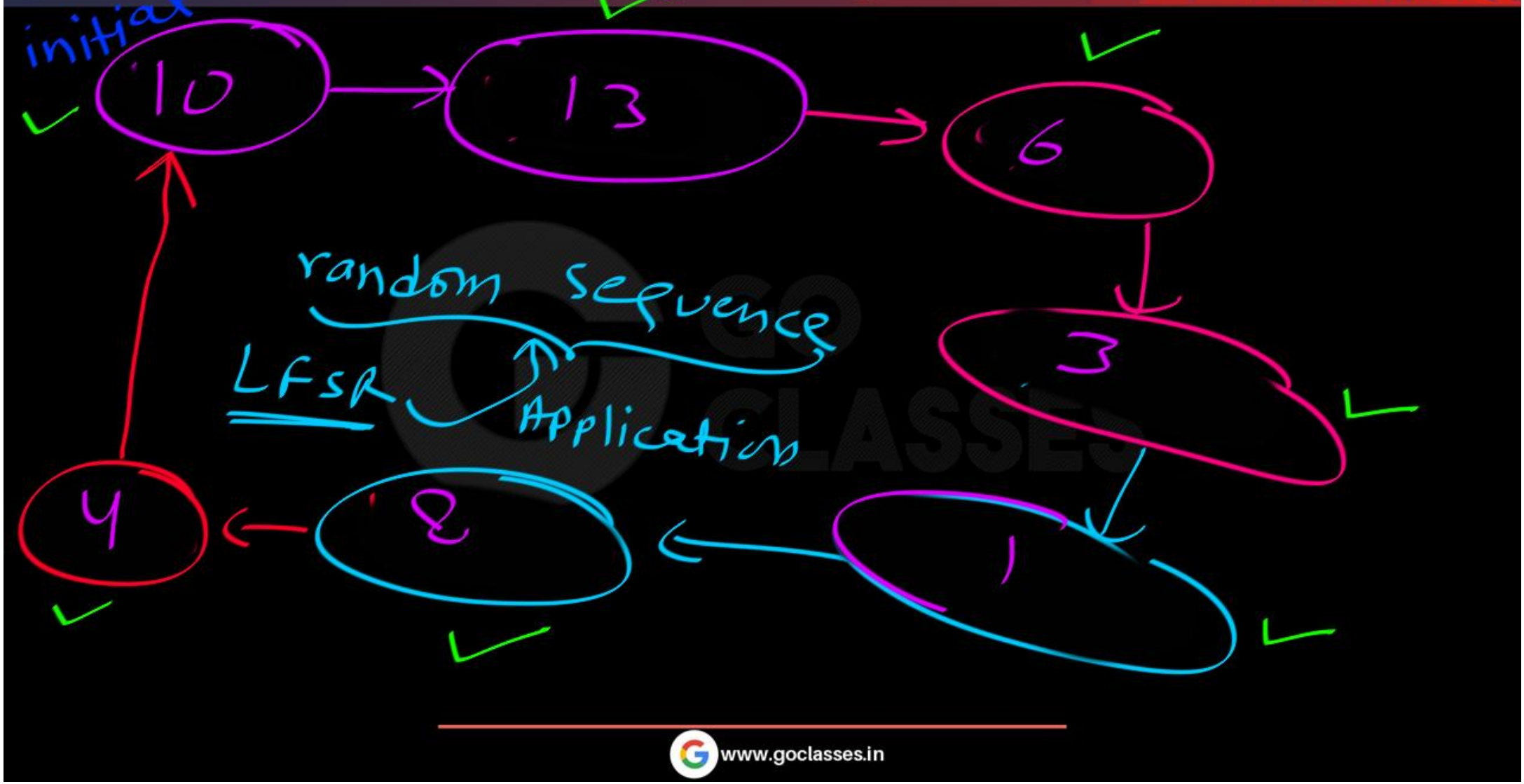


https://gateoverflow.in/120382/Isro-2007-ece-shift-register
https://gateoverflow.in/120043/Isro-2010-ece-shift-register


Clock
A
B
C
D
B⊕C⊕D


A, B, C, D
after one Clock Pulse
B⊕C⊕D, A, B, C


initial
1,0,1,0
1,1,0,1
0,1,1,0
0,0,1,1
0,0,0,1
1,0,0,0
0,1,0,0


initial
10
13
6
3
1
8
4
random sequence
LFSR
Application

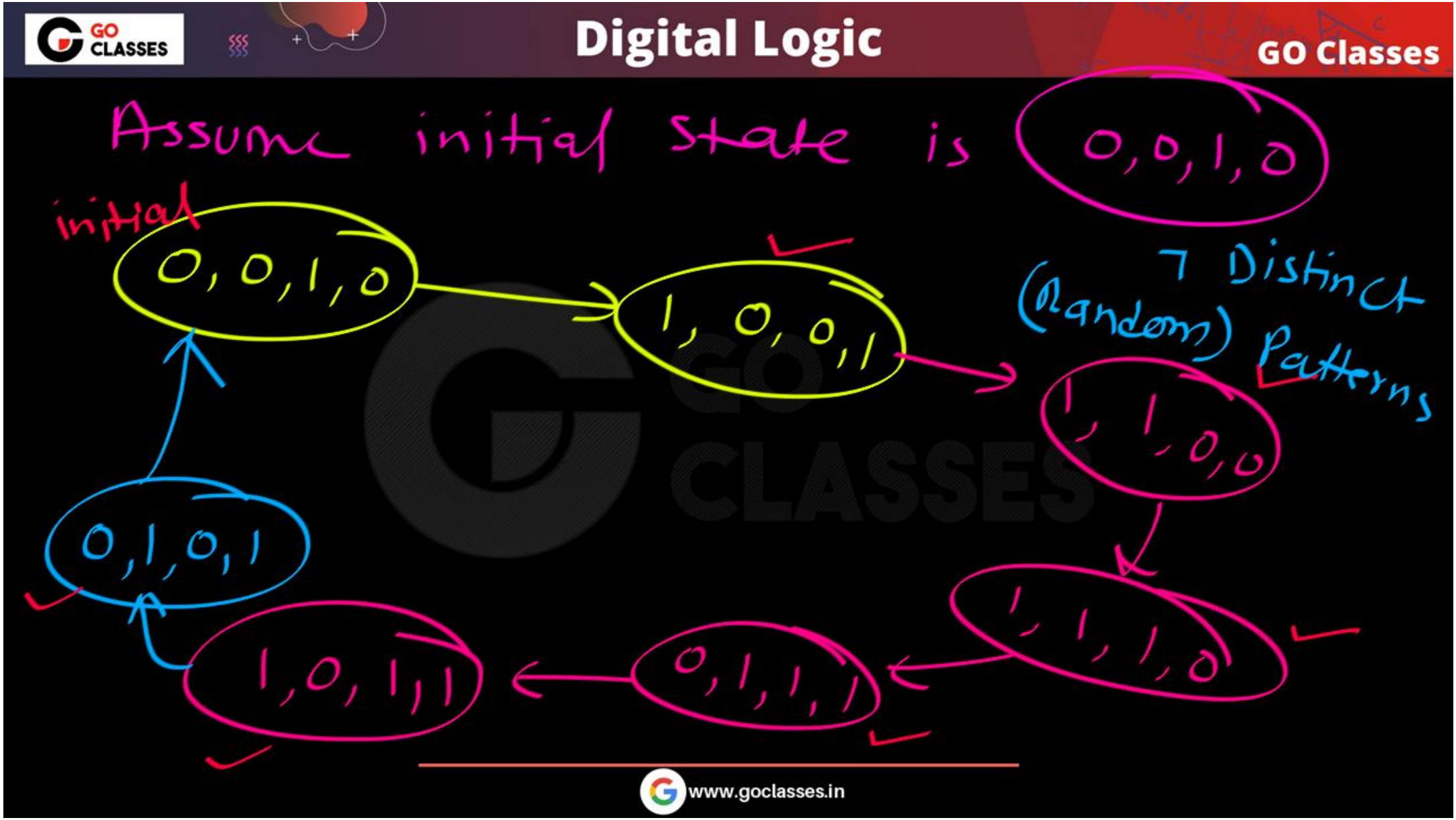
$a_1 \oplus a_2 \oplus a_3 = 1$ iff odd no. of inputs are 1.

$1 \oplus 1 \oplus 0 = 0$

$1 \oplus 1 \oplus 1 = 1$

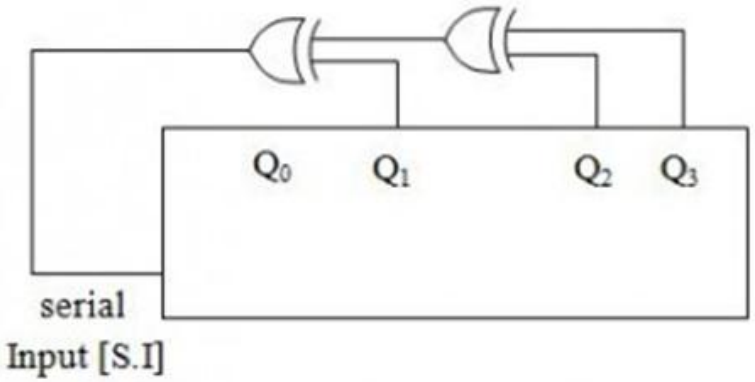
$1 \oplus 0 \oplus 0 = 1$

Exor becomes 1 iff odd no. of inputs are 1.


Assume initial state is
0,0,1,0
initial
0,0,1,0
1,0,0,1
7 Distinct
(Random) Patterns
1,1,0,0
0,1,0,1
1,1,1,0
1,0,1,1
0,1,1,1

1 Distinct Pattern

A shift Register that shifts the bit one position to the right at each clock pulse is initialized to 1100 [Q_0 Q_1 Q_2 Q_3] as shown in fig.



After which clock pulse will initial pattern reappear at the output

(A) 4 clock pulses

(B) 5 clock pulses

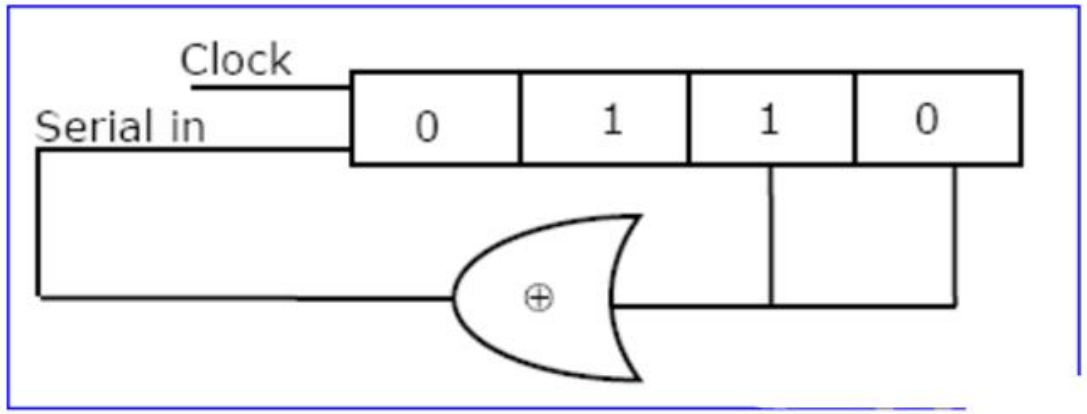
(C) 6 clock pulses

(D) 7 clock pulses

https://gateoverflow.in/176156/Digital-test-question-about-shift-register

GATE ECE 1992

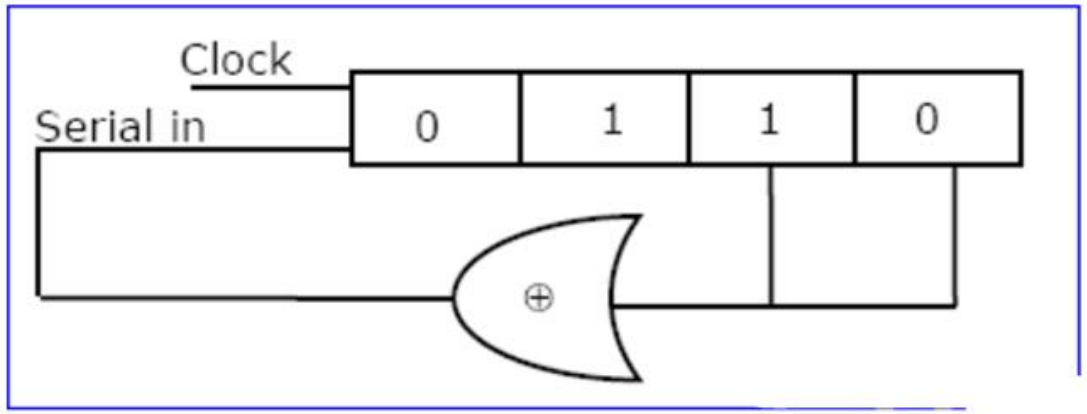
The initial contents of the 4 bit serial in serial out, right shift, shift register shown in figure, are 0110. After three clock pulses are applied, the contents of the shift register will be



