Sequential circuits

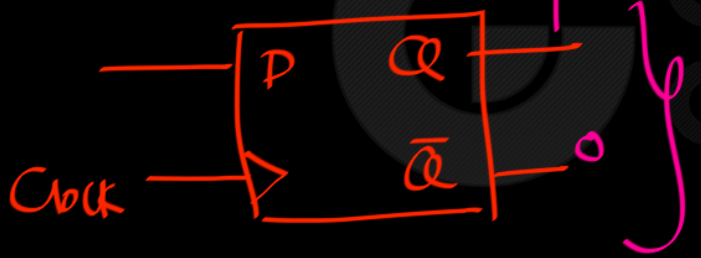
* The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are *combinatorial* in nature, i.e., the output(s) depends only on the *present* values of the inputs and *not* on their past values.
* In *sequential* circuits, the "state" of the circuit is crucial in 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.

Sequential CKt

- State
- State table
- State Diagram
- Next state Equation
- output Equation

Some Terminologies :

Output of flipflop : the Q output of the flipflop.



State of this ff = output of this ff : Q = 1

This ff has stored "1".

for a flipflop :

output of this ff = Q = bit stored in this ff = state of this ff.

Some Terminologies :

Circuits with Flip-Flop = Sequential Circuit

State of a flip flop : the bit stored in it = flip flop output

State of a sequential circuit : flip-flop output combination

State of a register : the register content = flip-flop output combination

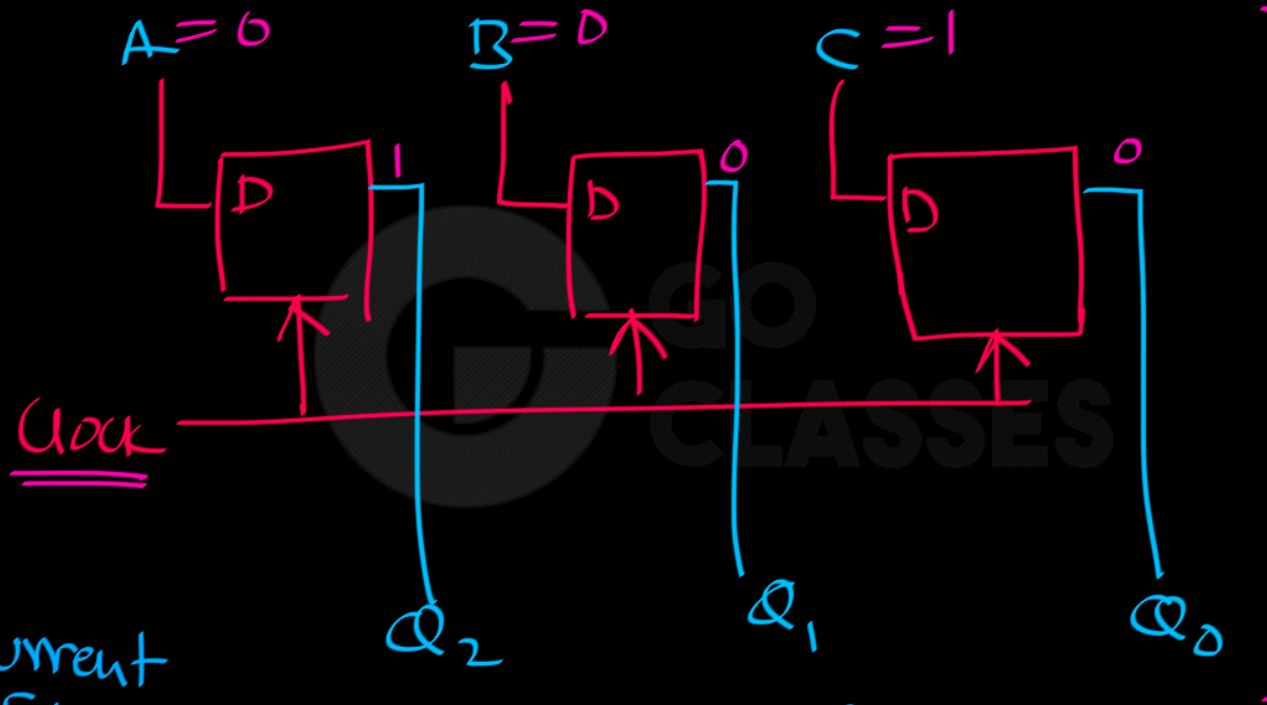
NOTE : In a flipflop (Or register) etc, the stored data is available on output of flipflop (Or register) and the output of flipflop(Or register) is the data stored in the flipflop(Or register)

Combinational ckt

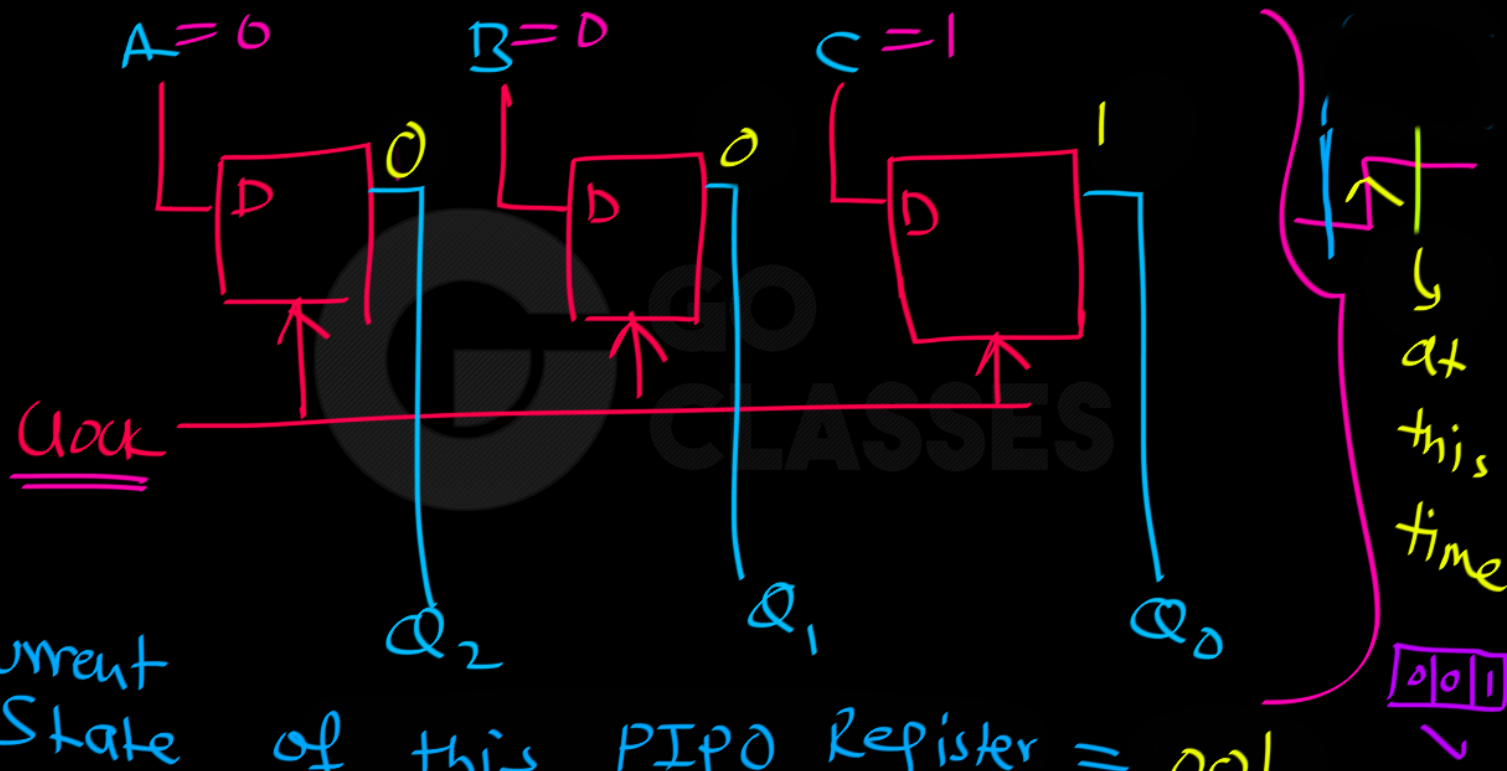
output = f(current input)

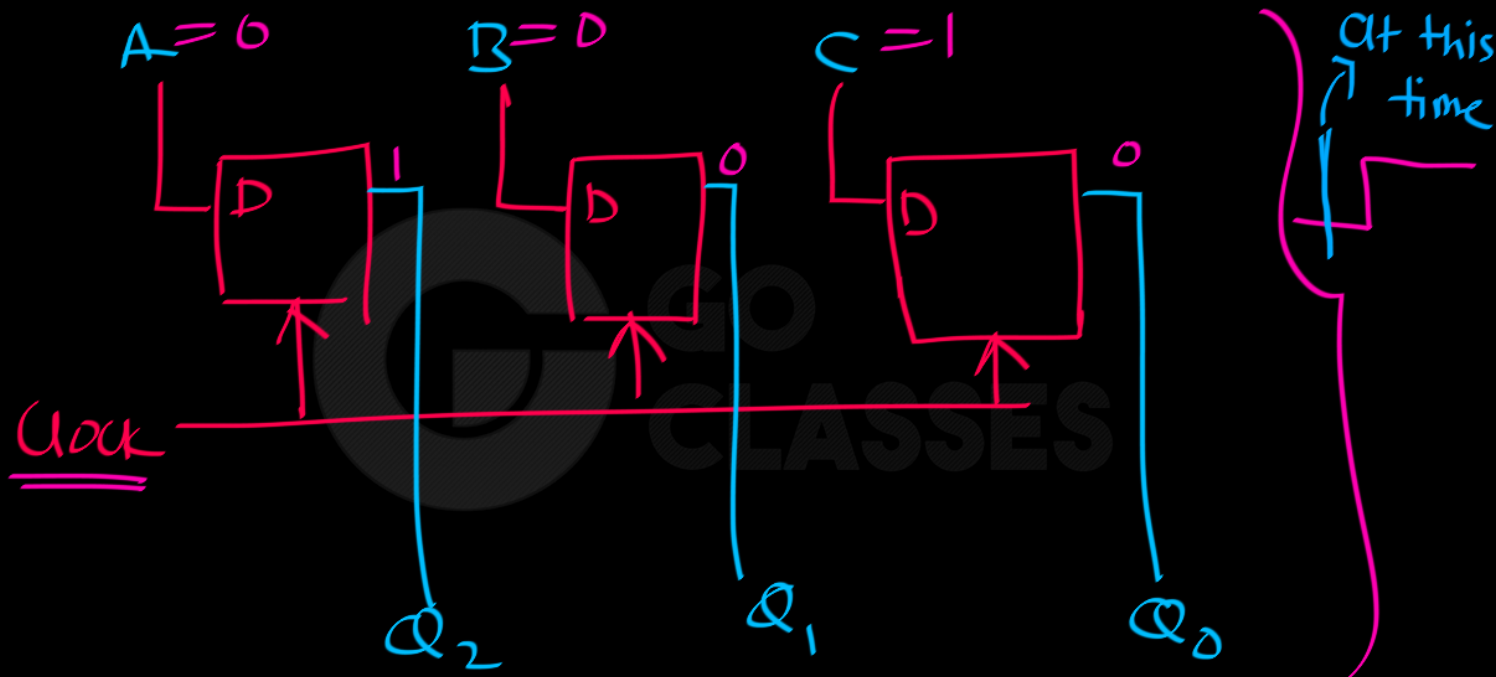
Sequential ckt :

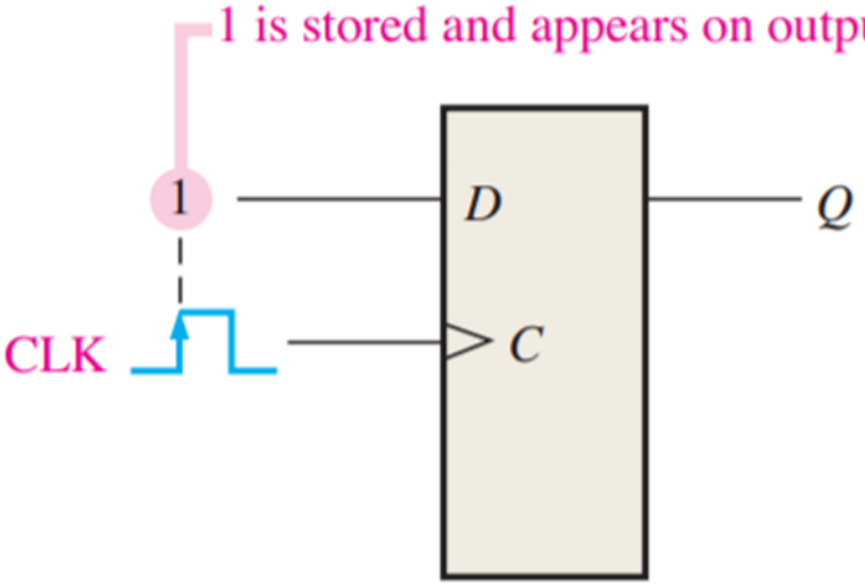
next output = f(current input, current state)



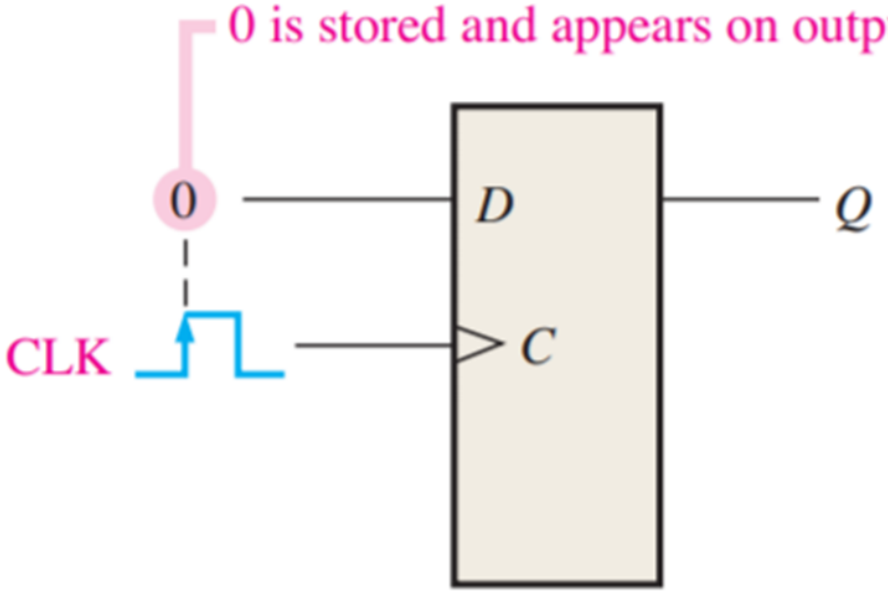
Current State of this PIPO Repister = 100


A = 0
B = 0
C = 1
D
D
D
0
0
1
Clock
Q_2
Q_1
Q_0
y at this time
0 0 1
Current State of this PIPO Register = 001


A = 0
B = 0
C = 1
D
1
D
0
D
0
Clock
Q2
Q1
Q0
At this time



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. A similar procedure applies to the storage of a 0 by resetting the flip-flop, as also illustrated in Figure 8–1.

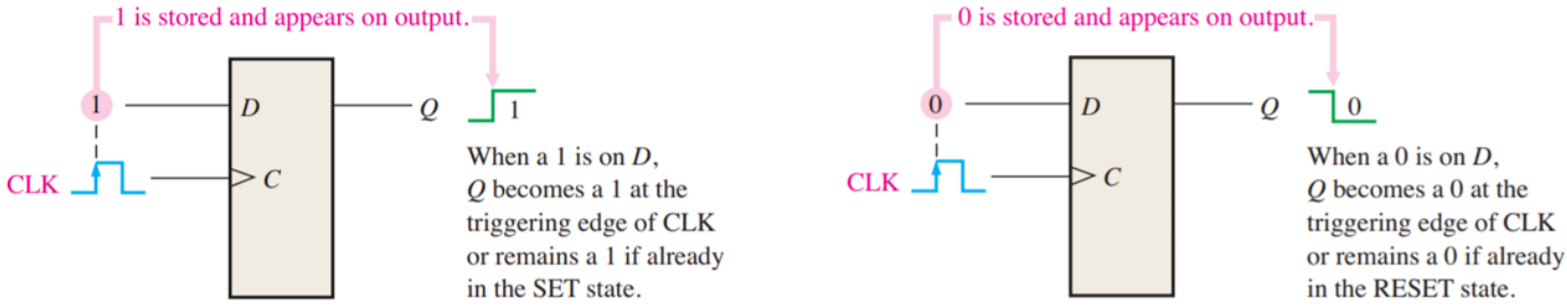
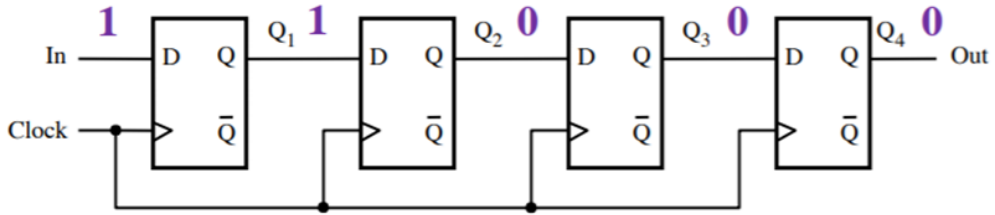
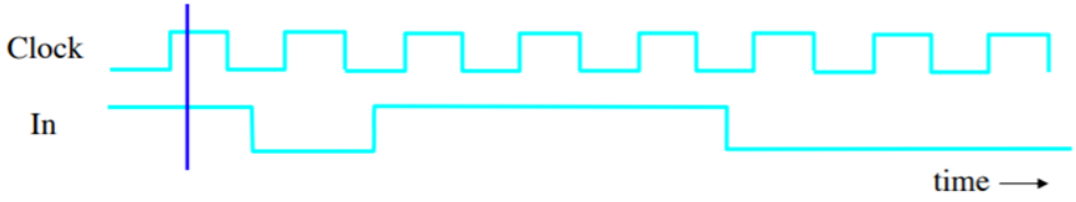


FIGURE 8–1 The flip-flop as a storage element.

Shift Register Simulation

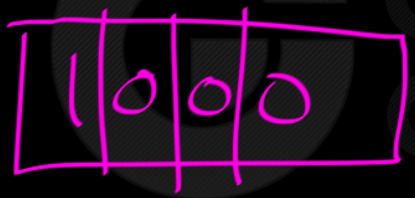
SISO
Right shift register



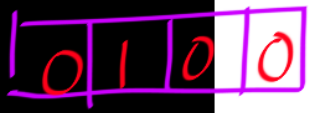


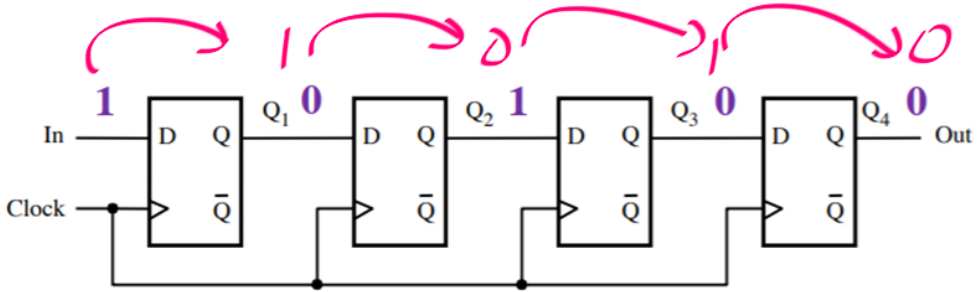
Register Content = Flip flop output

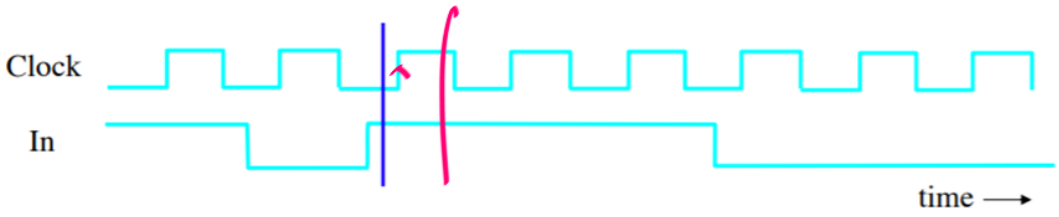
Combination = State of the register

= 

Shift Register Simulation

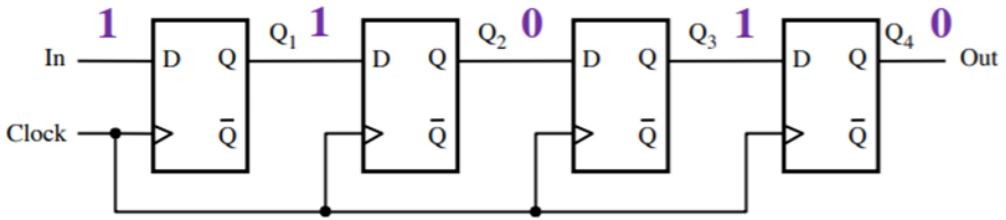


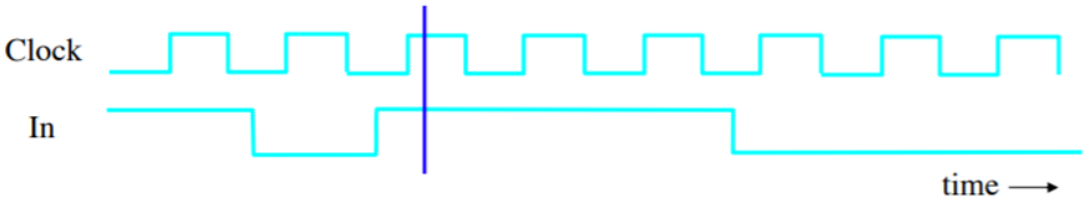




Shift Register Simulation

1	0	1	0





Some Terminologies :

Present state: before clock

Next state: after clock

Some Terminologies :

If all flip flops are positive edge triggered flipflops then :

Present state: before clock rising edge
Next state: after clock rising edge

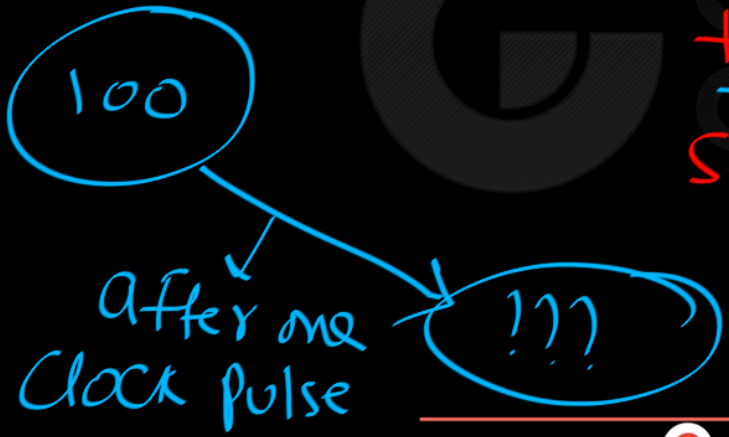
Some Terminologies :

If all flip flops are negative edge triggered flipflops then :

Present state: before clock falling edge
Next state: after clock falling edge

Sequential ckt = ckt with flipflops

output combination of these ffs is called State of the seq. ckt.



7.9.7 State Diagram

A *state diagram* is a graph that shows the flip-flop's operations in terms of how it transitions from one state to another. The nodes are labeled with the states and the directed arcs are labeled with the input signals that cause the transition to go from one state to the next. Figure 21 shows the state diagram for the SR flip-flop. For example, to go from state $Q = 0$ to the state $Q = 1$, the two inputs S and R have to be 1 and 0 respectively. Similarly, if the current state is $Q = 0$ and we want to remain in that state, then SR need to be 00 or 01.

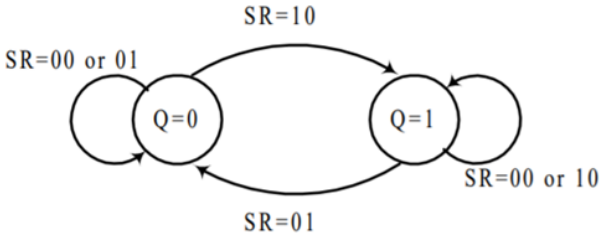


Figure 21. State diagram for the SR flip-flop.

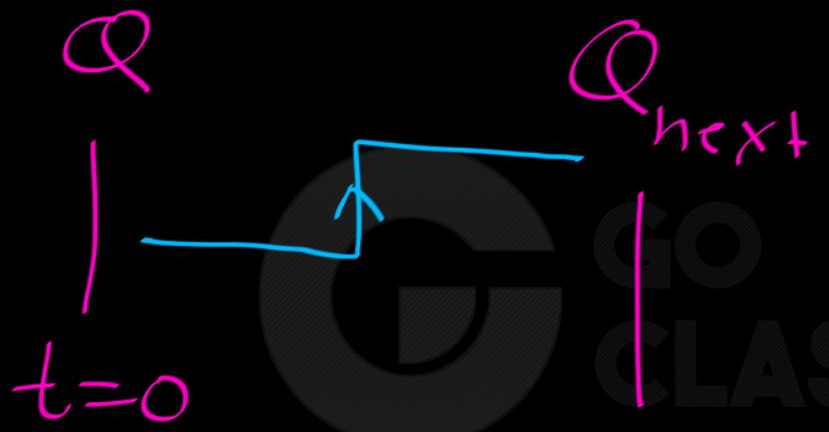
SR-ff:

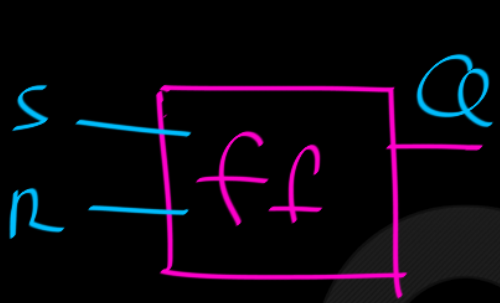
Q_{next} = next state = next output = next Q value

S	R	Q_n
0	0	Q
0	1	0
1	0	1
1	1	forbidden

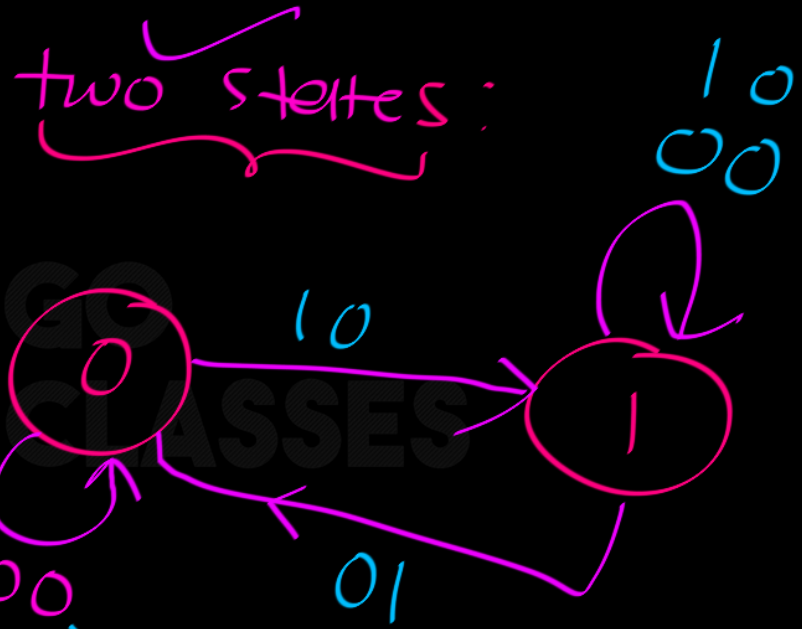
Q_n → present state = present output = present Q value

State table


Q
Q next
t=0



two states :



SR-ff

State Diagram

SR-ff state table:

S	R	Q_n
0	0	Q
0	1	0
1	0	1
1	1	forbidden

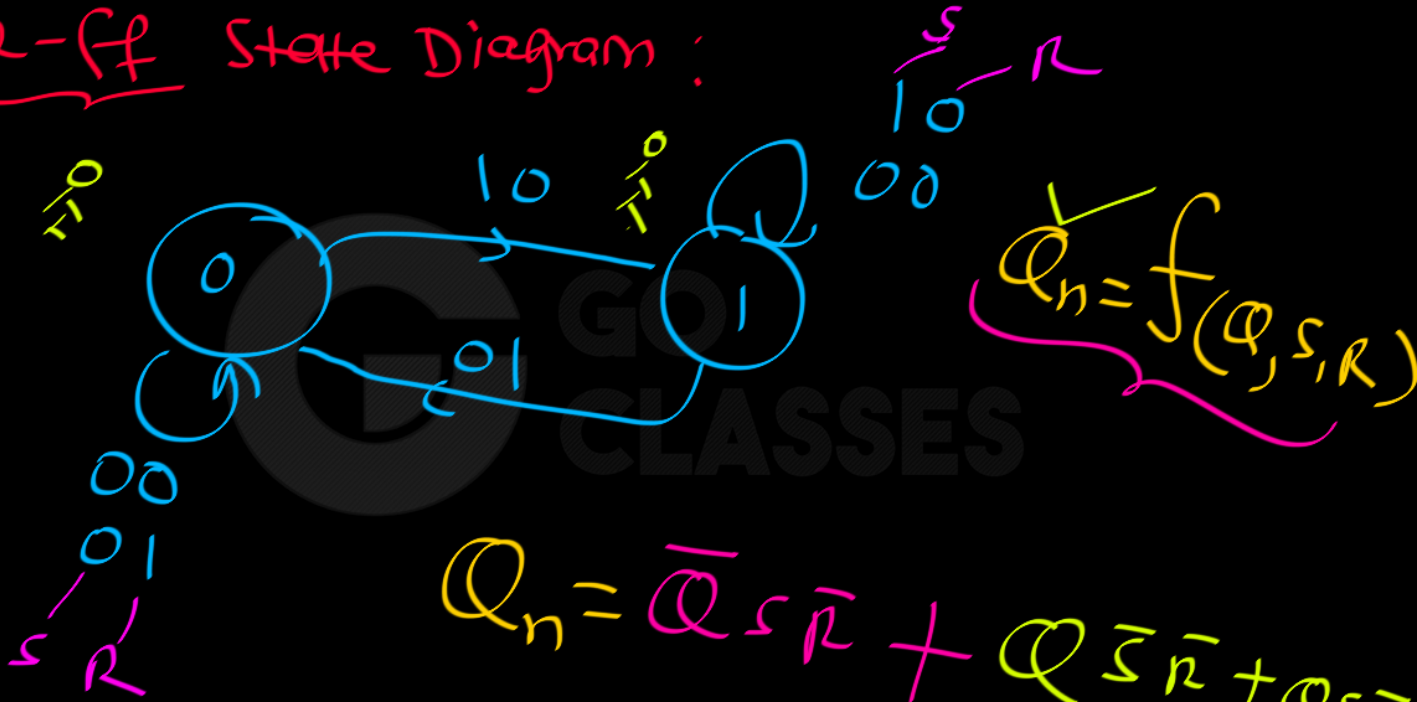
Next state Equation

$Q_n = f(Q, S, R)$

Q_n — Next state; Q — Present state; S, R — inputs

$Q_n = \bar{S}\bar{R}Q + \bar{S}R\,0 + S\bar{R}\,1 = \bar{S}\bar{R}Q + S\bar{R}$

SR-ff State Diagram :



$$Q_n = f(Q, S, R)$$

$$Q_n = \bar{Q} S \bar{R} + Q \bar{S} \bar{R} + Q S \bar{R}$$

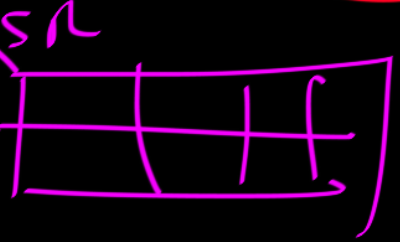
$Q_n = \bar{Q}\,s\,\bar{r} + Q\,\bar{s}\,\bar{r} + Q\,s\,\bar{r}$

$Q_n = s\,\bar{r} + Q\,\bar{s}\,\bar{r}$

SR-ff: $s=1=r$ forbidden

Don't care Combination

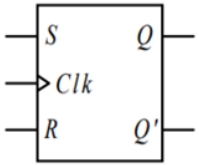
k-map



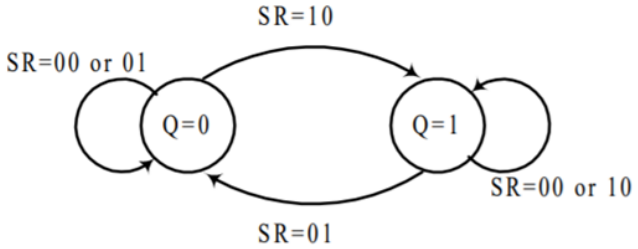
$Q_n = ?$

Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table

SR



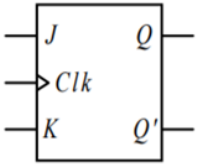
S	R	Q	Qnext
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	×
1	1	1	×



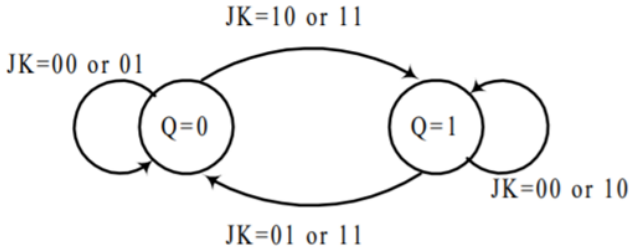
$$Q_{next} = S + R'Q$$
$$SR = 0$$

Q	Qnext	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

JK



J	K	Q	Qnext
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$
\begin{aligned}
Q_{next} &= J'K'Q + JK' + JKQ' \\
&= J'K'Q + JK'Q + JK'Q' + JKQ' \\
&= K'Q(J'+J) + JQ'(K'+K) \\
&= K'Q + JQ'
\end{aligned}
$$

Q	Qnext	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

JK ff:

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

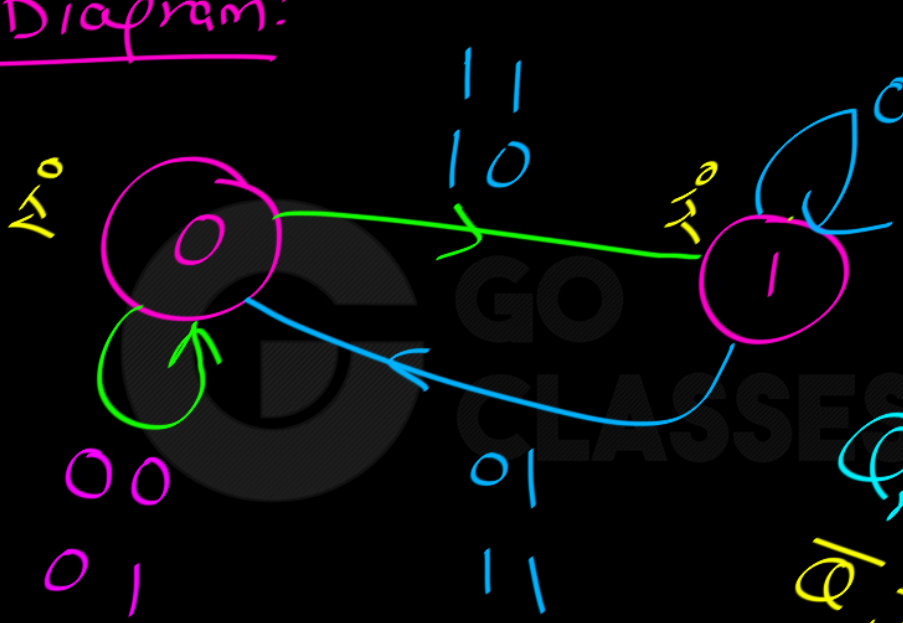
State table

$Q_n = f(Q, J, K)$

$Q_n = f(\text{present state}, \text{input})$

$Q_n = \bar{J}K\bar{Q} + \cancel{\bar{J}K0} + J\bar{K}\cdot 1 + JK\bar{Q}$

State Diagram:

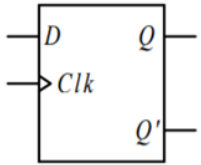


Next State Equation:

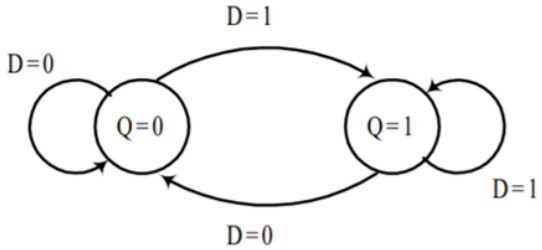
$$Q_n = \bar{Q}JK + Q\bar{J}K + Q\bar{J}\bar{K} + Q\bar{J}\bar{K}$$

using K-map, we can get minimized equation for Next state Q_n BUT we have seen another way to write next state equation.

