Digital Electronics And Computer Architecture

Department of Computer Science and Engineering

Difference between **combinational** and **sequential** logic

Sr. No.	Combinational Logic	Sequential Logic
1.	The output are all times depend on the combination of input variable.	The output is depend on present input as well depend on past history of input variable
2.	It does not use any memory	It has memory so output can vary based on input.
3.	It is easy to design	It is harder to design
4.	Faster	Slower

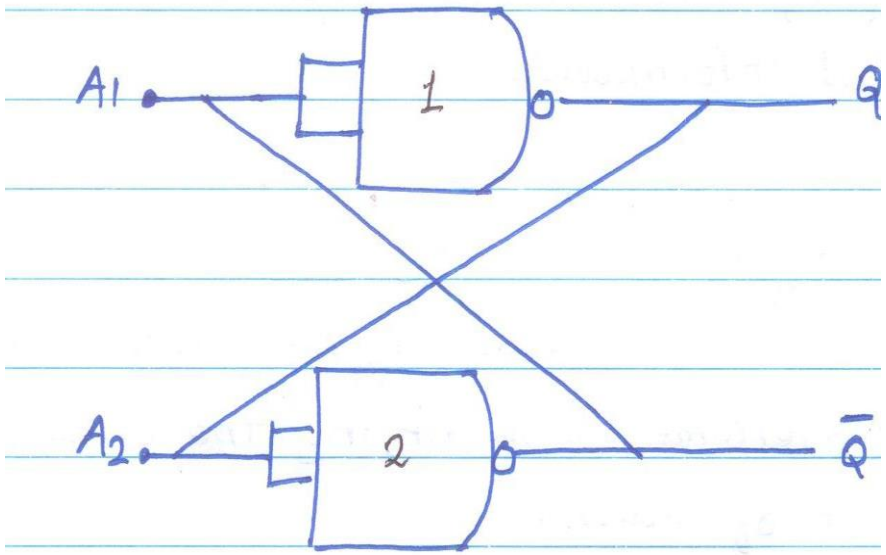
Difference between **asynchronous** and **synchronous** circuits

- The SEQUENCIAL Circuit are classified in two main categories, known as Asynchronous and Synchronous.
- Sequential circuit depending on timing of their signals.
- Asynchronous Circuit: A sequential circuit whose behavior depends upon the sequence in which the input signal change is referred to as an asynchronous circuit.
- Synchronous Circuit: A sequential circuit whose behavior can be defined from knowledge of its signal at discrete instants of time is referred to as a synchronous circuit.

Sequential Logic

- It has a memory element in the circuit that stores the result of the previous set of inputs and the current output depends on inputs **in the past** as well as present inputs.
- The basic element in sequential logic is the **bistable latch** or **flip-flop**, which acts as a memory element for one bit of data.