a. 0000
b. 0101
c. 1010
d. 1111

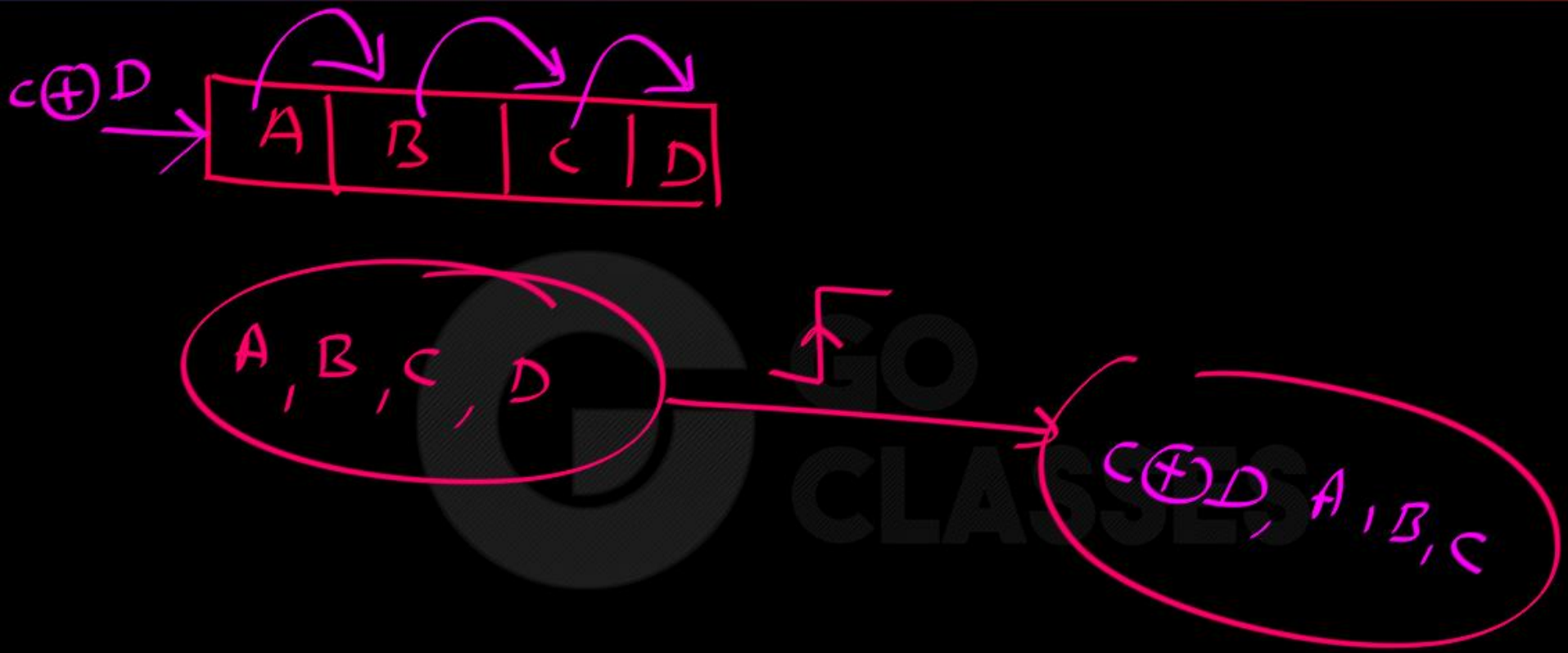
GATE ECE 1992

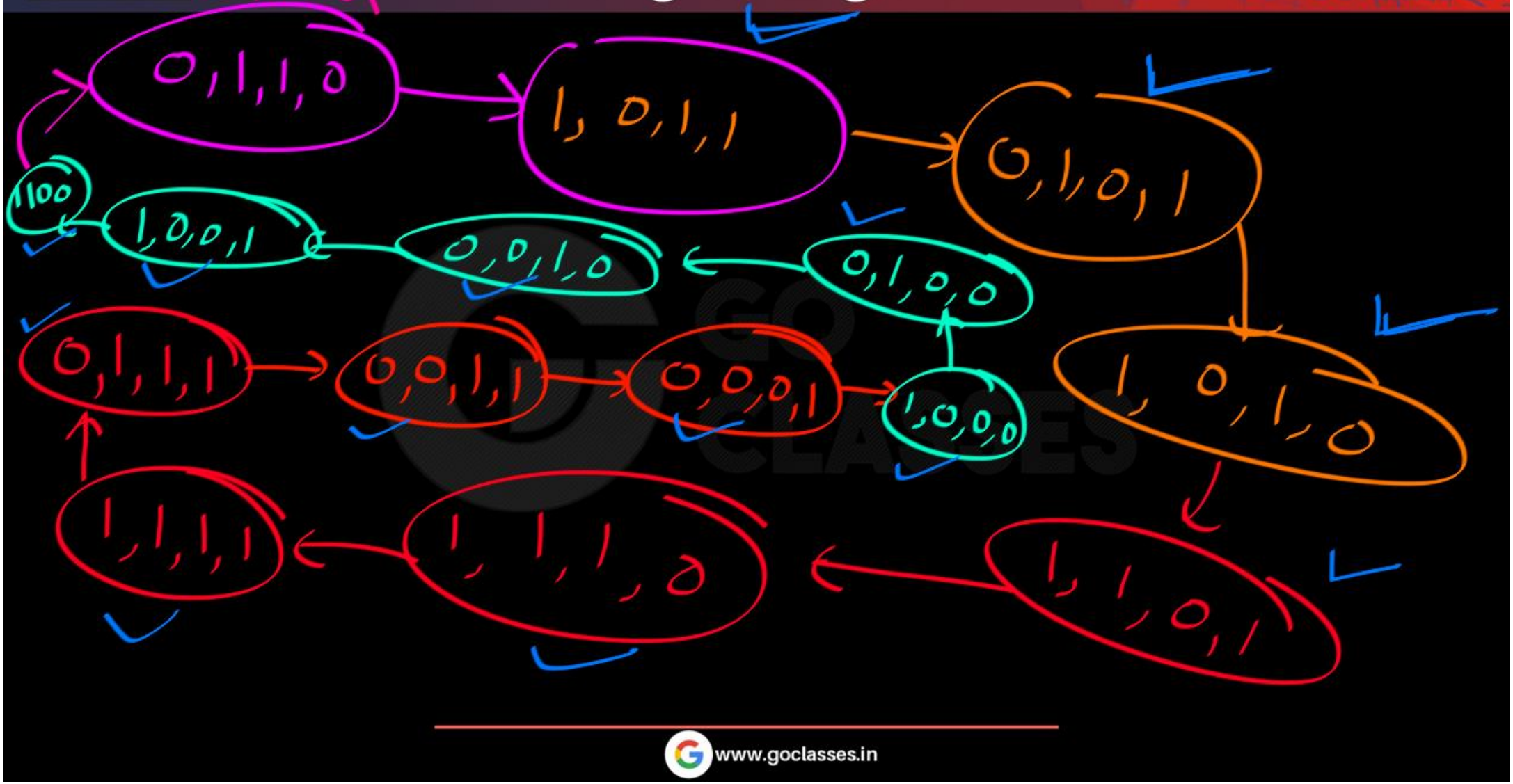
The initial contents of the 4 bit serial in serial out, right shift, shift register shown in figure, are 0110. After three clock pulses are applied, the contents of the shift register will be

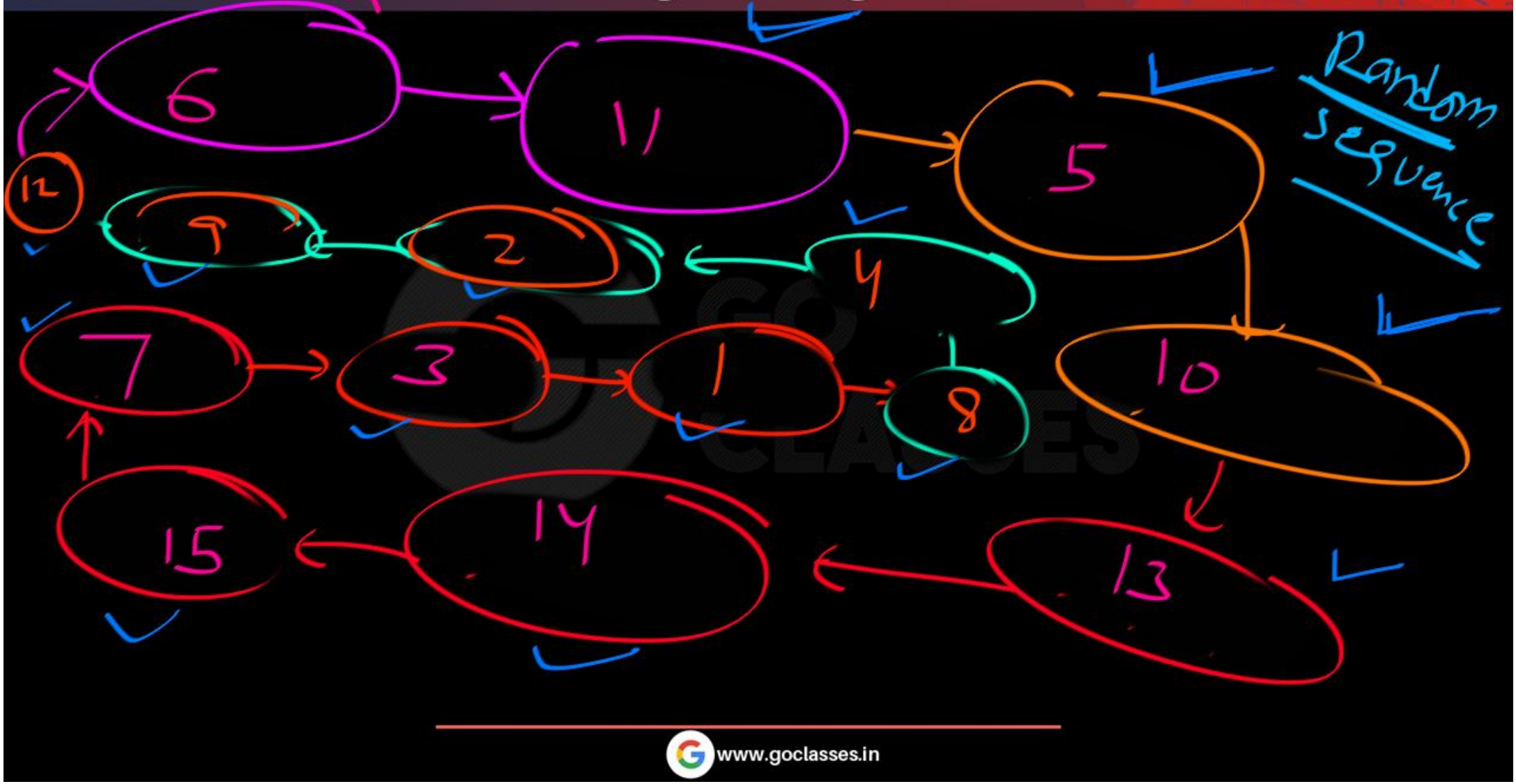


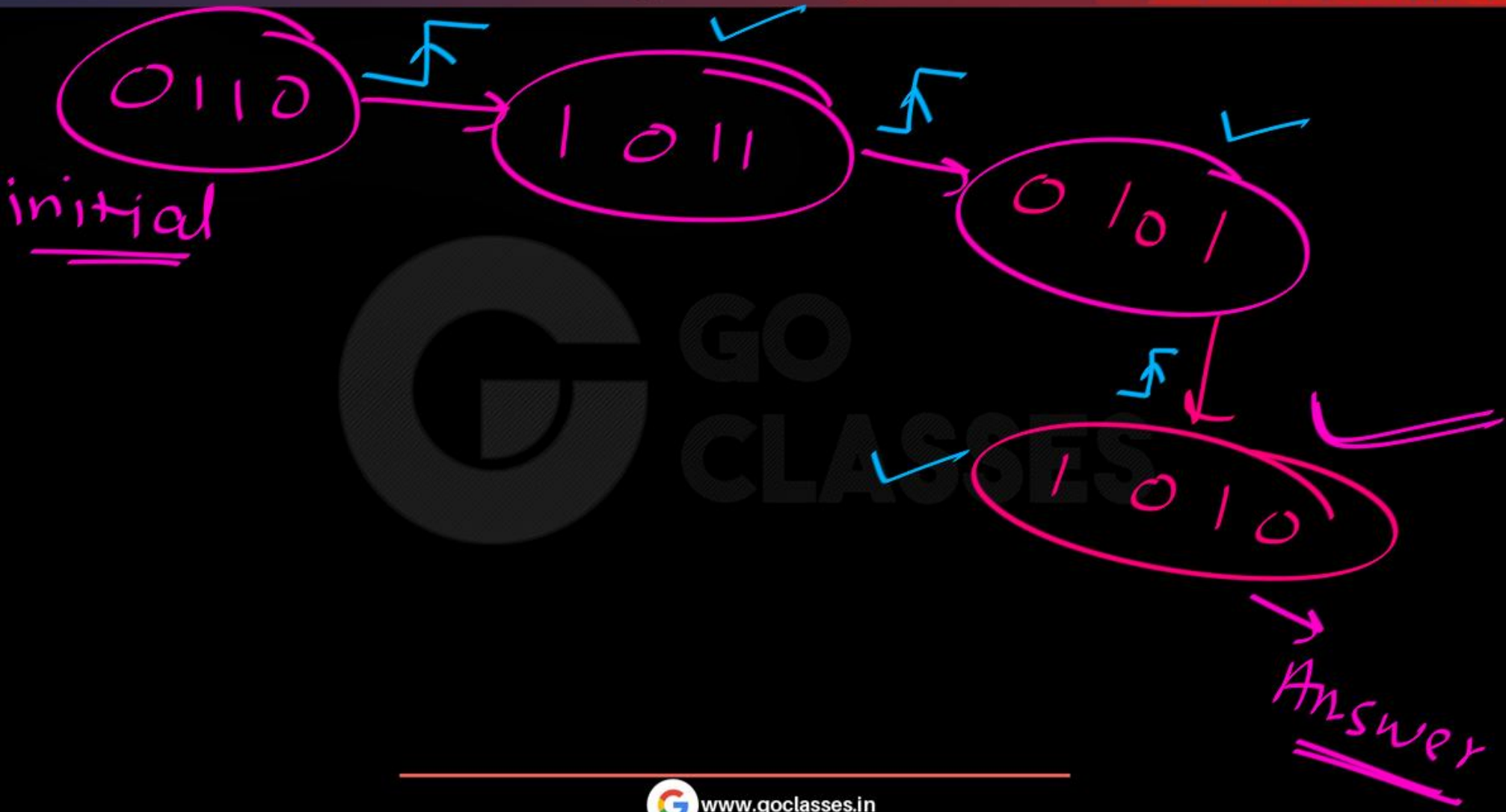
a. 0000
b. 0101
c. 1010
d. 1111

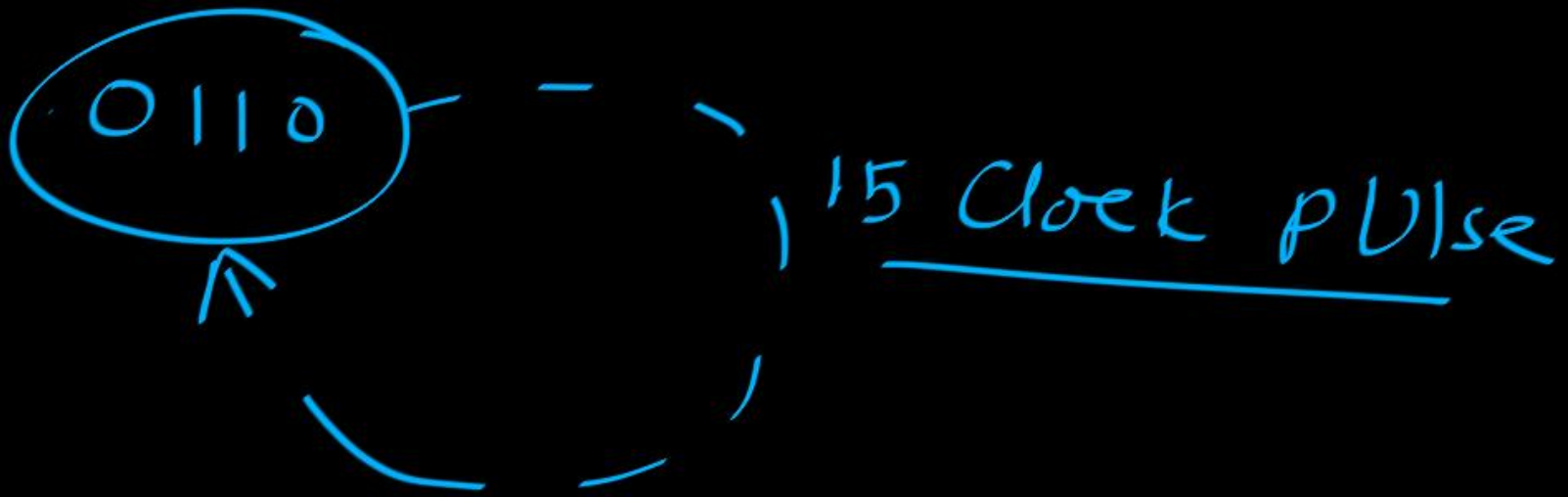
LFSR
linear feedback shift register
Randomness