D



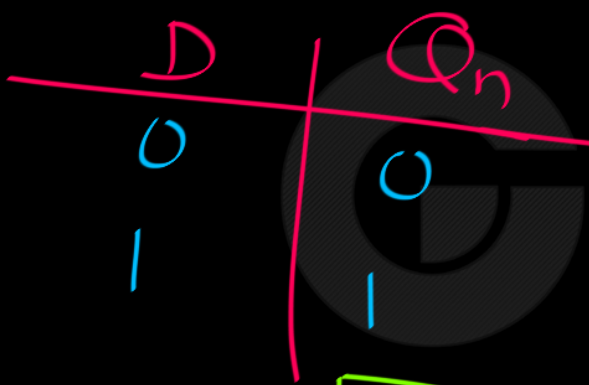
D	Q	Qnext
0	×	0
1	×	1



$Q_{next} = D$

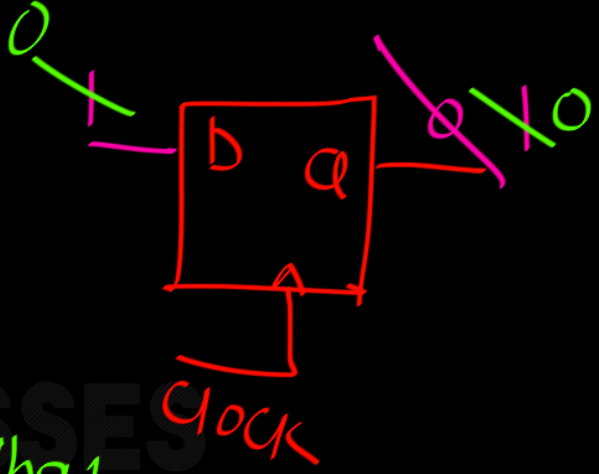
Q	Qnext	D
0	0	0
0	1	1
1	0	0
1	1	1

D-ff (Data ff)



$Q_n = D$

State table

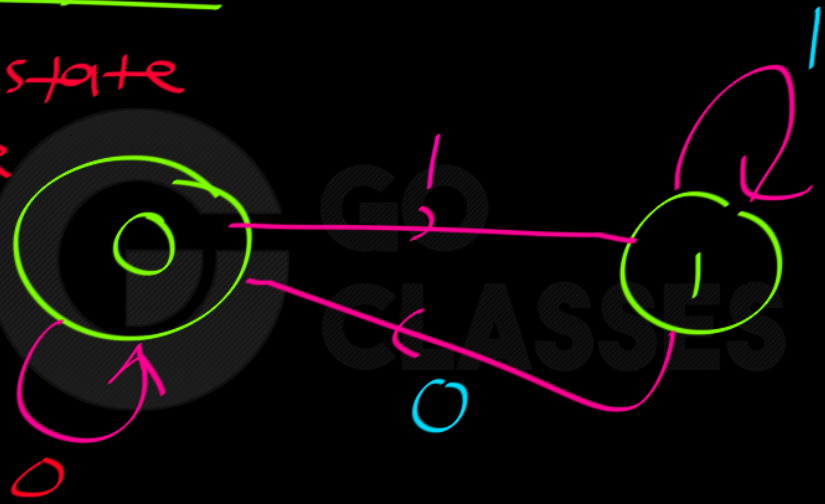


Whatever Data you want to store, Just apply that Data on input.

State Diagram:

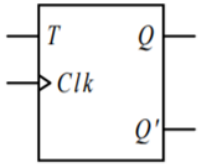
Q = Present state

Q_n = Next state

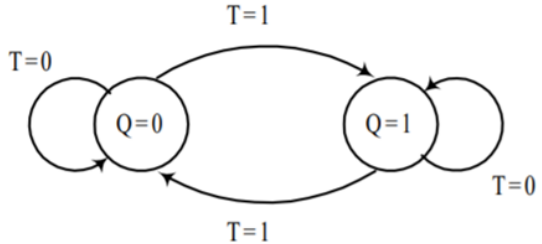


$\therefore Q_n = \bar{Q}D + QD = D$

T



T	Q	Q_{next}
0	0	0
0	1	1
1	0	1
1	1	0



$$Q_{next} = TQ' + T'Q = T \oplus Q$$

Q	Q_{next}	T
0	0	0
0	1	1
1	0	1
1	1	0

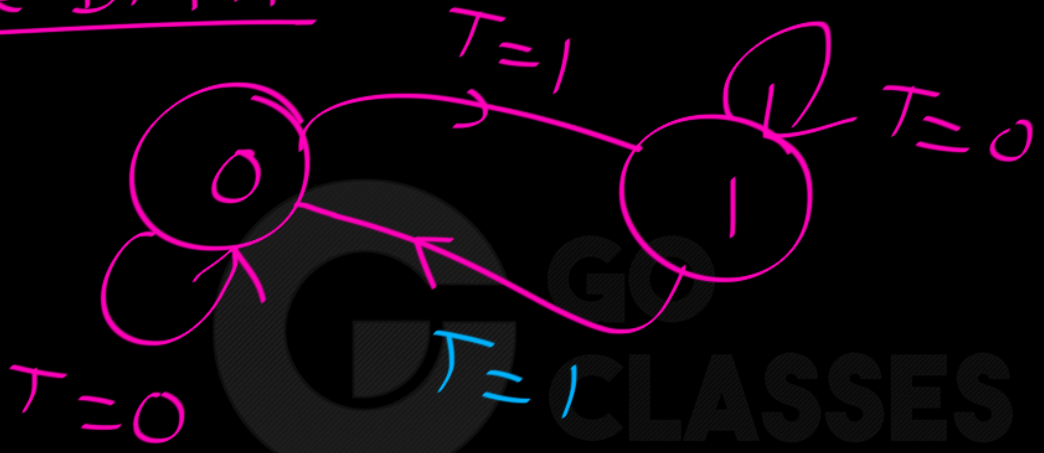
T-ff (Toggle ff)

T	Q_n
0	Q
1	$\overline{Q}$

State Table

$$Q_n = \overline{T}Q + T\overline{Q}$$
$$= T \oplus Q$$

State Diagram:



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

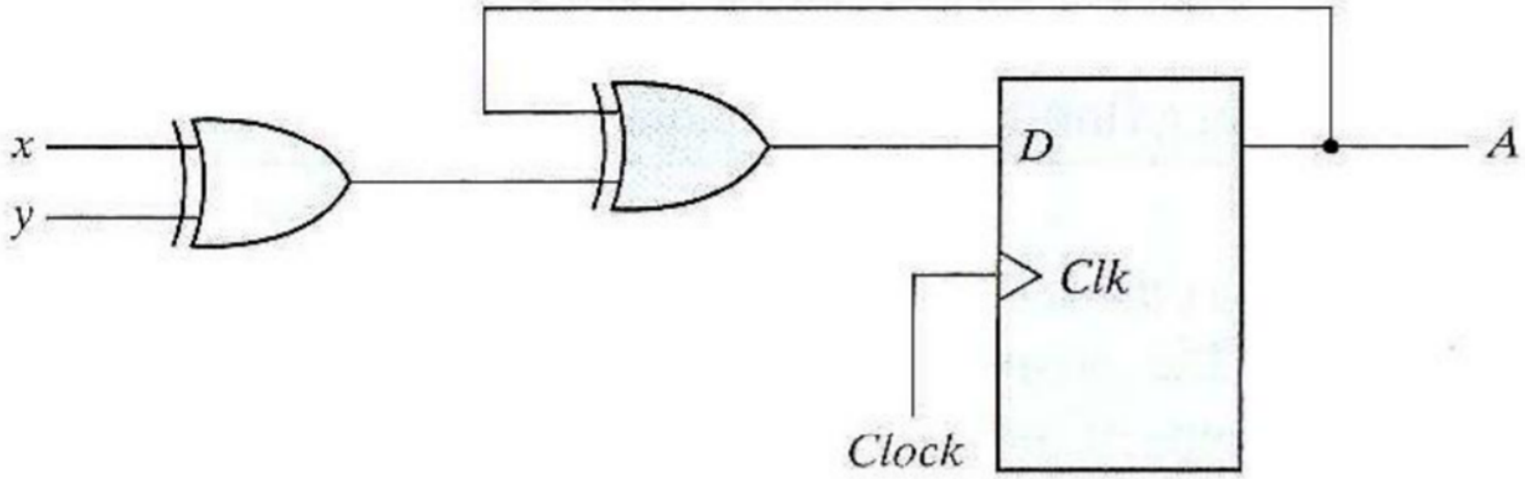
1 flip-flop => 2 states (each state is of 1 bit)

2 flip-flops => 4 states (each state is of 2 bit)

3 flip-flops => 8 states flops => 8 states (each state is of 3 bit)

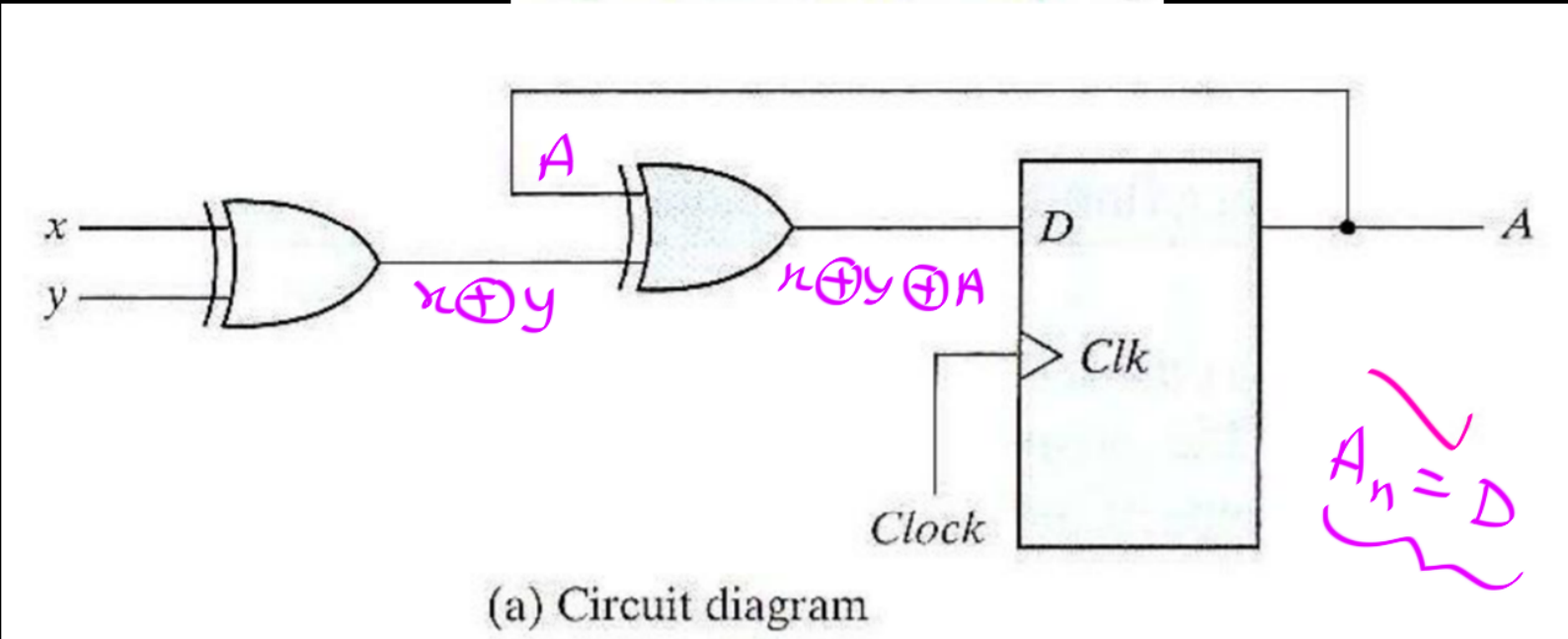
4 flip-flops => 16 states (each state is of 4 bit)

Sequential circuit with *D* flip-flop



(a) Circuit diagram

Sequential circuit with *D* flip-flop



(a) Circuit diagram

$D = x \oplus y \oplus A$

$A_n = f(A, x, y)$

A_n ↙ next state

A, x, y ↙ present state

State Table:

x	y	A_n
0	0	A
0	1	$\bar{A}$
1	0	$\bar{A}$
1	1	A

$A_n = D$

$0 \oplus A = A$

$1 \oplus A = \bar{A}$

$D = x \oplus y \oplus A$

$A_n = f(A, x, y)$

A_n ← next state, A ← present state

State Table:

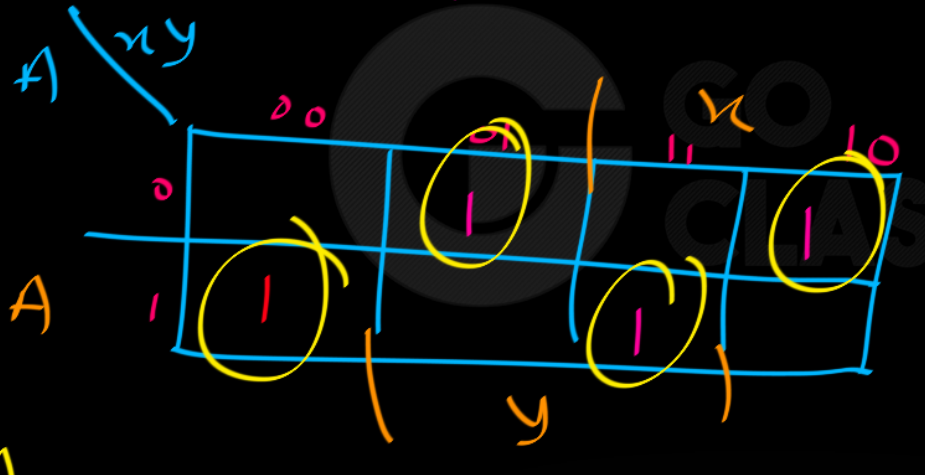
x	y	A_n
0	0	A
0	1	$\bar{A}$
1	0	$\bar{A}$
1	1	A

$A_n = \bar{x}\bar{y}A + \bar{x}y\bar{A} + x\bar{y}\bar{A} + xyA$

To get minimized SOP equation of A_n:

use k-map method

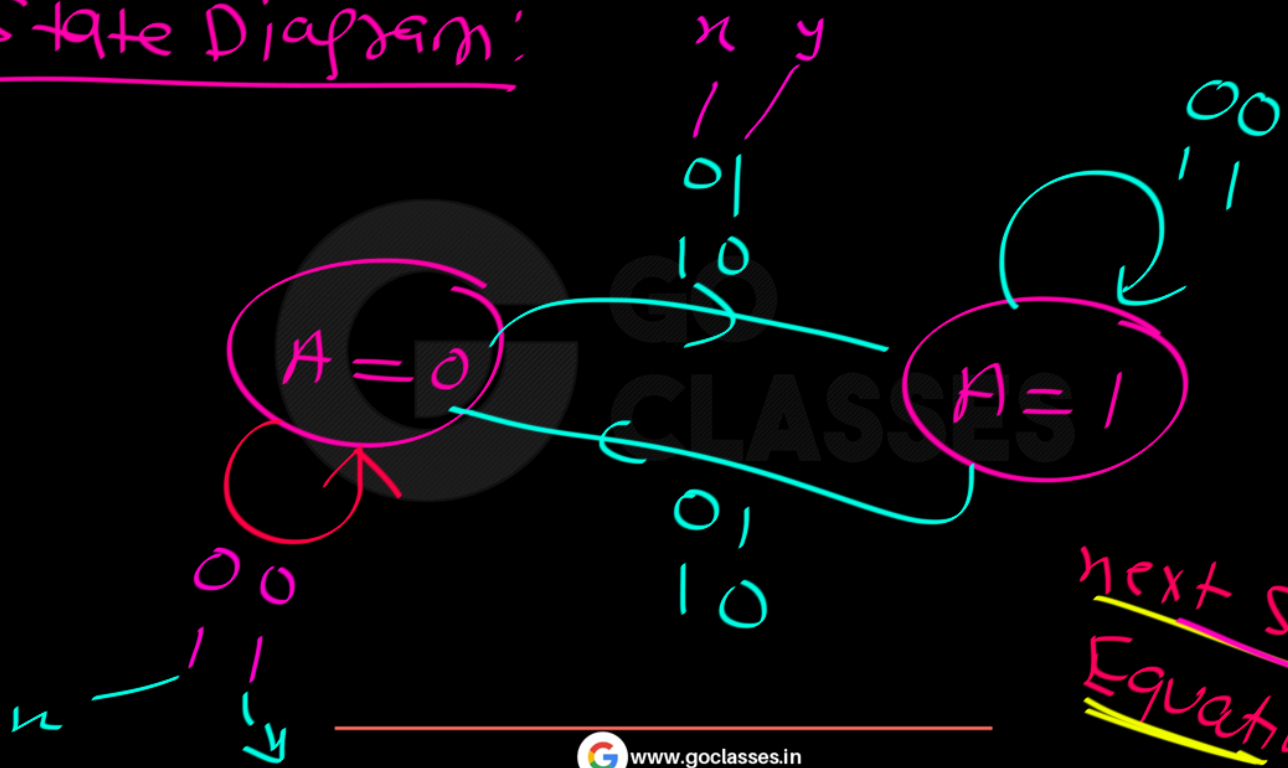
$A_n = f(A, x, y)$



k-map for A_n

$A_n = \bar{x}\bar{y}A + \bar{x}y\bar{A} + x\bar{y}\bar{A} + xyA$

State Diagram:

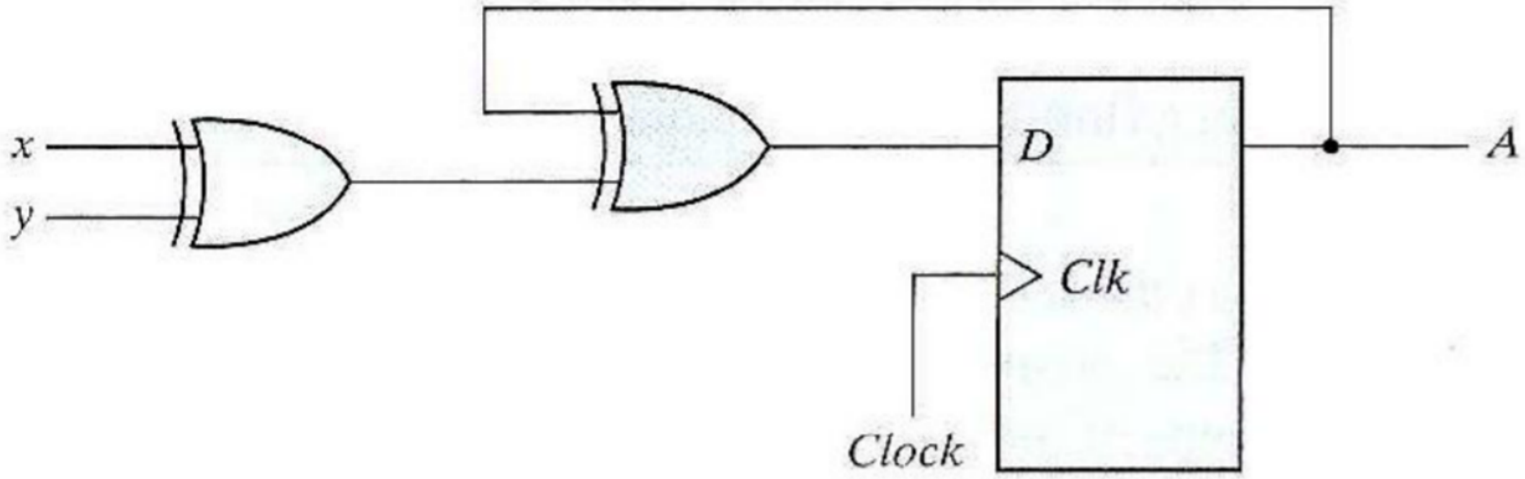


next state Equation ??

$$A_n = \bar{A}x\bar{y} + \bar{A}\bar{x}y + Axy + A\bar{x}\bar{y}$$

next state

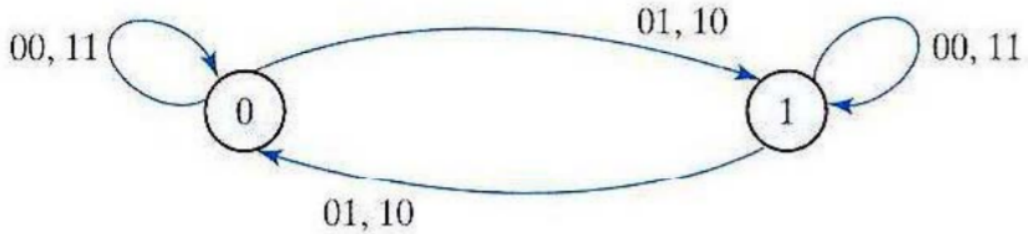
Sequential circuit with *D* flip-flop



(a) Circuit diagram

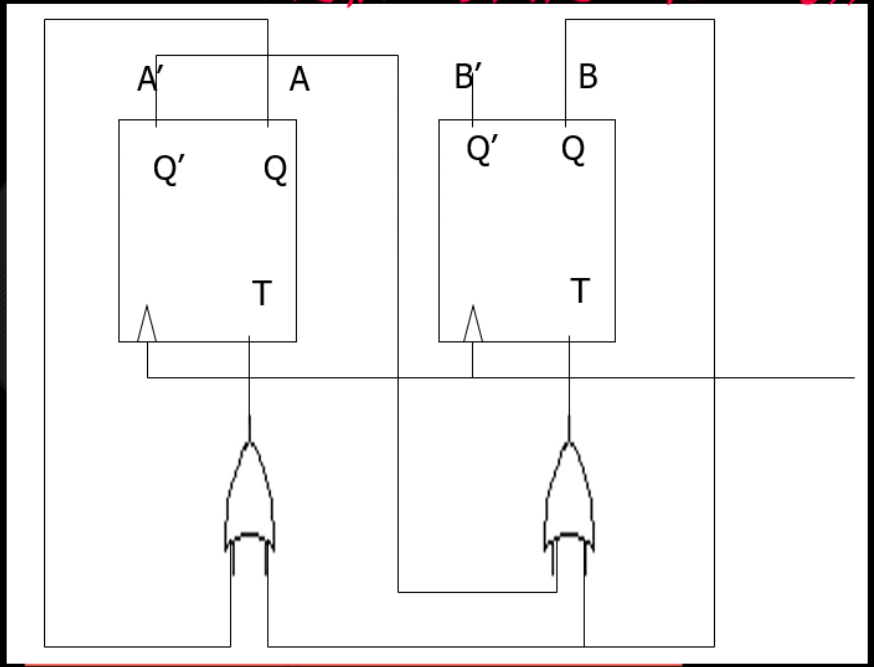
Present state	Inputs		Next state
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table



(c) State diagram

Derive the state table and state diagram of the sequential circuit of the Figure below.
What is the function of the circuit? Next State Equation ??

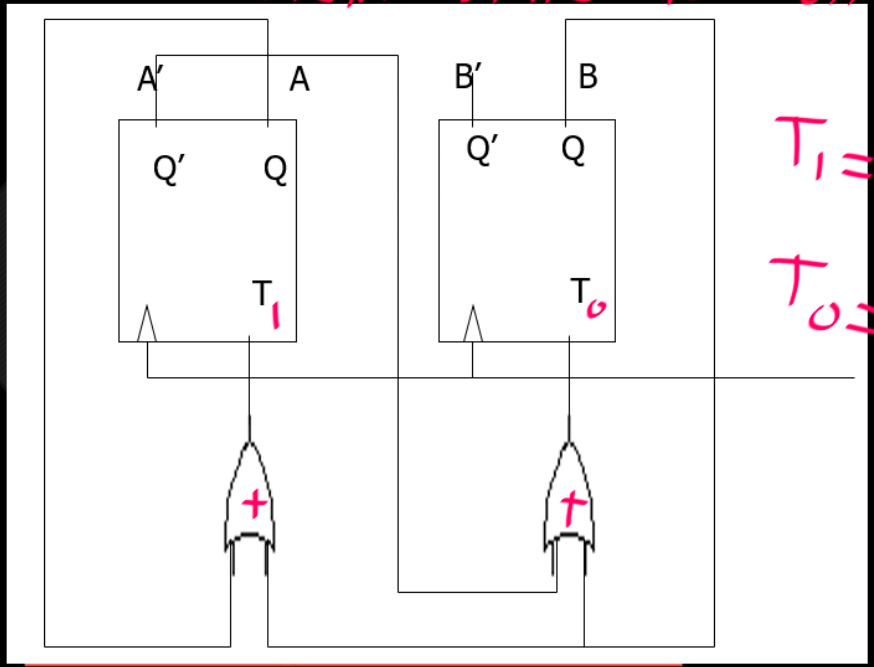


Derive the state table and state diagram of the sequential circuit of the Figure below.
What is the function of the circuit?

Next State Equation ??

4 states Possible:

A B
(0,0)
(0,1)
(1,0)
(1,1)



$T_1 = A + B$

$T_0 = \bar{A} + B$

State Table: Next State in sequential Circuit = f(present state, external inputs)

So

Next State = f(present state)

← no external inputs in prev. seq. ckt

because we have two flipflops, so State of the seq. ckt has two bits (A, B).

So

$$(A_n, B_n) = f(A, B)$$

(A_n, B_n): next state of seq. ckt

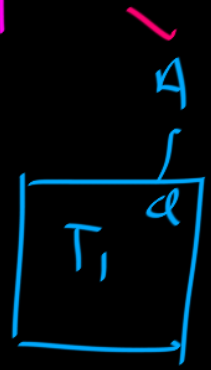
(A, B): present state of seq. ckt.

State table :

A	B	A_n	B_n	T_1	T_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	0	0	1	0
1	1	0	0	1	1

$T_1 = A + B$

$T_0 = \bar{A} + B$



State table :

A	B	A_n	B_n
0	0	0	1
0	1	1	0
1	0	0	0
1	1	0	0

$A_n = \bar{A}B$

$B_n = \bar{A}\bar{B}$

$(A_n, B_n) = f(A, B)$

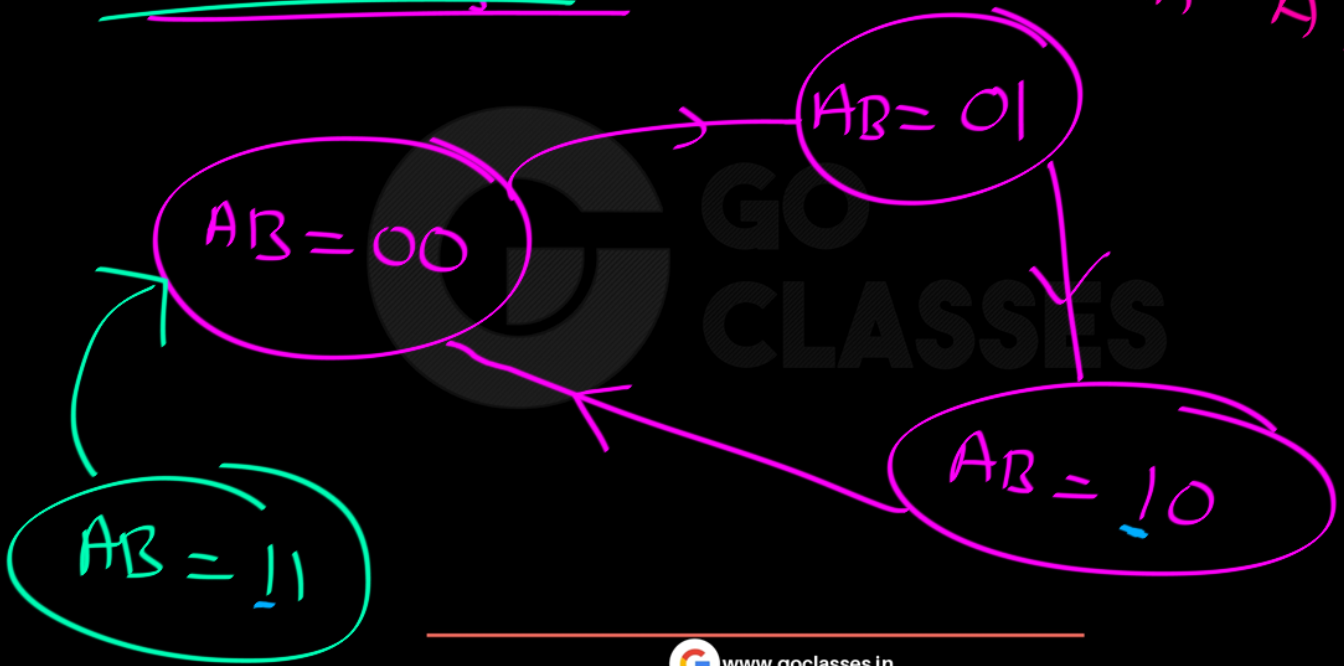
$(A_n, B_n) = (\bar{A}B, \bar{A}\bar{B})$

T - FF

T	Q_n
0	Q
1	$\bar{Q}$

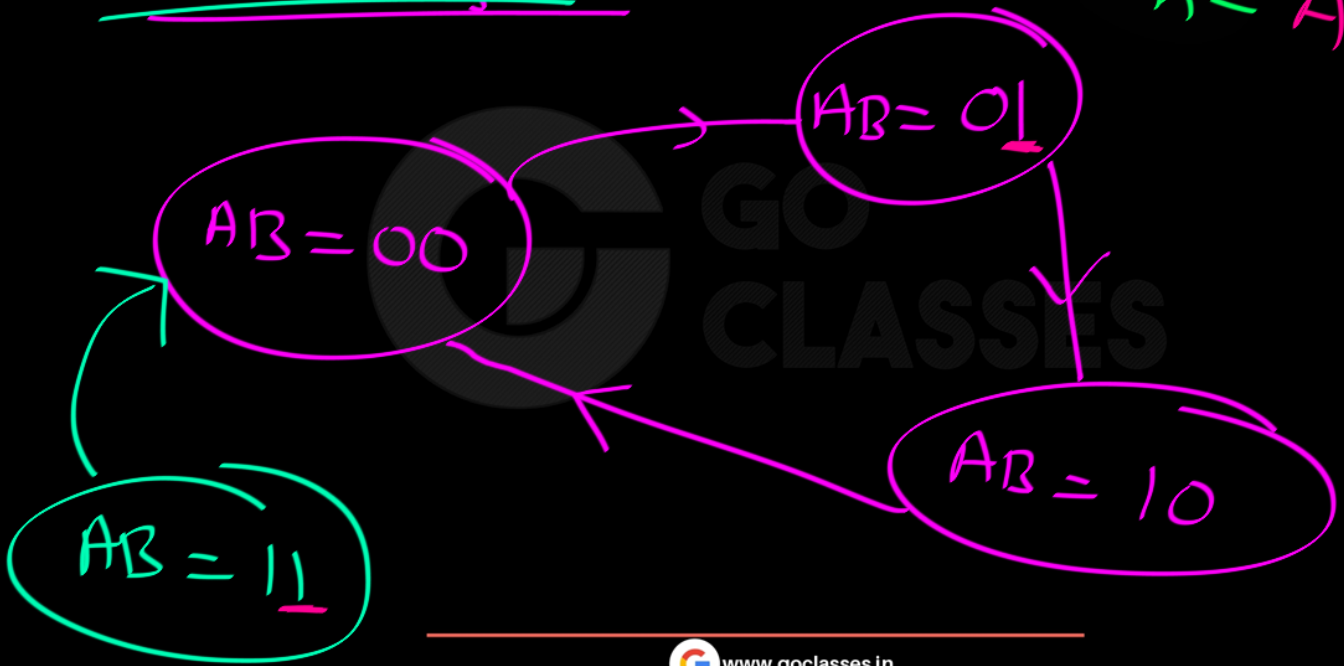
State Diagram:

$A_n = \bar{A} B$



State Diagram:

$B_n = \overline{A}\,\overline{B}$



Next State Equation: $(A_n, B_n) = f(A, B)$

(A_n, B_n) → next state of the given seq. ckt

(A, B) → present state of the given seq ckt

$$A_n = f(A, B) \quad ; \quad B_n = f(A, B)$$

Next Topic :

FSM → finite state machine

- Mealy
- Moore

- Combinational and Sequential Circuits

– In a combinational circuit, the outputs depend only on the applied input values and not on the past history.

– In a sequential circuit, the outputs depend not only on the applied input values but also on the internal state.

- The internal states also change with time.

- The number of states is finite, and hence a sequential circuit is also referred to as a Finite State Machine (FSM).
- Most of the practical circuits are sequential in nature.

How are the States Stored and Changed in Sequential Circuits ?

- In a sequential circuit, we need some mechanism to store the internal states.

– We use flip-flops or latches to store the internal states.

– If we use k flip-flops, there can be a maximum of 2^k internal states. So, number of states possible for a sequential circuit is finite, hence, sequential circuit is also referred to as a Finite State Machine (FSM).

• Depending upon the way the states change, sequential circuits can be categorized into two types:

a) Synchronous: The states change in synchronism with a clock pulse.

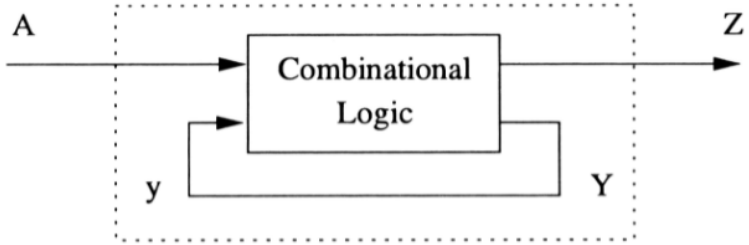
b) Asynchronous: There is no clock pulse for synchronism; all state changes occur depending on the delays of the circuit elements (e.g. gates).