One bit memory cell



Latch

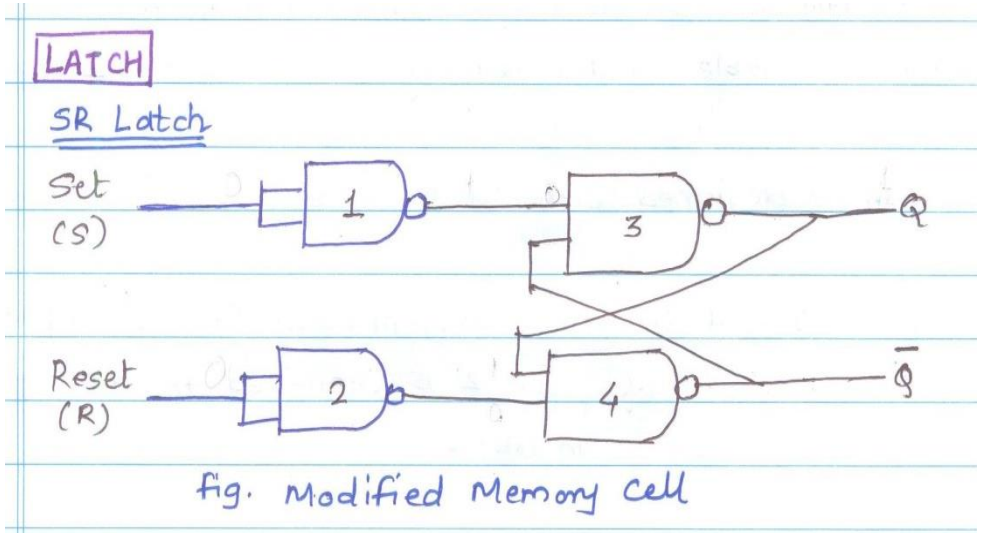
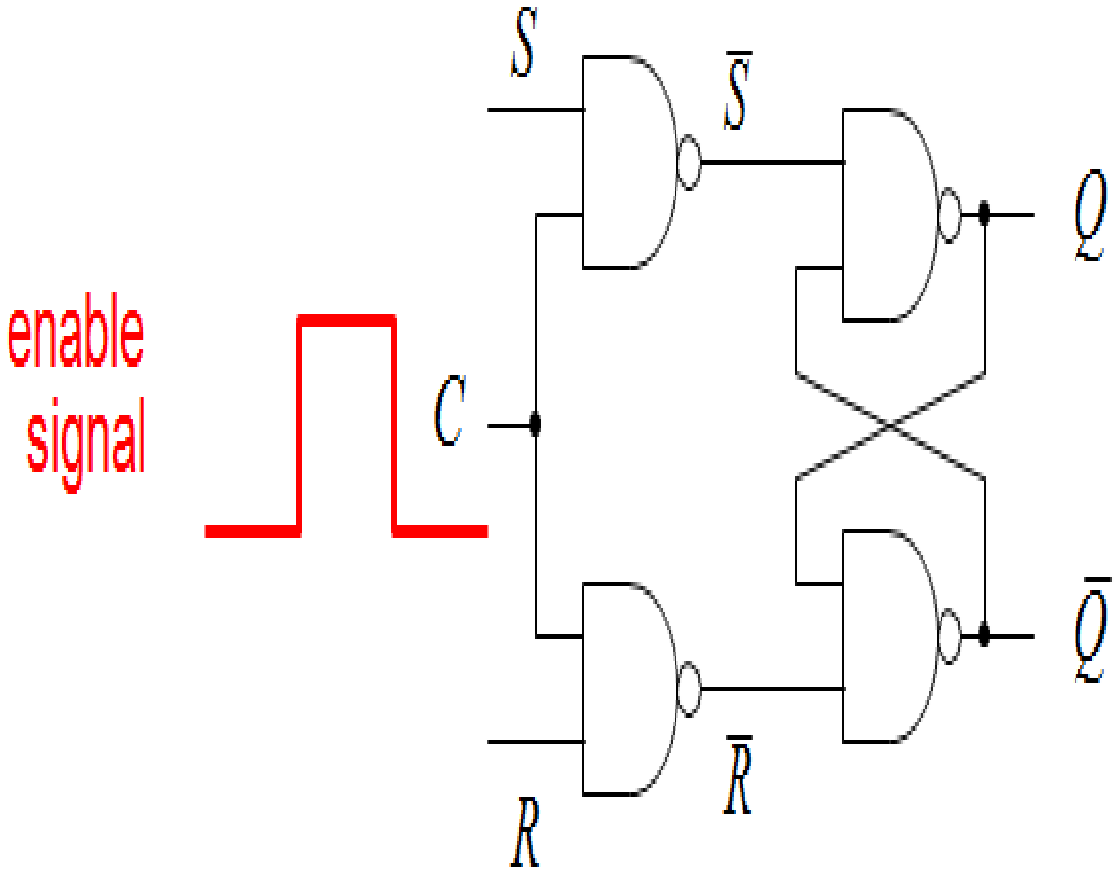