C ⊕ D
A
B
C
D
A, B, C, D
C ⊕ D, A, B, C

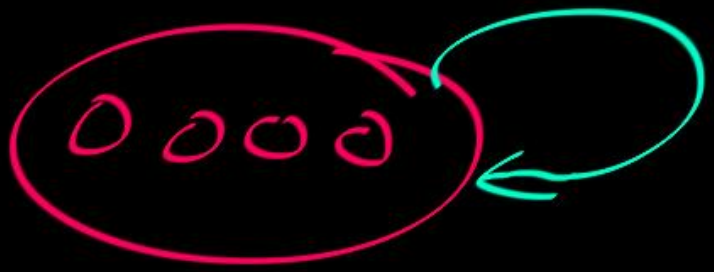

0,1,1,0
1,0,1,1
0,1,0,1
1100
1,0,0,1
0,0,1,0
0,1,0,0
0,1,1,1
0,0,1,1
0,0,0,1
1,0,0,0
1,0,1,0
1,1,1,1
1,1,1,0
1,1,0,1


6
11
5
Random sequence
12
9
2
4
7
3
1
8
10
15
14
13


0110
initial
1011
0101
1010
Answer



15 Distinct patterns Generated



4.31.2 Shift Registers: GATE CSE 1991 | Question: 06,a

top

https://gateoverflow.in/532

Using D flip-flop gates, design a parallel-in/serial-out shift register that shifts data from left to right with the following input lines:

i. Clock CLK

ii. Three parallel data inputs A, B, C

iii. Serial input S

iv. Control input load/$\overline{\text{SHIFT}}$.

4.31.2 Shift Registers: GATE CSE 1991 | Question: 06,a

top

https://gateoverflow.in/532

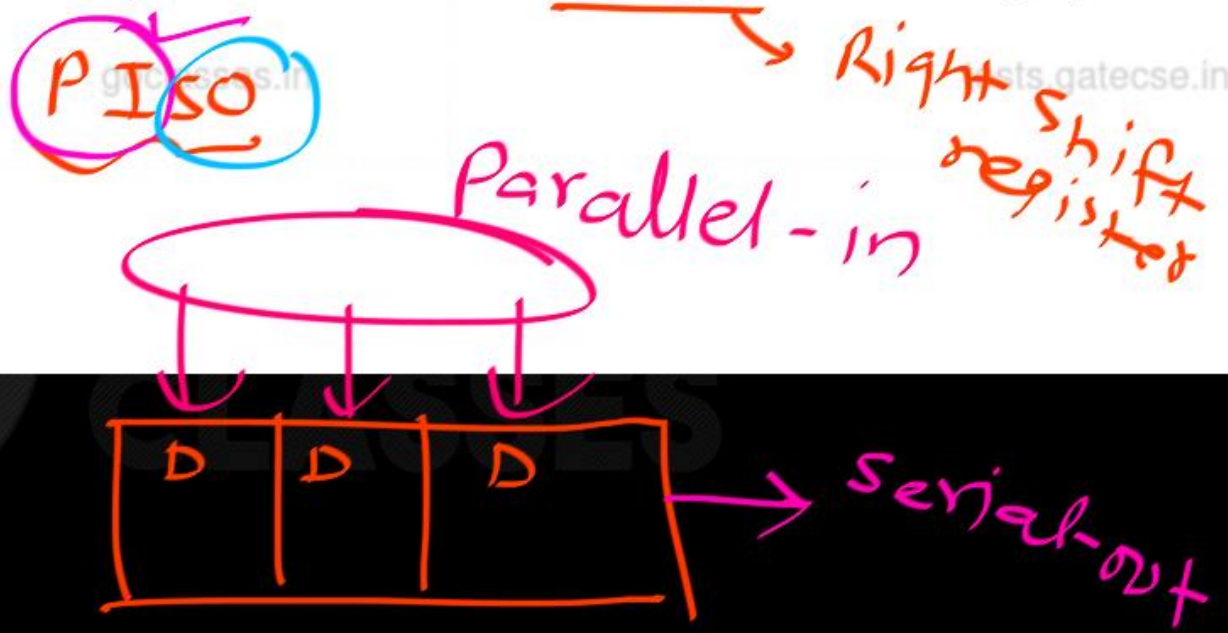
Using D flip-flop gates, design a parallel-in/serial-out shift register that shifts data from left to right with the following input lines:

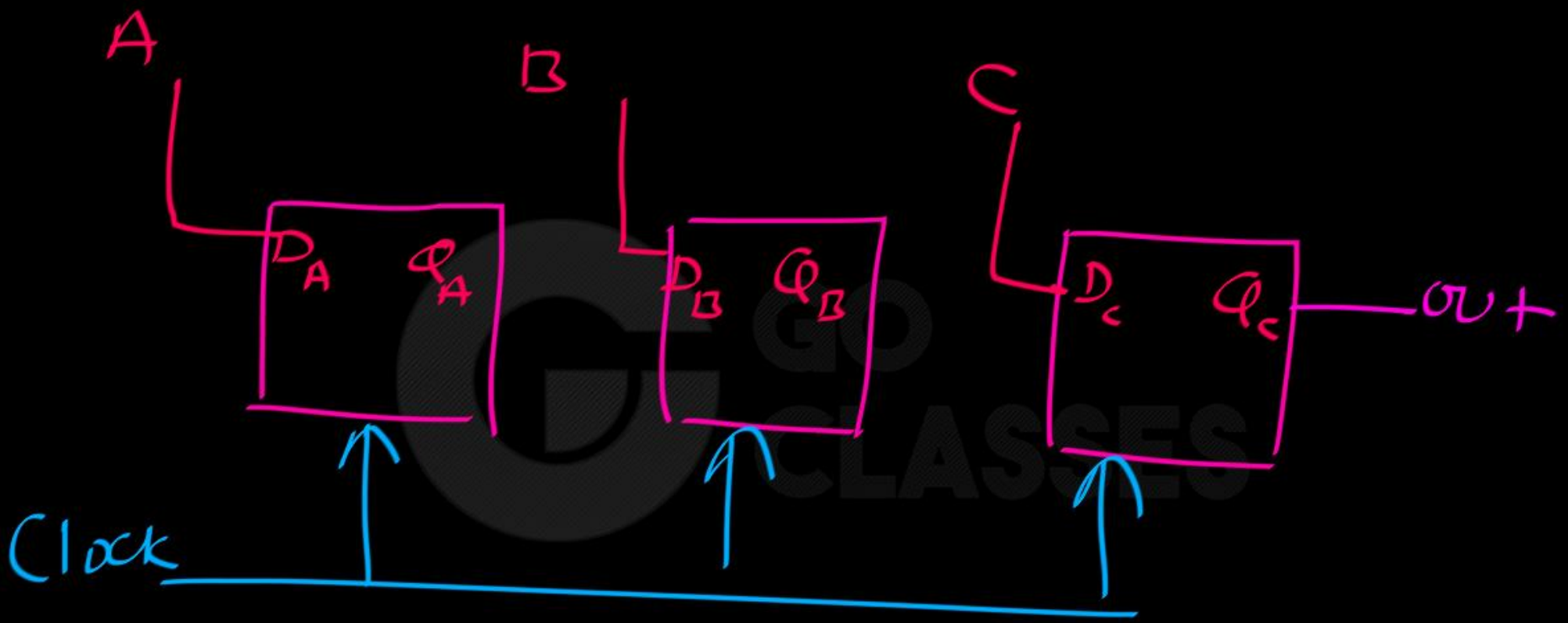
i. Clock CLK

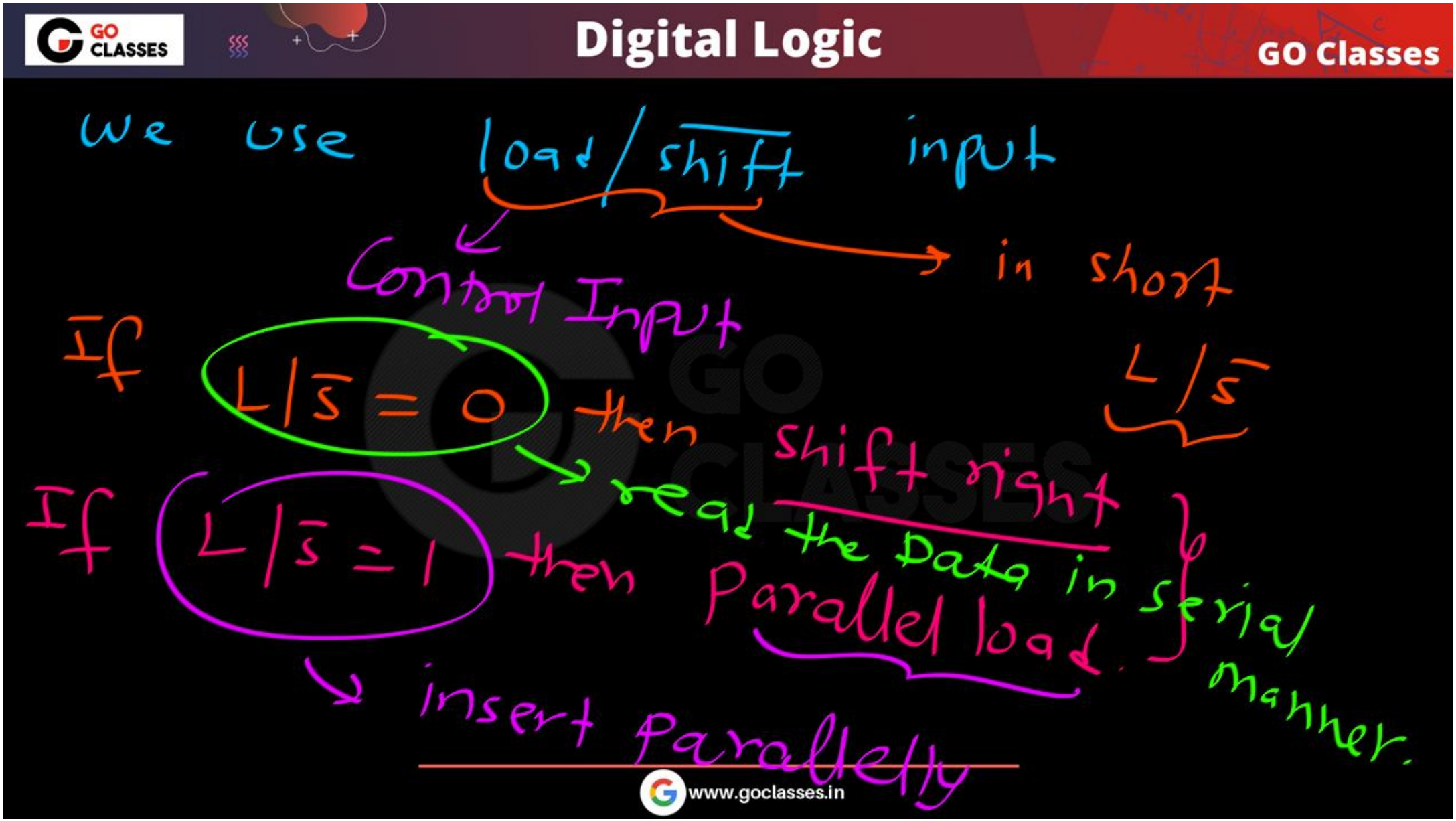
ii. Three parallel data inputs A, B, C

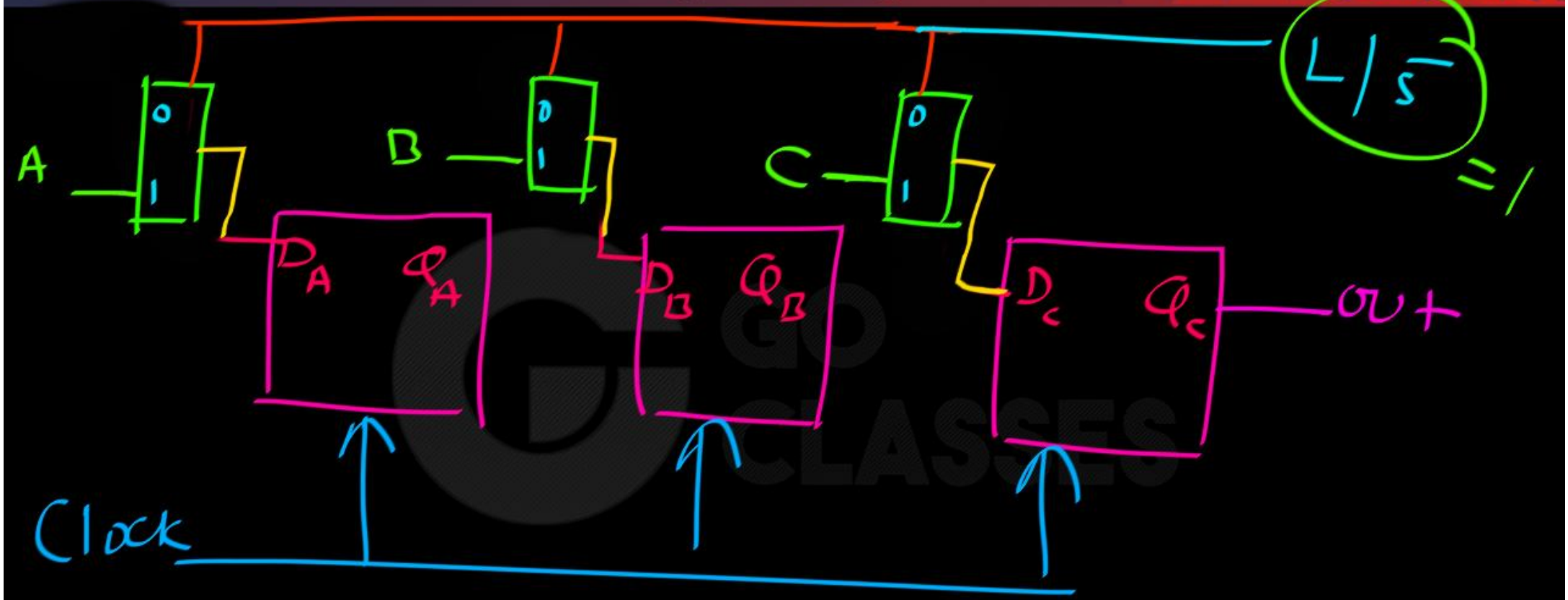
iii. Serial input S

iv. Control input load/$\overline{\text{SHIFT}}$.