Synchronous vs. Asynchronous

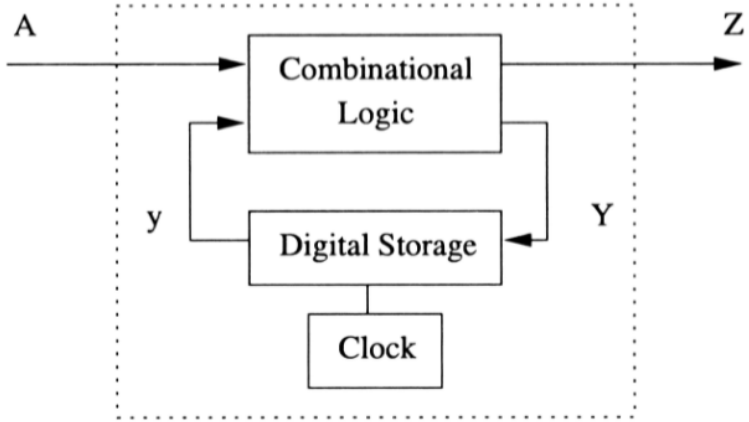
- Synchronous circuits have sequential elements whose outputs change at the same time.
- Asynchronous circuits have sequential elements whose outputs change at different times.

- Disadvantages of Asynchronous Circuits
  - Difficult to analyze operations
  - Intermediate states that are not part of the desired design may be generated

Asynchronous



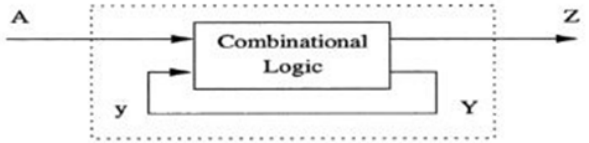
Synchronous



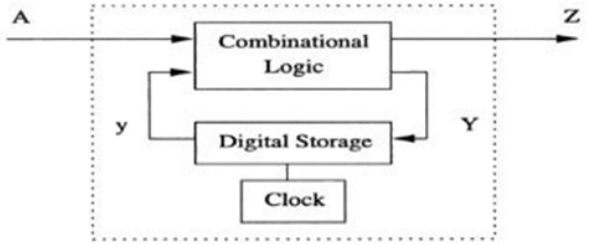
Types of Sequential Circuits

In Asynchronous sequential circuits the output of the logic circuit can change state at any time, as soon as any input changes its state whereas in the case of synchronous systems a signal namely clock signal is used to determine/control the exact time at which any output can change its state. These are also called as clocked sequential circuits.

Asynchronous



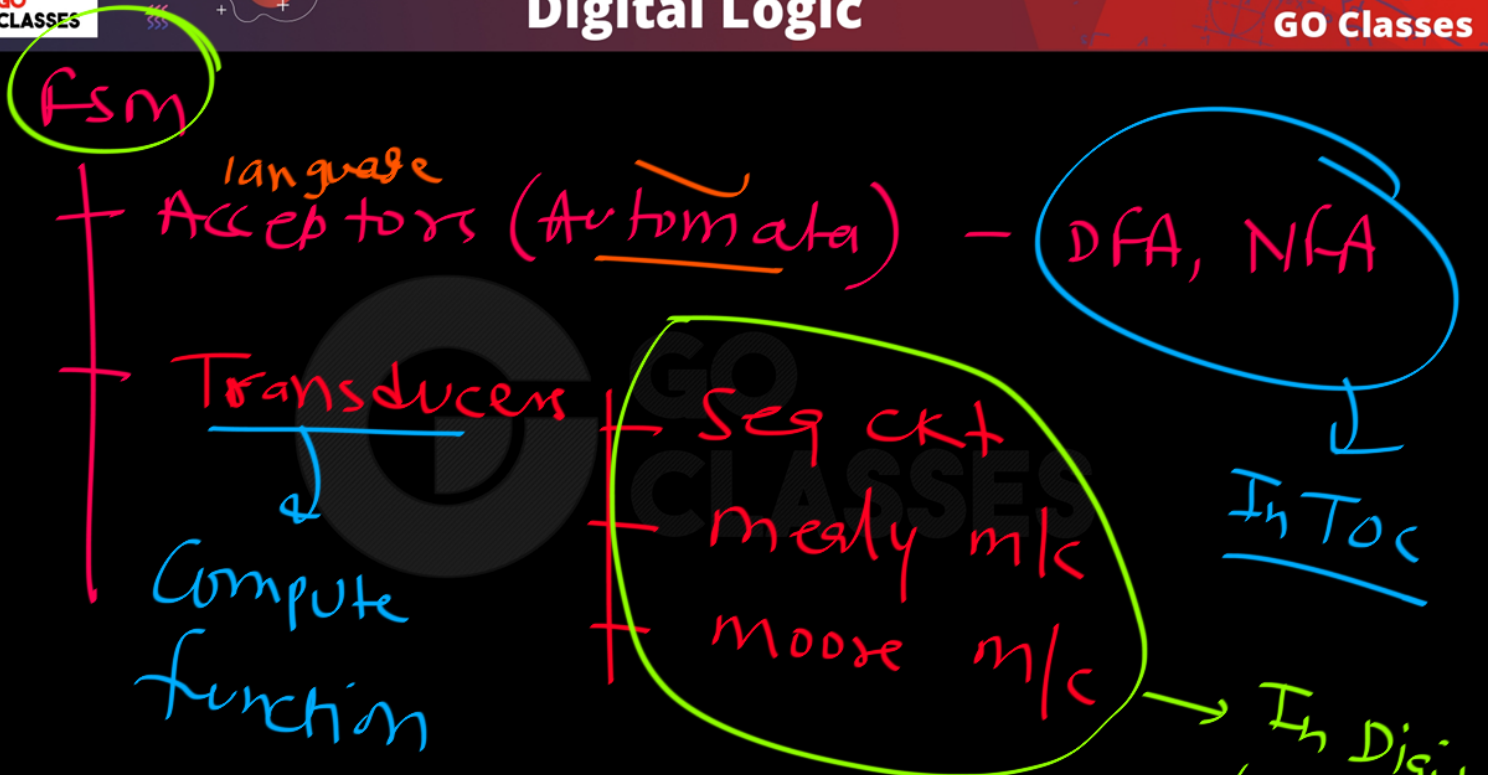
Synchronous



Finite State Machine (FSM) (synchronous sequential machine)

In a sequential circuit, We use flip-flops to store the internal states.

– If we use k flip-flops, there can be a maximum of 2^k internal states. So, number of states possible for a sequential circuit is finite, hence, sequential circuit is also referred to as a Finite State Machine (FSM).


FSM
language
Acceptors (Automata) – DFA, NFA
In ToC
Transducers
Compute function
Seq ckt
Mealy m/c
Moore m/c
In Digital

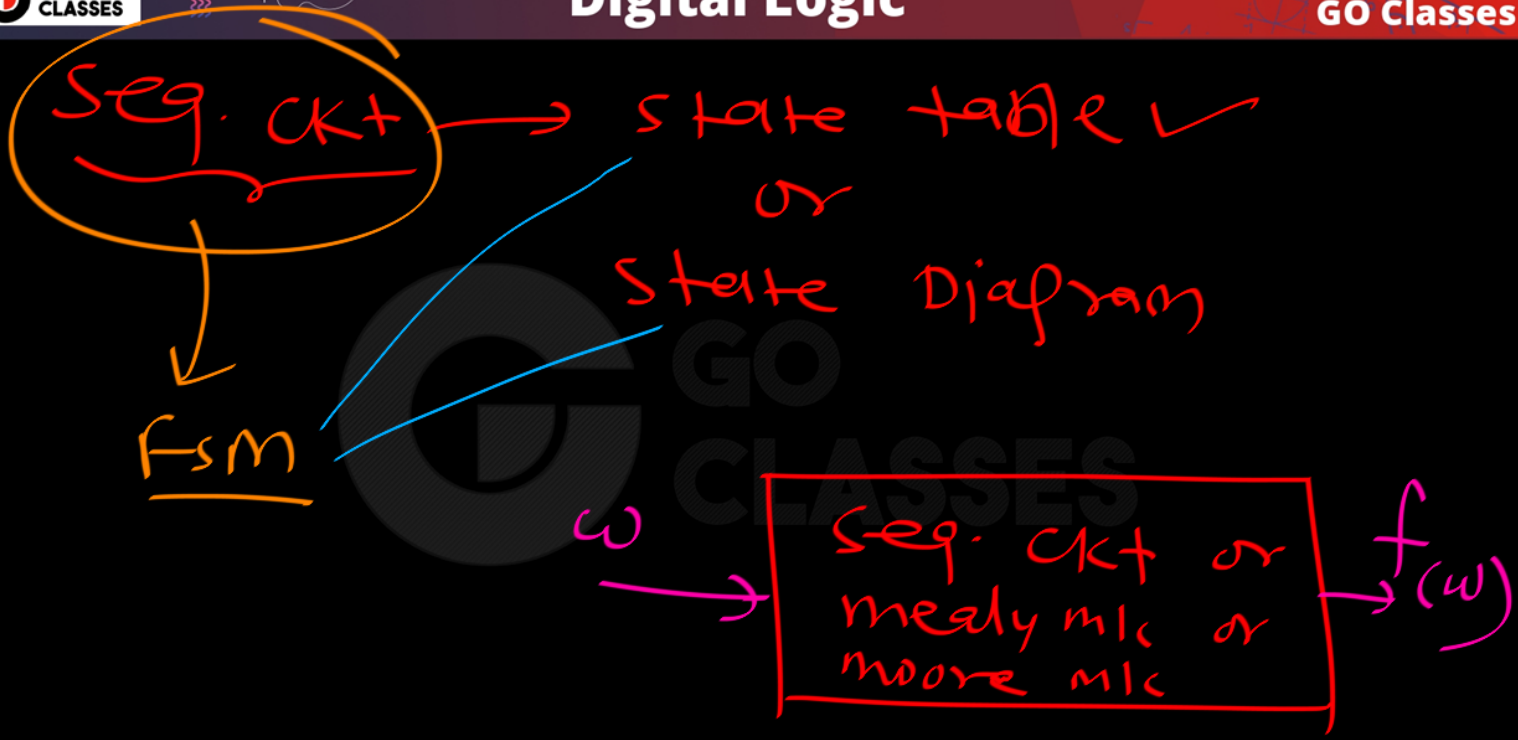
Finite State Machine (FSM) (synchronous sequential machine) :

- A FSM can be represented either in the form of a state table or in the form of a state transition diagram.

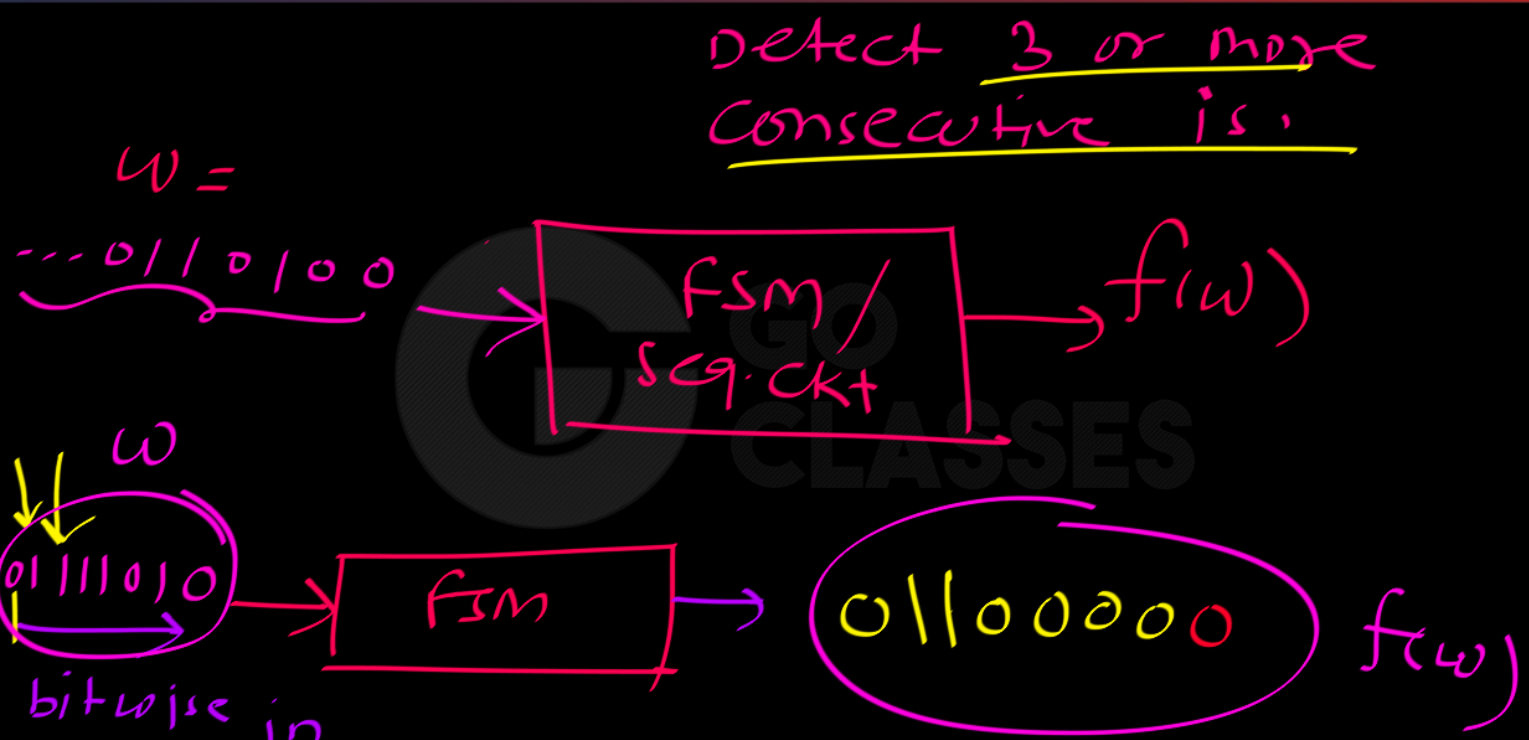
Finite State Machine (FSM):

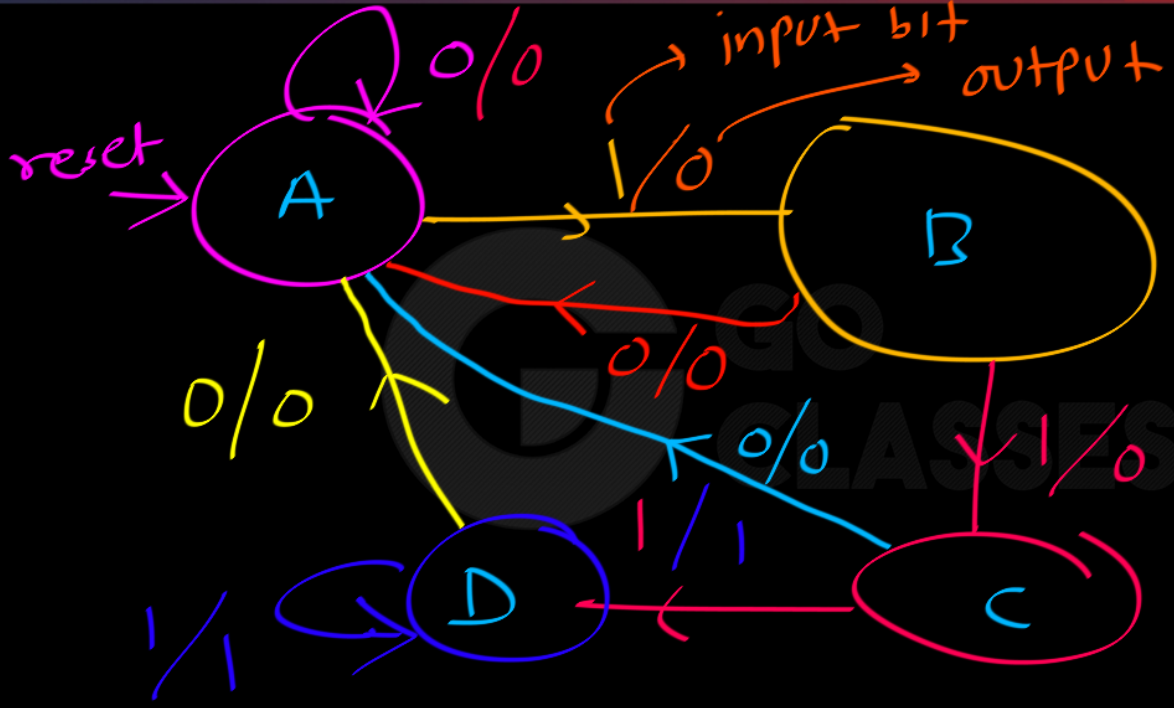
- A FSM can be represented either in the form of a state table or in the form of a state transition diagram.
- Example: Specification of Problem :
  - A circuit to detect 3 or more 1's in a serial bit stream.
  - The bits are applied serially in synchronism with a clock.
  - The output will become 1 whenever it detects 3 or more consecutive 1's in the stream, else 0.

In every clock, one input is Applied.

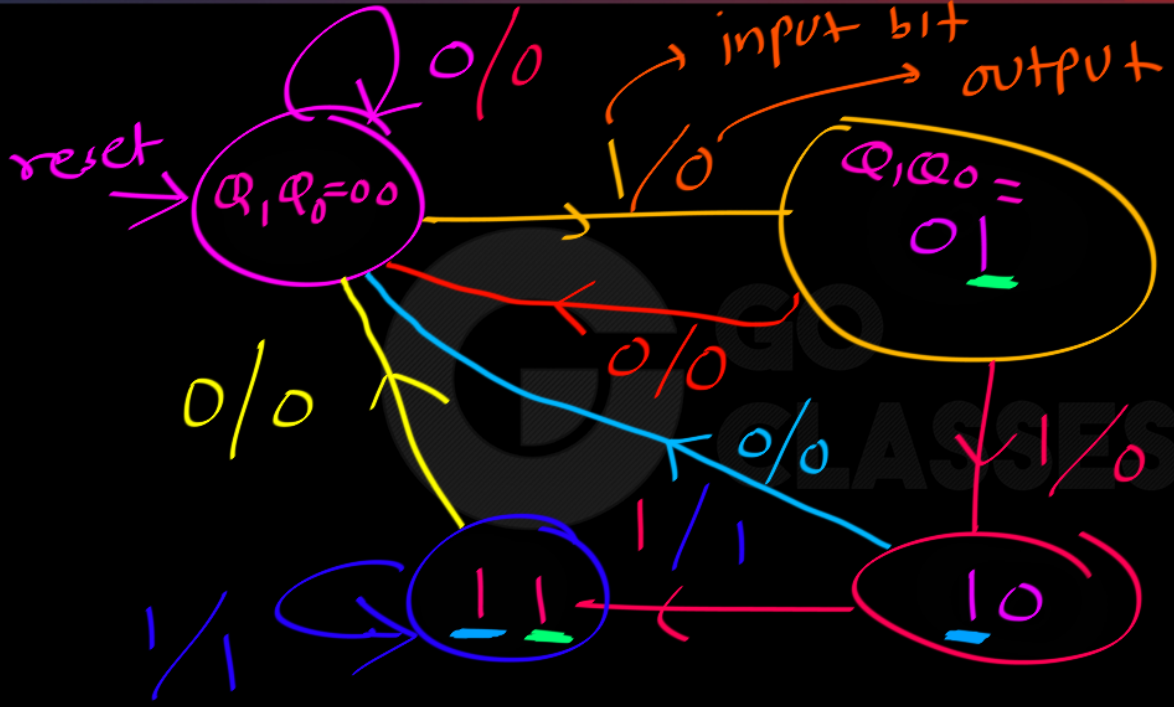

Seq. ckt
state table
or
state Diagram
FSM
w
Seq. ckt or
mealy mlc or
moore mlc
f(w)

Detect 3 or more consecutive 1's.





State Diagram of FSM (seq. ckt) that we need to Design.



State Diagram of FSM (seq. ckt) that we need to Design.

from state Diagram, next state input equation : $(Q_{1n}, Q_{0n}) = f(x, Q_1, Q_0)$

present state

$$Q_{1n} = \bar{Q}_1 Q_0 x + Q_1 \bar{Q}_0 x + Q_1 Q_0 x$$

$$Q_{1n} = x(Q_0 + Q_1)$$

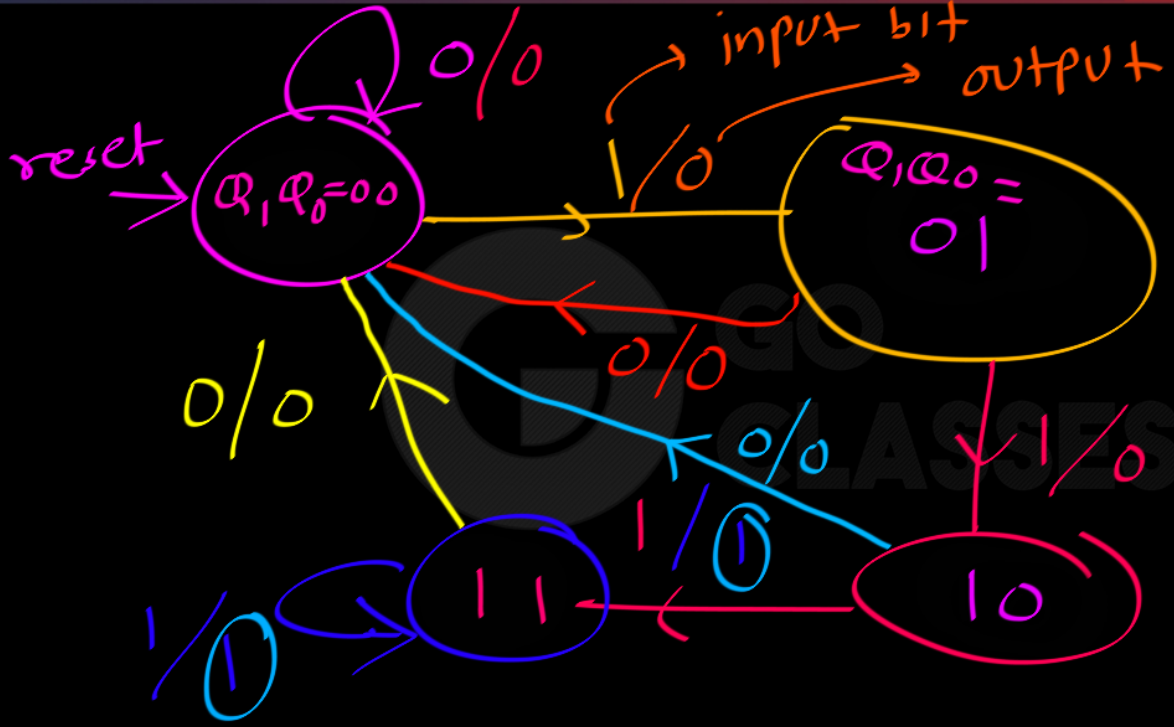
$$Q_{on} = \bar{Q}_1 \bar{Q}_0 x + Q_1 \bar{Q}_0 x + Q_1 Q_0 x$$

$$Q_{on} = \bar{Q}_1 \bar{Q}_0 x + Q_1 x$$

$$= x(Q_1 + \bar{Q}_1 \bar{Q}_0)$$

$$Q_{on} = x(Q_1 + \bar{Q}_0)$$

$$a + \bar{a}b = a + b$$


reset
$Q_1Q_0=00$
$Q_1Q_0=01$
$Q_1Q_0=11$
$Q_1Q_0=10$
0/0
1/0
input bit
output
State Diagram of FSM (seq. ckt) that we need to Design.

from state Diagram;

Next output equation = $Z = f(x, Q_1, Q_0)$ (x: input)

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

output variable = $Q_1 x$ → output equation

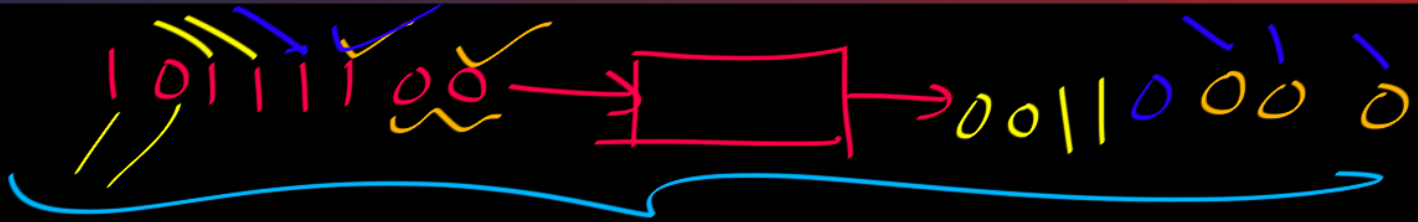
Output equation:

$$z = Q_1 x$$

$$Q_{0n} = x(Q_1 + \overline{Q_0})$$

$$Q_{1n} = x(Q_0 + Q_1)$$


D1
Q1
D0
Q0



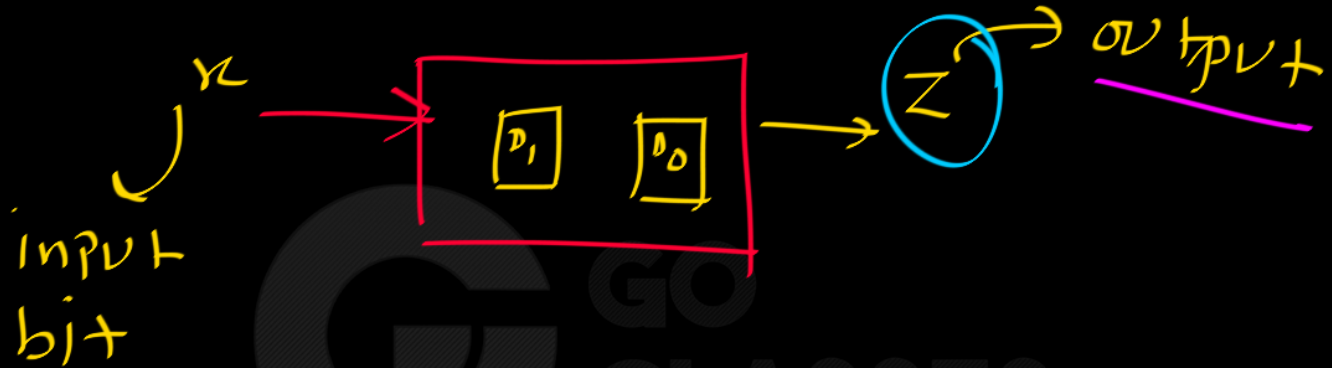
so we need 4 states

2 ffs needed

Seq ckt:

4 states

	Q_1	Q_0
A	0	0
B	0	1
C	1	0
D	1	1



Next output equation Z

" State " Q_{1n}, Q_{0n}

Output equation:

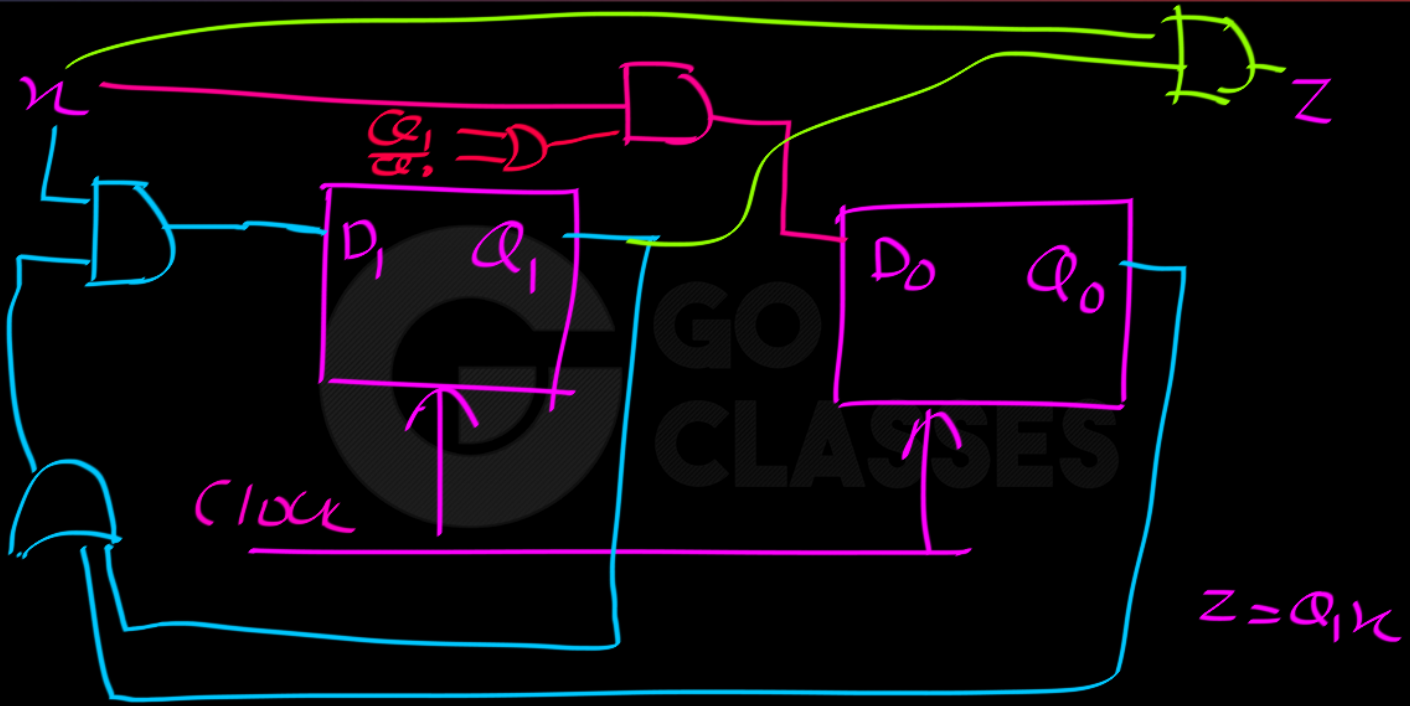
$$z = Q_1 x$$

$$Q_{0n} = x(Q_1 + \overline{Q_0}) = D_0$$

$$Q_{1n} = x(Q_0 + Q_1) = D_1$$

In D-ff

$$Q_n = D$$


x
Q_1
$\overline{Q_1}$
D
D_1
Q_1
D_0
Q_0
Clock
z
$z = Q_1 x$

Finite State Machine (FSM):

- A FSM can be represented either in the form of a state table or in the form of a state transition diagram.
- Example:

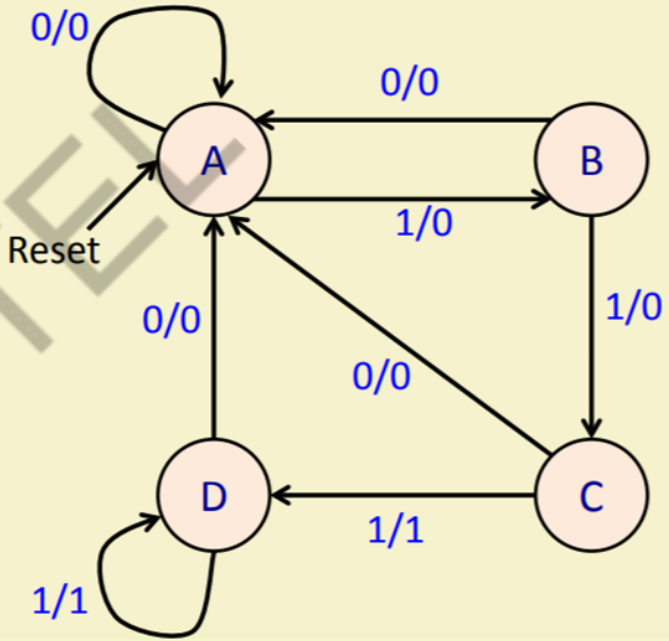
– A circuit to detect 3 or more 1's in a serial bit stream.
– The bits are applied serially in synchronism with a clock.
– The output will become 1 whenever it detects 3 or more consecutive 1's in the stream, else 0.

State Table

Reset	PS	Input	NS	Output
1	-	-	A	0
0	A	0	A	0
0	A	1	B	0
0	B	0	A	0
0	B	1	C	0
0	C	0	A	0
0	C	1	D	1
0	D	0	A	0
0	D	1	D	1

Normal operation of ckt

State Transition Diagram



finite State machine (FSM)

TOC — DFA ✓, NFA ✓, PDA ✗, TM ✗

Digital logic : Sequential ckt ✓, Mealy m/c, Moore m/c,

FSM:

Anything that has finite number of states AND No Additional Memory.