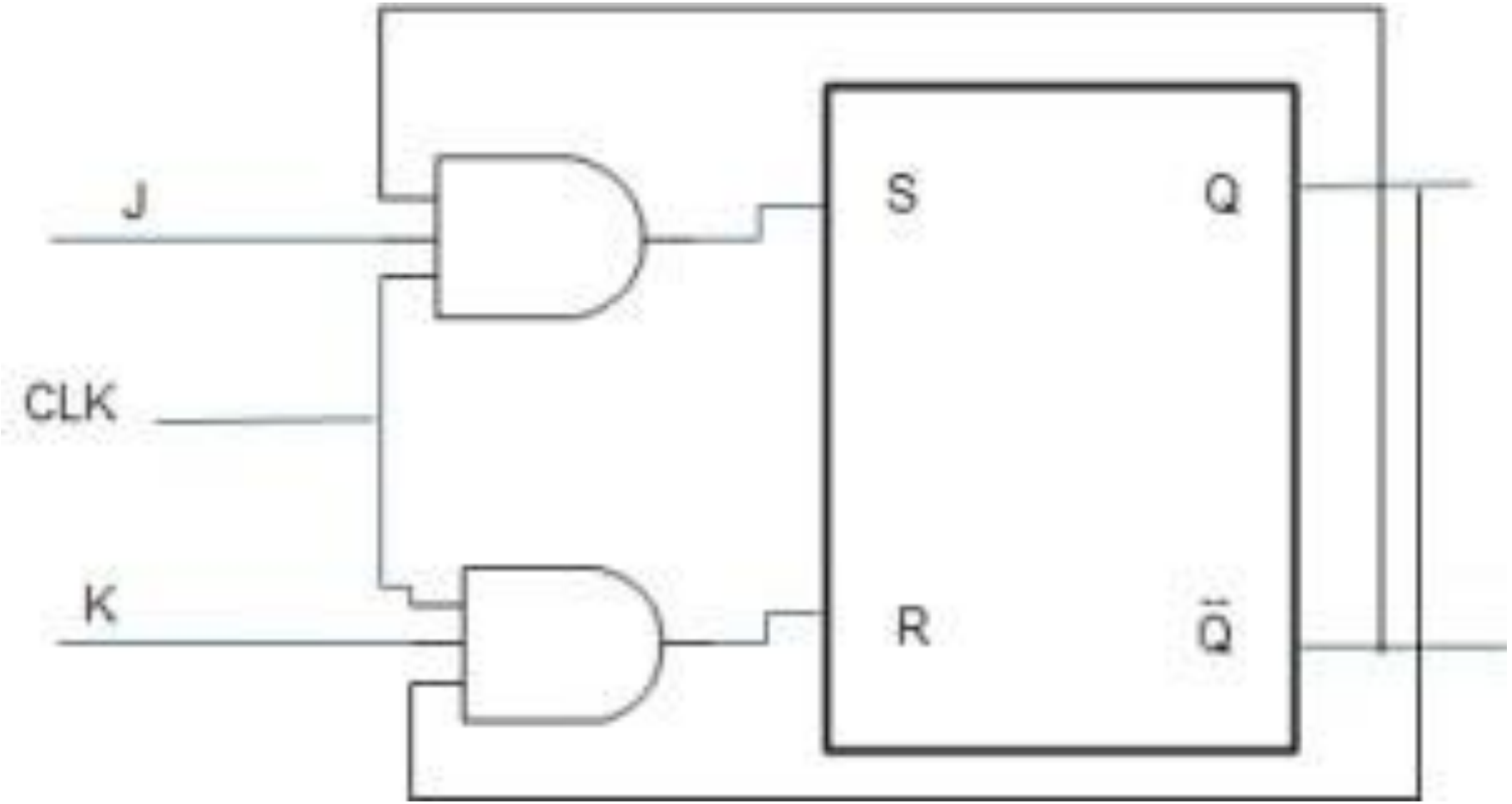
fig. Modified Memory Cell

Flip Flops (The set–reset or SR Latch)