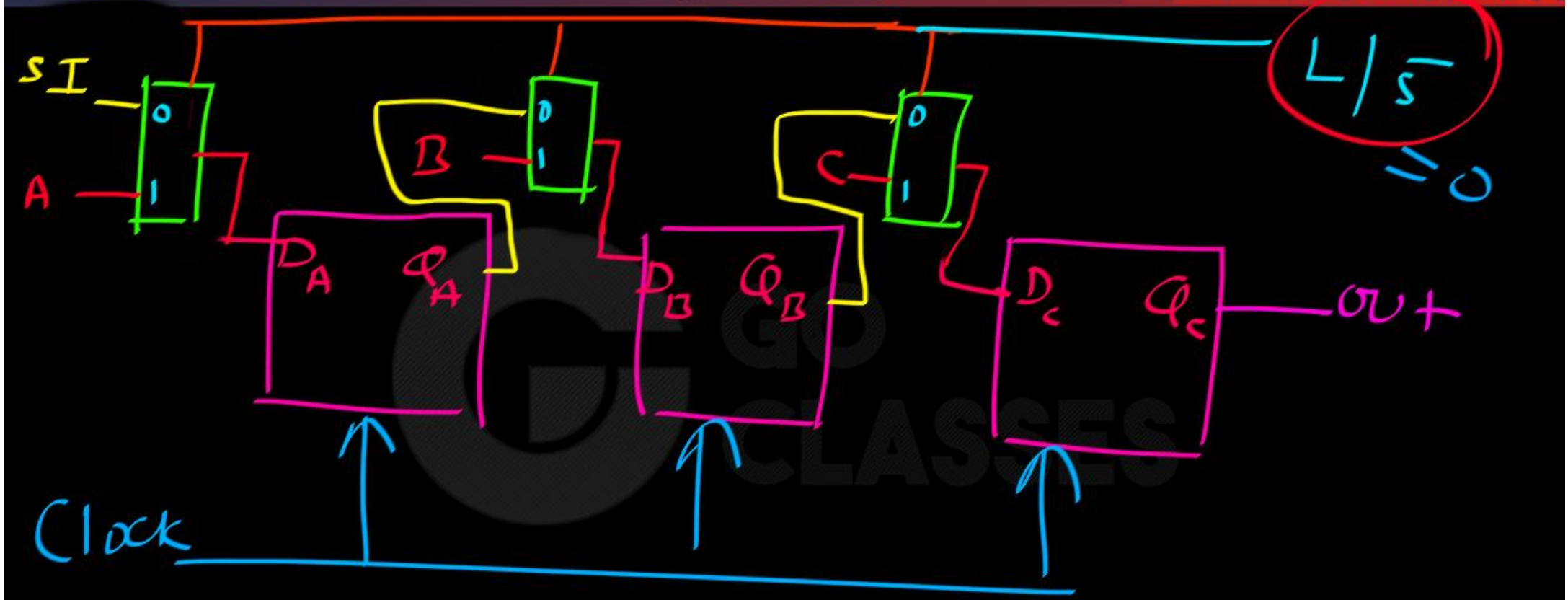

A
B
C
D_A
Q_A
D_B
Q_B
D_C
Q_C
Out
Clock


We use load/shift input
Control Input
in short
L/S̄
If L/S̄ = 0 then shift right
read the Data in serial manner.
If L/S̄ = 1 then Parallel load
insert Parallelly

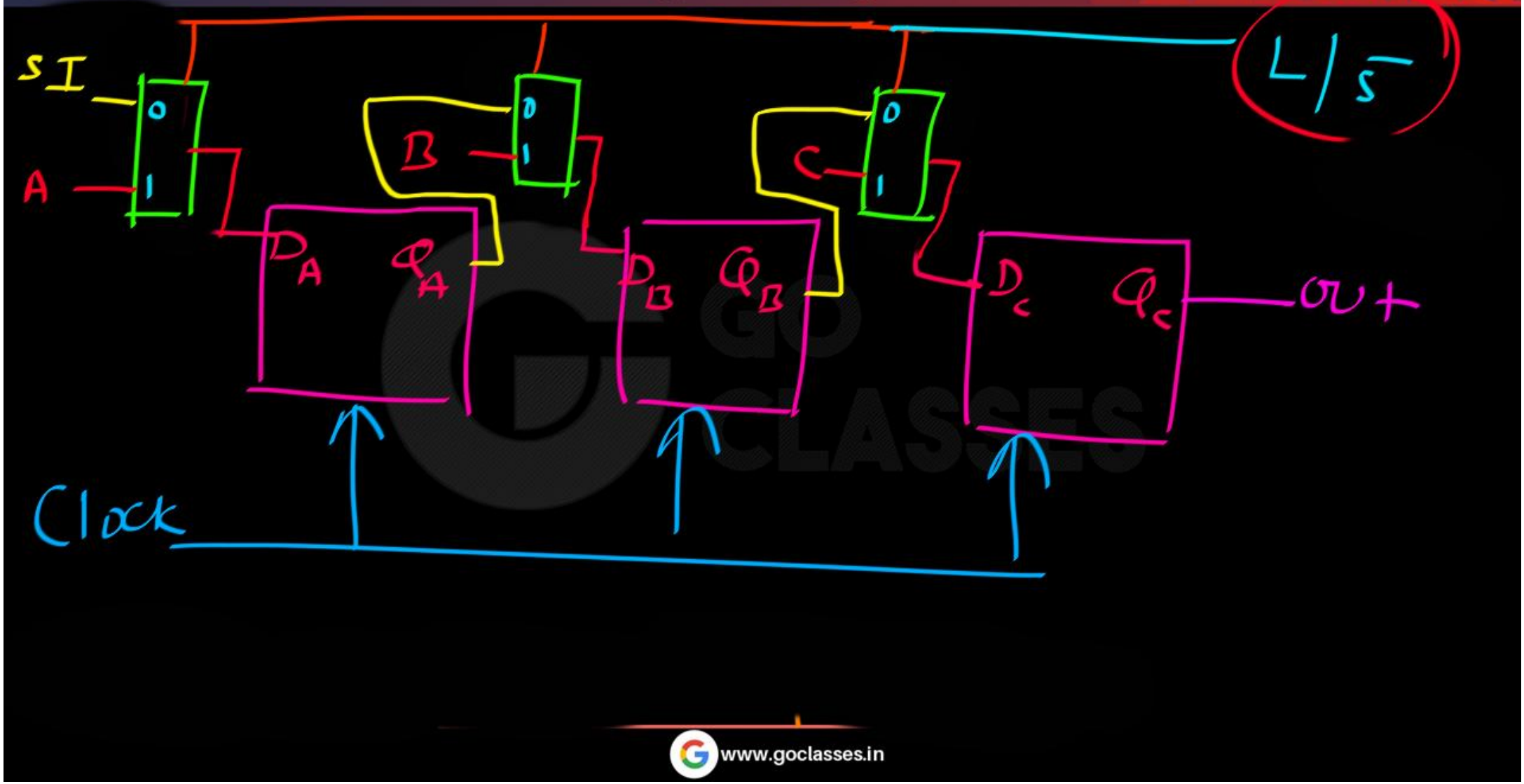


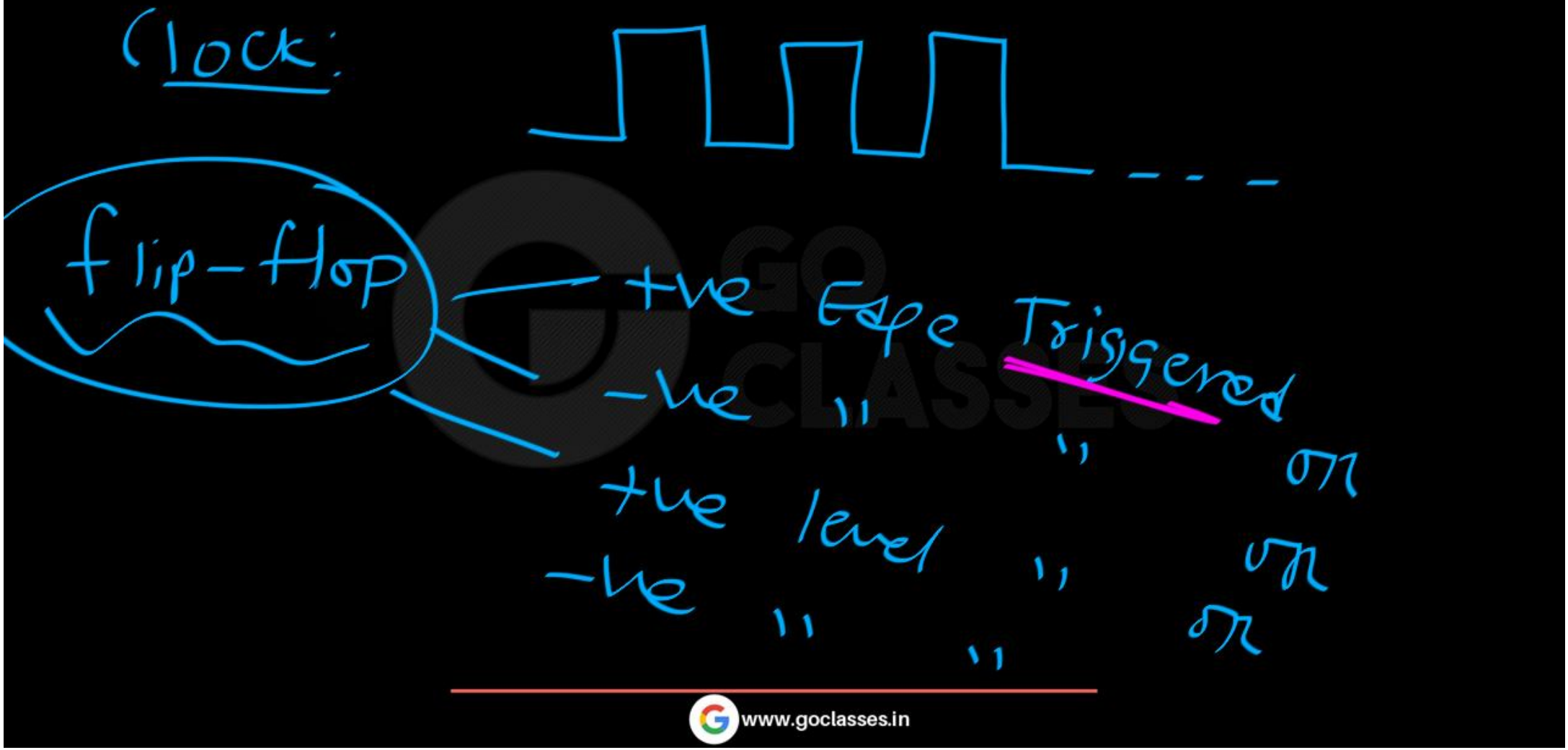
If $L/\bar{S} = 1$ then Parallel load.