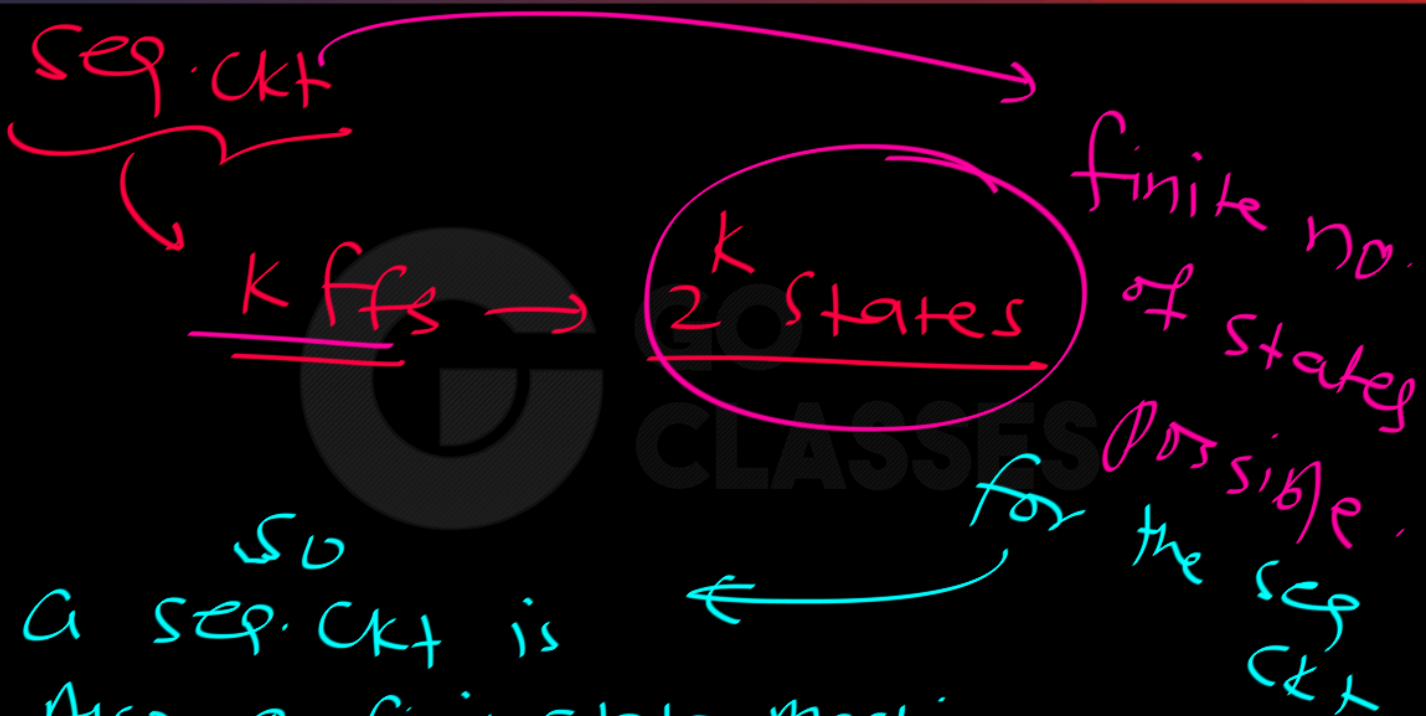
Sequential Ckt : Ckt with flipflops

finite number of flipflops

2 ffs

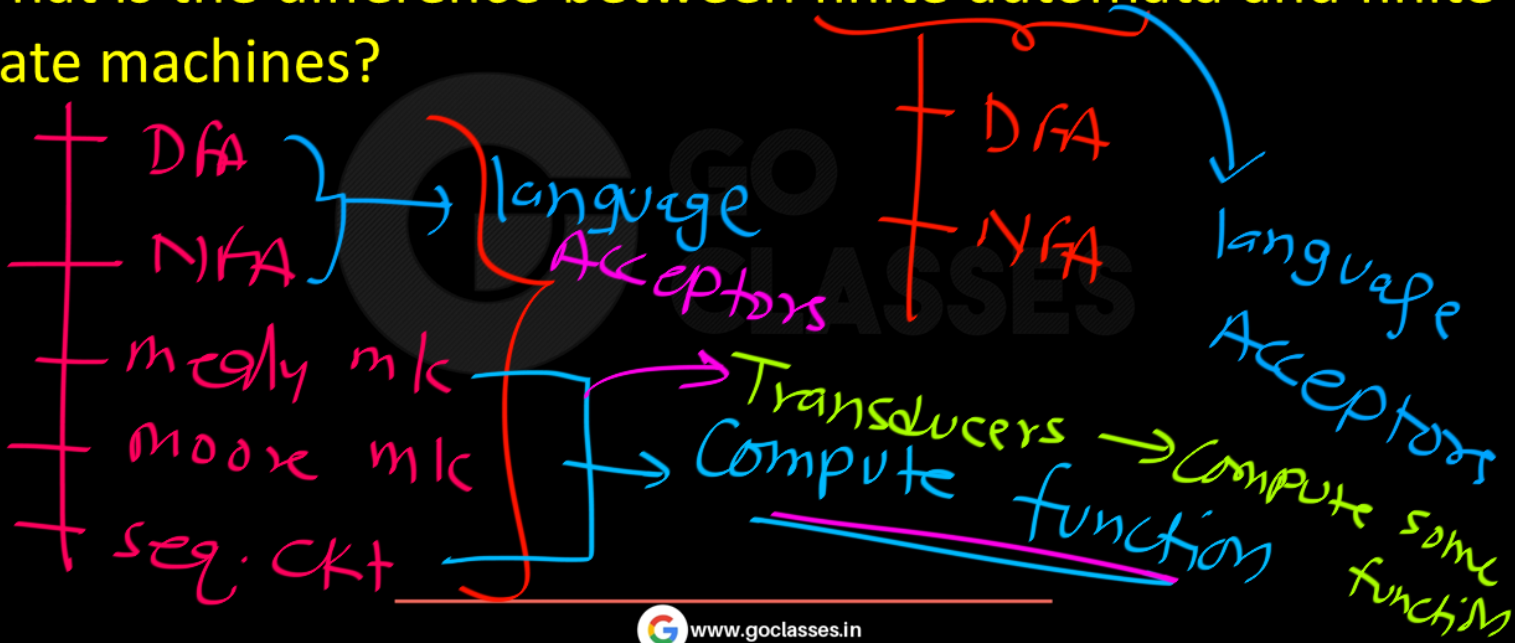
4 possible states of seq. ckt

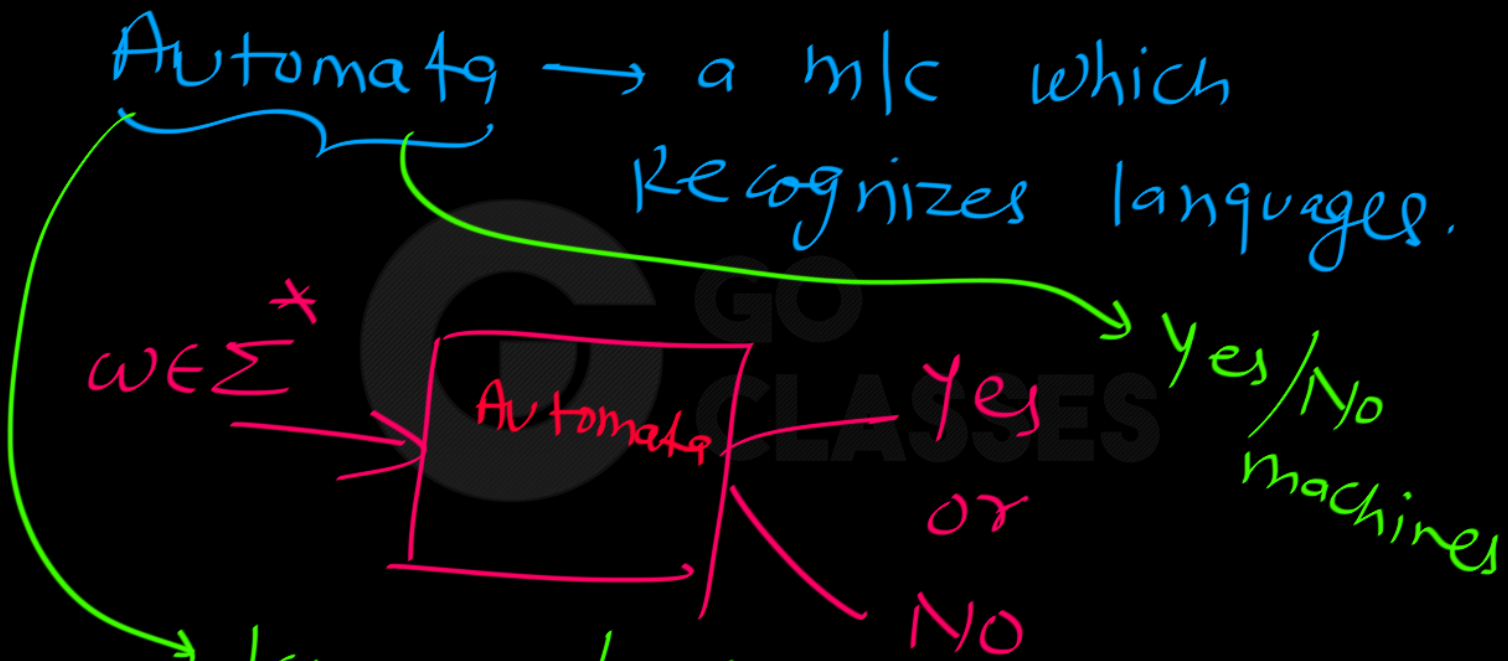
- 00
- 01
- 10
- 11


seq. ckt
k ffs → 2^k states
finite no. of states possible for the seq ckt
So
a seq. ckt is
Also a finite state machine.

Q :

What is the difference between finite automata and finite state machines?




Automata → a m/c which
Recognizes languages.
$w \in \Sigma^*$
Automata
Yes
or
No
Yes/No
machines
language / string acceptors

Mealy m/c or Moore m/c or Seq. Ckt :

Finite State Transducers → Compute functions

Transducer M which compute is Complement of a binary string :

$w =$ 10010

$f(w) =$ 01101

Q :

What is the difference between finite automata and finite state machines?

FSM is a broad term that includes DFA, NFA, Mealy and Moore machines, (finite-state) transducers etc; simply everything with a finite state space and without auxiliary memory.

Q : What is the difference between finite automata and finite state machines?

FSM is a broad term that includes DFA, NFA, Mealy and Moore machines, (finite-state) transducers etc; simply everything with a finite state space and without auxiliary memory.

Finite accepters play a central role in the study of formal languages, but in other areas, such as digital design, transducers are more important.

Finite State Machine (FSM) (synchronous sequential machine) (FINITE-STATE TRANSDUCERS)(FST)

Finite State Machine (FSM):

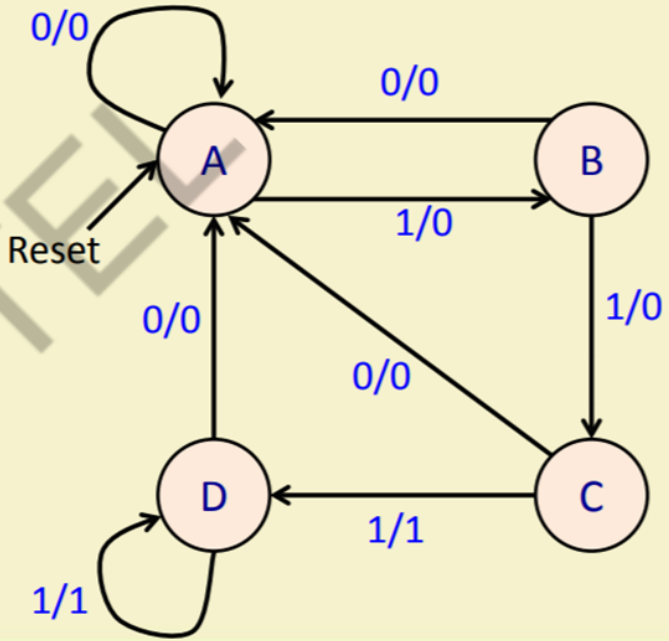
- A FSM can be represented either in the form of a state table or in the form of a state transition diagram.
- Example:

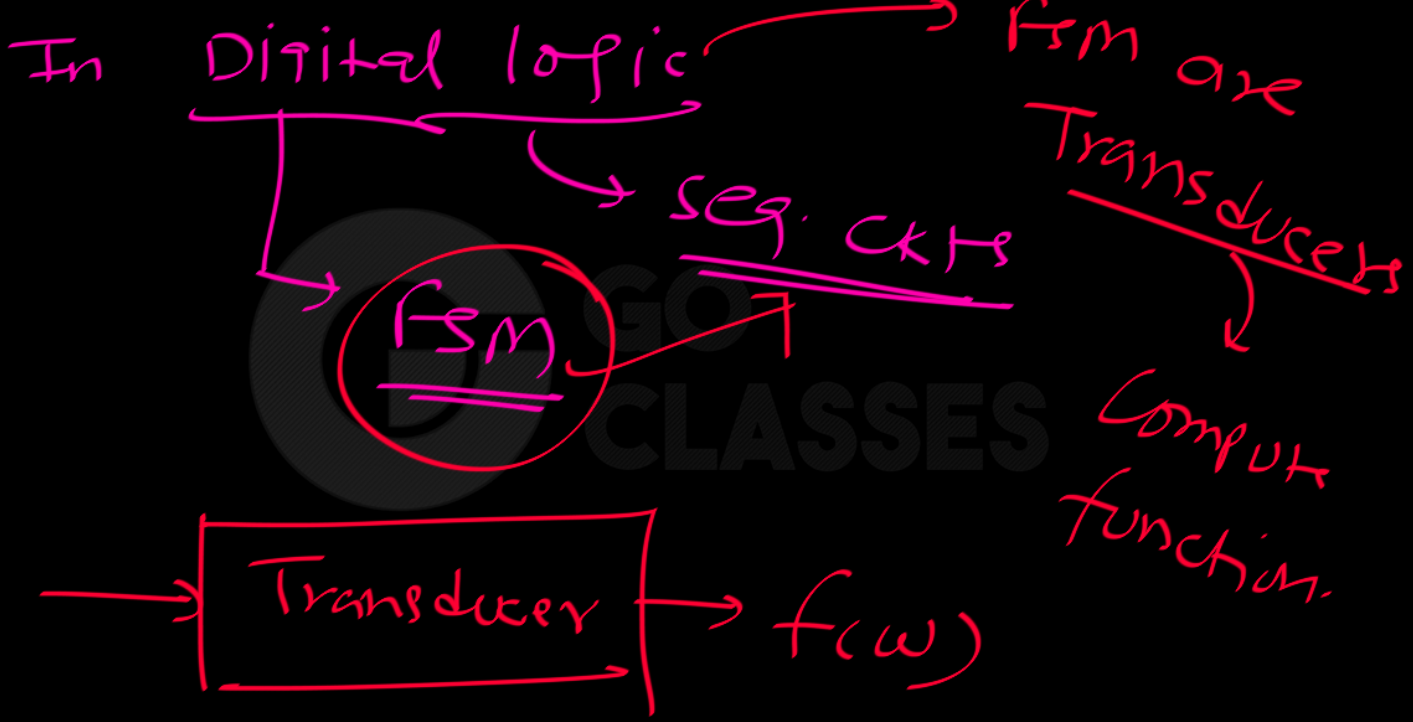
– A circuit to detect 3 or more 1's in a serial bit stream.
– The bits are applied serially in synchronism with a clock.
– The output will become 1 whenever it detects 3 or more consecutive 1's in the stream, else 0.

State Table

Reset	PS	Input	NS	Output
1	-	-	A	0
0	A	0	A	0
0	A	1	B	0
0	B	0	A	0
0	B	1	C	0
0	C	0	A	0
0	C	1	D	1
0	D	0	A	0
0	D	1	D	1

State Transition Diagram




In Digital logic
FSM are Transducers
seq. ckts
FSM
Comput function.
w
Transducer
f(w)

A GENERAL FRAMEWORK

Finite-state transducers (fst's) have many things in common with finite accepters. An fst has a finite set Q of internal states and operates in a discrete time frame with transitions from one state to another made in the interval between two instances t_n and t_{n+1}. An fst is associated with a read-once-only input file that contains a string from an **input alphabet** Σ, and an output mechanism that produces a string from an **output alphabet** Γ in response to a given input. It will be assumed that in each time step one input symbol is used, while output symbol is produced (we also say printed).

ToC

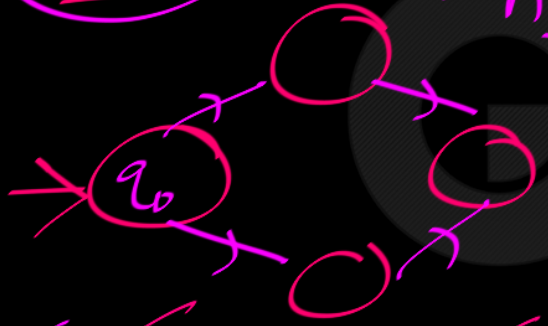
Yes / No machine

Automata

↓

language

Recognizers



$(Q, q_0, \delta, \Sigma, F)$ — To accept strings

(Transducer) FSM

FSM $(Q, q_0, \Sigma, \tau, \delta, \omega)$

Deterministic

input Alphabet

state Transition function

output function

output Alphabet

Automata

(DFA, NFA)

↳ Theoretical Concepts

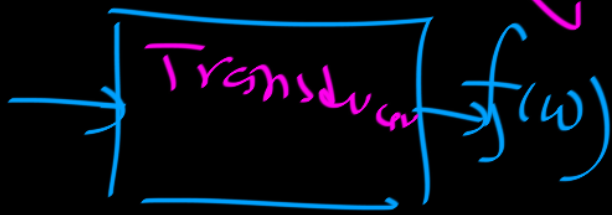
$(Q, q_0, \Sigma, \delta, F)$

Transducers

FSM { Mealy m/c, Moore m/c, Seq. Ckt }

Compute function

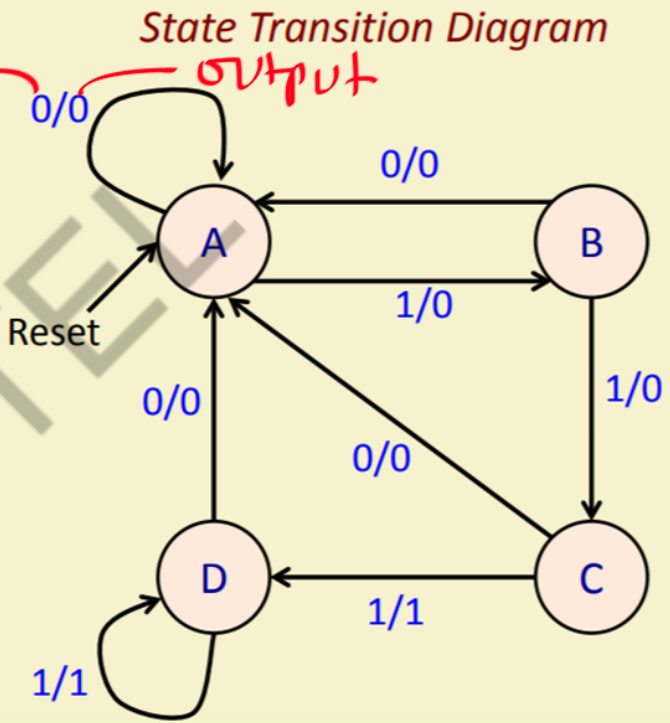
ω



State Table

Reset	PS	Input	NS	Output
1	-	-	A	0
0	A	0	A	0
0	A	1	B	0
0	B	0	A	0
0	B	1	C	0
0	C	0	A	0
0	C	1	D	1
0	D	0	A	0
0	D	1	D	1

State Transition Diagram



finite Automata

- DFA
- NFA

Since an fst just translates certain strings into other strings, we can look at the fst as an implementation of a function. If M is an fst, we will let F_M denote the function represented by M, so

$$F_M : D \rightarrow R,$$

where D is a subset of Σ^* and R is a subset of Γ^*. For most of the discussion we assume that $D = \Sigma^*$.

Interpreting an fst as a function implies that it is deterministic, that is, the output is uniquely determined by the input. Nondeterminism is an important issue in language theory, but plays no significant role in the study of finite-state transducers.

"Power" of a Machine

Power of a DFA, NFA, DPDA, NPDA/CFG:

Set of **languages** it can recognize/produce.

Power of a Moore Machine:

Set of **functions** it can perform.

Language	Function
Set of strings	Set of <*string*, *string*> (input/output) pairs

$FSM(Q, q_0, \Sigma, \gamma, \delta, \omega)$

- Deterministic
- finite state m/c (Transducer Definition)

two types of FSM

- Transducer
  - mealy m/c
  - moore m/c
  - ⋮

- We will study two machines created by G.H Mealy (1955) and by E.F Moore (1956).
- Initially both models were intended for sequential circuit design.

Mealy and Moore FSM Types

- A deterministic FSM can be mathematically defines as a 5-tuple

  $M = (\Sigma, \Gamma, S, s_0, \delta, \lambda)$

  where Σ is the set of input combinations, Γ is the set of output combinations, S is a finite set of states, $s_0 \in S$ is the initial state, δ is the state-transition function, and λ is the output function.

- Here, $\delta : S \times \Sigma \rightarrow S$
  - Present state (PS) and present input determines the next state (NS).
- For Mealy machine, $\lambda : S \times \Sigma \rightarrow \Gamma$ (output depends on state + inputs)
- For Moore machine, $\lambda : S \rightarrow \Gamma$ (output depends only on the state)

FSM $(Q, q_0, \Sigma, \tau, \delta, \omega)$ — ω: output function

Mealy m/c

output = f(present state, input)

$\omega : Q \times \Sigma \to \tau$

Moore m/c

output = f(present state)

$\omega : Q \to \tau$

FSM $(Q, q_0, \Sigma, \gamma, \delta, \omega)$

$\delta : Q \times \Sigma \rightarrow Q$

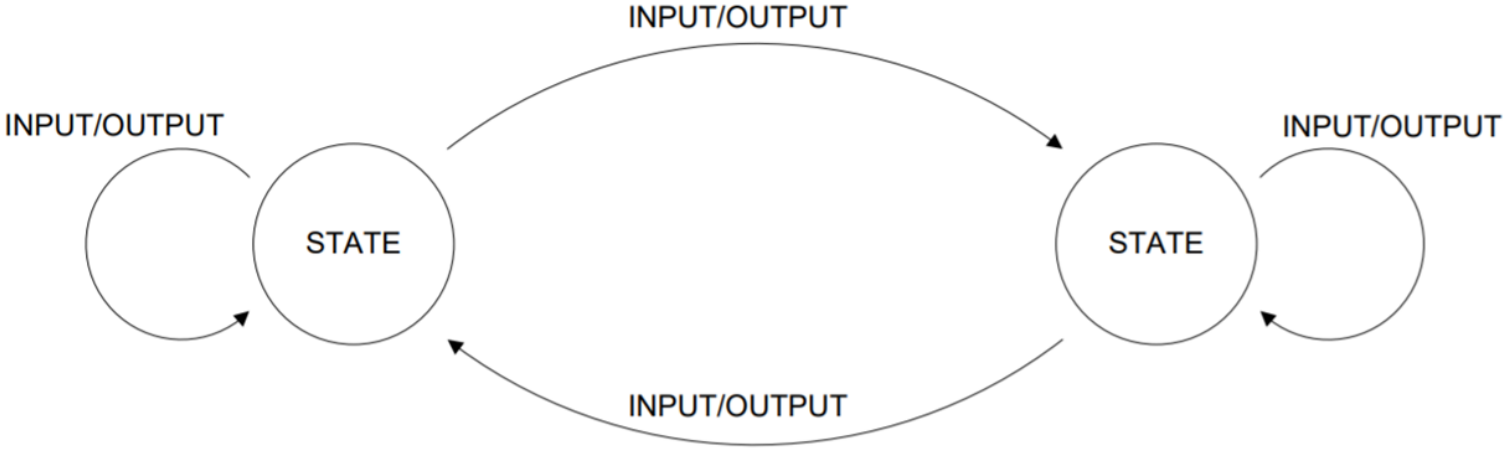
↓

State Transition function

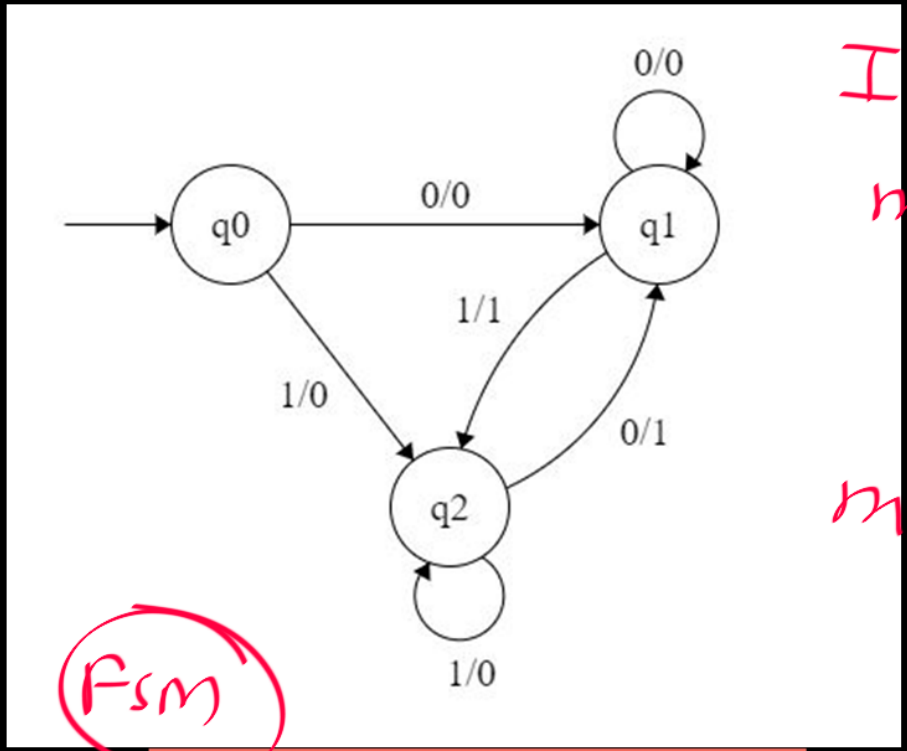
Same for Mealy, Moore m/c

Mealy State Machines:

- Outputs determined by the current state **and** the current inputs.
  - Outputs are *conditional* (directly dependent on input signals)



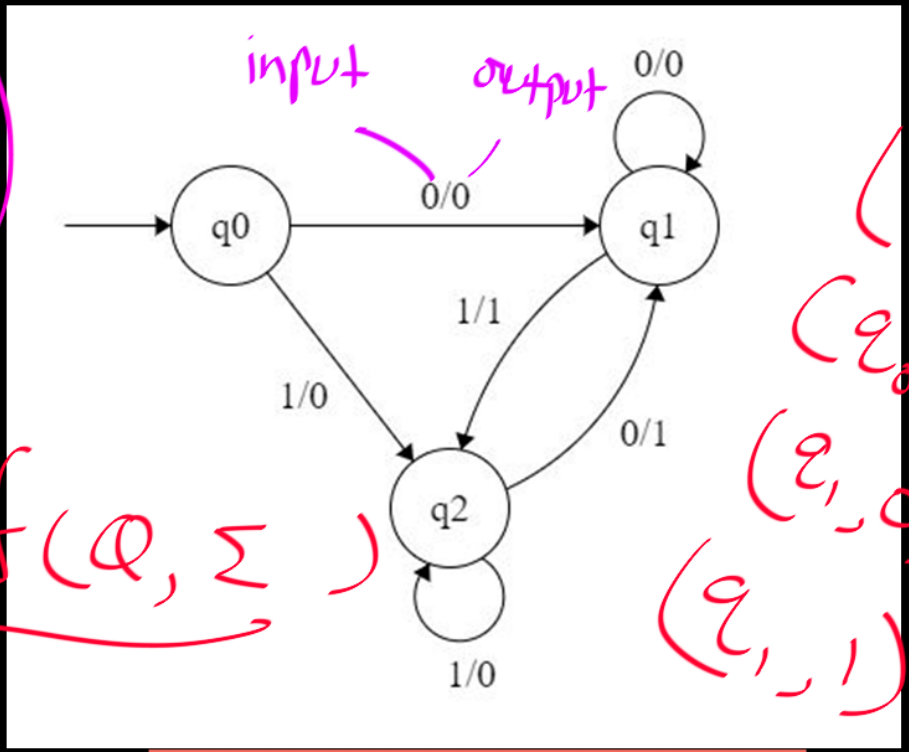
GENERIC MEALY STATE MACHINE



Is it mealy m/c OR moore m/c?

FSM

mealy m/c



output = f(Q, Σ)

state input

(q0, 0) → 0

(q0, 1) → 0

(q1, 0) → 0

(q1, 1) → 1

mealy m/c $(Q, q_0, \Sigma, \gamma, \delta, \omega)$

$Q = \{q_0, q_1, q_2\}$

q_0

$\Sigma = \{0, 1\}$

$\gamma = \{0, 1\}$

$\delta : (Q \times \Sigma) \to Q$

State Transition function

$(q_0, 0) \to q_1$

$(q_1, 1) \to q_2$

- - - - -

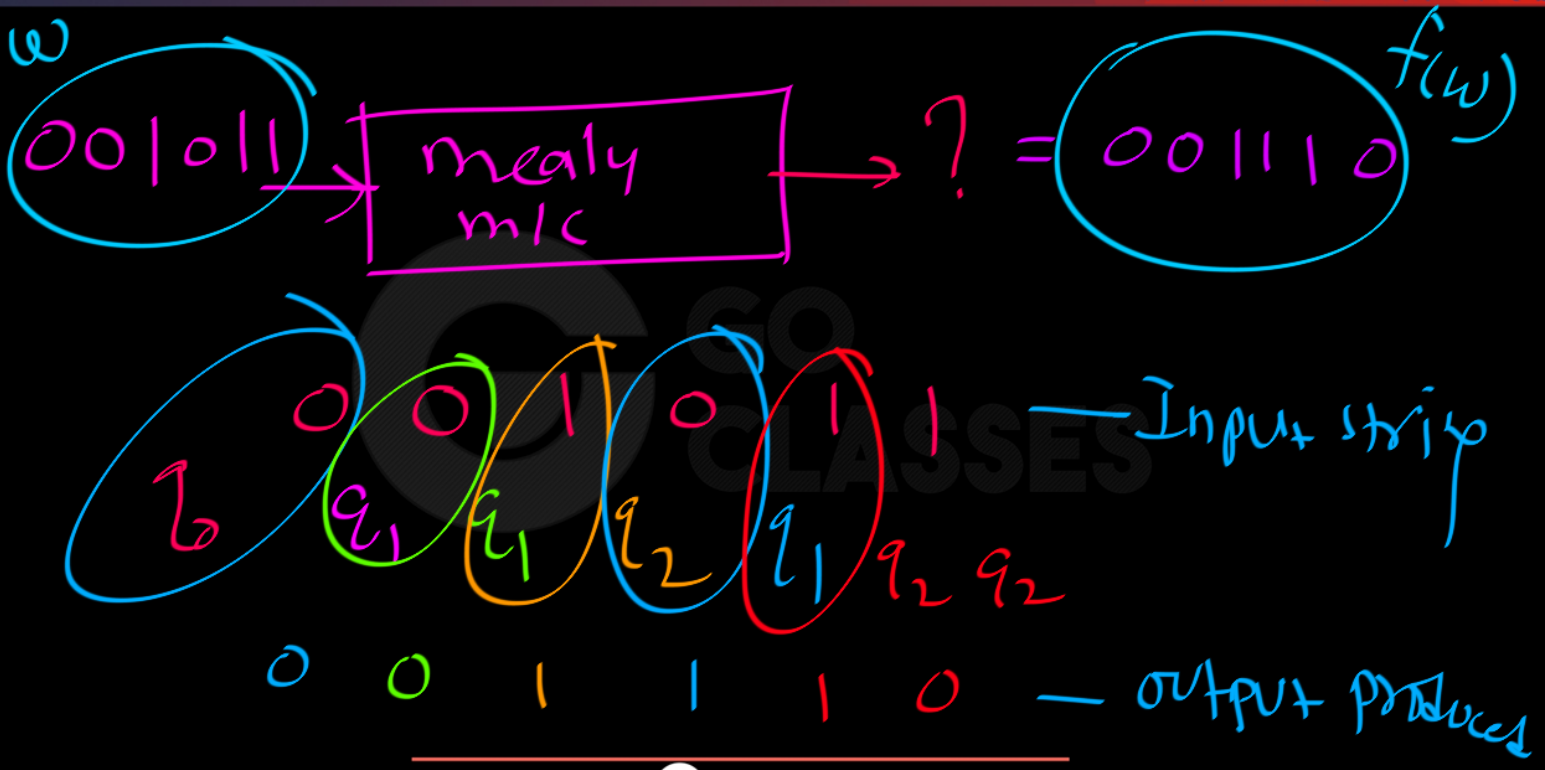
ω → output function

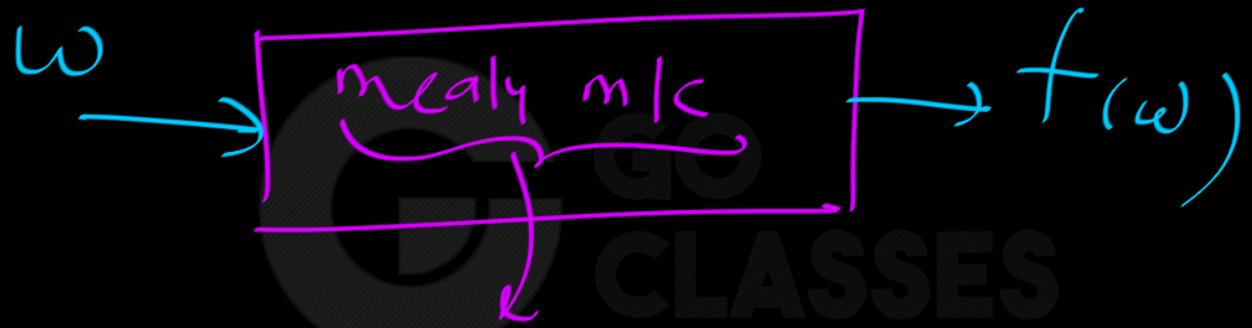
In mealy m/c $\omega = f(Q, \Sigma)$

$\omega : Q \times \Sigma \rightarrow \Gamma$

$(q_0, 0) \rightarrow 0$ | $(q_2, 1) \rightarrow 0$

$(q_1, 1) \rightarrow 1$ | - - - -

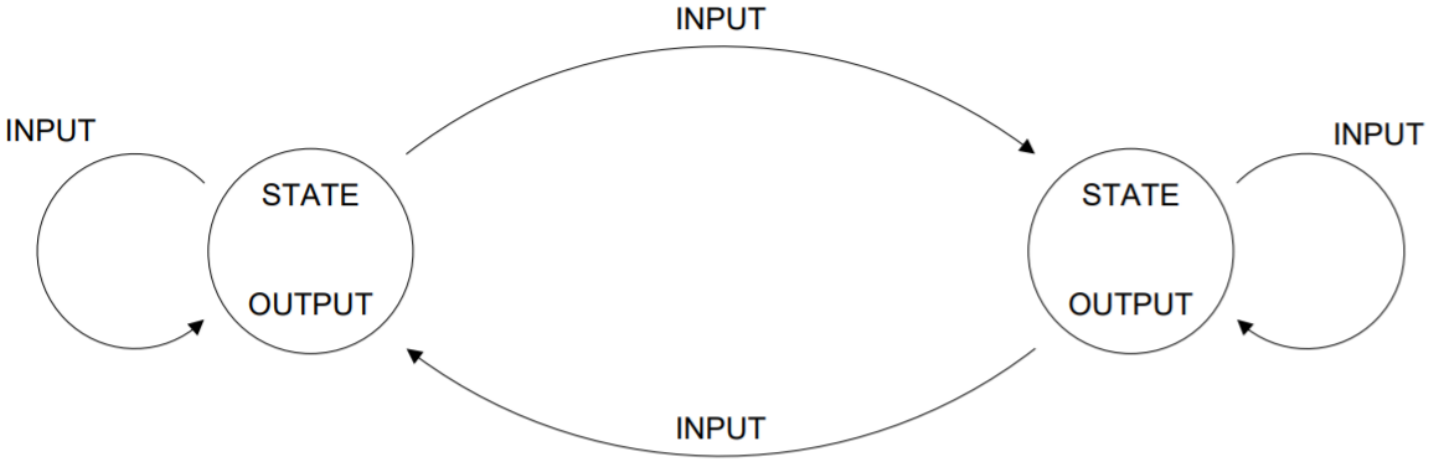

w
0 0 1 0 1 1
mealy m/c
?
0 0 1 1 1 0
f(w)
0 0 1 0 1 1
q0 q1 q1 q2 q1 q2 q2
Input strip
0 0 1 1 1 0
output produced



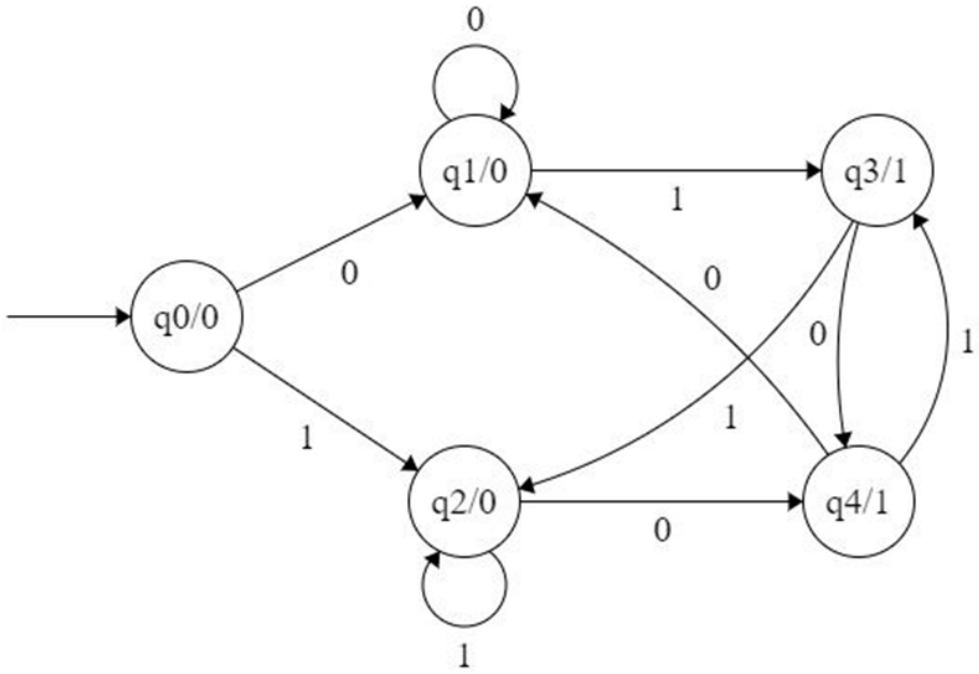
a Transducer → Computes function

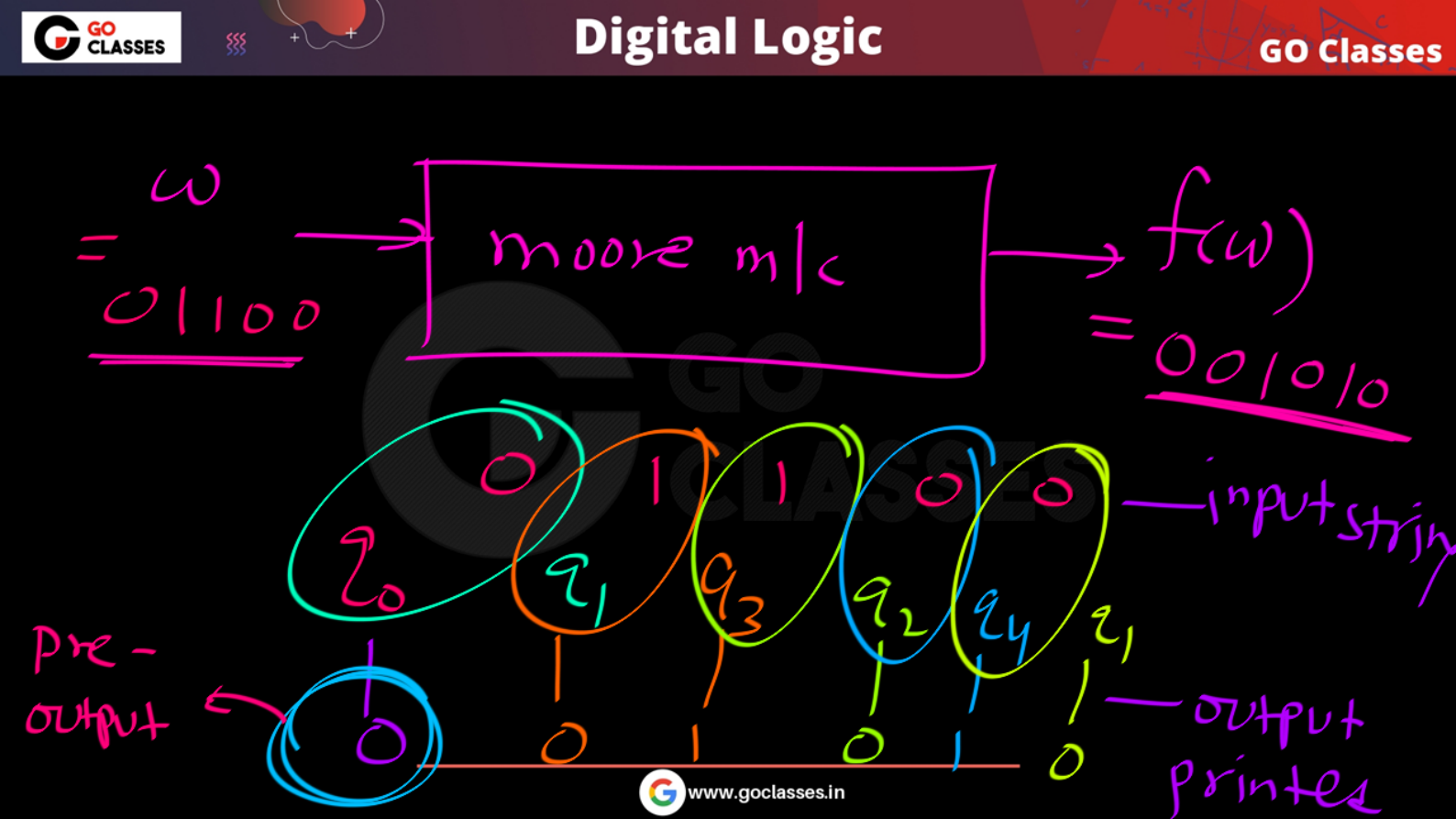
Moore State Machines:

- Outputs determined **solely** by the current state
  - Outputs are *unconditional* (not directly dependent on input signals)



GENERIC MOORE STATE MACHINE


0
q1/0
q3/1
1
0
0
q0/0
0
1
1
q2/0
q4/1
0
1


w
= 01100
moore m/c
f(w)
= 00010
0 1 1 0 0
input string
q0 q1 q3 q2 q4 q1
Pre-output
0 0 1 0 1 0
output printed

Note: without reading anything from input string, in moore m/c, should we produce output?