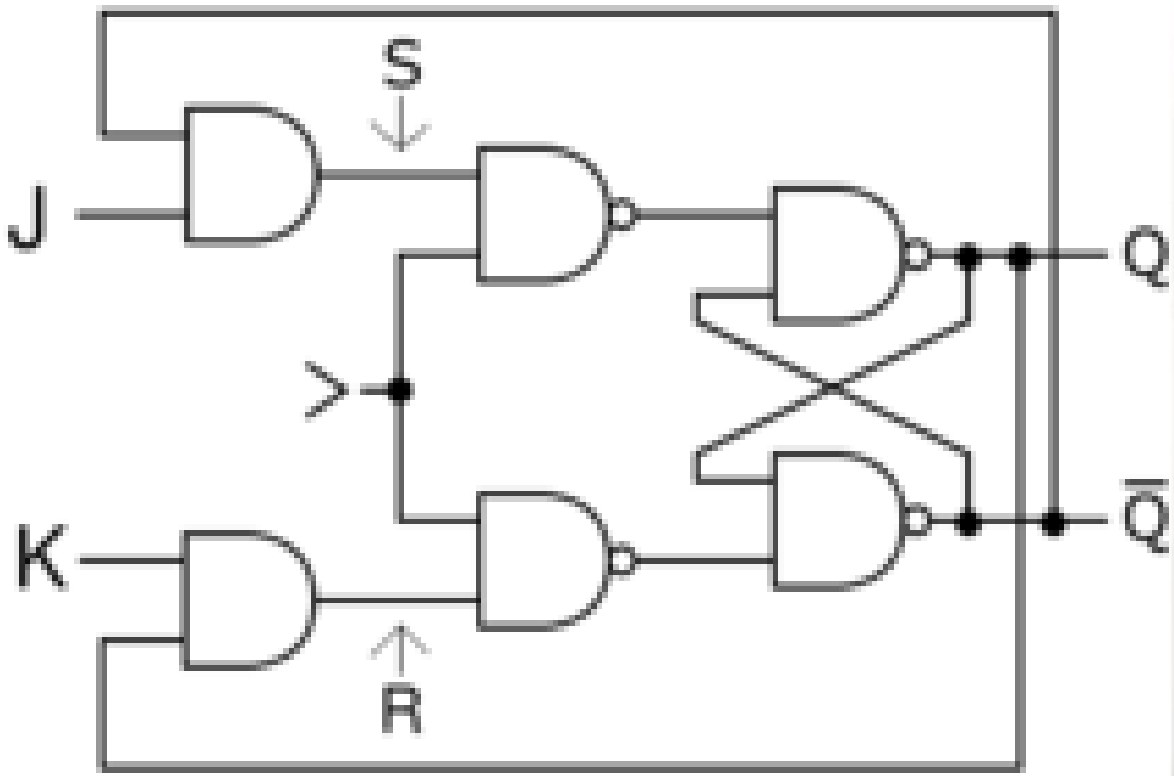
S	R	C(Clock)	Q(OUTPUT)
0	0	1	Q(Previous State)
0	1	1	0
1	0	1	1
1	**1**	**1**	***(indeterminate or ambiguous)**
X	X	0	Q(Previous State)