Digital Logic

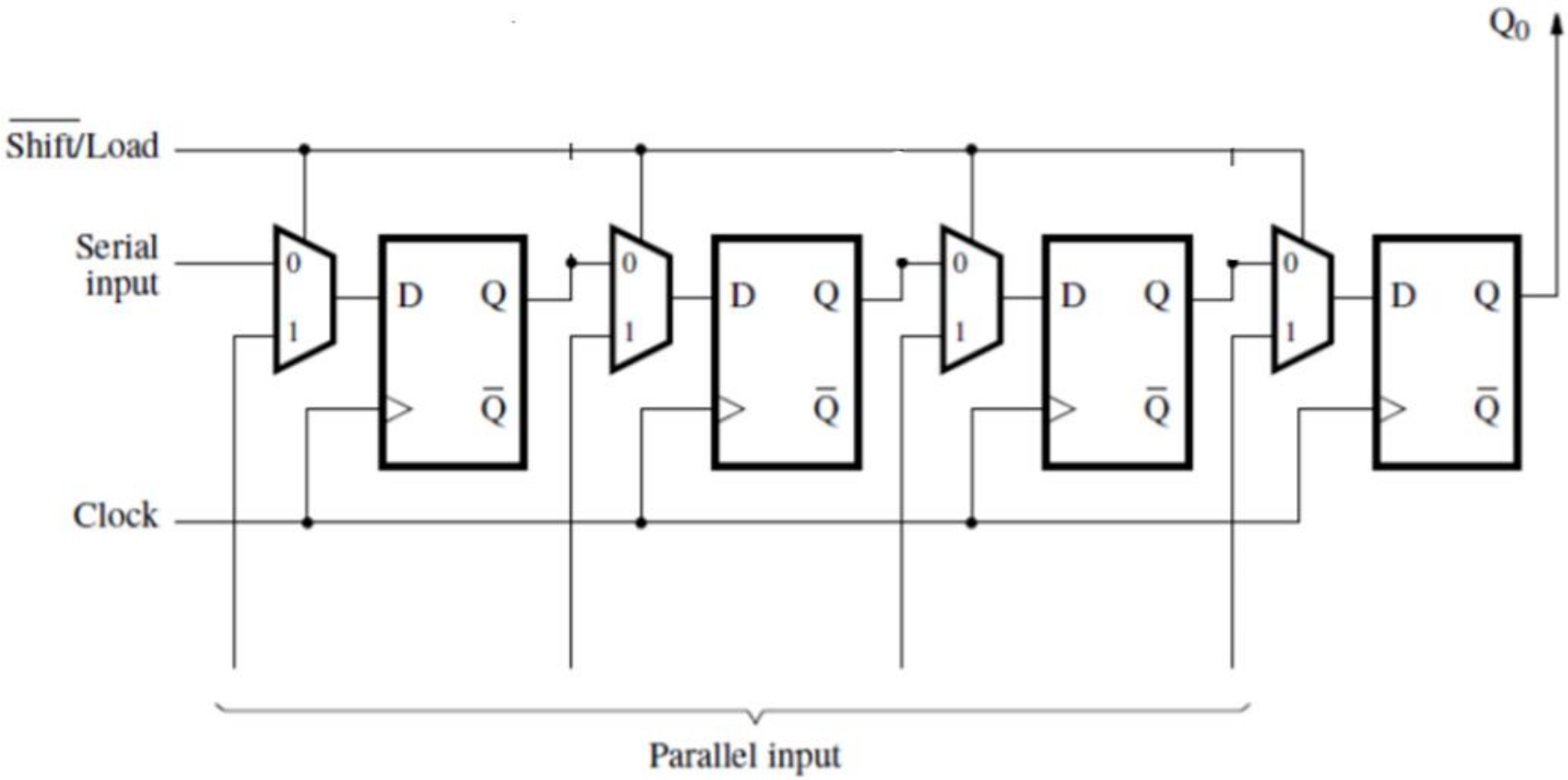


If $L/\bar{S} = 0$ then Right shift

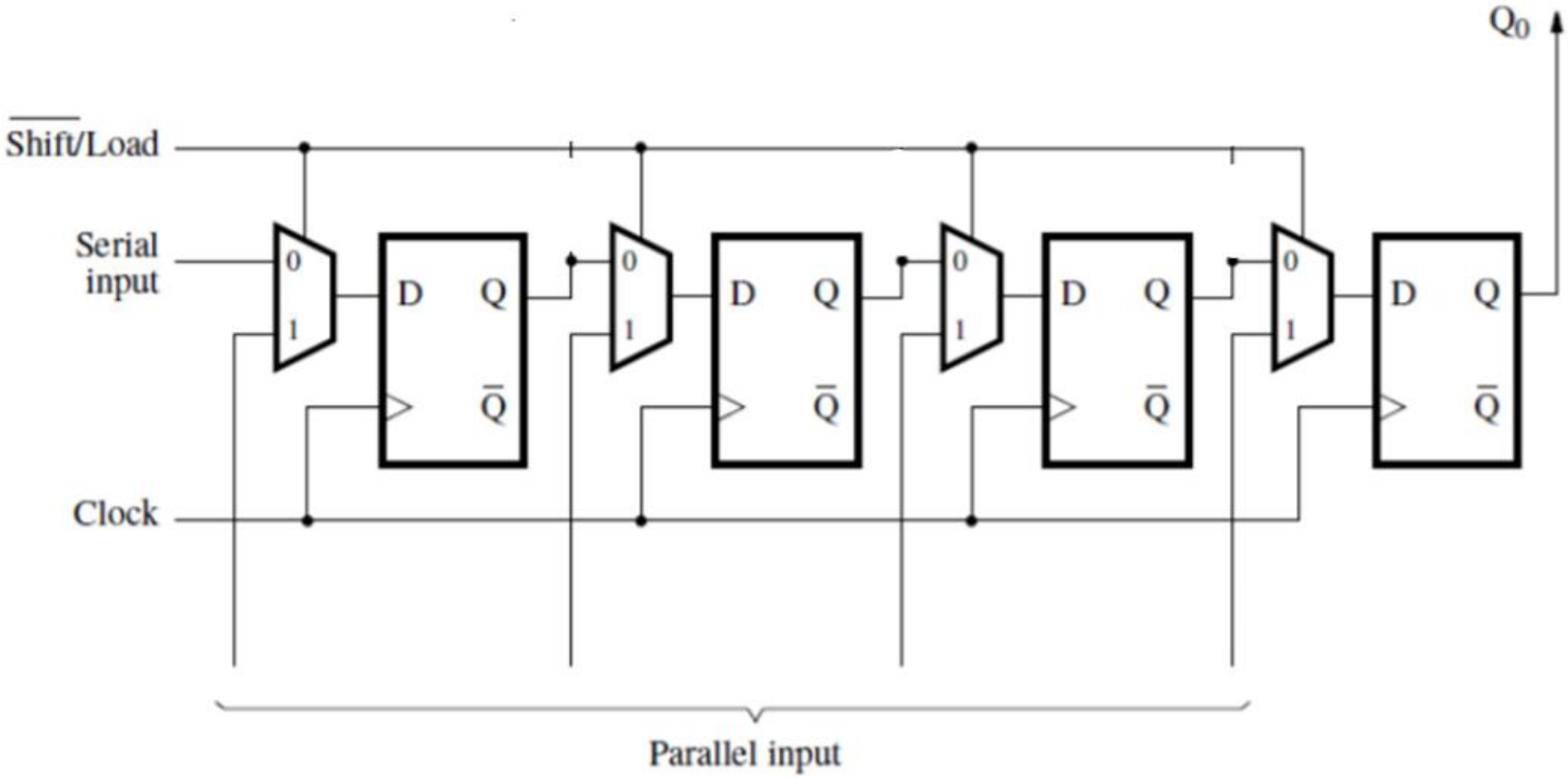

SI
A
B
C
L/S̄
0
1
D
Q
DA
QA
DB
QB
DC
QC
Out
Clock

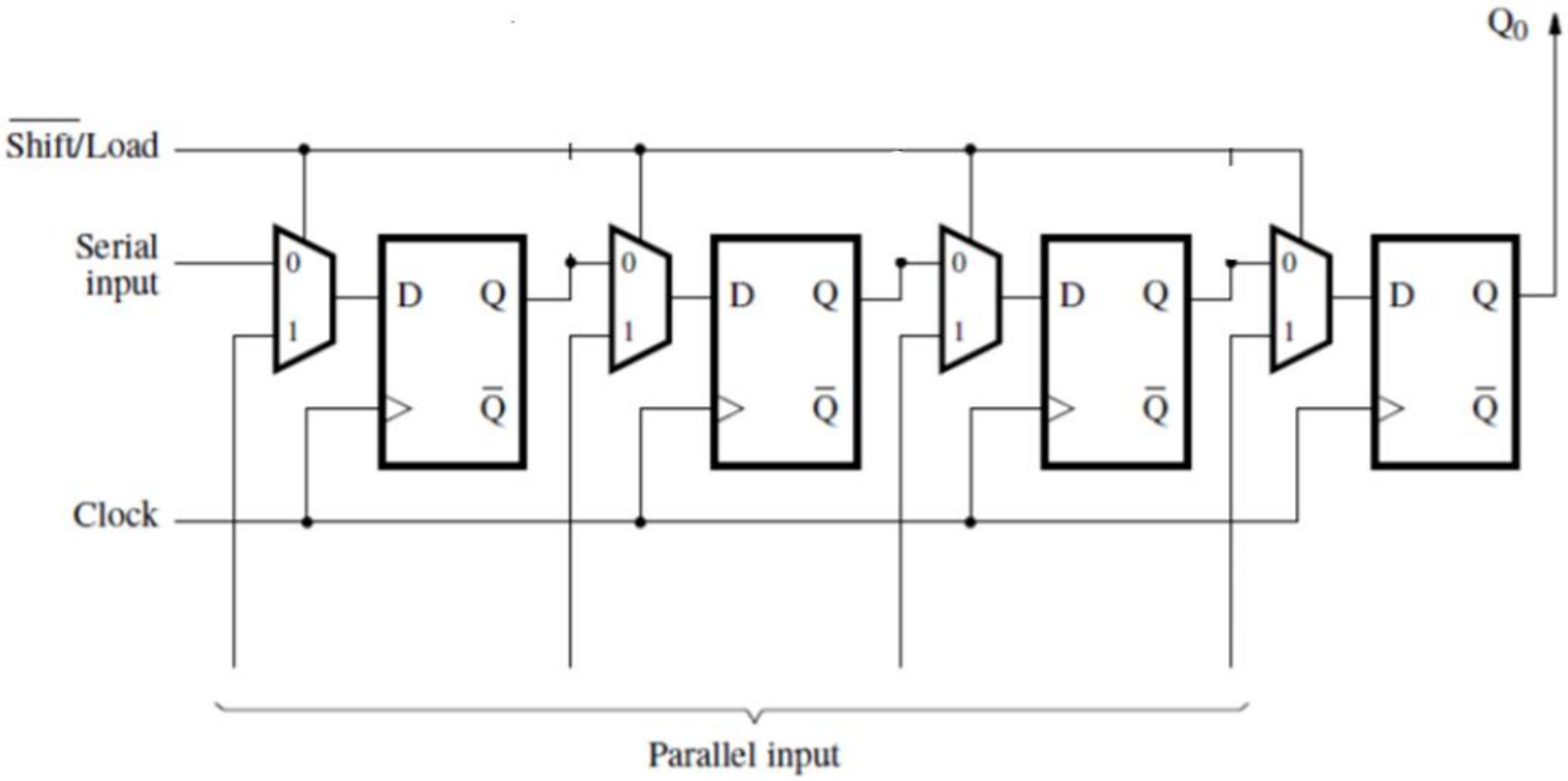

Clock:
flip-flop
+ve Edge Triggered
-ve " "
+ve level
-ve " "
or
or
or

When Load=0, this behaves like a shift register.



When Load=1, this behaves like a parallel-access register.




Q_0
Shift/Load
Serial input
Clock
0
1
D
Q
$\bar{Q}$
Parallel input

Some Applications of Registers

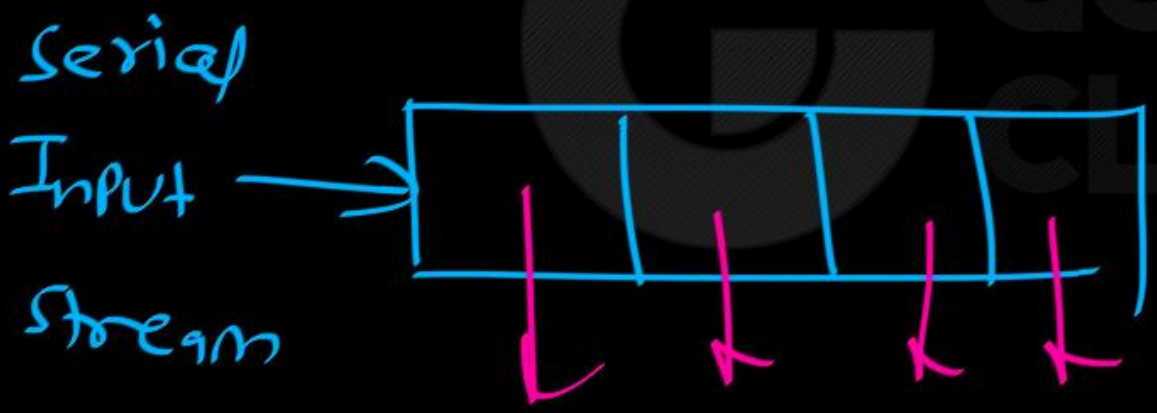
1. *Parallel-to-serial and serial-to-parallel conversion for data transmission*
   - Many device-to-device communication protocols use serial data transmission.
   - Advantages: less number of wires, higher reliability, lower cost, etc.
   - Data generators and receivers typically handle data in parallel.
   - We require conversion, which can be done using shift registers.

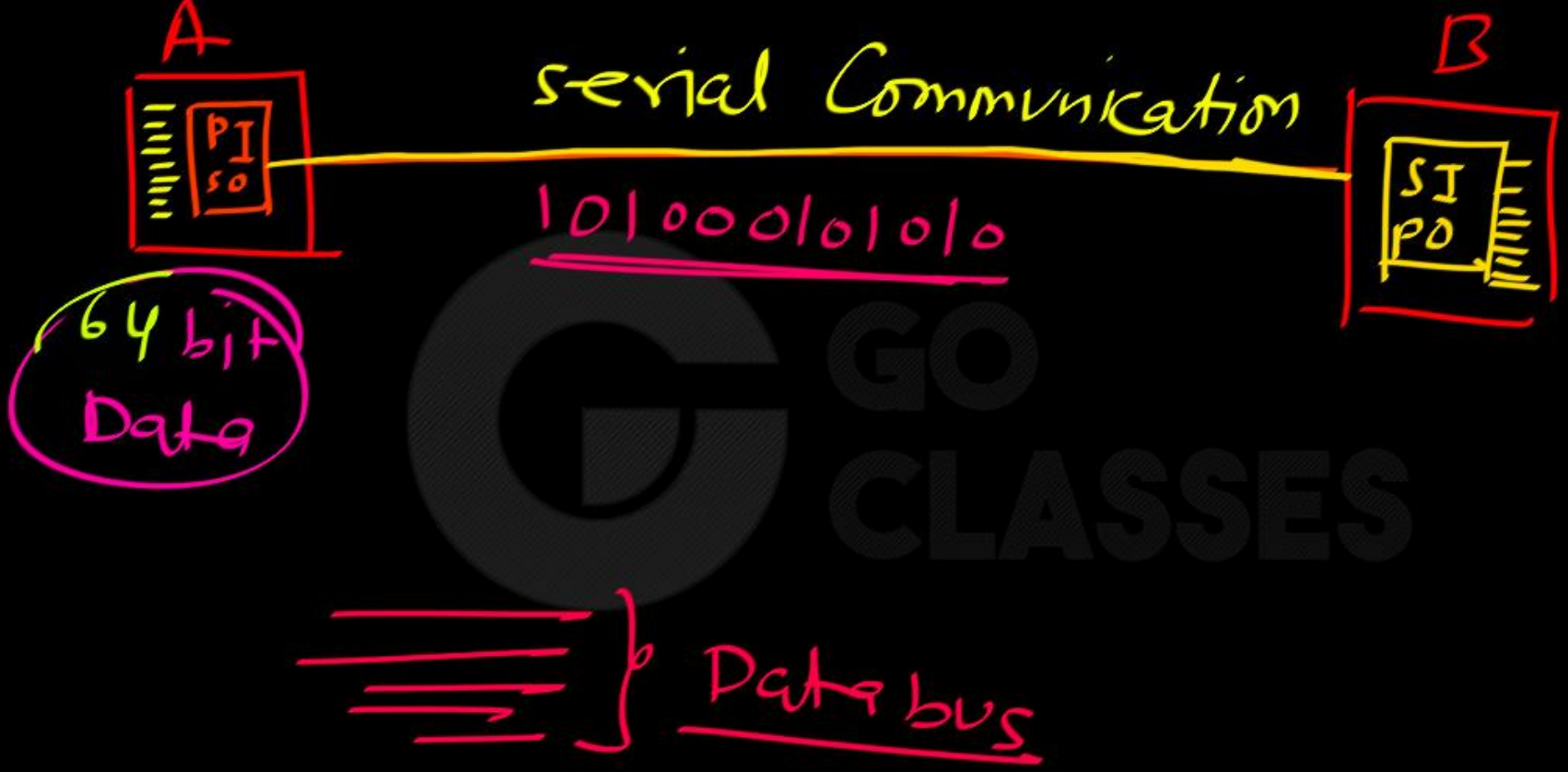
Parallel-to-serial Conversion:

Use PISO Register.

Serial- to- Parallel Conversion:

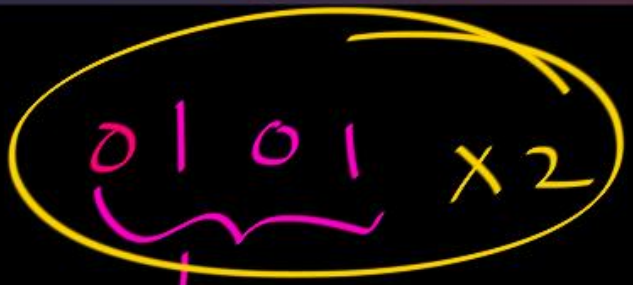
Use SIPO Register.