Note: without reading anything from input string, in moore m/c, should we produce output?

Ans: Depends on the Author.

Some Authors

↓

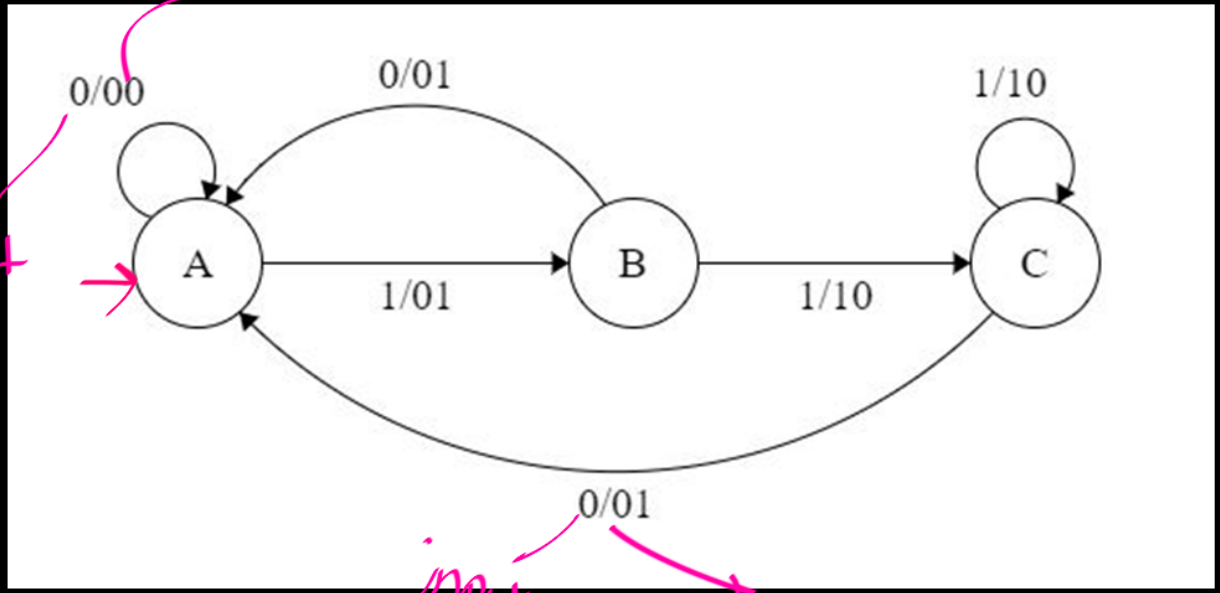
They print this pre-output

Some Authors

↓

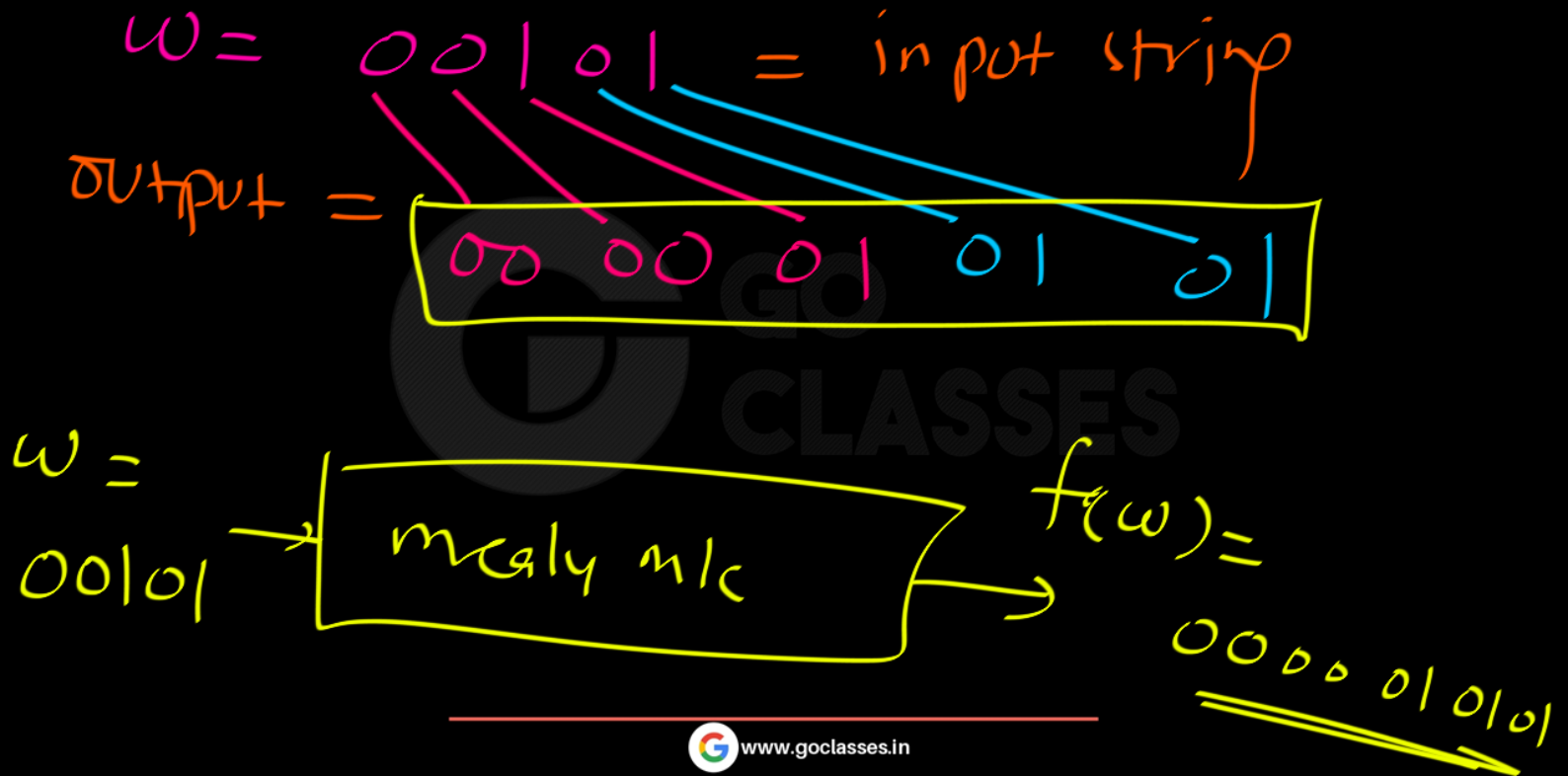
They don't Print/Produce Pre-output.

output



input

input → output


w = 00101 = input string
output =
00 00 01 01 01
w =
00101
mealy m/c
f(w) =
00001 01 01 01

Finite State Machine (FSM)(FST):

There are several types of fst's that have been extensively studied. The main difference between them is on how the output is produced.

1. MEALY MACHINES
2. MOORE MACHINES

1. MEALY MACHINES :

A Mealy Machine is a state machine where the output is a function of the present state and the current input.

In a Mealy machine, the output produced by each transition depends on the internal state prior to the transition and the input symbol used in the transition, so we can think of the output produced during the transition.

A Mealy machine is defined by the sextuple

$$M = (Q, \Sigma, \Gamma, \delta, \theta, q_0),$$

where

Q is a finite set of internal states,
Σ is the input alphabet,
Γ is the output alphabet,
$\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
$\theta : Q \times \Sigma \rightarrow \Gamma$ is the output function,
$q_0 \in Q$ is the initial state of M.

The machine starts in state q_0 at which time all input is available for processing. If at time t_n the Mealy machine is in state q_i, the current input symbol is a, and $\delta(q_i, a) = q_j$, $\theta(q_i, a) = b$, the machine will enter state q_j and produce output b. It is assumed the entire process is terminated when the end of the input is reached. Note that there are no final states associated with a transducer.

Transition graphs are as useful here as they are for finite accepters. In fact, the only difference is that now the transition edges are labeled with a/b, where a is the current input symbol and b is the output produced by the transition.

Mealy machine

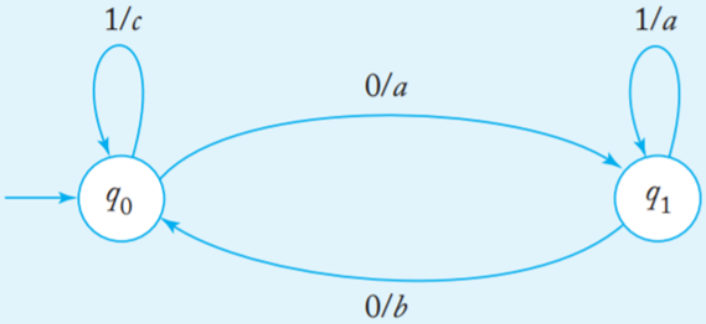
A Mealy machine consists of the following

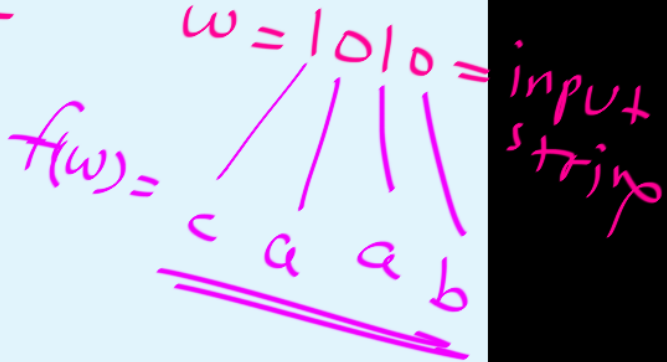
1. A finite set of states q_0, q_1, q_2, ... where q_0 is the initial state.
2. An alphabet of letters Σ = {a,b,c,...} from which the input strings are formed.
3. An alphabet Γ={x,y,z,...} of output characters from which output strings are generated
4. A pictorial representation with states and directed edges labeled by an input letter along with an output character. The directed edges also show how to go from one state to another corresponding to every possible input letter

The fst with $Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{a, b, c\}$, initial state q_0, and

$$\begin{aligned}
\delta(q_0, 0) &= q_1, \quad \delta(q_0, 1) = q_0, \\
\delta(q_1, 0) &= q_0, \quad \delta(q_1, 1) = q_1, \\
\theta(q_0, 0) &= a, \quad \theta(q_0, 1) = c, \\
\theta(q_1, 0) &= b, \quad \theta(q_1, 1) = a
\end{aligned}$$

is represented by the graph in Figure A.1. This Mealy machine prints out *caab* when given the input string 1010.





Note

- It is to be noted that since, similar to Moore machine, in Mealy machine no state is designated to be a final state, so there is no question of accepting any language by Mealy machine.

Example

- Consider the Mealy machine shown aside, having the states q0, q1, q2, q3 , where q0 is the start state and

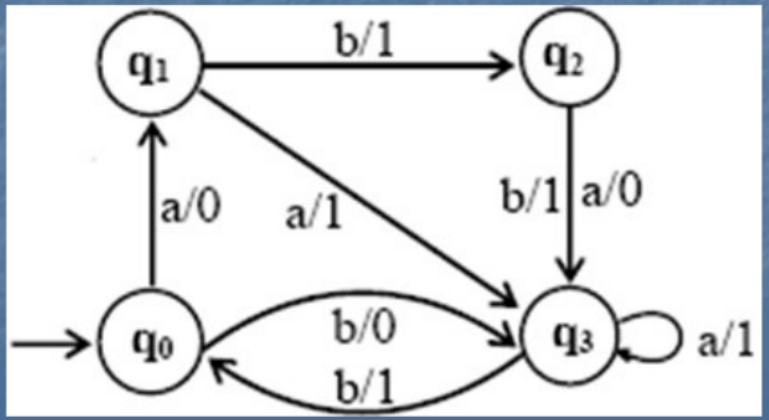
  $\Sigma = \{a,b\}$,

  $\Gamma = \{0,1\}$

output function

$(q_1, b) \rightarrow 1$

$(q_3, a) \rightarrow 1$



Example Cont.....

- Running the string abbabbba over the above machine, the corresponding output string will be 01111010, which can be determined by the following table as well

Input		a	b	b	a	b	b	b	a
States	q_0	q_1	q_2	q_3	q_3	q_0	q_3	q_0	q_1
output		0	1	1	1	1	0	1	0

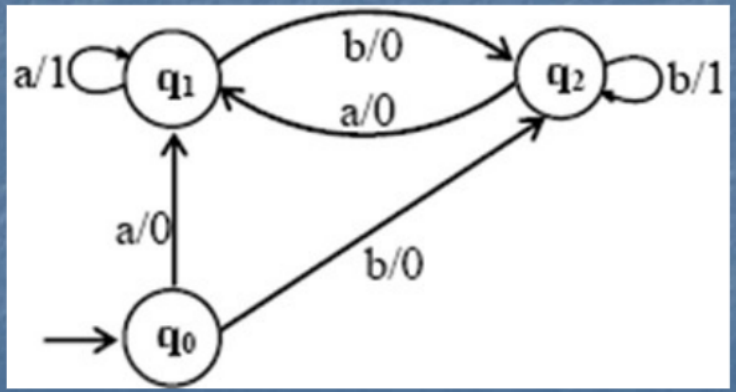
Example

What Does this mealy m/c Do ?

- Consider the Mealy machine shown aside, having the states q0, q1, q2, where q0 is the start state and

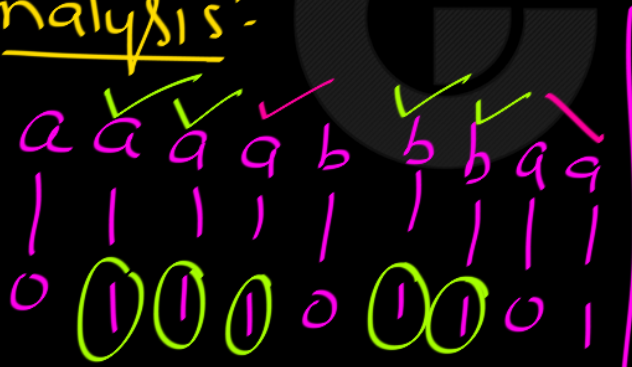
  $\Sigma = \{a,b\}$,

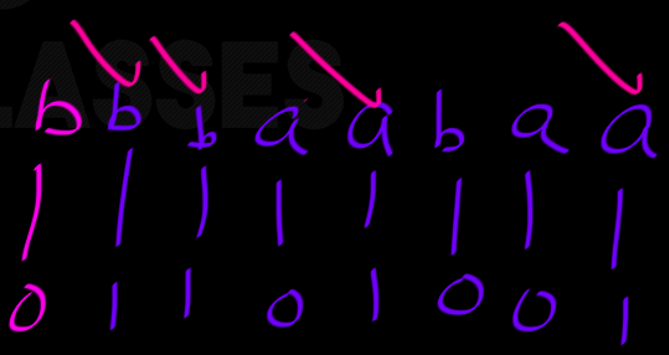
  $\Gamma = \{0,1\}$




input = a b a b b b a a
output = 0 0 0 0 1 1 0 1

Analysis:-


a a a a b b b a a
0 1 0 1 0 0 1 0 1

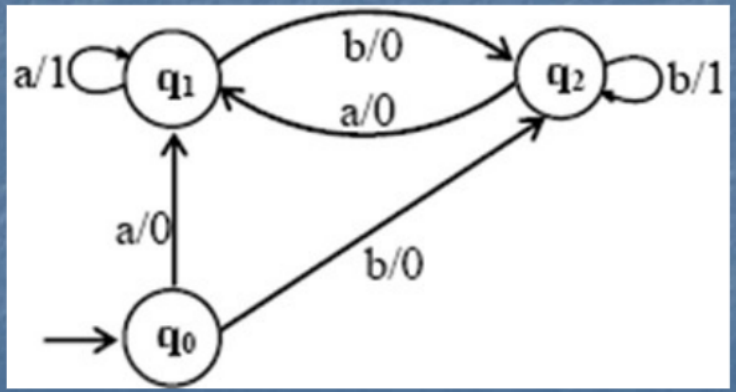

b b b a a b a a
0 1 1 0 1 0 0 1

Example

- Consider the Mealy machine shown aside, having the states q0, q1, q2, where q0 is the start state and

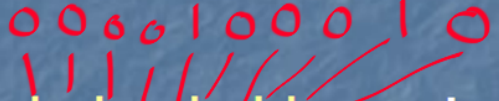
  $\Sigma = \{a,b\}$,

  $\Gamma = \{0,1\}$



Example Cont.....

- It is observed that in the above Mealy machine, if in the output string the nth character is 1, it shows that the nth letter in the input string is the second in the pair of double letter.



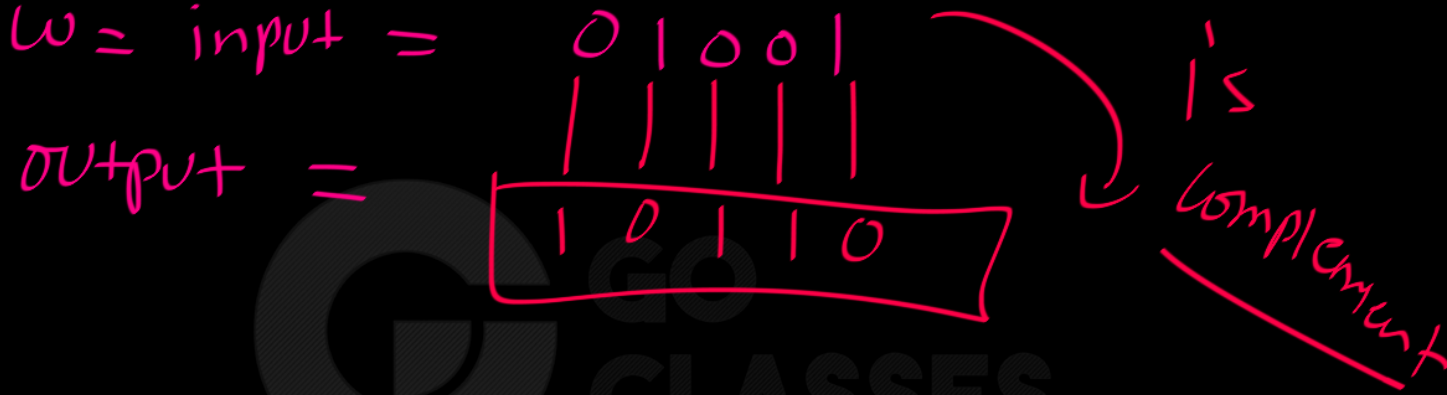
- For babaababba as input string the machine will print 0000100010.

Example

- Consider the Mealy machine shown aside, having the states q0, where q0 is the start state and

  $\Sigma = \{0, 1\}$,

  $\Gamma = \{0,1\}$

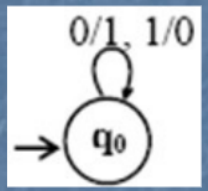

w = input = 01001
output = 10110
1's complement

Example

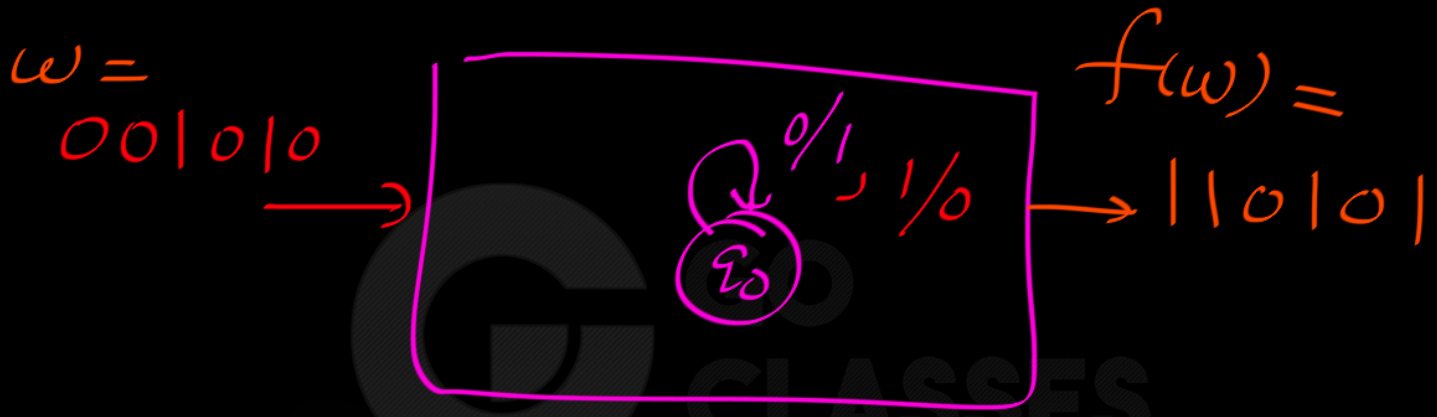
- Consider the Mealy machine shown aside, having the states q0, where q0 is the start state and

  $\Sigma = \{0, 1\}$,

  $\Gamma = \{0,1\}$



- If 0011010 is run on this machine then the corresponding output string will be 1100101.
- This machine is called **Complementing machine.**



$f(w)$ = 1's comp of w.

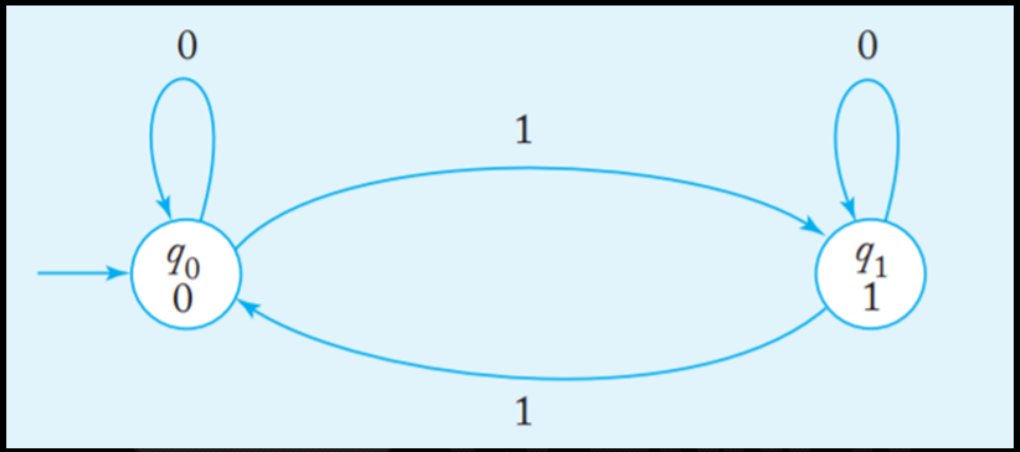
2. MOORE MACHINES :

Moore machines differ from Mealy machines in the way output is produced. In a Moore machine every state is associated with an element of the output alphabet. Whenever the state is entered, this output symbol is printed. The output is produced only when a transition occurs; thus, the symbol associated with the initial state is not printed at the start, but may be produced if this state is entered at a later stage.

2. MOORE MACHINES :

NOTE that :

The output is produced only when a transition occurs; thus, the symbol associated with the initial state is not printed at the start, but may be produced if this state is entered at a later stage.

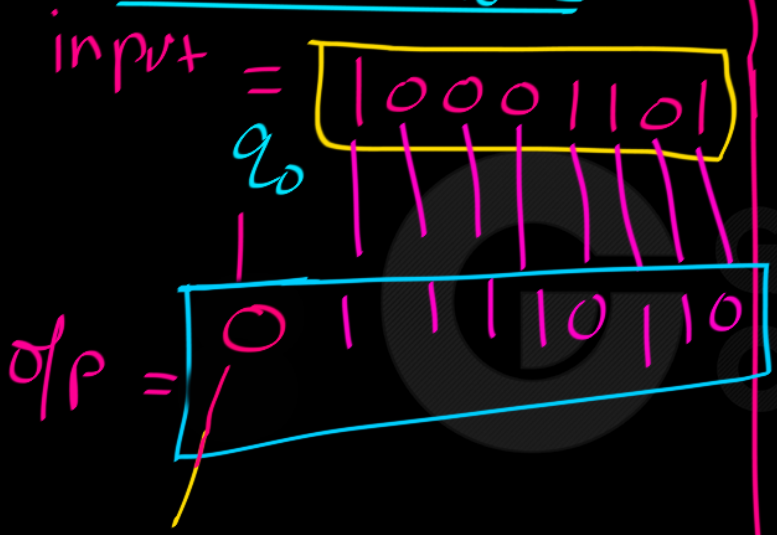


input = 10001101

output = ?

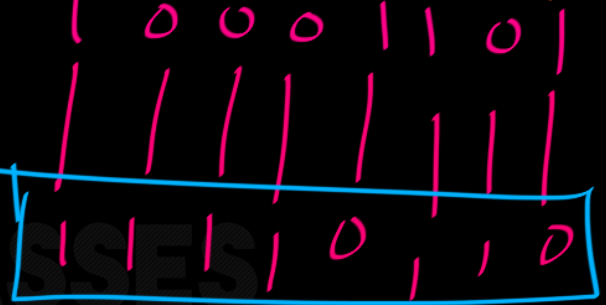
→ Print pre-output

Some Authors



Pre-output of Moore m/c

Some Authors → Don't Print pre-output



A Moore machine is defined by the sextuple

$$M = (Q, \Sigma, \Gamma, \delta, \theta, q_0),$$

where

Q is a finite set of internal states,
Σ is the input alphabet,
Γ is the output alphabet,
$\delta : Q \times \Sigma \to Q$ is the transition function,
$\theta : Q \to \Gamma$ is the output function,
$q_0 \in Q$ is the initial state.

The machine starts in state q_0, at which time all input is available for processing. If at time t_n the Moore machine is in state q_i, the current input symbol is a, and $\delta(q_i, a) = q_j$, $\theta(q_j) = b$, the machine will enter state q_j and produce output b.

In the transition graph of a Moore machine, each vertex now has two labels: the state name and the output symbols associated with the state.

Moore machine Definition

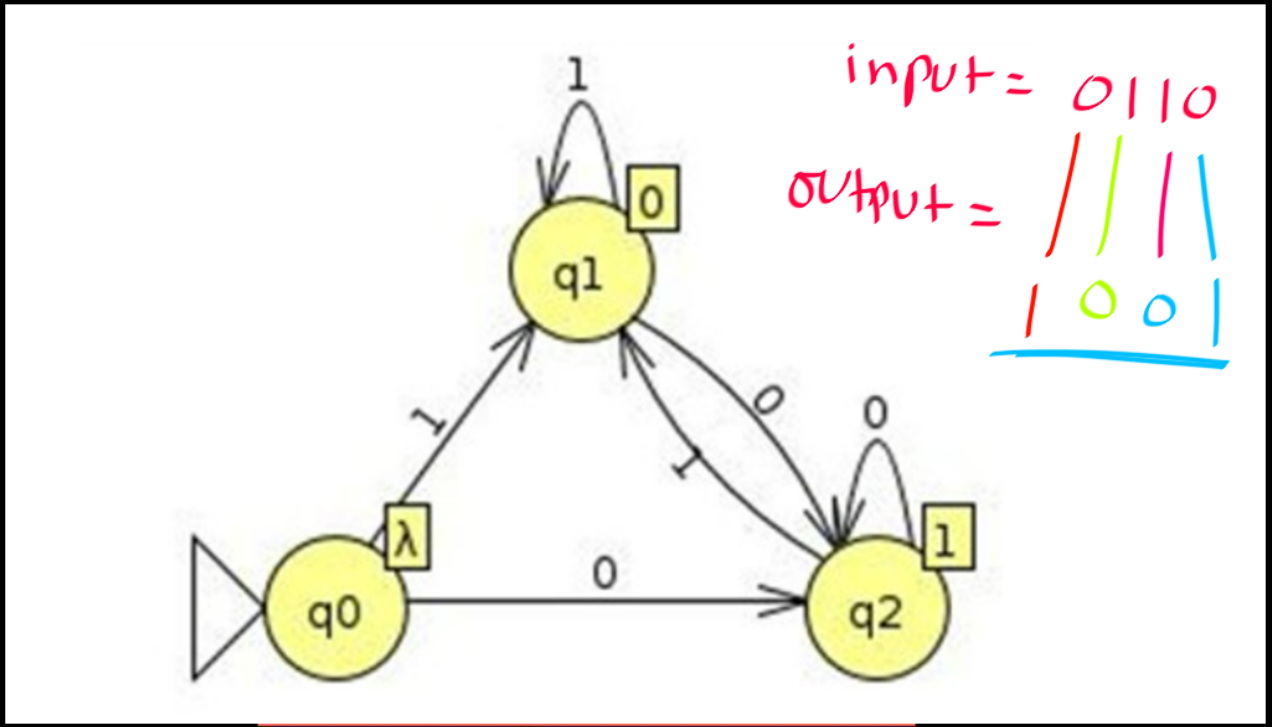
A Moore machine consists of the following

1. A finite set of states q_0, q_1, q_2, … where q_0 is the initial state.
2. An alphabet of letters Σ = {a,b,c,…} from which the input strings are formed.
3. An alphabet Γ={x,y,z,…} of output characters from which output strings are generated.

Moore machine continued ...

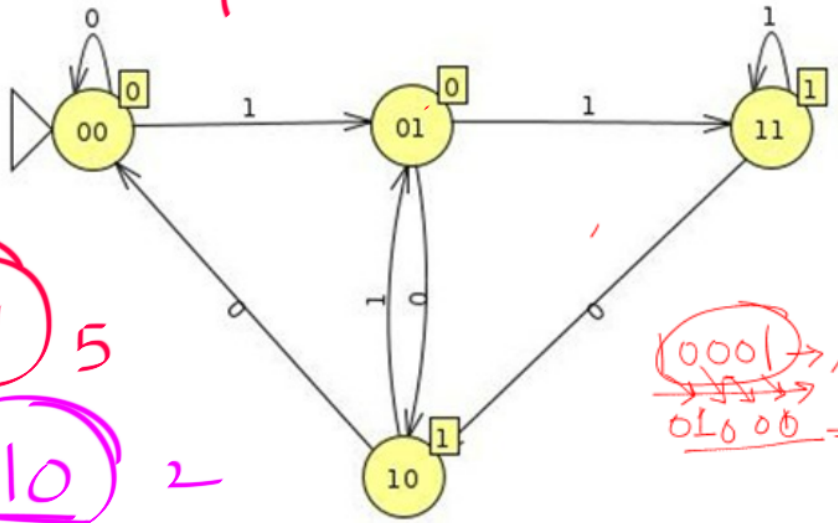
4. A transition table that shows for each state and each input letter what state is entered the next.

5. No final state, so no question of acceptance of input strings.


1
0
q1
1
0
0
1
λ
1
q0
0
q2
input = 0110
output = 1001

Example: Divide an Input in to half

input = 100 → output = 010



input = 101 5

output = 010 2

2/2 = 1

10 = 2

01 = 1

0001 → / 2

01000 =

Constructing the incrementing machine

- In the previous example of complementing machine, it has been observed that the input string and the corresponding output string are 1's complement of each other.
- There is a question whether the Mealy machine can be constructed, so that the output string is increased, in magnitude, by 1 than the corresponding input string?