JK Flip Flop using SR Flip Flop



JK(Jack Kilby) Flip Flop



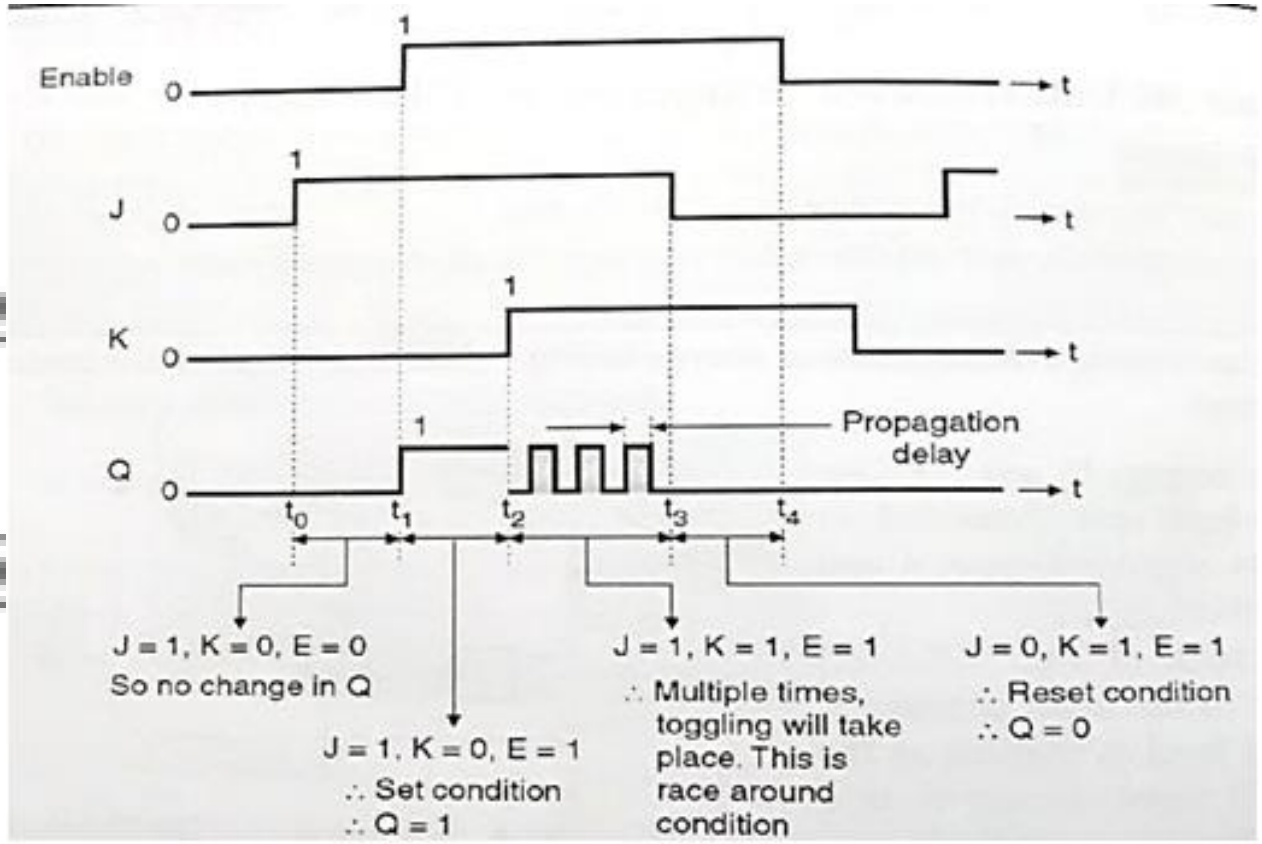