A
PI
SO
serial Communication
B
SI
PO
1010001010101o
64 bit
Data
Data bus

Shift Register

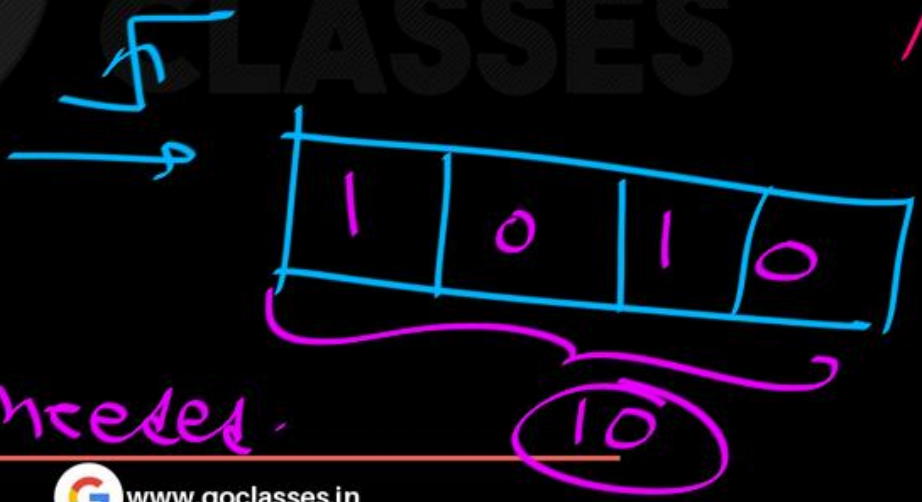
- Shift registers can be used to perform multiplication by powers of 2.
  - For example, multiplying a binary number by 2 is equivalent to shifting it left by 1 position
  - Multiplying by 2^i is equivalent to shifting *i* bit positions to the left and padding the lower significant bits by 0s.
- For dividing by powers of 2
  - Shift right by 1 position
  - Dividing by 2^i is equivalent to shifting *i* bit positions to the right and padding the upper significant bits by 0s.



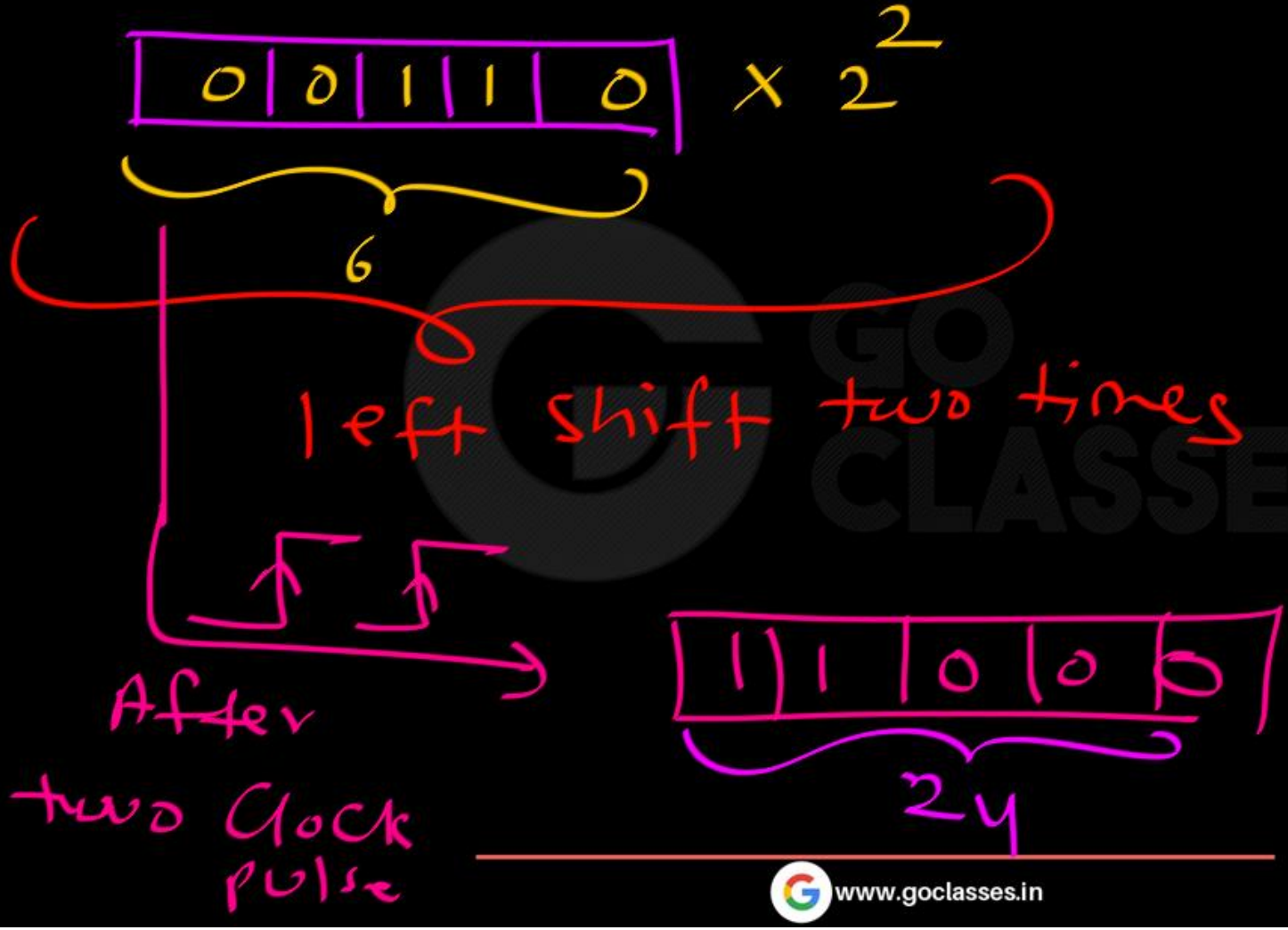
multiplier HW/Circuit

Consume many clock cycles.

5 X 2 = 10



one clock pulse needed.


0 0 1 1 0
x 2^2
6
left shift two times
After two Clock pulse
1 1 0 0 0
2y

Multiplication with 2^k : k time left shift

Divide by 2^k : k time Right shift

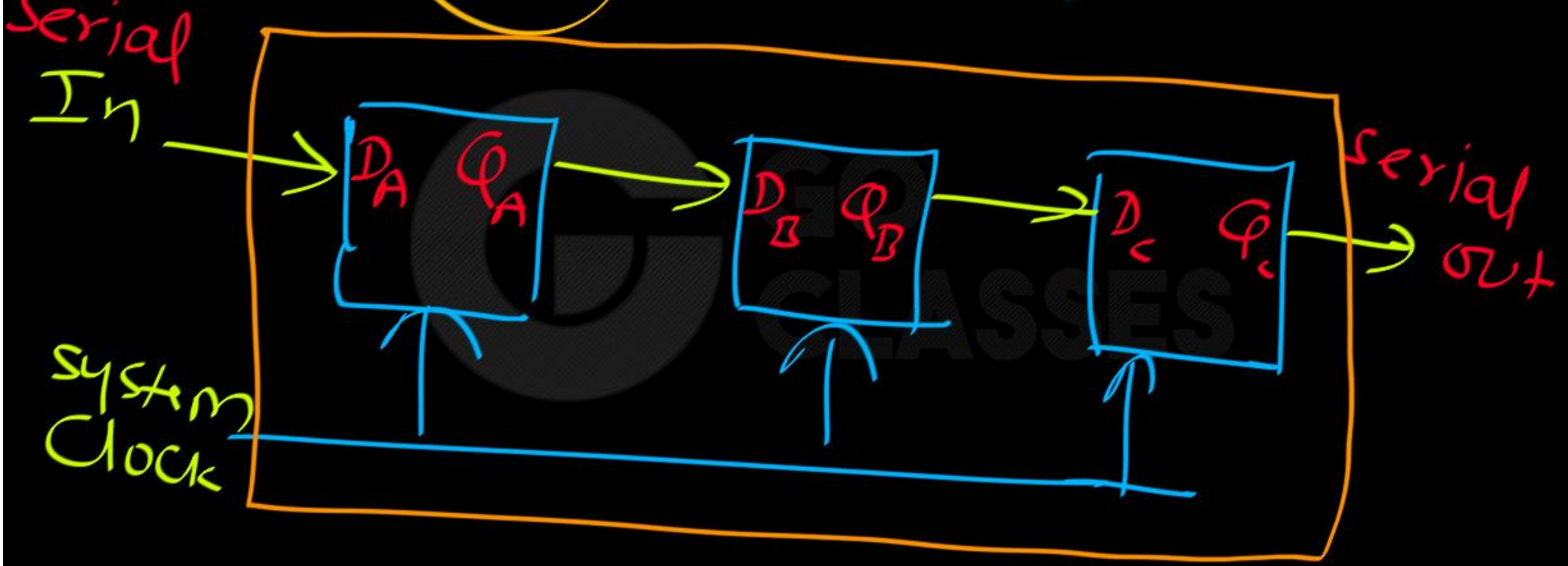
Eg:

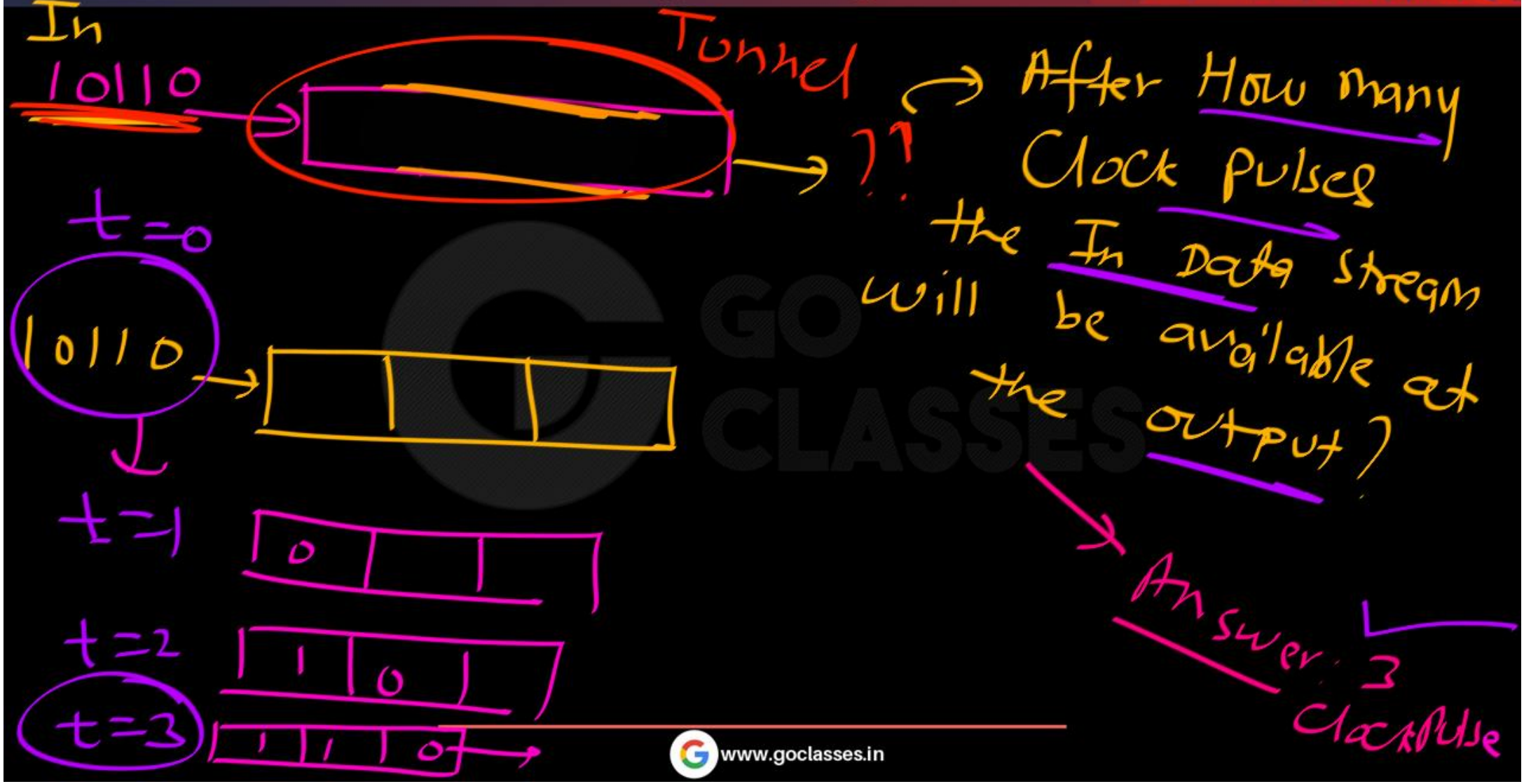
(3) Random Sequence Generation

→ LFSR → Random sequences

(4) Delaying a pulse : Application of SISO shift register

3 bit SISO shift Register

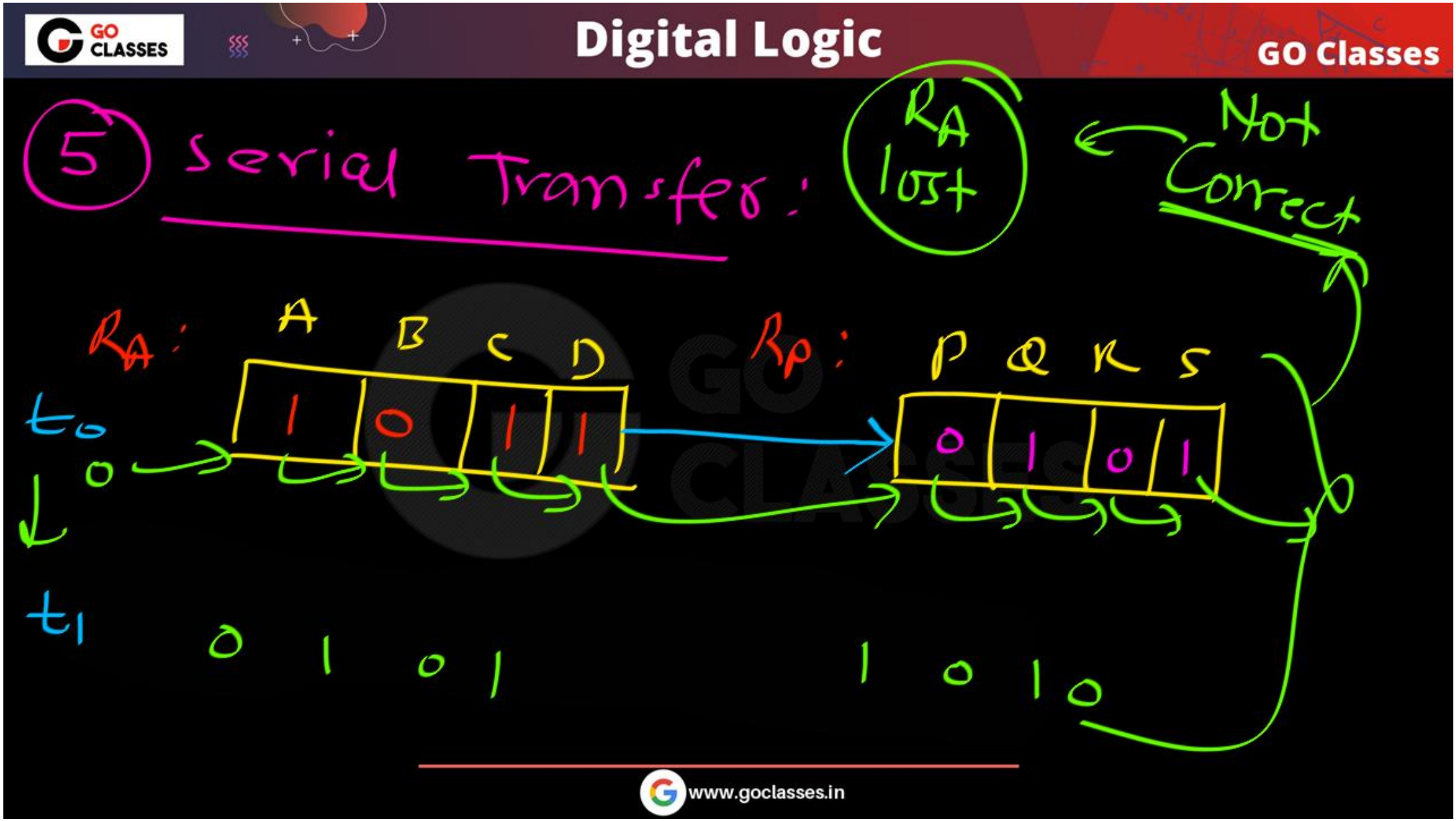



In
10110
Tunnel
??
After How many Clock pulses the In Data stream will be available at the output?
t=0
10110
t=1
0
t=2
1 0
t=3
1 1 0
Answer: 3 Clock Pulse