Constructing the incrementing machine

- In the previous example of complementing machine, it has been observed that the input string and the corresponding output string are 1's complement of each other.
- There is a question whether the Mealy machine can be constructed, so that the output string is increased, in magnitude, by 1 than the corresponding input string?
- The answer is yes.

Input :⟶ binary string ω

output = $|\omega|+1$

Eg :

input	output
100	101
1010	1011
101	110
111	1000

Observation:

LSB

1 0 0 0
+ 1
1 0 0 1

1 0 1 0
+ 1
1 0 1 1

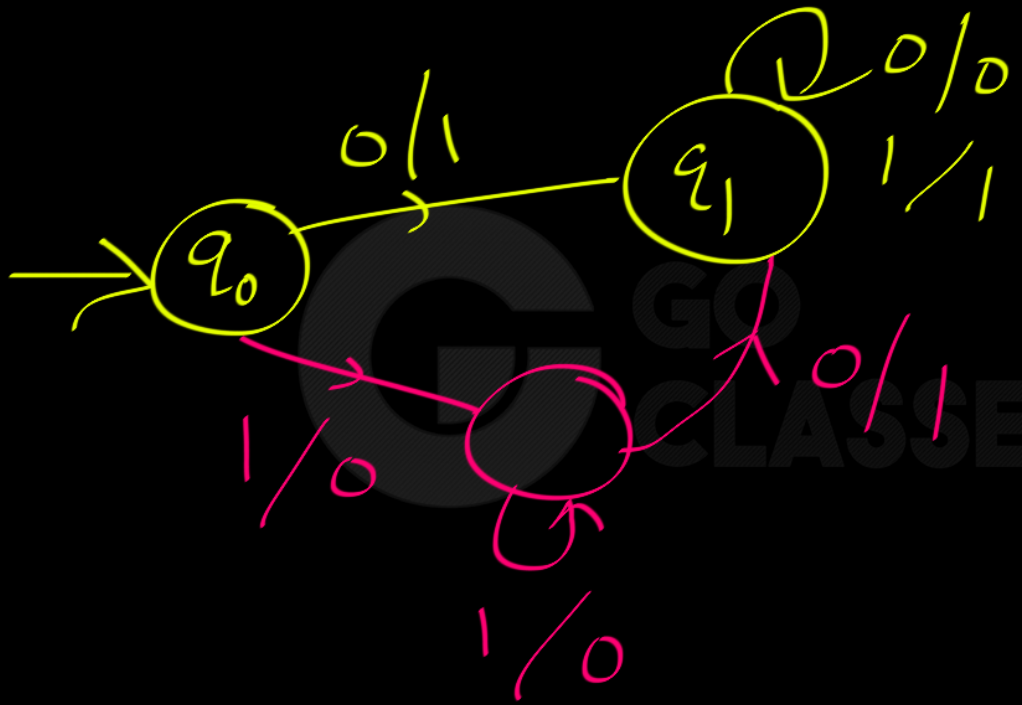
1 0 1 1
+ 1
1 1 0 0

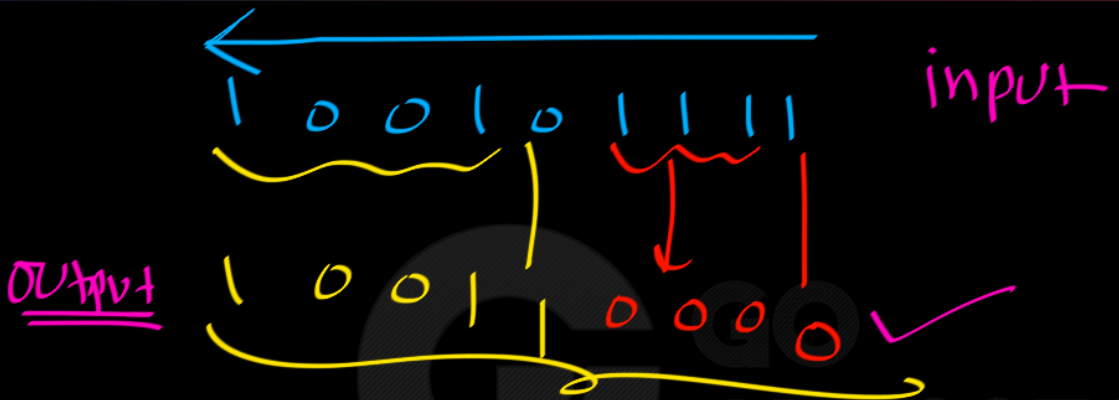
1 0 1
+ 1
1 1 0

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

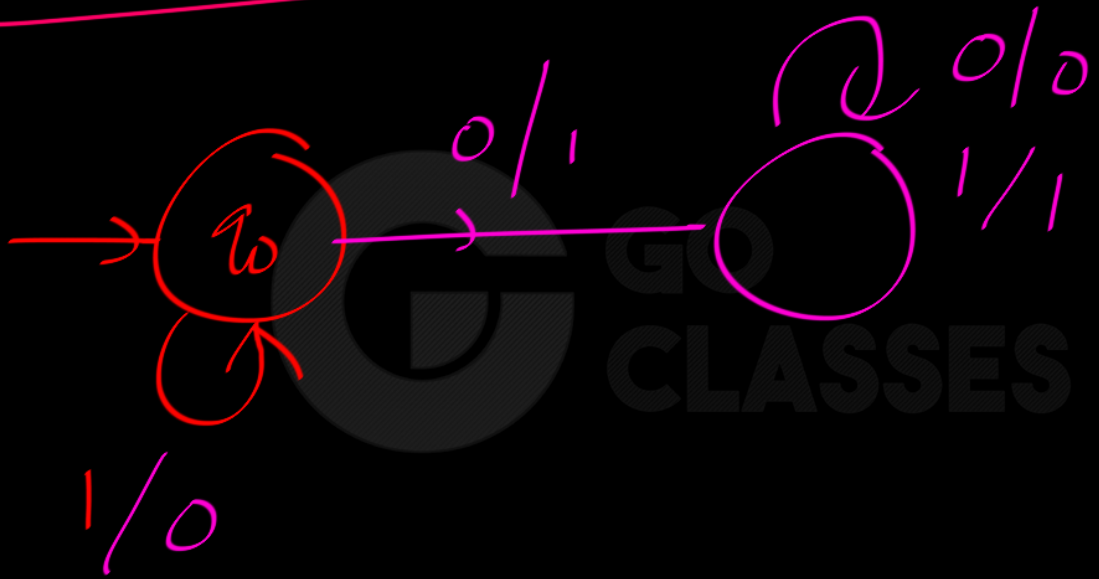
Assume mealy m/c reads input from right to left.

input 1000


0/1
0/0
1/1
q_0
q_1
1/0
0/1
1/0


input
output
1 0 0 1 0 1 1 1 1
1 0 0 1 0 0 0 0

two states are enough :

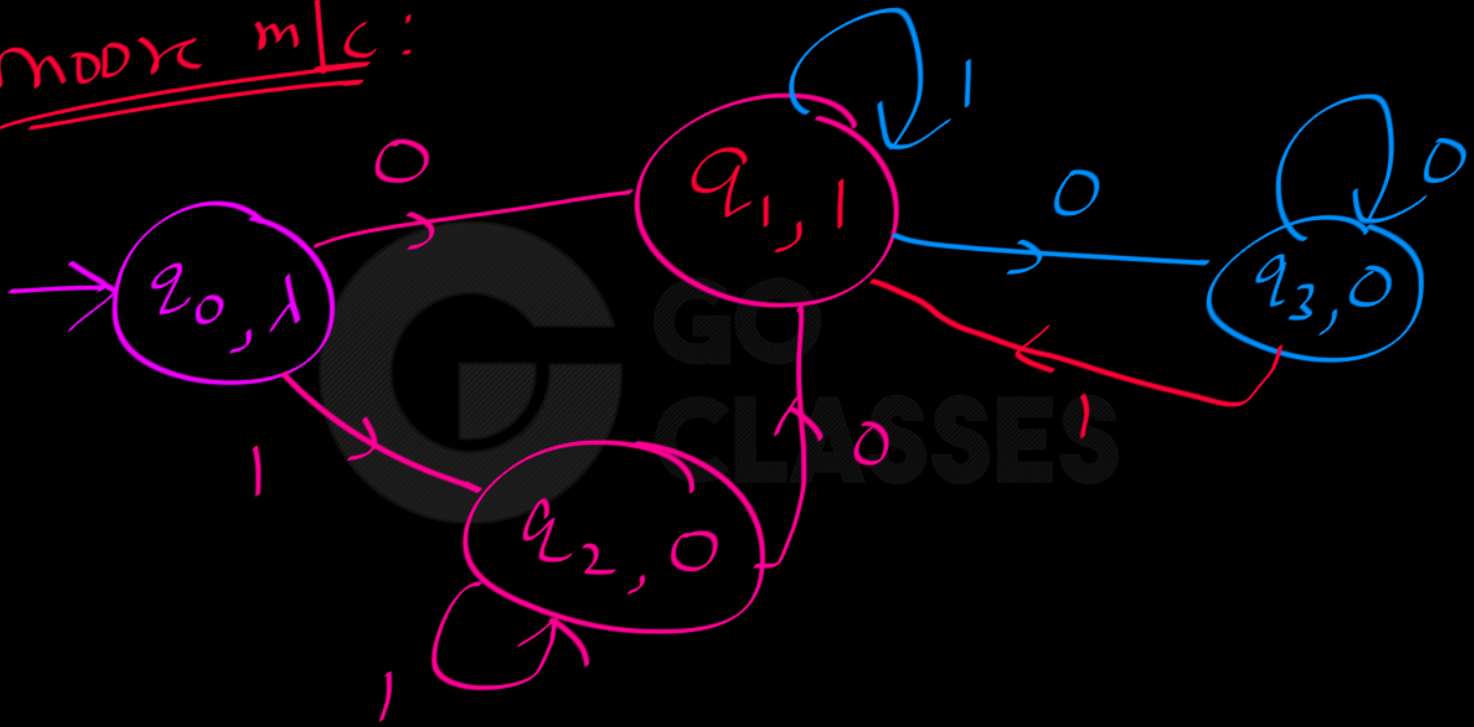


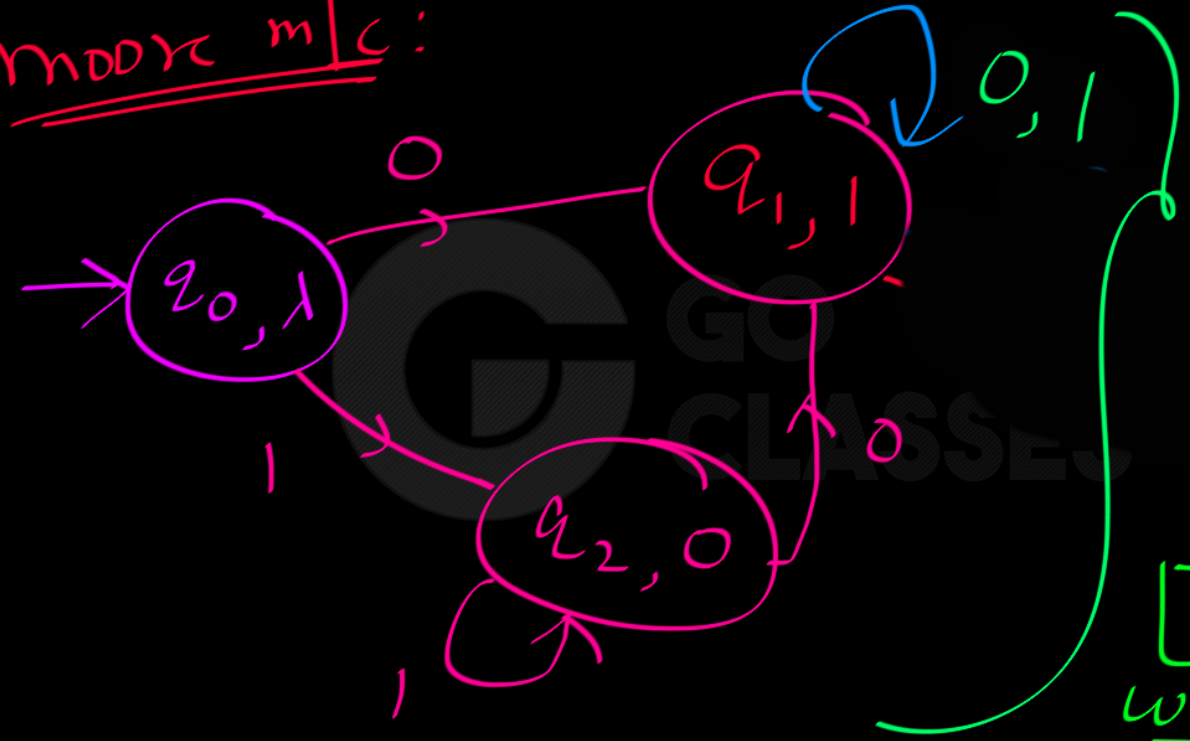
Moore m/c to increment a binary number by 1 :

Assume moore m/c will read input from right to left.

1 0 1 0 1 1 1 1 0

Moore m/c:




Moore m/c:
$q_0, 1$
$q_1, 1$
$q_2, 0$
0
1
0
1
0, 1
Mistake
1 0 0 1 0 1
1 1 1 1 1 0
Wrong.

- This machine is called the incrementing machine. Following is how to construct the incrementing machine. Before the incrementing machine is constructed, consider how 1 is added to a binary number.
- Since, if two numbers are added, the addition is performed from right to left, so while increasing the binary number by 1, the string (binary number) must be read by the corresponding Mealy machine from right to left, and hence the output string (binary number) will also be generated from right to left.
- Consider the following additions

- a) 100101110 + 1 = 100101111
- b) 1001100111 + 1 = 1001101000

It may be observed from the above that

- a) If the right most bit of binary number, to be incremented, is 0, the output binary number can be obtained by converting the right most bit to 1 and remaining bits unchanged.
- b) If the right most bit of binary number is 1 then the output can be obtained, converting that 1 along with all its concatenated 1's to 0's, then converting the next 0 to 1 and remaining bits unchanged.

Observation

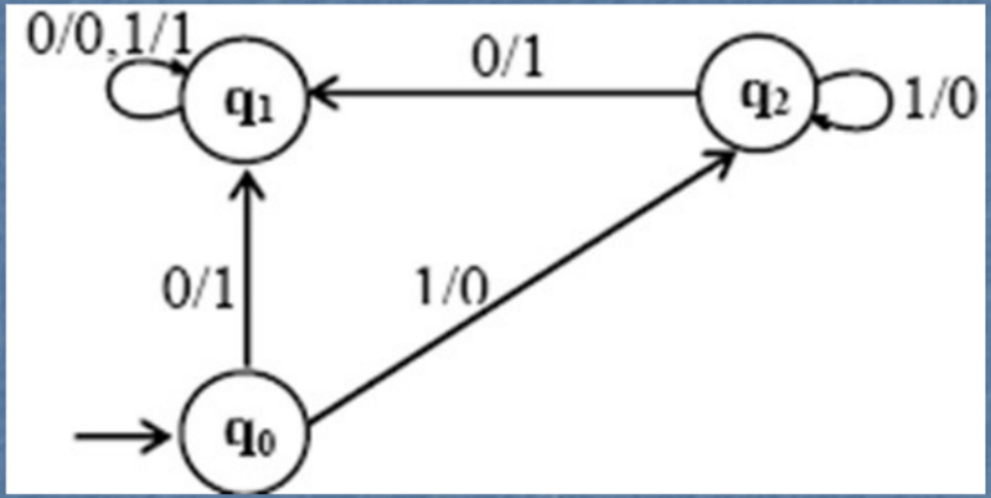
- The observations (a) and (b) help to construct the following Incrementing (Mealy) machine.

- The Mealy machine, having the states q0, q1, q2 where q0 is the start state and

  $\Sigma = \{0, 1\}$,

  $\Gamma = \{0,1\}$

Incrementing (Mealy) machine



Observation

- It may be observed that, in the incrementing machine, if 0 is read at initial state **q0**, that 0 is converted to 1 and a no change state is entered where all 0's and all 1's remain unchanged.
- If 1 is read at initial state, that 1 is converted to 0 and the state q2 is entered, where all 1's are converted to 0's and at that state if 0 is read that 0 is converted to 1 and the machine goes to no change state.
- If the strings 100101110 and 1001100111 are run over this machine, the corresponding output strings will be 100101111 and 1001101000 respectively.

Note

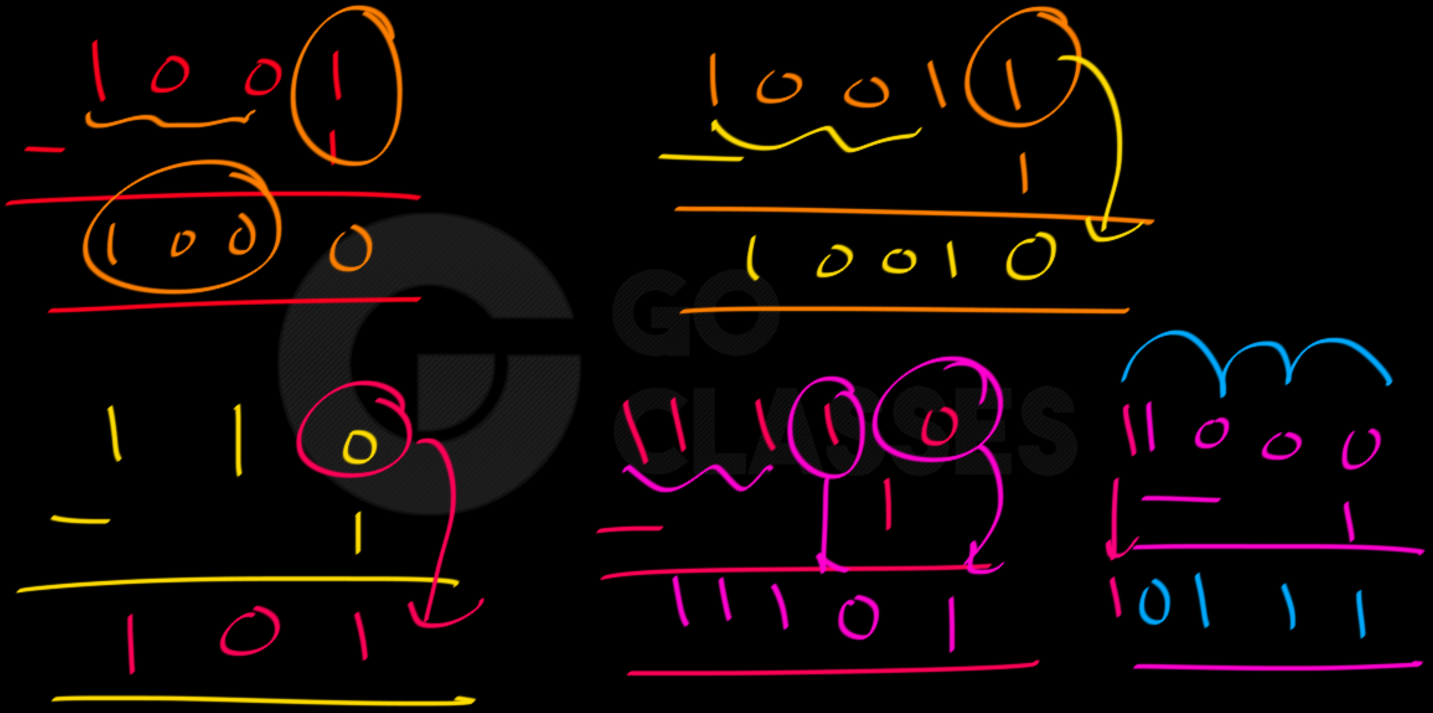
- It is to be noted that if the string 111111 is run over the incrementing machine, the machine will print out 000000, which is not increased in magnitude by 1. Such a situation is called an overflow situation, as the length of output string will be same as that of input string.
- It may also be noted that there exists another incrementing machine with two states.

Applications of Incrementing and Complementing machines

- 1's complementing and incrementing machines which are basically Mealy machines are very much helpful in computing.
- The incrementing machine helps in building a machine that can perform the addition of binary numbers.
- Using the complementing machine along with incrementing machine, one can build a machine that can perform the subtraction of binary numbers.

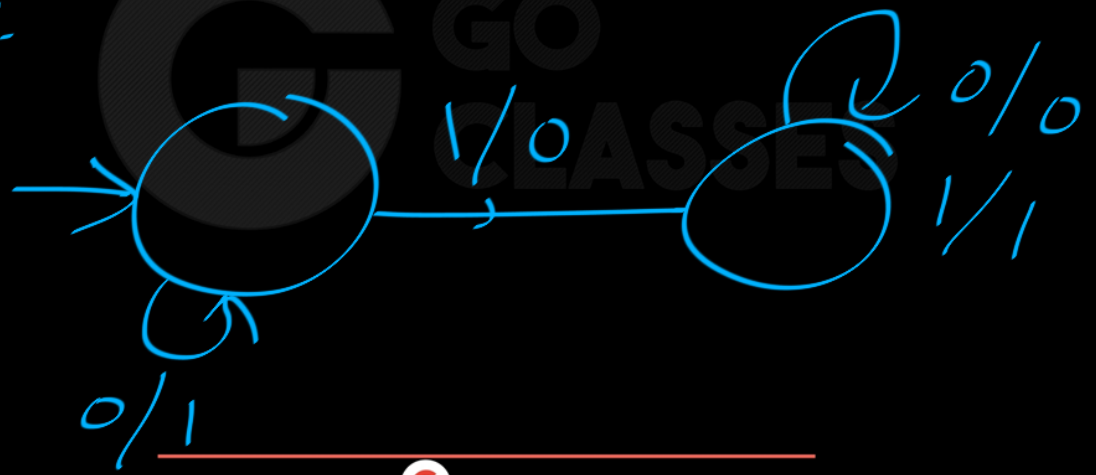
Example :

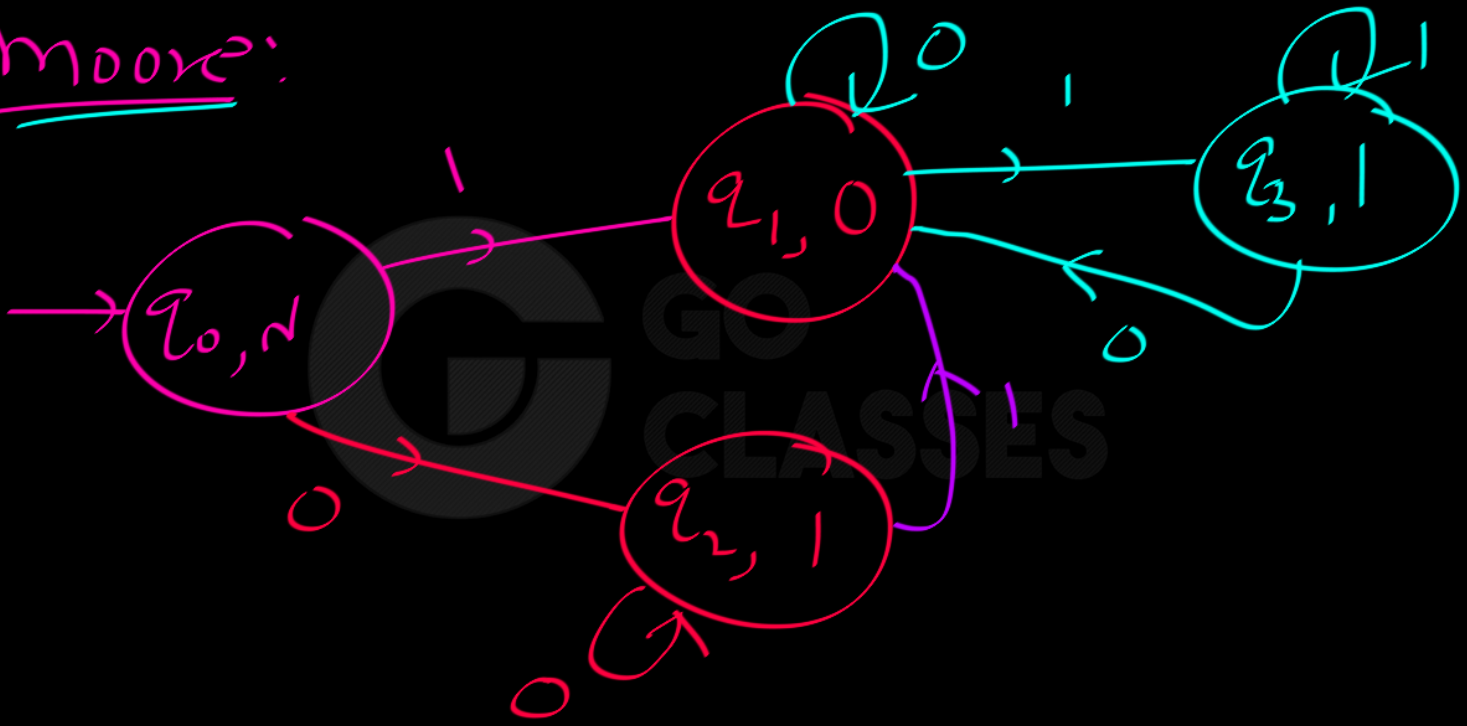
Design Moore , Mealy machine to decrement 1 from a given binary input.

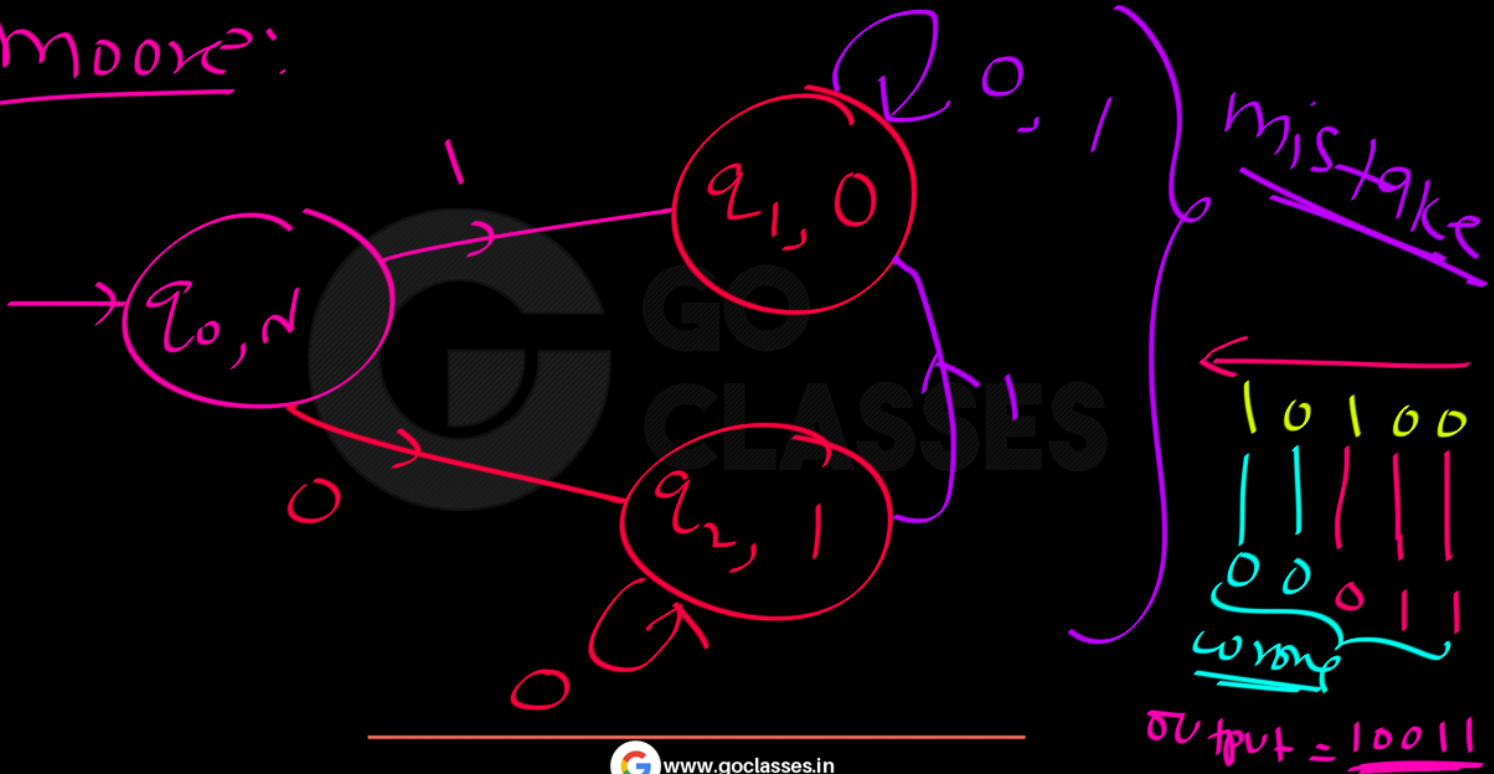


Assume mealy / moore m/c will read input string from right :

mealy:




Moore:
q_0, N
$q_1, 0$
$q_2, 1$
$q_3, 1$
1
0
1
0
1
0


Moore:
q_0, N
$q_1, 0$
$q_2, 1$
1
0
0
0, 1
mistake
1 0 1 0 0
0 0 0 1 1
wrong
output = 10011

Example 2 :

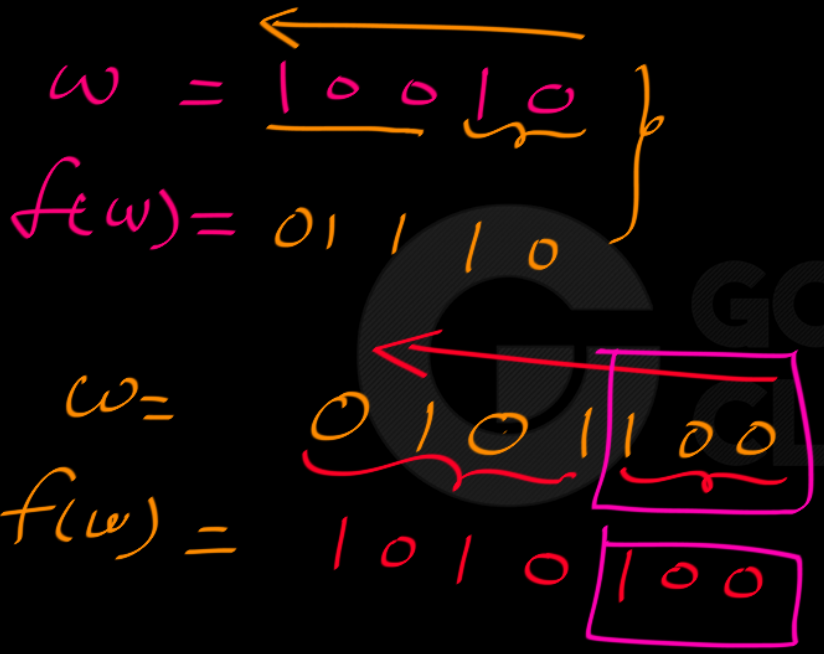
Design Moore , Mealy machine to generate 1's complement of a any given binary input.

Example 3 :

Design Moore , Mealy machine to generate 2's complement of a any given binary input, assuming the input string is read from right to left.