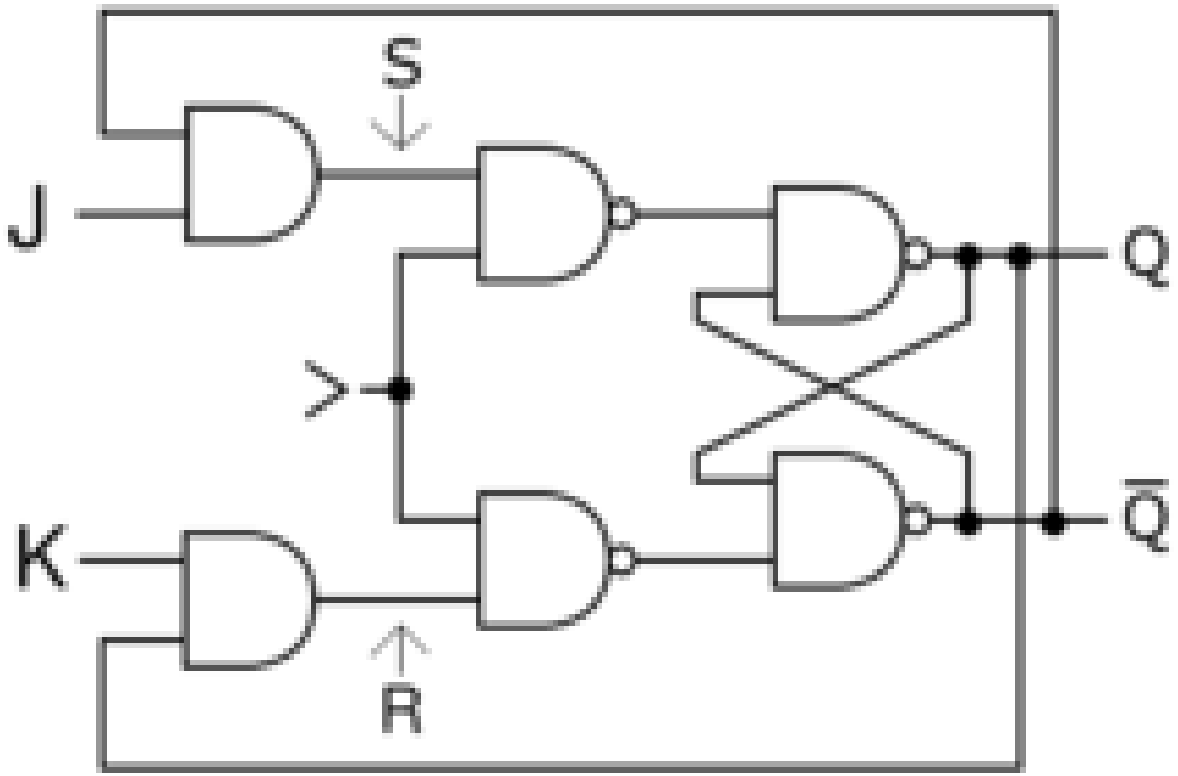
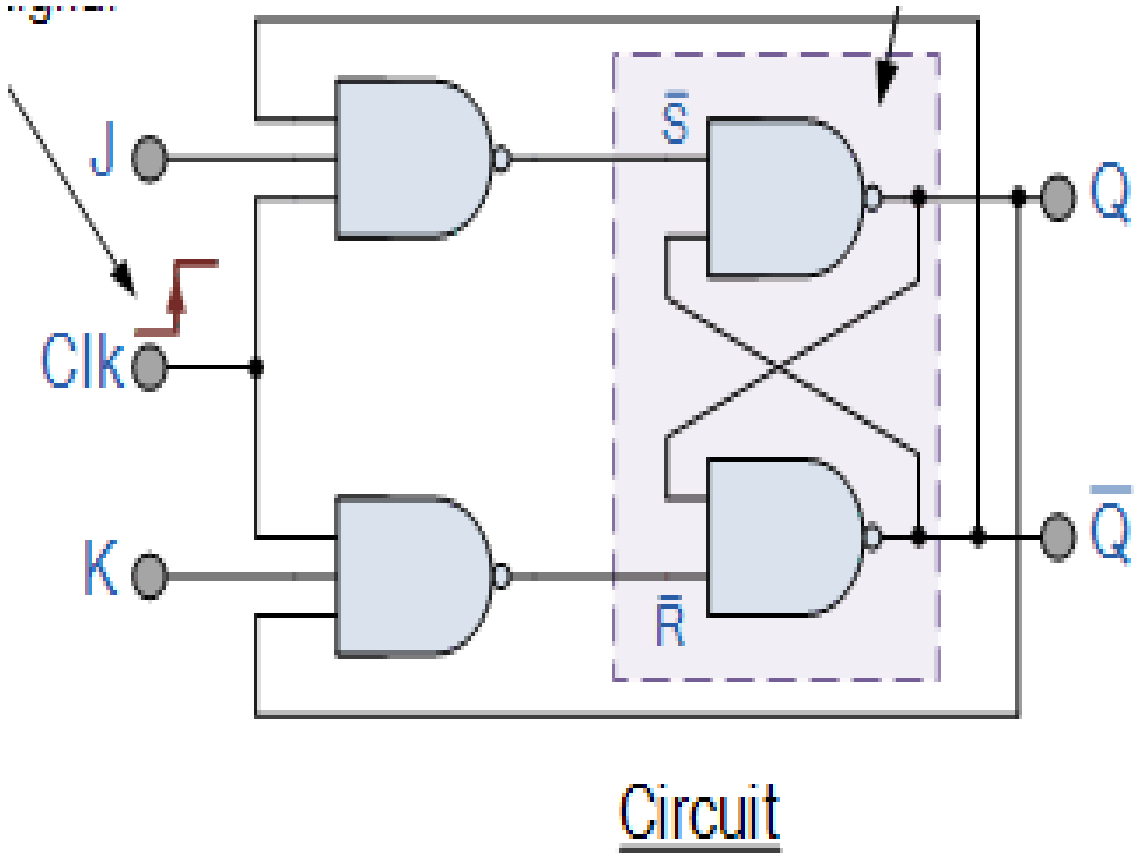