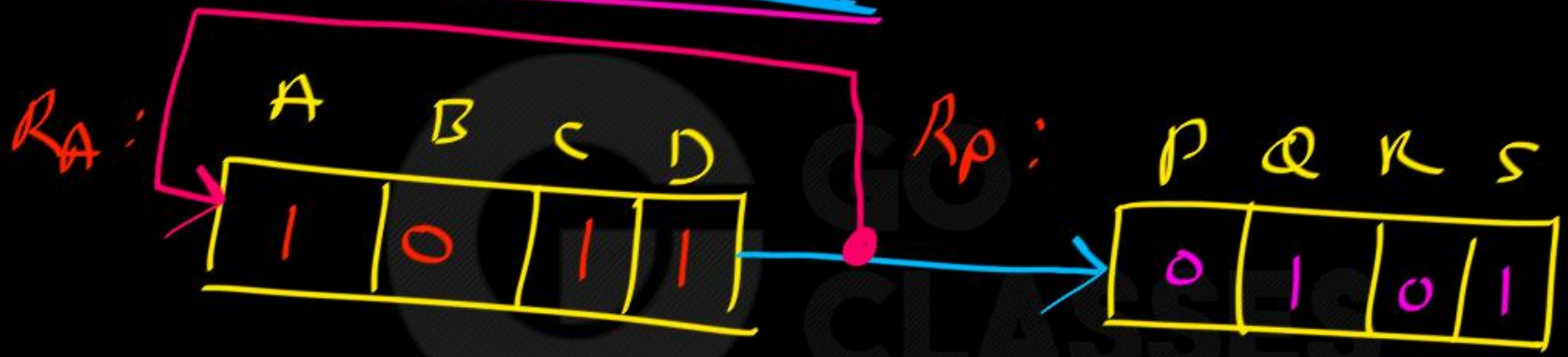
5 Serial Transfer :

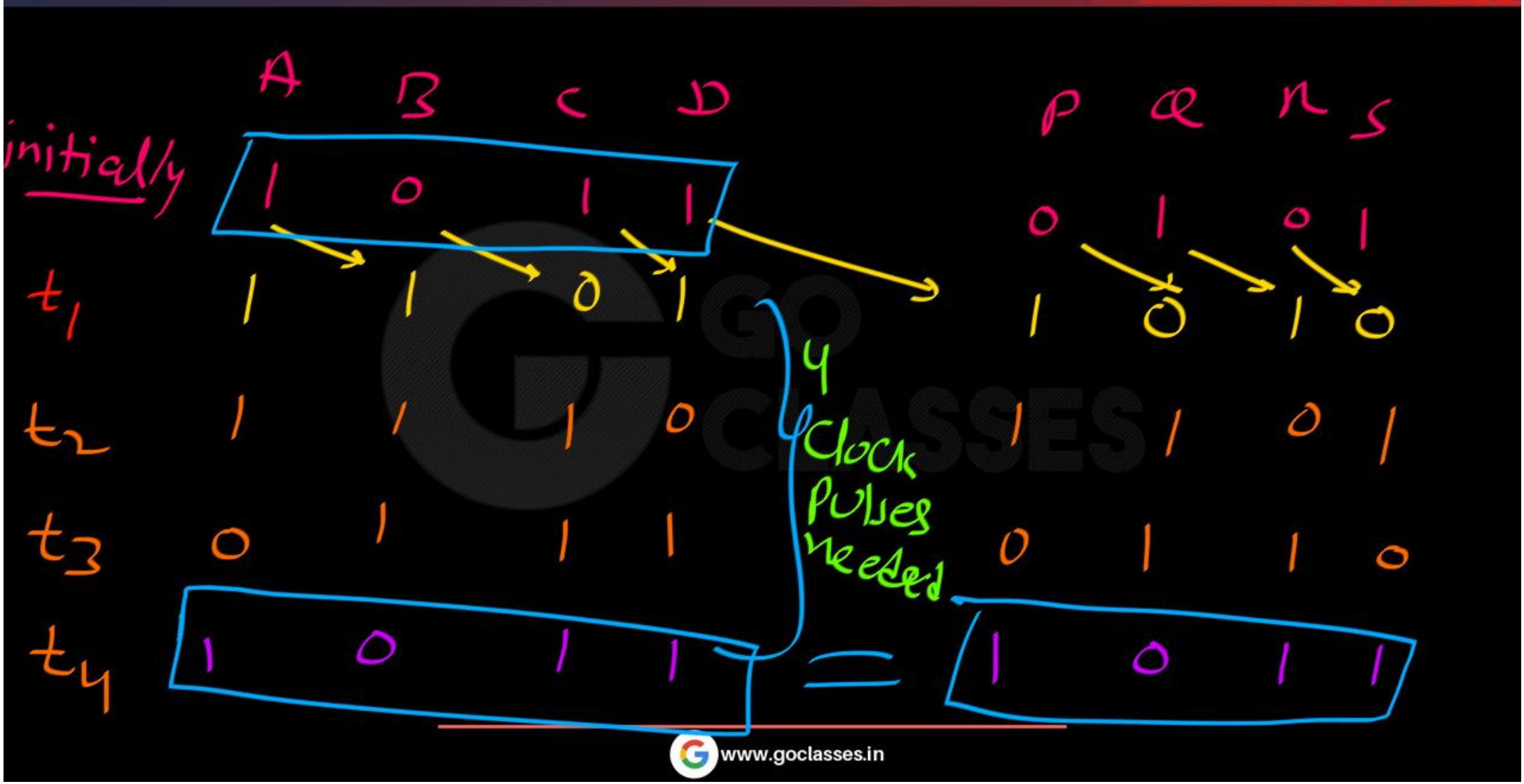
Objective : Copy R_A into R_B

$R_A, R_B \rightarrow$ SISO shift Register


5 Serial Transfer:
RA lost
Not Correct
RA:
A B C D
1 0 1 1
RP:
P Q R S
0 1 0 1
t0
0
t1
0 1 0 1
1 0 1 0

5 Serial Transfer :

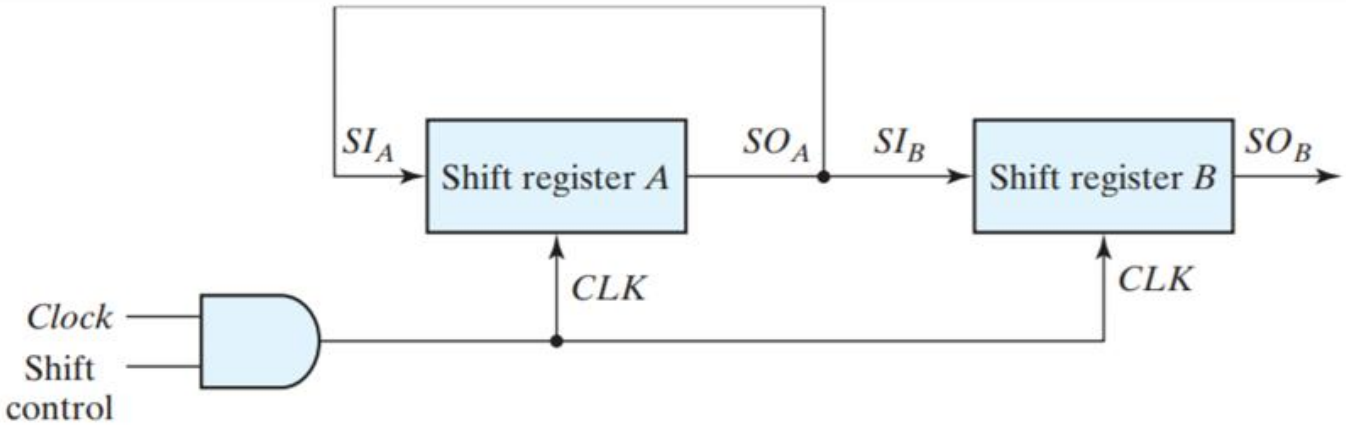



A B C D
P Q R S
Initially
1 0 1 1
0 1 0 1
t_1
1 1 0 1
1 0 1 0
t_2
1 1 1 0
1 1 0 1
t_3
0 1 1 1
0 1 1 0
t_4
1 0 1 1
4 Clock Pulses Needed
=
1 0 1 1

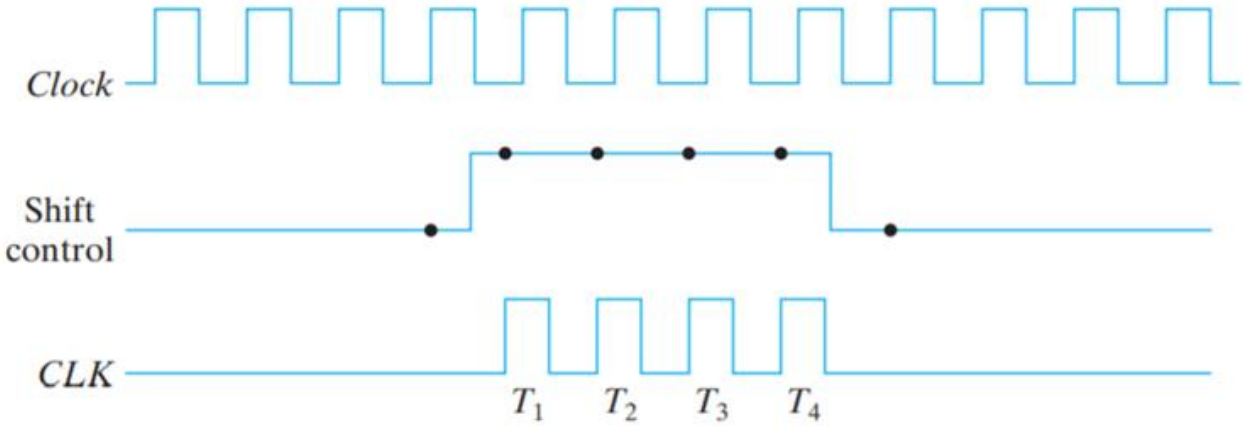
Serial Transfer

The datapath of a digital system is said to operate in serial mode when information is transferred and manipulated one bit at a time. Information is transferred one bit at a time by shifting the bits out of the source register and into the destination register. This type of transfer is in contrast to parallel transfer, whereby all the bits of the register are transferred at the same time.

The serial transfer of information from register A to register B is done with shift registers, as shown in the block diagram of Fig. 6.4(a). The serial output (SO) of register A is connected to the serial input (SI) of register B. To prevent the loss of information stored in the source register, the information in register A is made to circulate by connecting the serial output to its serial input. The initial content of register B is shifted out through its serial output and is lost unless it is transferred to a third shift register. The shift control input determines when and how many times the registers are shifted. For illustration here, this is done with an AND gate that allows clock pulses to pass into the CLK terminals only when the shift control is active. (This practice can be problematic because it may compromise the clock path of the circuit, as discussed earlier.)



(a) Block diagram



(b) Timing diagram

FIGURE 6.4
Serial transfer from register *A* to register *B*

Table 6.1
Serial-Transfer Example

Timing Pulse	Shift Register *A*				Shift Register *B*			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

GATE ECE 1996

A pulse train can be delayed by a finite number of clock periods using a

a. Serial In Serial Out shift register
b. Serial In Parallel Out shift register
c. Parallel In serial Out shift register
d. Parallel In parallel Out shift register

https://gateoverflow.in/32445/Pulse-train-can-be-delayed-finite-number-periods-using-clock

GATE ECE 1996

A pulse train can be delayed by a finite number of clock periods using a

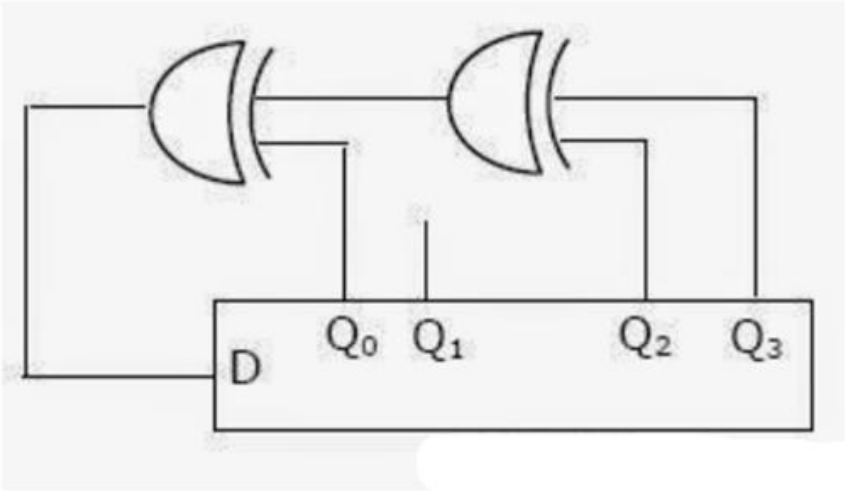
a. Serial In Serial Out shift register
b. Serial In Parallel Out shift register
c. Parallel In serial Out shift register
d. Parallel In parallel Out shift register

https://gateoverflow.in/32445/Pulse-train-can-be-delayed-finite-number-periods-using-clock


Input Data stream
0 1 1 0 0
Delaying the input pulse
SISO

GATE
1996
ECE

A 4 bit shift register, which shifts 1 bit to the right at every clock pulse, is initialized to values 1000 for ($Q_0Q_1Q_2Q_3$). The D input is derived from Q_0, Q_2 and Q_3 through two XOR gates as shown in figure.