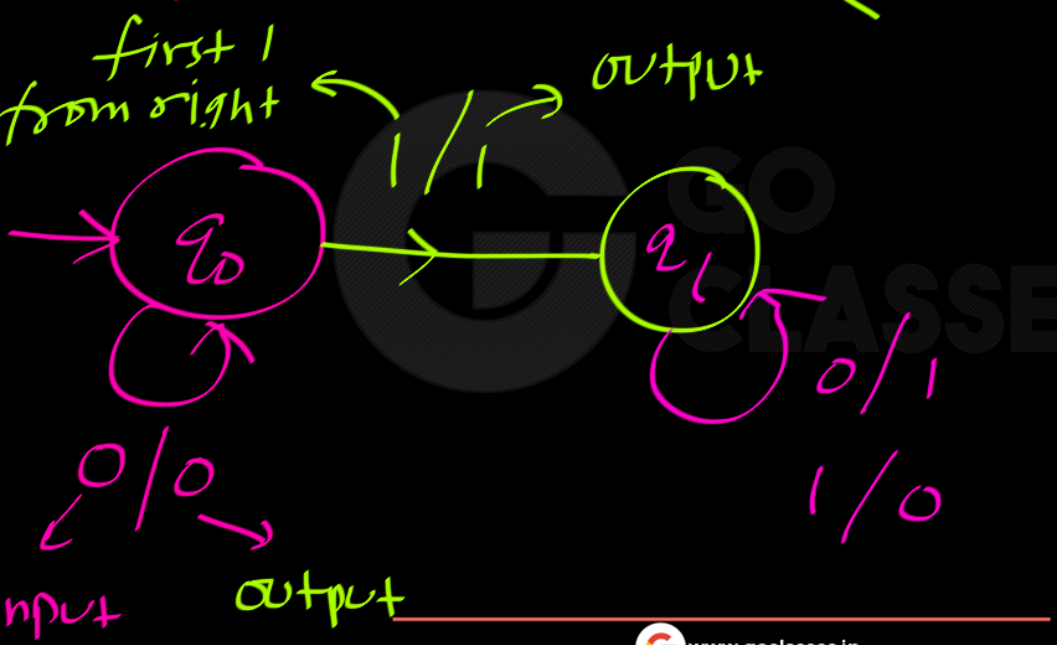

w = 0101 1
Mealy
f(w) = 1 0 1 0 1


w =
0 1 0 1 1
Moore
f(w) = 1 0 1 0 1


w = 10010
f(w) = 01110
w = 0101100
f(w) = 10100

mealy m/c:

w = 1 0 1 0 0 1 0 0

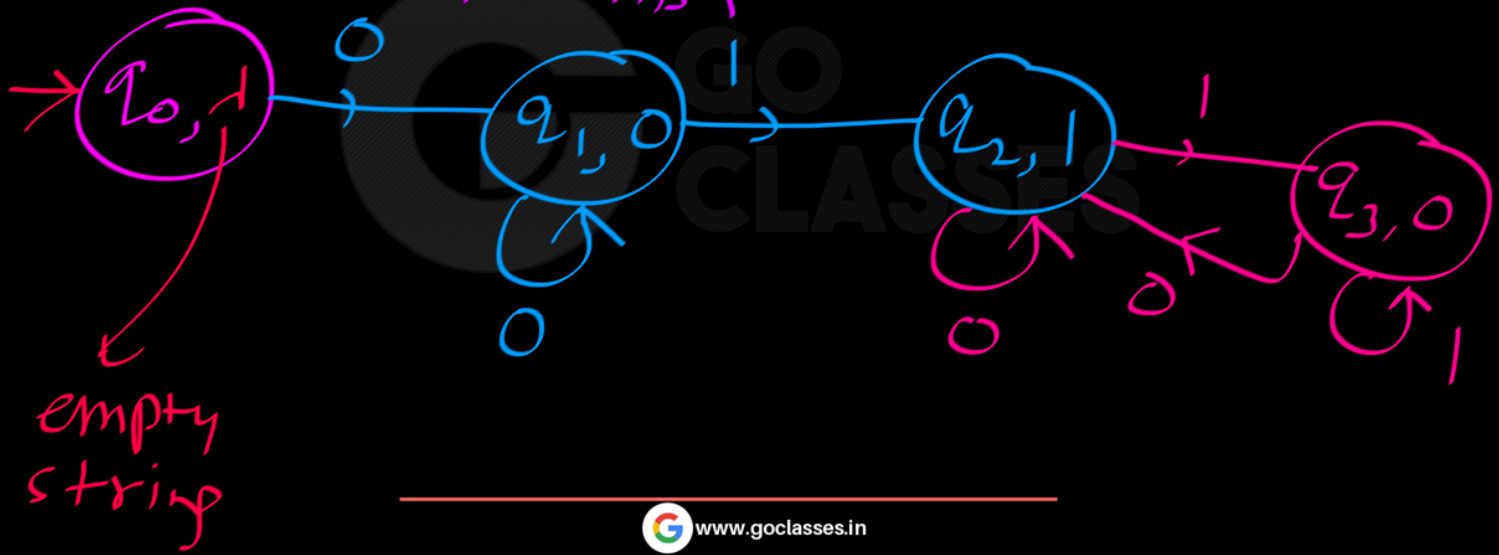


Moore m/c:

$w =$ 1 0 1 0 1 1 0 0

first 1 from RHS

1 0 0

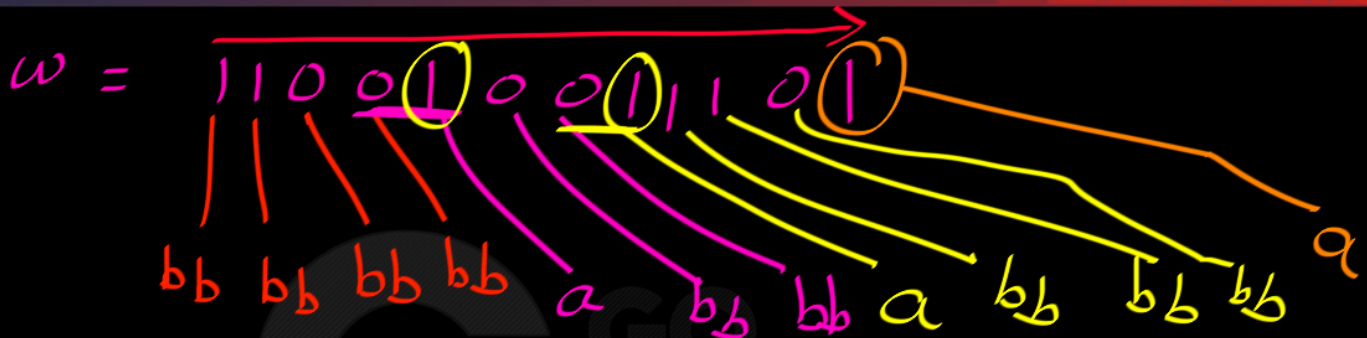


Example 4 :

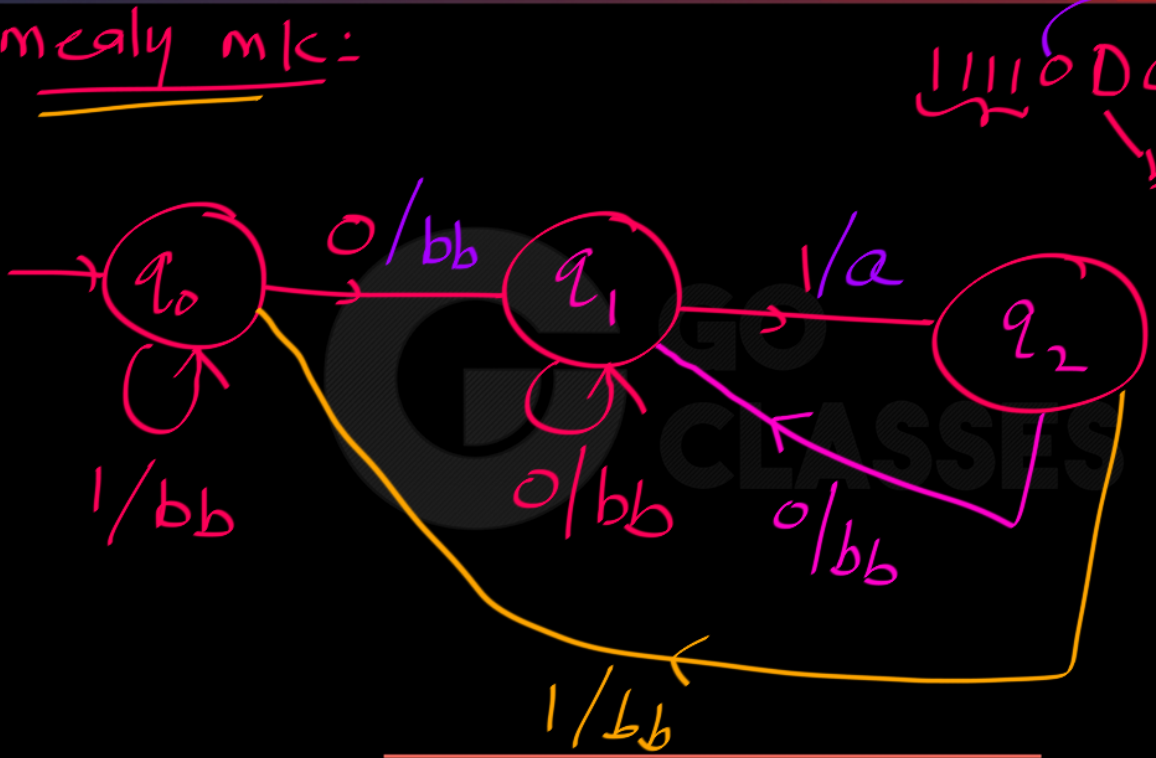
Design Moore , Mealy machine that prints "a" whenever the sequence "01" is encountered in any input binary string, else prints bb.

Sequence Detection

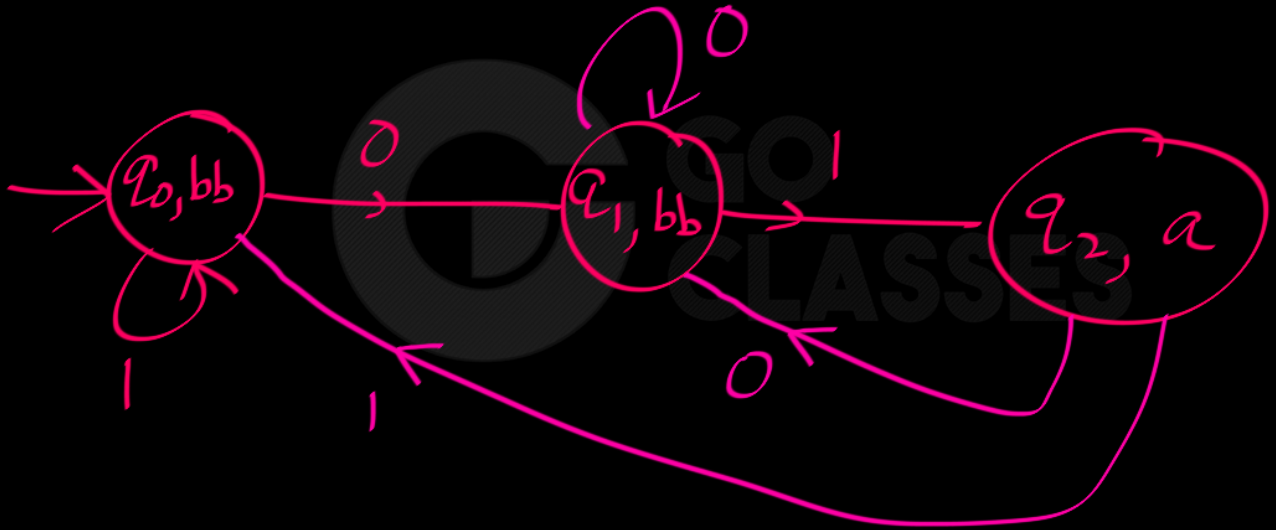
0 0 1 0 1 1 0 1


w = 1 1 0 0 1 0 0 1 1 0 1
bb bb bb bb a bb bb a bb bb bb a

mealy mlc:



Moore m/c:



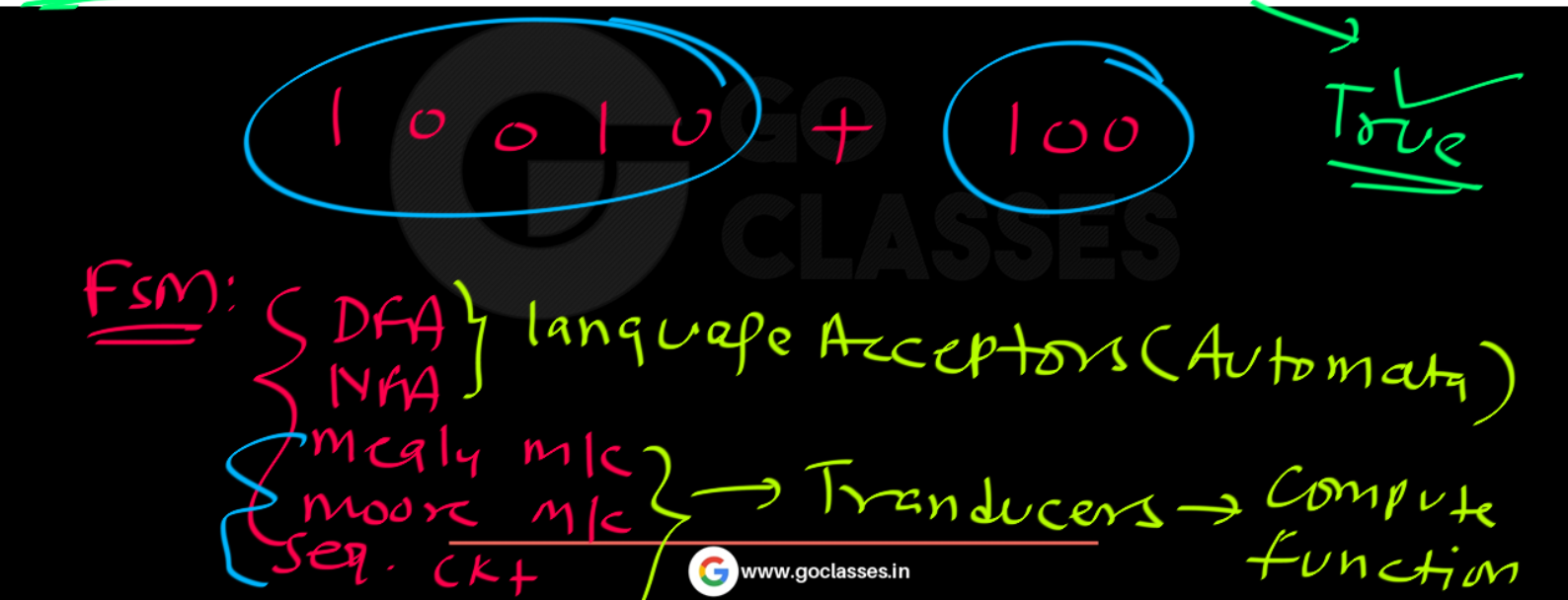
5.5.4 Finite Automata: GATE CSE 1994 | Question: 3.3

top

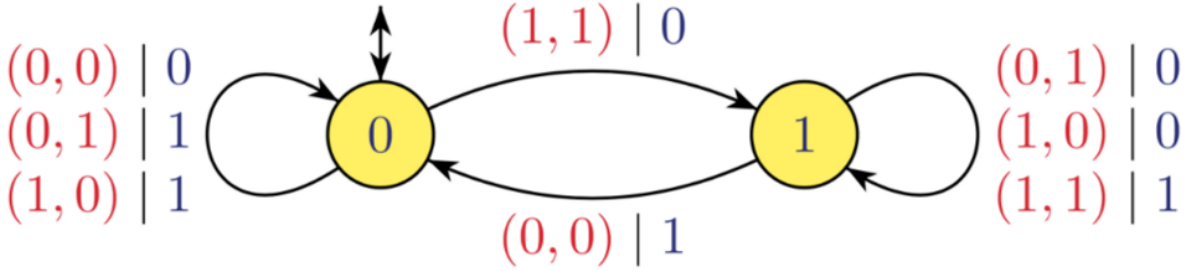
https://gateoverflow.in/2480

State True or False with one line explanation

A FSM (Finite State Machine) can be designed to add two integers of any arbitrary length (arbitrary number of digits).



Something like this:



Edit. The two numbers are written in inverse binary notation and you may need to add an extra zero in the front. For instance, suppose you want to add 22 and 13. In binary notation, 22 is 10110 and 13 is 1101. In inverse binary notation, 22 is 01101 and 13 is 1011. Add 0 at the end of 01101 and 00 at the end of 1011 and then write the two numbers as follows

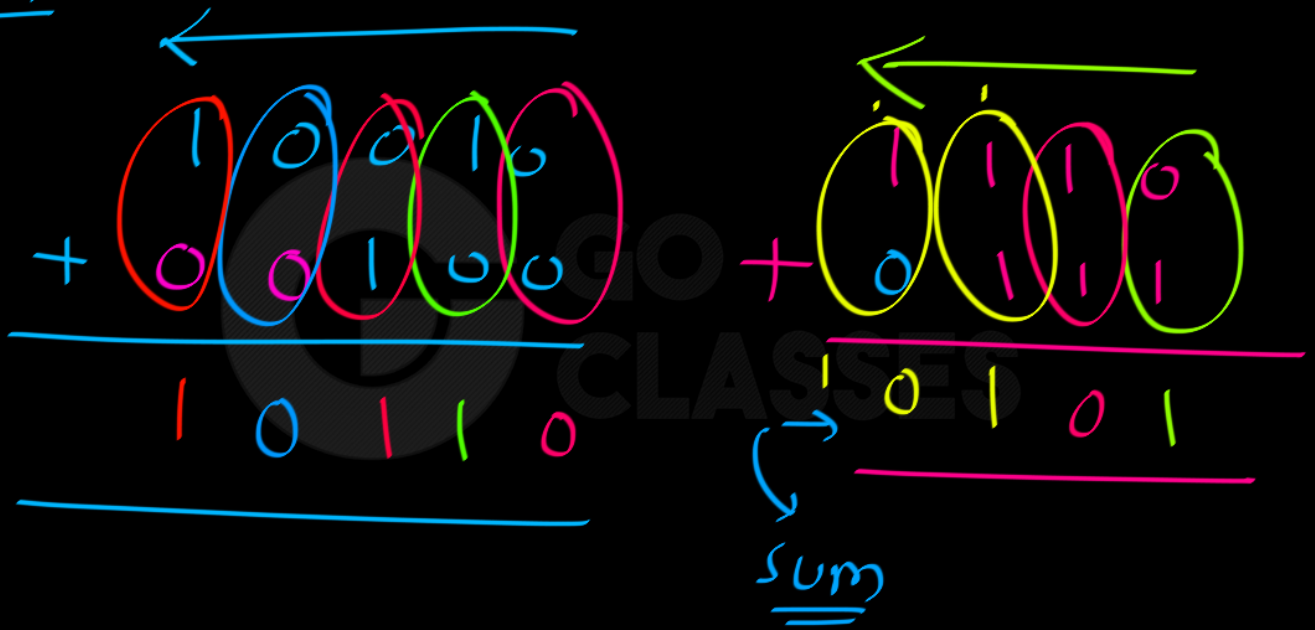
$$\begin{array}{cccccc} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{array}$$

Starting from the initial state, you now have the following path

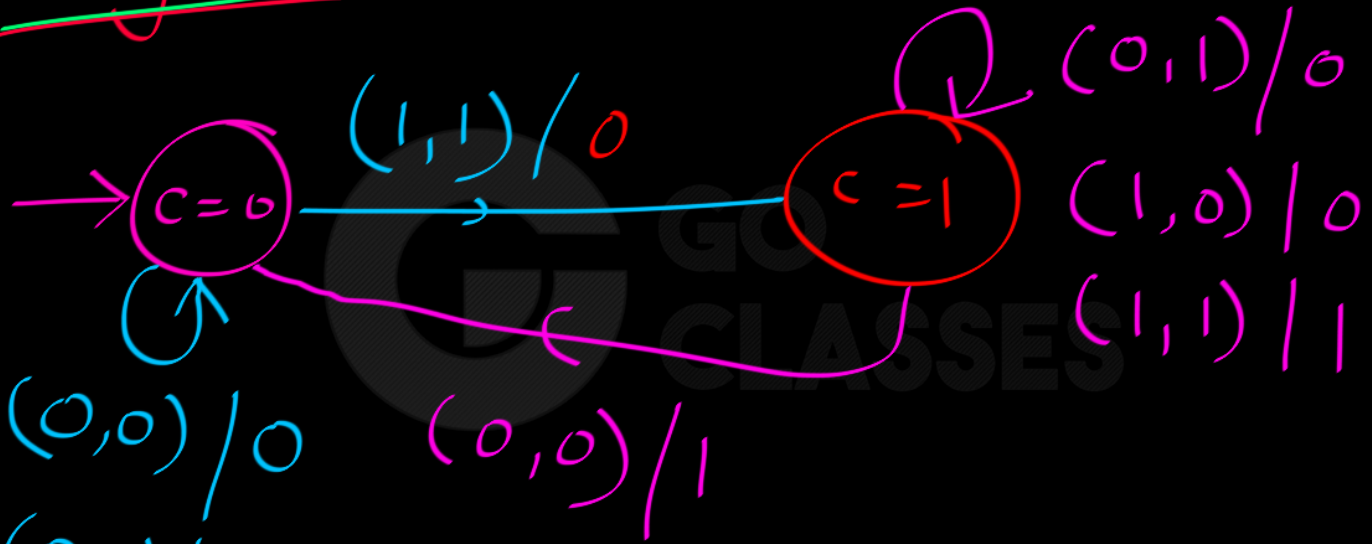
$$0 \xrightarrow{(0,1)|1} 0 \xrightarrow{(1,0)|1} 0 \xrightarrow{(1,1)|0} 1 \xrightarrow{(0,1)|0} 1 \xrightarrow{(1,0)|0} 1 \xrightarrow{(0,0)|1} 0$$

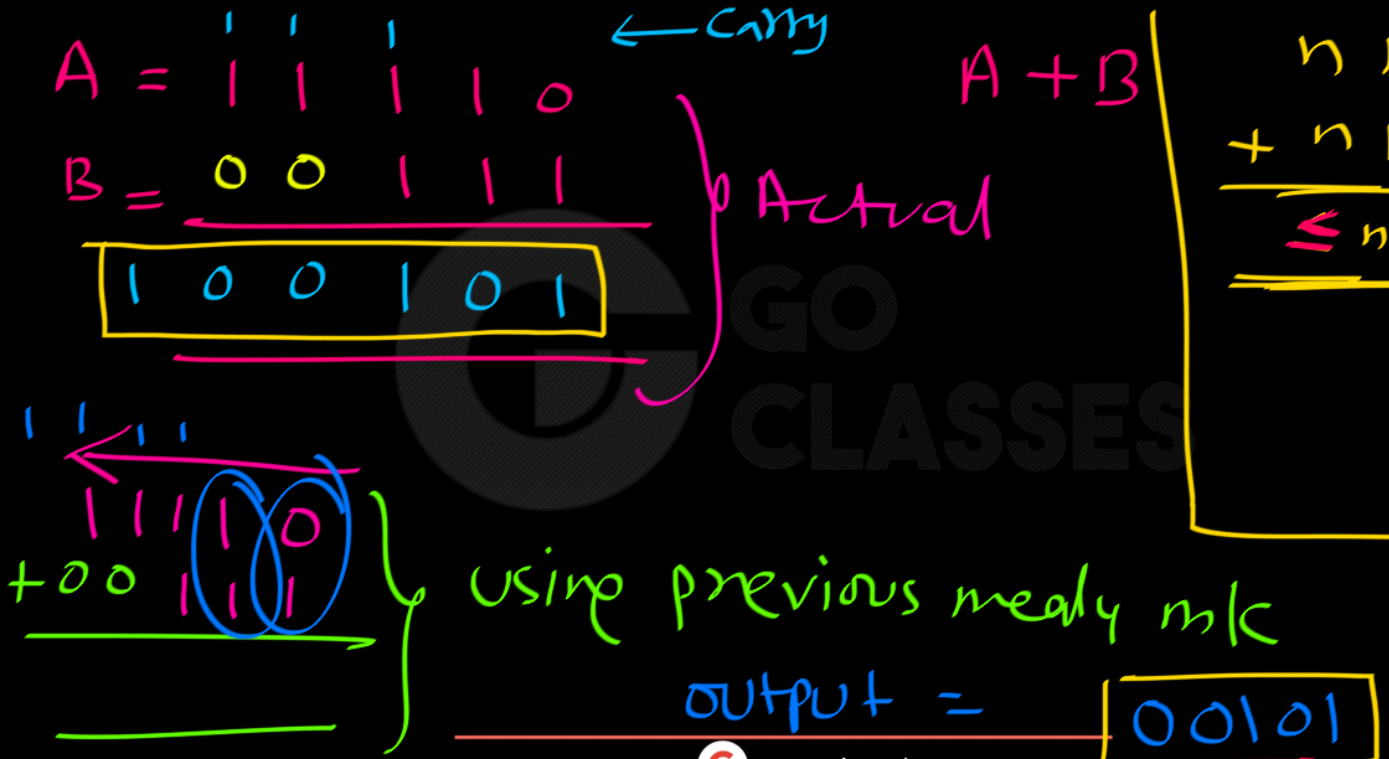
Giving the output 110001 which is 35 in reverse binary notation.

Fsm :



Mealy m/c :




Carry
A = 1 1 1 1 1 0
B = 0 0 1 1 1 1
1 0 0 1 0 1
Actual
A + B
n bit
+ n bit
≤ n+1 bits
1 1 1 1 1
1 1 1 1 1 0
+ 0 0 1 1 1 1
using previous mealy m/c
output = 00101

Solution:

input for mealy machine

0 11110

0 00111

1 00101

full Adder

C_{in}	a	b	sum	C_{out}
0	0	0	0	0
0	0	1	1	0
1	0	1	0	1
1	1	1	1	1
1	1	0	0	1

C_{out} ← a, C_{in} ← a, b, sum

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder
||
Add two bits

5.5.5 Finite Automata: GATE CSE 1995 | Question: 2.23 top

https://gateoverflow.in/2636

A finite state machine with the following state table has a single input x and a single out z.

present state	next state, z	
	x=1	x=0
A	D,0	B,0
B	B,1	C,1
C	B,0	D,1
D	B,1	C,0

If the initial state is unknown, then the shortest input sequence to reach the final state C is:

A. 01
B. 10
C. 101
D. 110

5.5.5 Finite Automata: GATE CSE 1995 | Question: 2.23 top

https://gateoverflow.in/2636

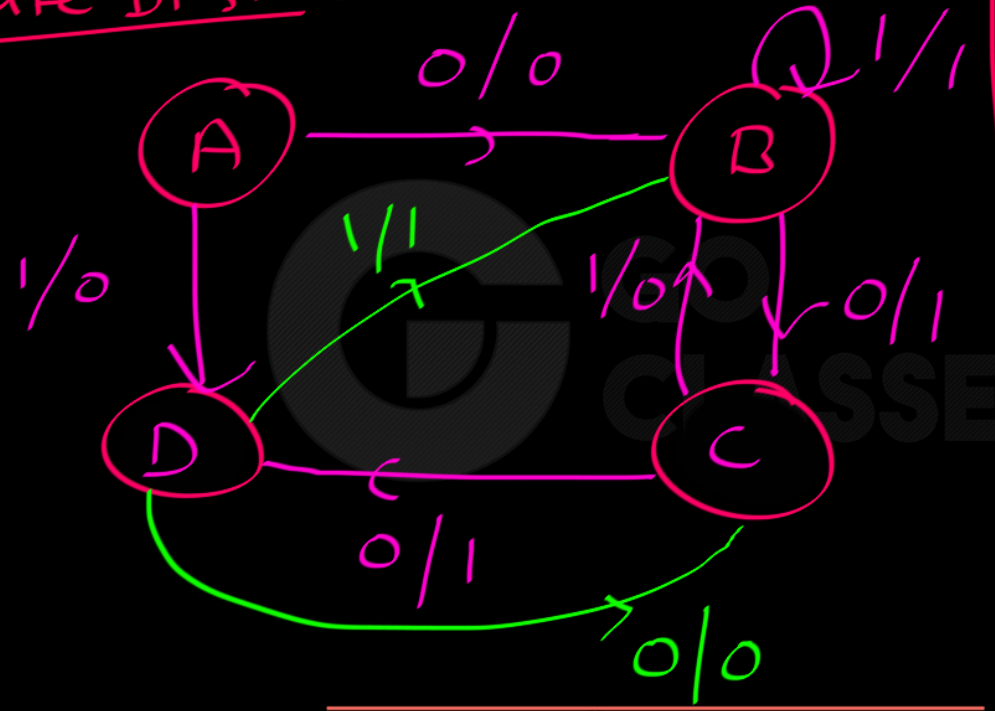
A finite state machine with the following state table has a single input x and a single out z.

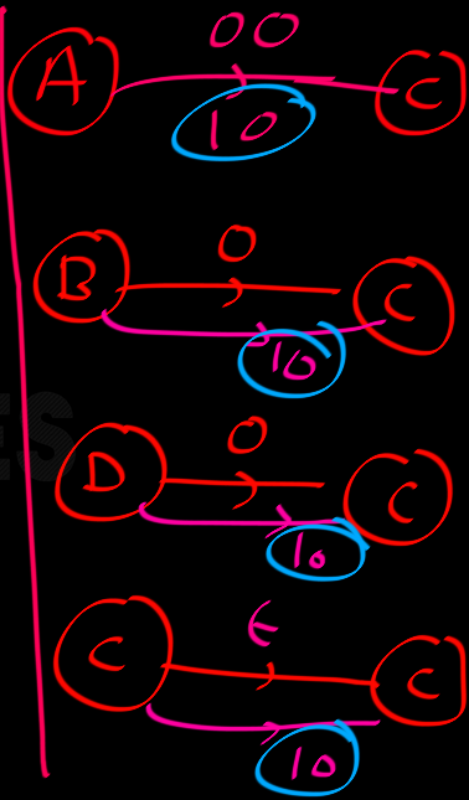
present state	next state, z	
	x=1	x=0
A	D,0	B,0
B	B,1	C,1
C	B,0	D,1
D	B,1	C,0

If the initial state is unknown, then the shortest input sequence to reach the final state C is:

A. 01
B. 10
C. 101
D. 110

State Diagram:



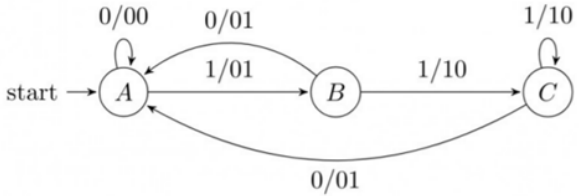


Note that the Question is asking for "Smallest Input string w" such that Regardless of the initial state (whichever is the initial state) we will surely go to C.

5.5.10 Finite Automata: GATE CSE 2002 | Question: 2.5 top

https://gateoverflow.in/835

The finite state machine described by the following state diagram with A as starting state, where an arc label is x and y stands for 1-bit input and y stands for 2-bit output

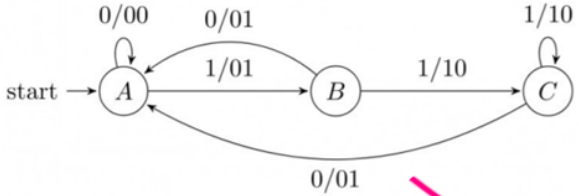


A. outputs the sum of the present and the previous bits of the input
B. outputs 01 whenever the input sequence contains 11
C. outputs 00 whenever the input sequence contains 10
D. none of the above

5.5.10 Finite Automata: GATE CSE 2002 | Question: 2.5 top

https://gateoverflow.in/835

The finite state machine described by the following state diagram with A as starting state, where an arc label is x and y stands for 1-bit input and y stands for 2-bit output



A. outputs the sum of the present and the previous bits of the input
B. outputs 01 whenever the input sequence contains 11
C. outputs 00 whenever the input sequence contains 10
D. none of the above

x/y
1-bit input
2-bit o/p
mealy m/c
output function : (A, 0) → 00 ; (C, 1) → 10

Input string: A A B C A B A
0 1 1 0 1 0 — binary string

output " : 0 0 0 1 1 0 0 1 0 1 0 1 — binary string

Mealy m/c: $(Q, A, \Sigma, \gamma, \delta, \omega)$

$Q = \{A, B, C\}$

$\Sigma = \{0, 1\}$

$\gamma = \{0, 1\}$

δ → state Transition function

ω → output function

State Transition function δ

$(A, 0) \longrightarrow A$

$(A, 1) \longrightarrow B$

$(C, 1) \longrightarrow C$

$(C, 0) \longrightarrow A$

output function ω

$(A, 0) \rightarrow 00$

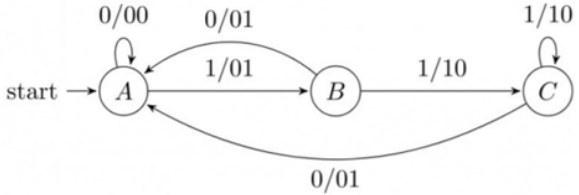
$(A, 1) \rightarrow 01$

$(C, 1) \rightarrow 10$

5.5.10 Finite Automata: GATE CSE 2002 | Question: 2.5 top

https://gateoverflow.in/835

The finite state machine described by the following state diagram with A as starting state, where an arc label is and x stands for 1-bit input and y stands for 2-bit output



A. outputs the sum of the present and the previous bits of the input
B. outputs 01 whenever the input sequence contains 11
C. outputs 00 whenever the input sequence contains 10
D. none of the above

I/P — 0 1 1 0 1 0

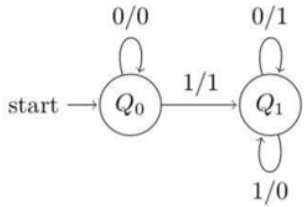
O/P — 0 0 0 1 1 0 0 1 0 1 0 1

O/P : 00, 01, 01, 01, 10, 10, 01, 00, 00, 01

5.5.16 Finite Automata: GATE CSE 2005 | Question: 63 top

https://gateoverflow.in/1386

The following diagram represents a finite state machine which takes as input a binary number from the least significant bit.



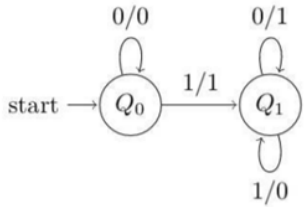
Which of the following is TRUE?

A. It computes 1's complement of the input number
B. It computes 2's complement of the input number
C. It increments the input number
D. it decrements the input number

5.5.16 Finite Automata: GATE CSE 2005 | Question: 63 top

https://gateoverflow.in/1386

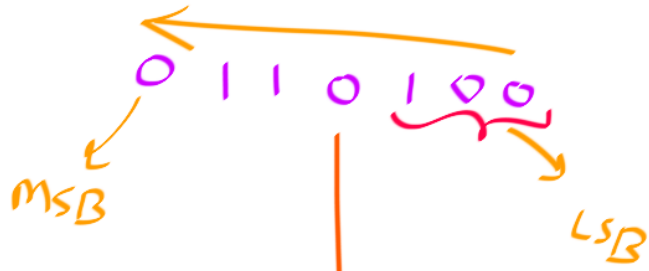
The following diagram represents a finite state machine which takes as input a binary number from the least significant bit.



Mealy m/c

o/p: $Q \times \Sigma \to \Gamma$

I/p:



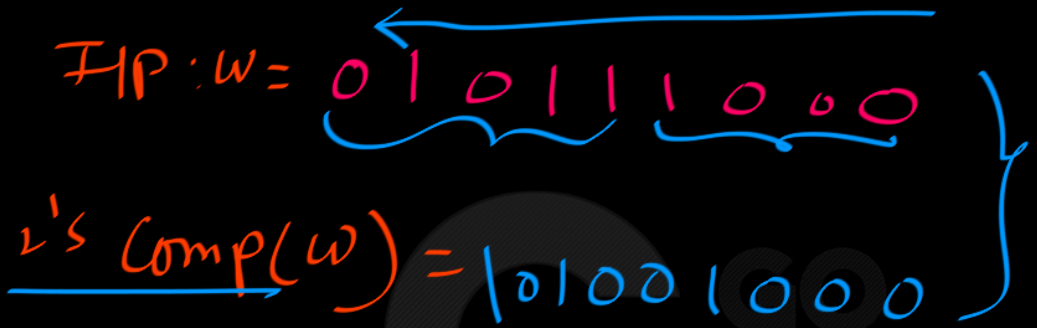
Which of the following is TRUE?

A. It computes 1's complement of the input number
B. It computes 2's complement of the input number
C. It increments the input number
D. it decrements the input number

o/p: 1 0 0 1 1 0 0

2's comp

1's comp: 1 0 0 1 0 1 1


FP: w = 0 1 0 1 1 1 0 0 0
2's Comp(w) = 1 0 1 0 0 1 0 0 0

5.5.20 Finite Automata: GATE CSE 2009 | Question: 27 top

https://gateoverflow.in/1313

Given the following state table of an FSM with two states A and B,one input and one output.

PRESENT STATE A	PRESENT STATE B	Input	Next State A	Next State B	Output
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	0	1

If the initial state is $A = 0, B = 0$ what is the minimum length of an input string which will take the machine to the state $A = 0, B = 1$ with $output = 1$.

A. 3
B. 4
C. 5
D. 6

5.5.20 Finite Automata: GATE CSE 2009 | Question: 27 top

https://gateoverflow.in/1313

Given the following state table of an FSM with two states A and B,one input and one output.

mealy m/c

PRESENT STATE A	PRESENT STATE B	Input	Next State A	Next State B	Output
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	0	1

If the initial state is $A = 0, B = 0$ what is the minimum length of an input string which will take the machine to the state $A = 0, B = 1$ with $output = 1$.