Fig5. Waveform showing Race around condition.

JK(Jack Kilby) Flip Flop





Circuit

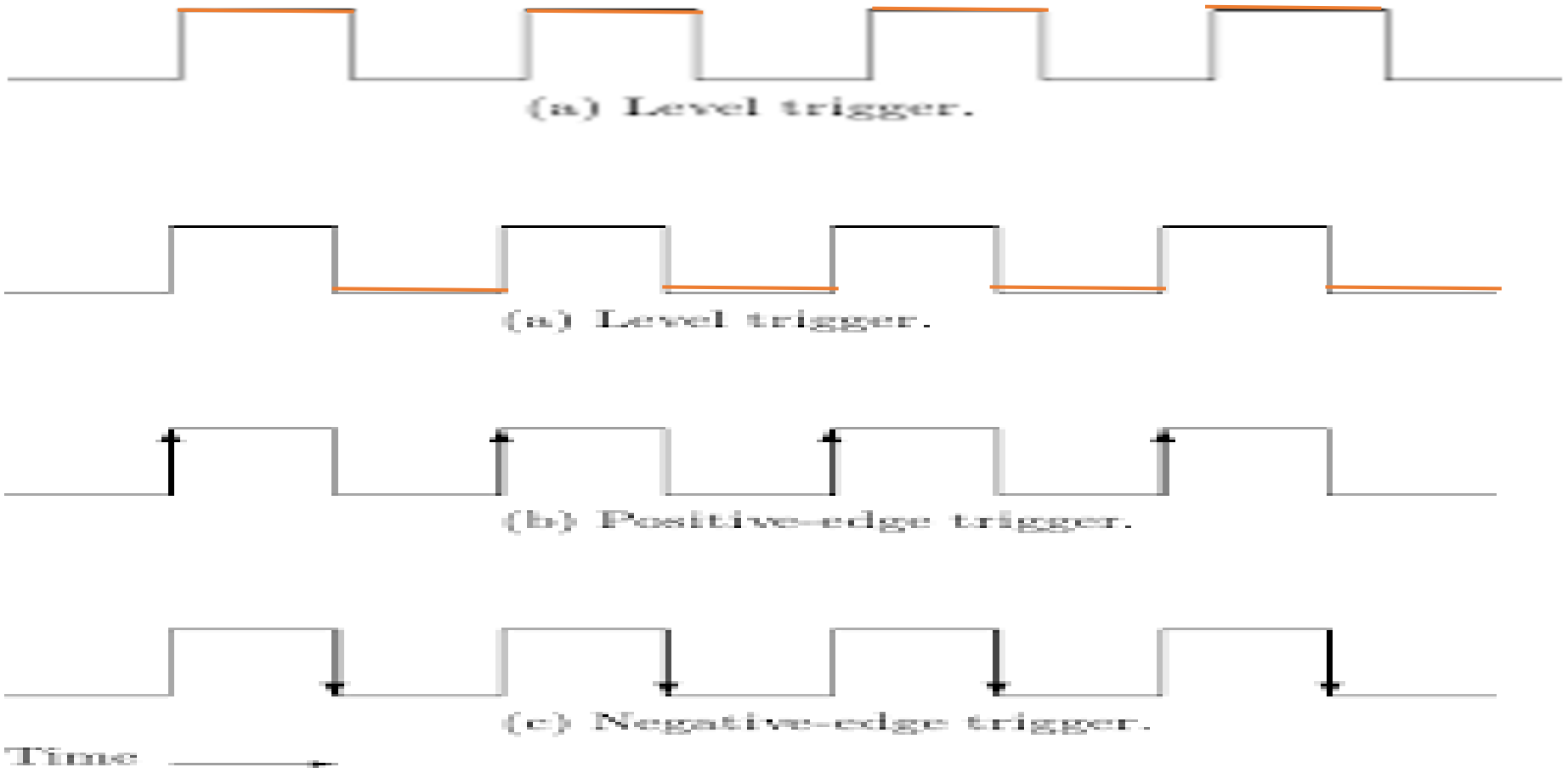
Truth Table of JK Flip Flop

Truth Table

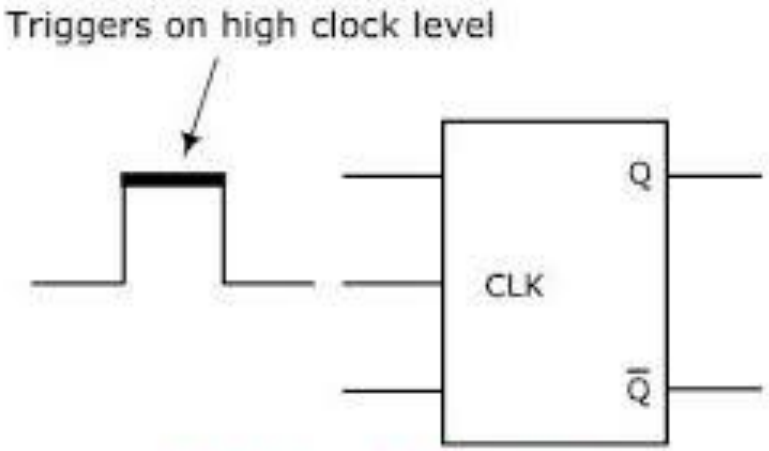
Trigger	Inputs		Output				Inference
			Present State		Next State		
CLK	J	K	Q	$\bar{Q}$	Q	$\bar{Q}$	
↑	x	x	-		-		Latched
↑	0	0	0	1	0	1	No Change
↑			1	0	1	0	
↑	0	1	0	1	0	1	Reset
↑			1	0	0	1	
↑	1	0	0	1	1	0	Set
↑			1	0	1	0	
↑	1	1	0	1	1	0	Multiple Toggle
↑			1	0	0	1	