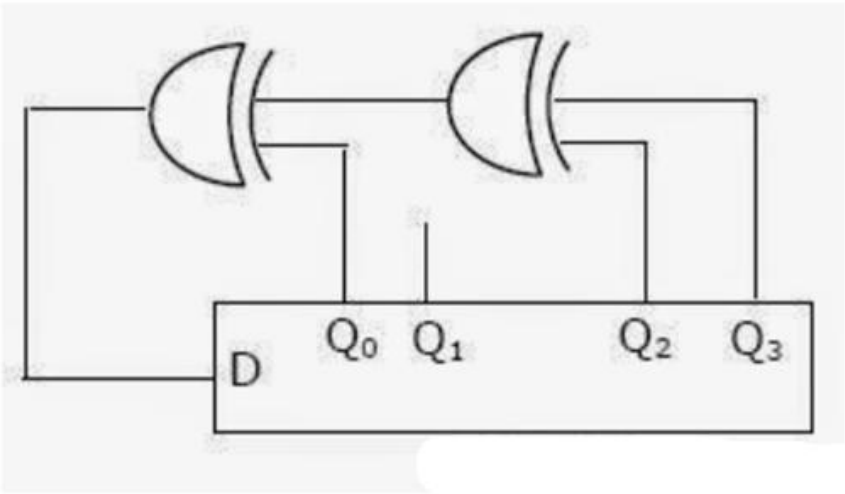


a. Write the 4 bit values ($Q_0Q_1Q_2Q_3$) after each clock pulse till the pattern (1000) reappears on ($Q_0Q_1Q_2Q_3$).

b. To what values should the shift register be initialized so that the pattern (1001) occurs after the first clock pulse?

GATE
1996
ECE

A 4 bit shift register, which shifts 1 bit to the right at every clock pulse, is initialized to values 1000 for ($Q_0Q_1Q_2Q_3$). The D input is derived from Q_0, Q_2 and Q_3 through two XOR gates as shown in figure.



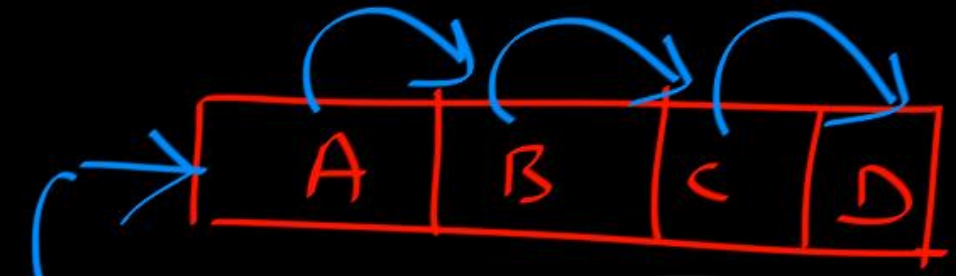
LFSR
↓
Used for
Randomness

6

a. Write the 4 bit values ($Q_0Q_1Q_2Q_3$) after each clock pulse till the pattern (1000) reappears on ($Q_0Q_1Q_2Q_3$).

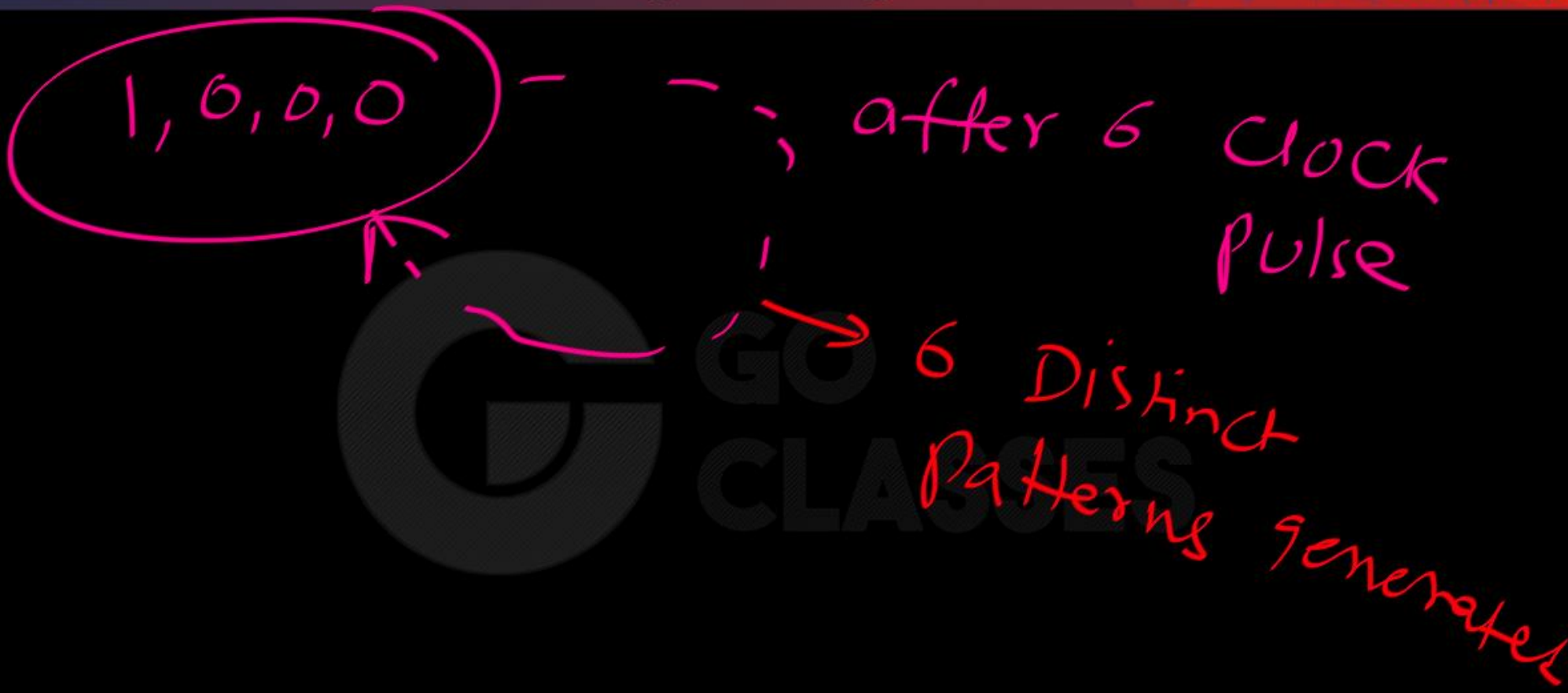
OOD

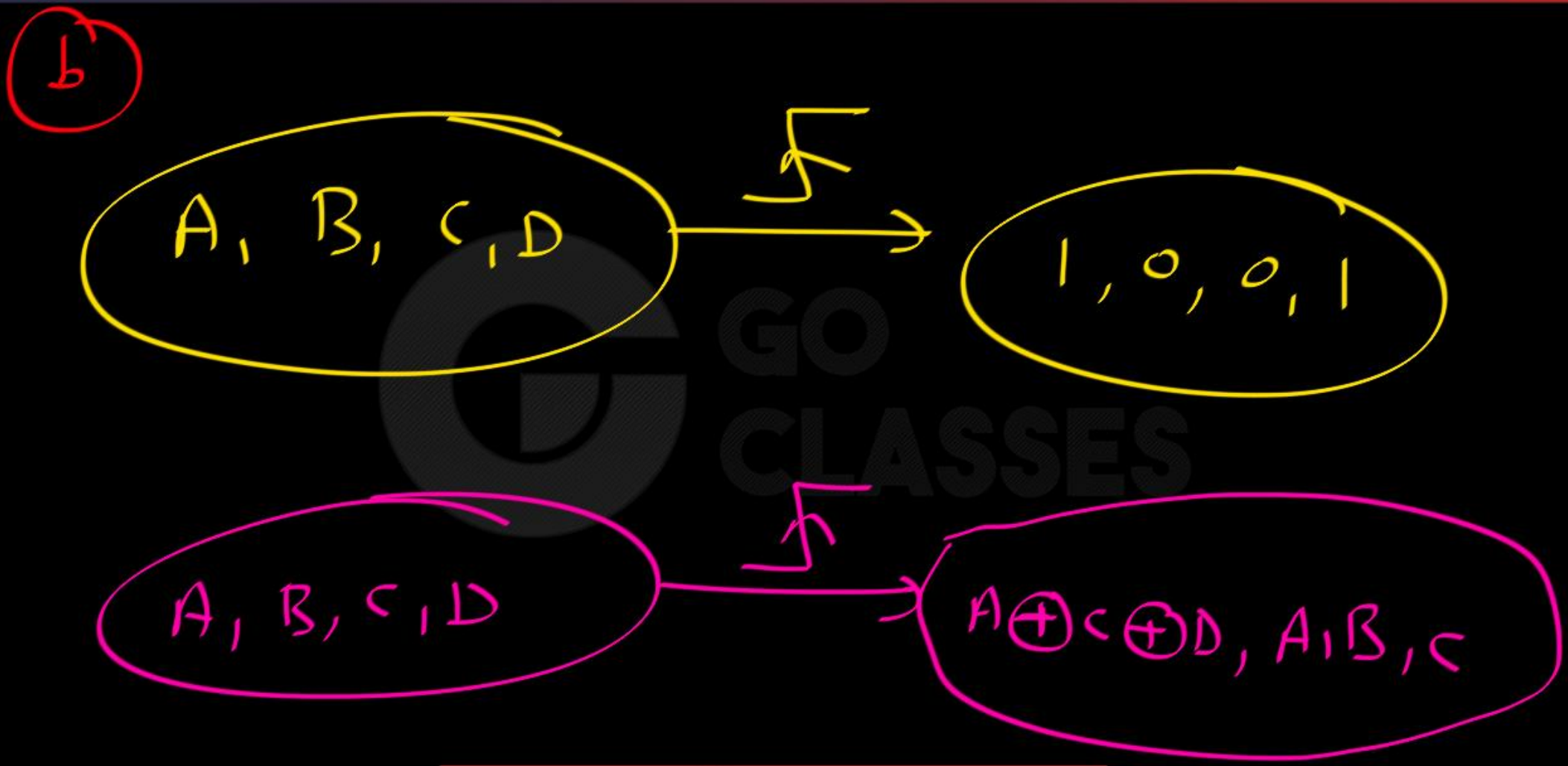
b. To what values should the shift register be initialized so that the pattern (1001) occurs after the first clock pulse?

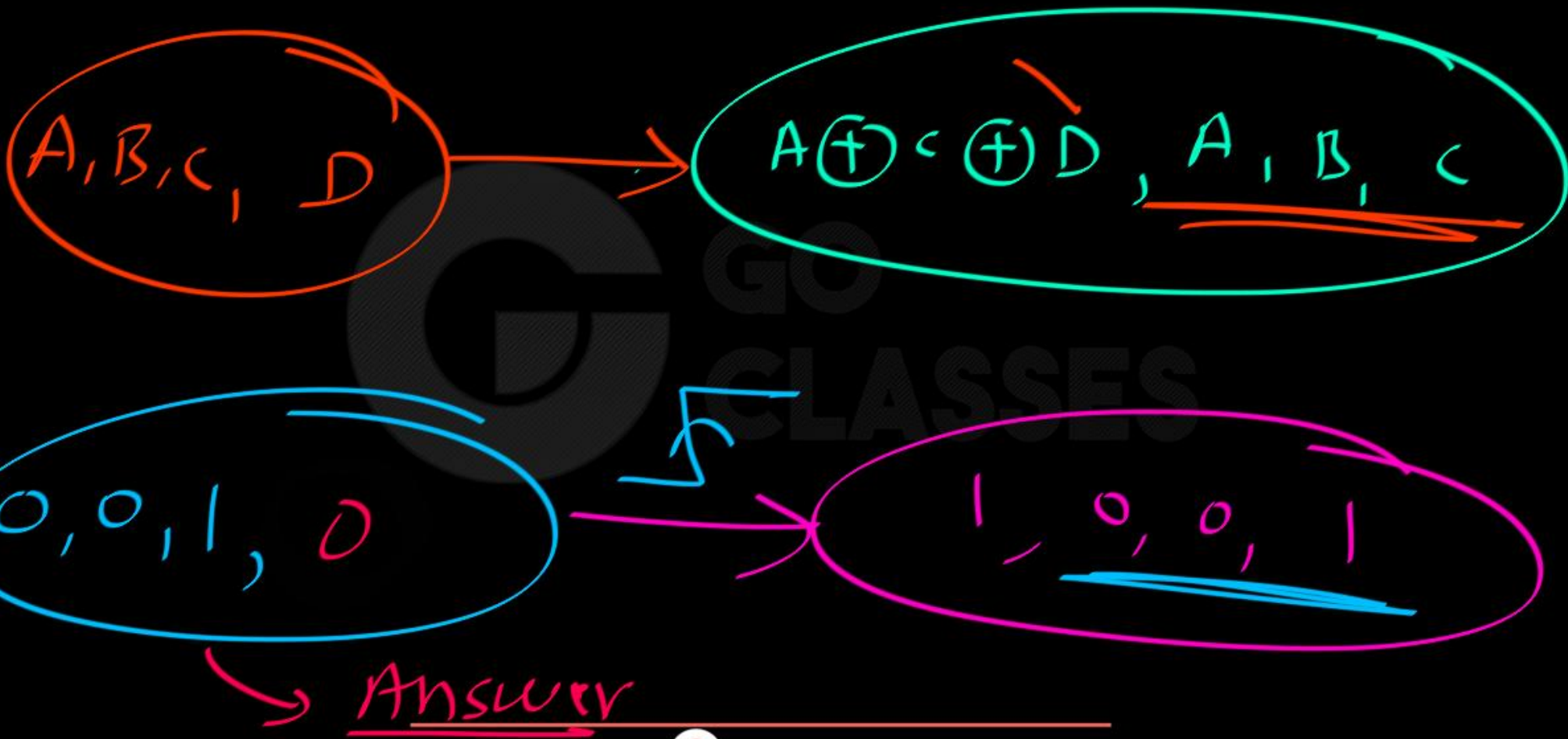


$A \oplus C \oplus D$


1,0,0,0
1,1,0,0
1,1,1,0
0,1,1,1
0,0,1,1
0,0,0,1


1,0,0,0
after 6 clock Pulse
6 Distinct Patterns generated


b
A, B, C, D
f
1, 0, 0, 1
A, B, C, D
f
A⊕C⊕D, A, B, C


A, B, C, D
A ⊕ C ⊕ D, A, B, C
0, 0, 1, 0
1, 0, 0, 1
Answer