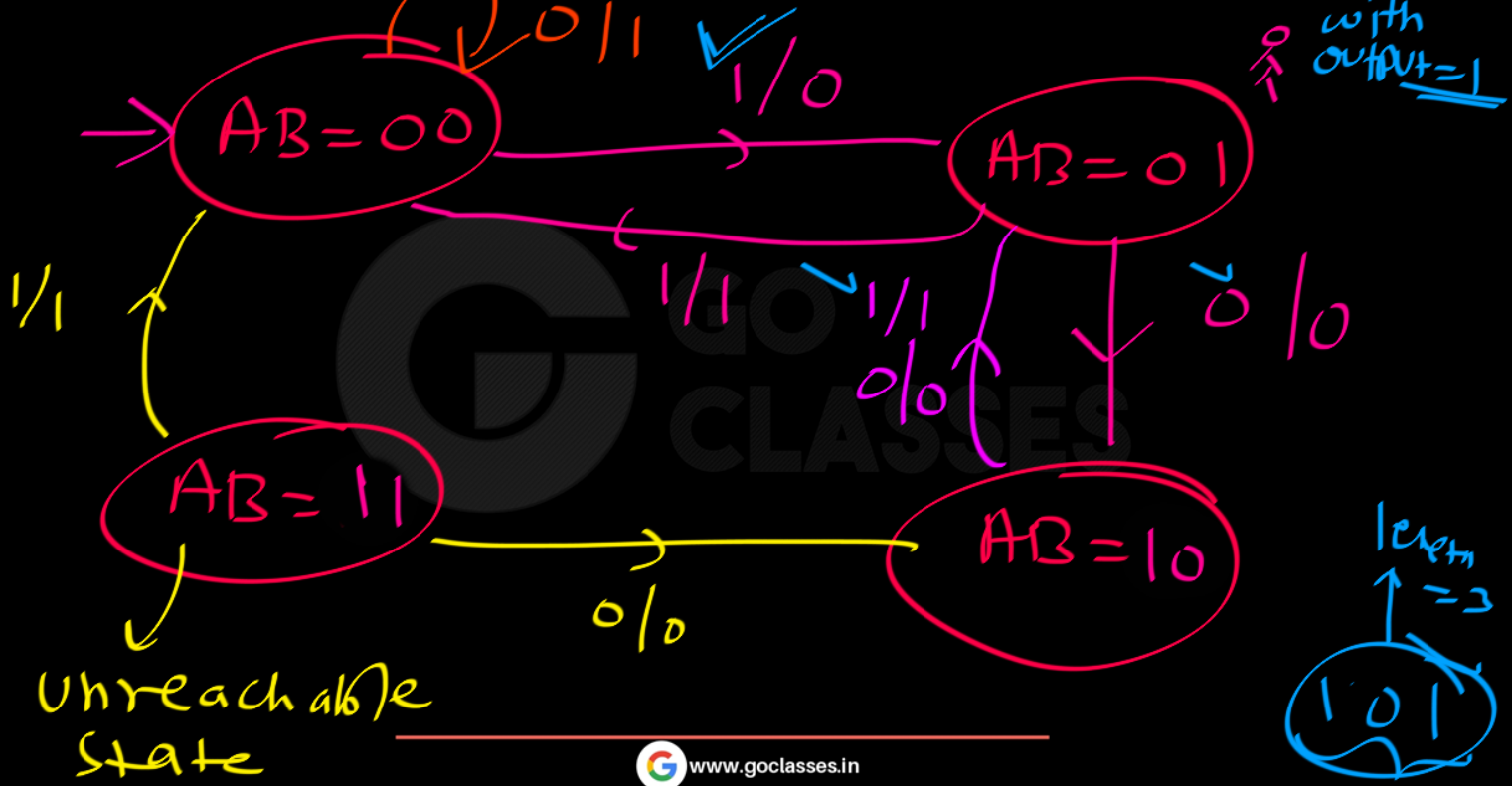
A. 3
B. 4
C. 5
D. 6

(00, 0) → 1

(00, 1) → 0

(Q, I/P) → O/P

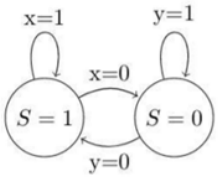
mealy m/c


0/1
1/0
AB=00
AB=01
with output=1
1/1
1/1
1/1
0/0
0/0
AB=11
AB=10
0/0
Unreachable state
Length =3
101

5.5.35 Finite Automata: GATE IT 2006 | Question: 37 top

https://gateoverflow.in/3576

For a state machine with the following state diagram the expression for the next state S^+ in terms of the current state S and the input variables x and y is



$S^+ = f(S, x, y)$

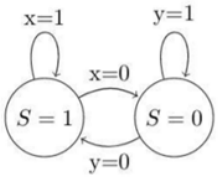
next state equation.

A. $S^+ = S'.y' + S.x$
B. $S^+ = S.x.y' + S'.y.x'$
C. $S^+ = x.y'$
D. $S^+ = S'.y + S.x'$

5.5.35 Finite Automata: GATE IT 2006 | Question: 37 top

https://gateoverflow.in/3576

For a state machine with the following state diagram the expression for the next state S^+ in terms of the current state S and the input variables x and y is



A. $S^+ = S'.y' + S.x$
B. $S^+ = S.x.y' + S'.y.x'$
C. $S^+ = x.y'$
D. $S^+ = S'.y + S.x'$

$S^+ = f(S, x, y)$

next state equation.

from state Diagram:

$S^+ = Sx + \bar{S}\bar{y}$

S^+ : next state variable

State Table:

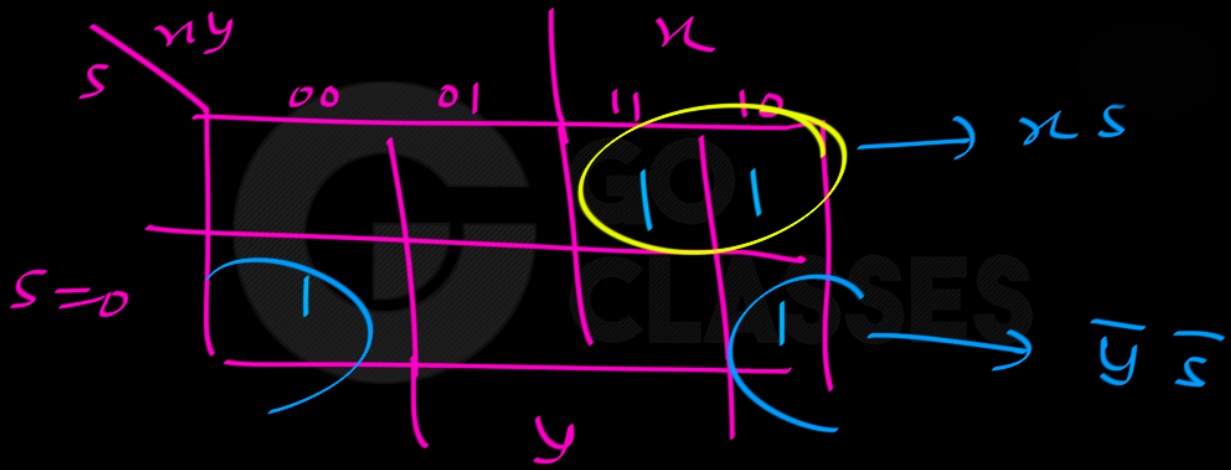
→ Don't cares

s	x	y	s^+
0	X	1	0
0	X	0	1 ✓
1	0	X	0
1	1	X	1 ✓

$$s^+ = \bar{s}\bar{y} + sx$$

k-map: for s^+:

$s^+ = f(s, x, y)$



$s^+ = xs + \bar{y}\,\bar{s}$

5.5.30 Finite Automata: GATE CSE 2021 Set 2 | Question: 28 top

https://gateoverflow.in/357512

Suppose we want to design a synchronous circuit that processes a string of 0's and 1's. Given a string, it produces another string by replacing the first 1 in any subsequence of consecutive 1's by a 0. Consider the following example.

Input sequence:	00100011000011100
Output sequence:	00000001000001100

A *Mealy Machine* is a state machine where both the next state and the output are functions of the present state and the current input.

The above mentioned circuit can be designed as a two-state Mealy machine. The states in the Mealy machine can be represented using Boolean values 0 and 1. We denote the current state, the next state, the next incoming bit, and the output bit of the Mealy machine by the variables s, t, b and y respectively.

Assume the initial state of the Mealy machine is 0.

What are the Boolean expressions corresponding to t and y in terms of s and b?

A. $t = s + b$, $y = sb$

B. $t = b$, $y = sb$

C. $t = b$, $y = s\bar{b}$

D. $t = s + b$, $y = s\bar{b}$

current state = s

next " = t = s+

I/p = b

O/p = y

5.5.30 Finite Automata: GATE CSE 2021 Set 2 | Question: 28 top

https://gateoverflow.in/357512

Suppose we want to design a synchronous circuit that processes a string of 0's and 1's. Given a string, it produces another string by replacing the first 1 in any subsequence of consecutive 1's by a 0. Consider the following example.

Input sequence:	00100011000011100
Output sequence:	00000001000001100

A *Mealy Machine* is a state machine where both the next state and the output are functions of the present state and the current input.

The above mentioned circuit can be designed as a two-state Mealy machine. The states in the Mealy machine can be represented using Boolean values 0 and 1. We denote the current state, the next state, the next incoming bit, and the output bit of the Mealy machine by the variables s, t, b and y respectively.

Assume the initial state of the Mealy machine is 0.

What are the Boolean expressions corresponding to t and y in terms of s and b?

A. $t = s + b$, $y = sb$

B. $t = b$, $y = sb$

C. $t = b$, $y = s\bar{b}$

D. $t = s + b$, $y = s\bar{b}$

current state = s

next " = t = s+

I/p = b

O/p = y

$w = 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0$

$f(w) = 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0$

output = f(current state, input)

next state = f(current state, input)

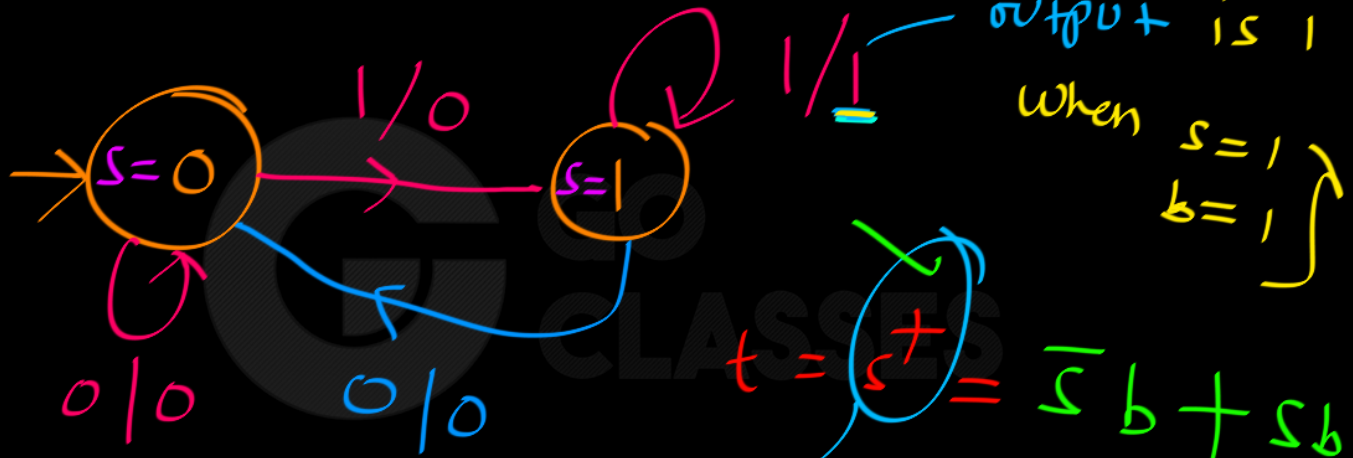
output $y = f(s, b)$

o/p variable

$t = s^+ = f(s, b)$

next state variable

Mealy mlc:


1/0
1/1
output is 1
when s=1
b=1
s=0
s=1
0/0
0/0
t = s+ = s̄b + sb
= b
next state variable
y = sb
output variable

Any variable (o/p variable, next state variable)

Equation can be written either

from State Diagram

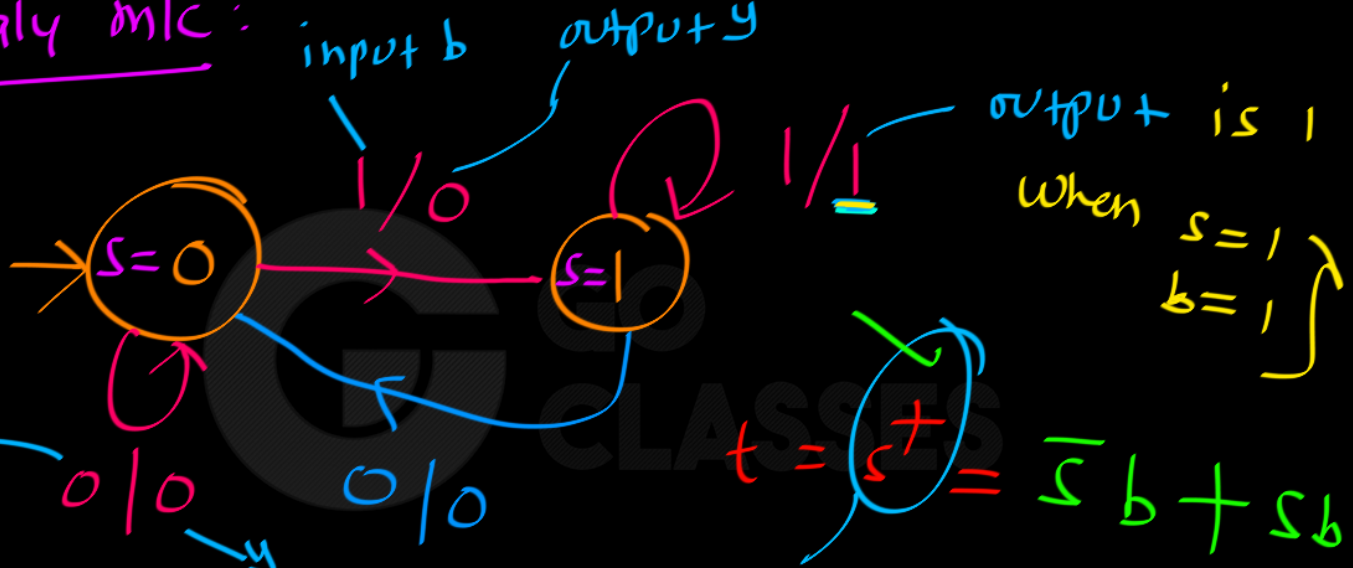
or state table

Equation of a variable

↓

Equation of any variable

Mealy mlc:


input b
output y
1/0
1/1
output is 1
when s=1 b=1
s=0
s=1
b
0/0
0/0
y
t = s+ = s̄b + sb
= b
next state variable
y = sb
output variable

State table :

next state variable

o/p variable

s	b	$t = s^+$	y
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	1

$s^+ = b$

$y = sb$

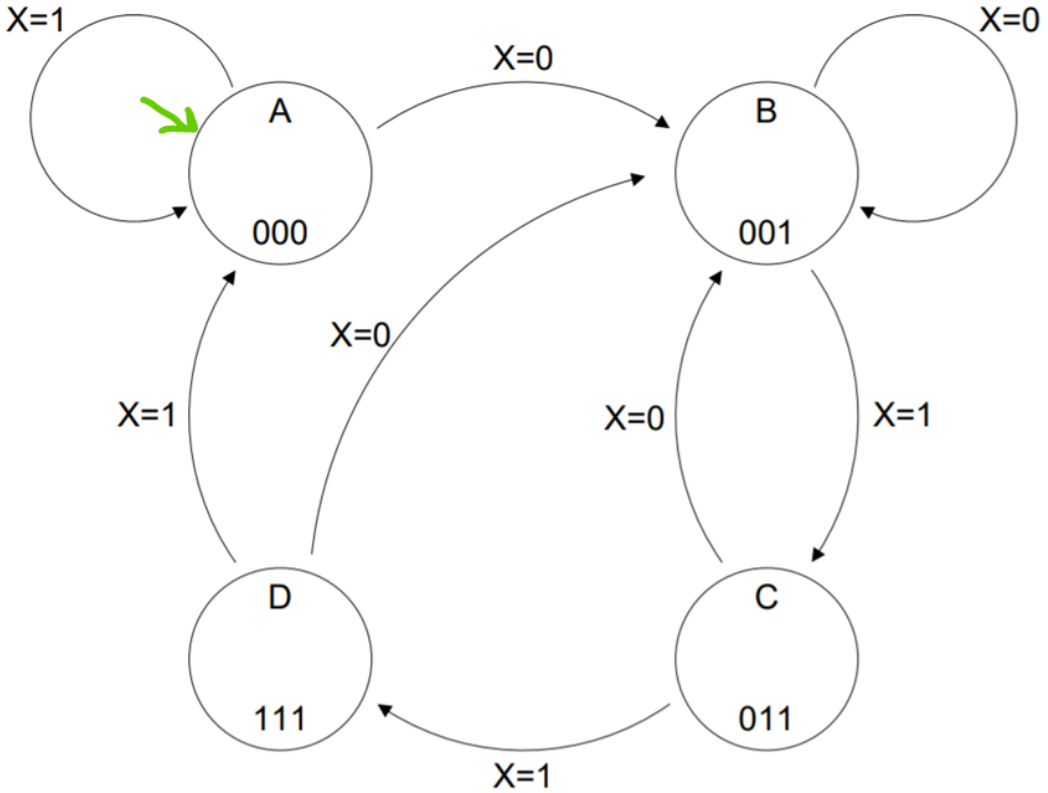
$t = s^+ = f(s, b)$

$y = f(s, b)$

Example 5 :

Design a simple sequence detector for the sequence 011. Include three outputs that indicate how many bits have been received in the correct sequence.

(For example, each output could be connected to an LED.)



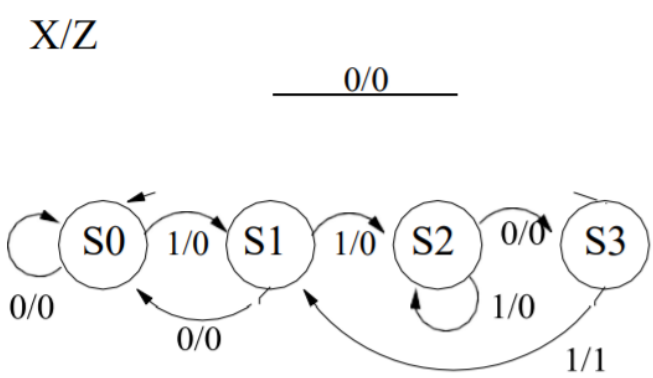
STATES
A=00
B=01
C=11
D=10

MOORE SEQUENCE DETECTOR FOR 011

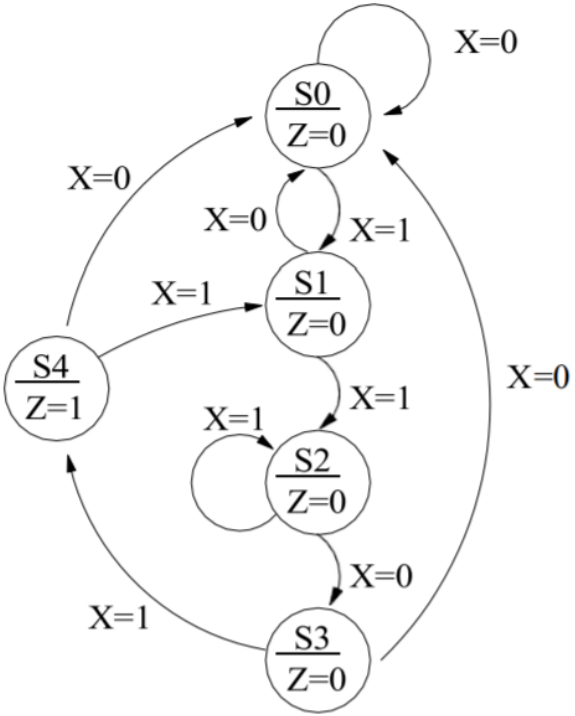
Example 6 :

Create a state diagram for a sequence detector that outputs a 1 when it detects the final bit in the serial data stream 1101. Assume overlapping is allowed.

Mealy Machine



Moore Machine

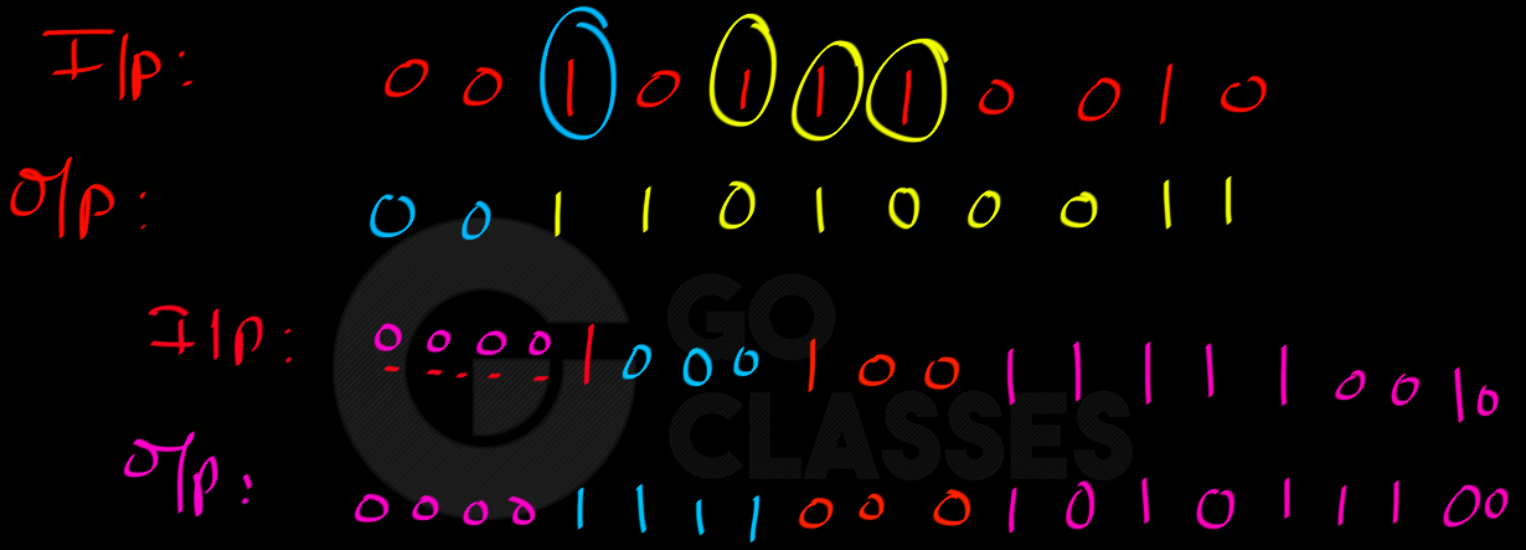


Peter linz

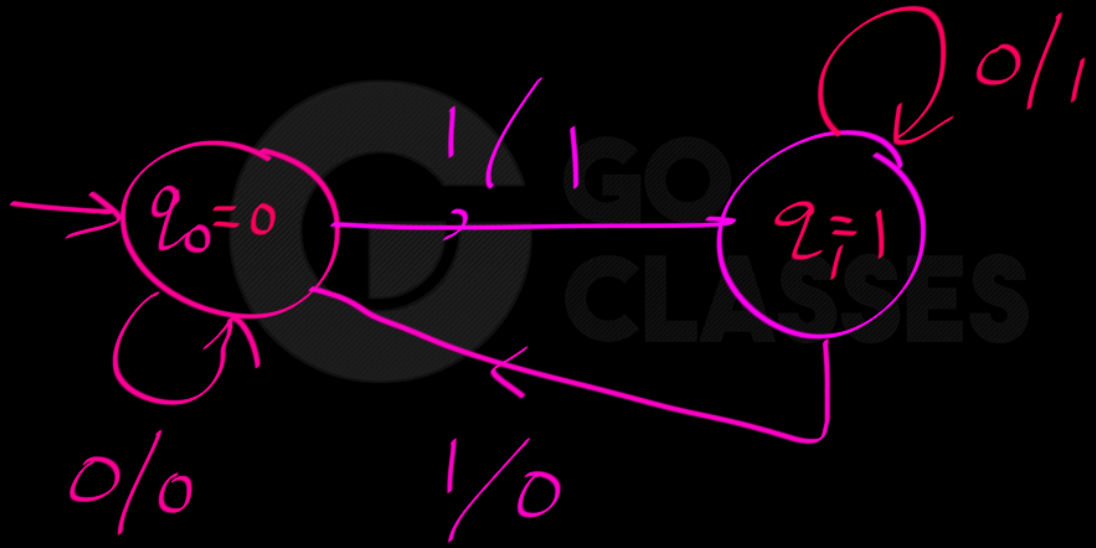
EXAMPLE A.2

and moore

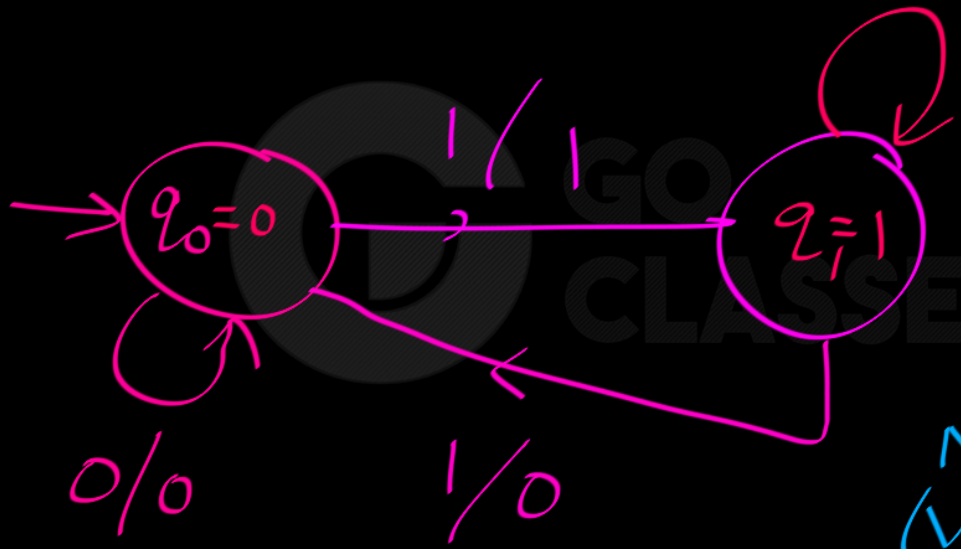
Construct a Mealy machine M that takes as input strings of 0's and 1's. Its output is to be a string of 0's until the first 1 occurs in the input, at which time it will switch to printing 1's. This is to continue until the next 1 is encountered in the input, when the output reverts to 0. The alternation continues every time a 1 is encountered. For example, $F_M(0010010) = 0011100$. This fst is a simple model for a flip-flop circuit. Figure A.2 shows a solution.


I/P: 0 0 1 0 1 0 0 0 1 0
O/P: 0 0 1 1 0 1 0 0 0 1 1
I/P: 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0
O/P: 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0

Mealy m/c :-



Mealy m/c :-



Output (variable) equation ?

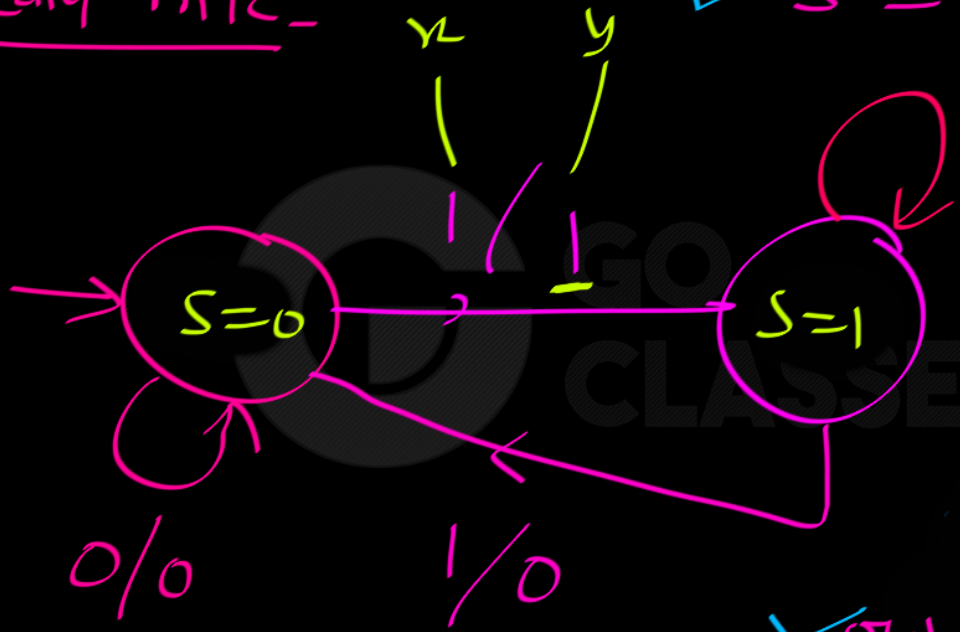
Next state (Variable) equation ?

Output variable $= y = f(s, x)$

s: current state, x: Input bit

Next state $s^+ = f(s, x)$

Mealy m/c :



$S^+ = \bar{S}x + S\bar{x}$

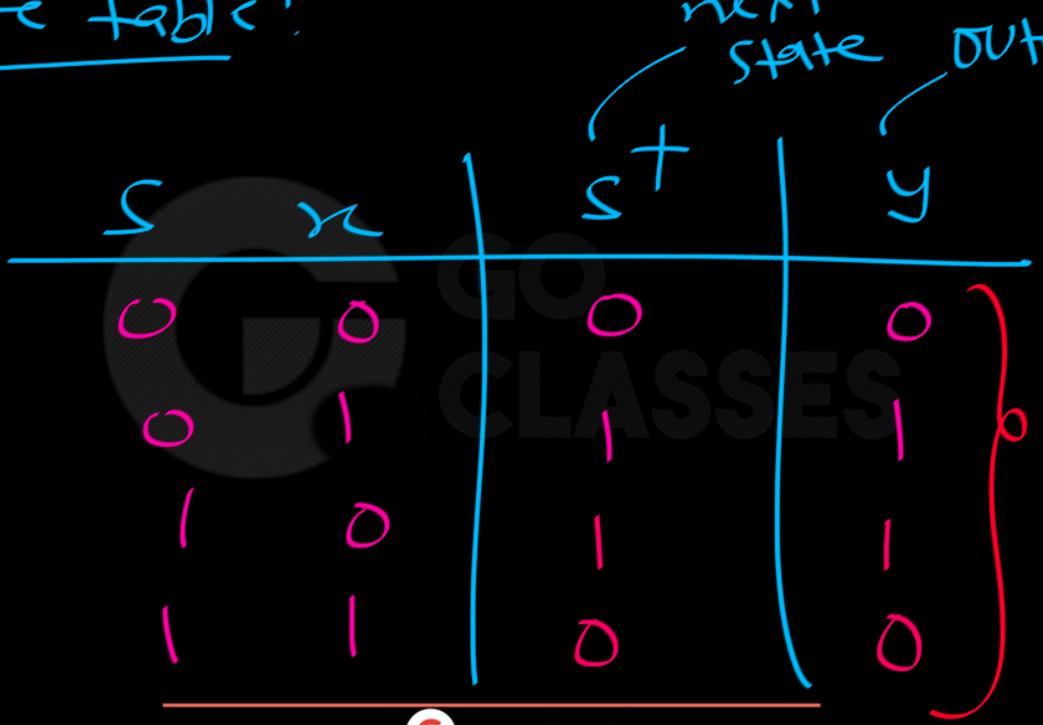
$= S \oplus x$

$y = \bar{S}x + S\bar{x}$

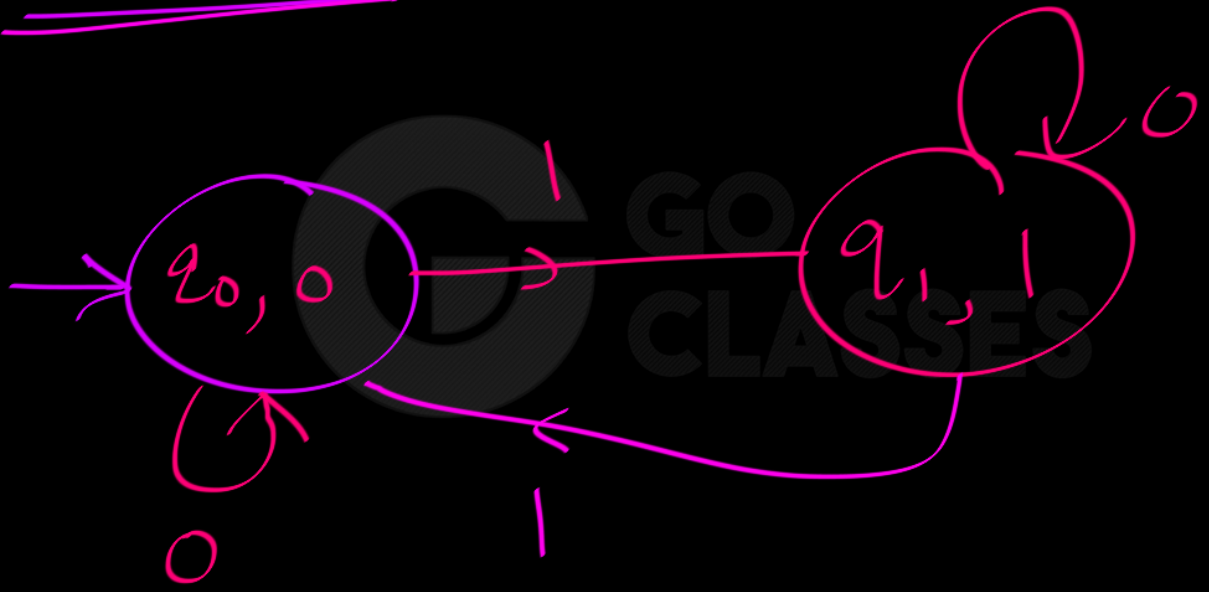
$\text{output} = \bar{S}x + S\bar{x}$

$= S \oplus x$

State table:



Moore m|c



A Moore machine solution for the problem in Example A.2 is given in Figure A.3.

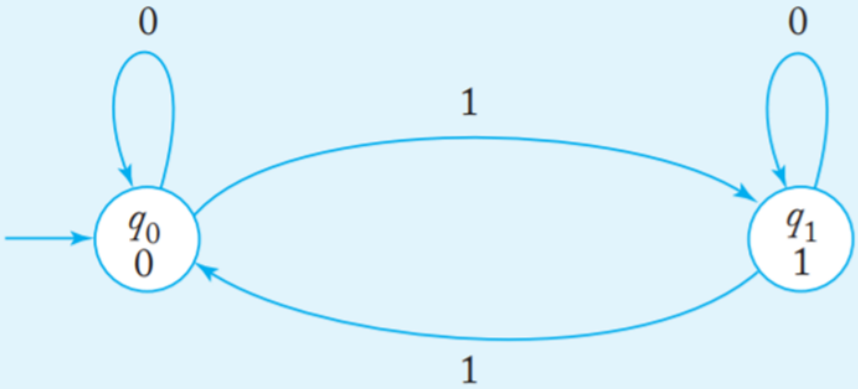


FIGURE A.3

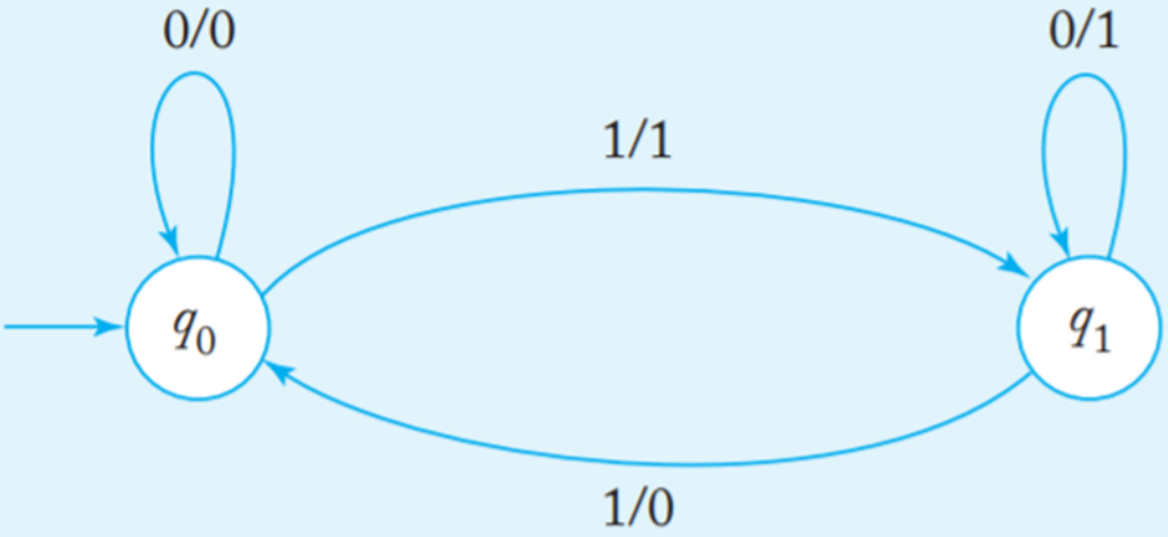


FIGURE A.2