Table I Truth table for positive-edge triggered JK flip-flop

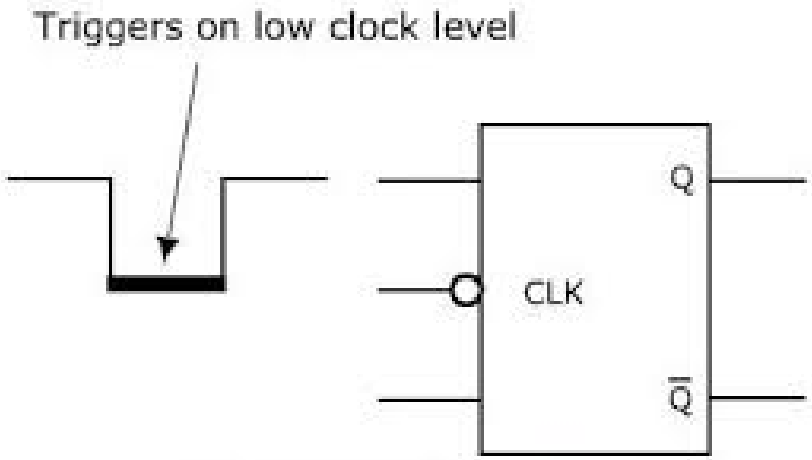
Level and Edge Triggering



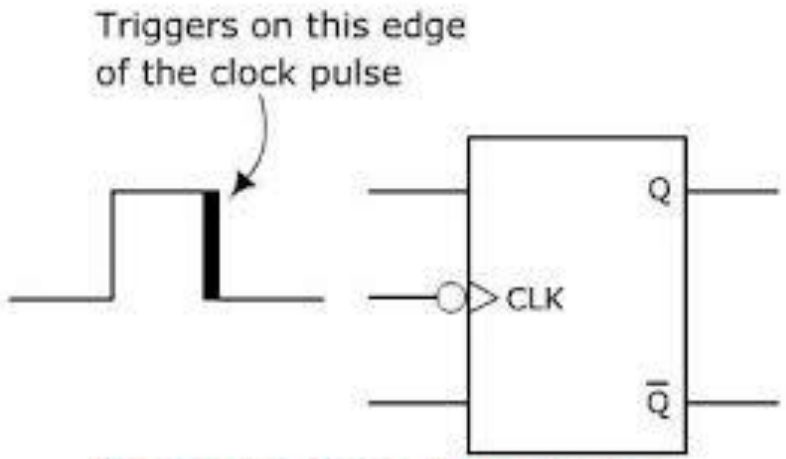
Level and Edge Triggering



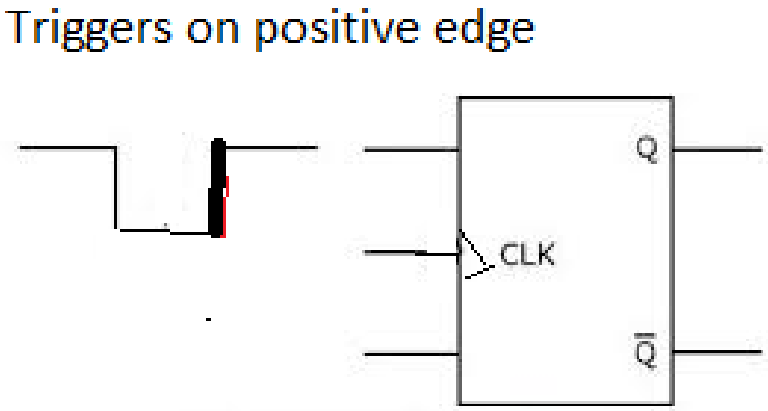
High Level Triggering



Low Level Triggering

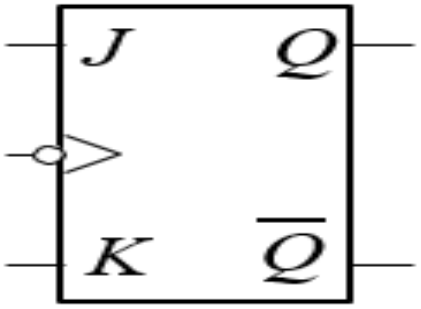


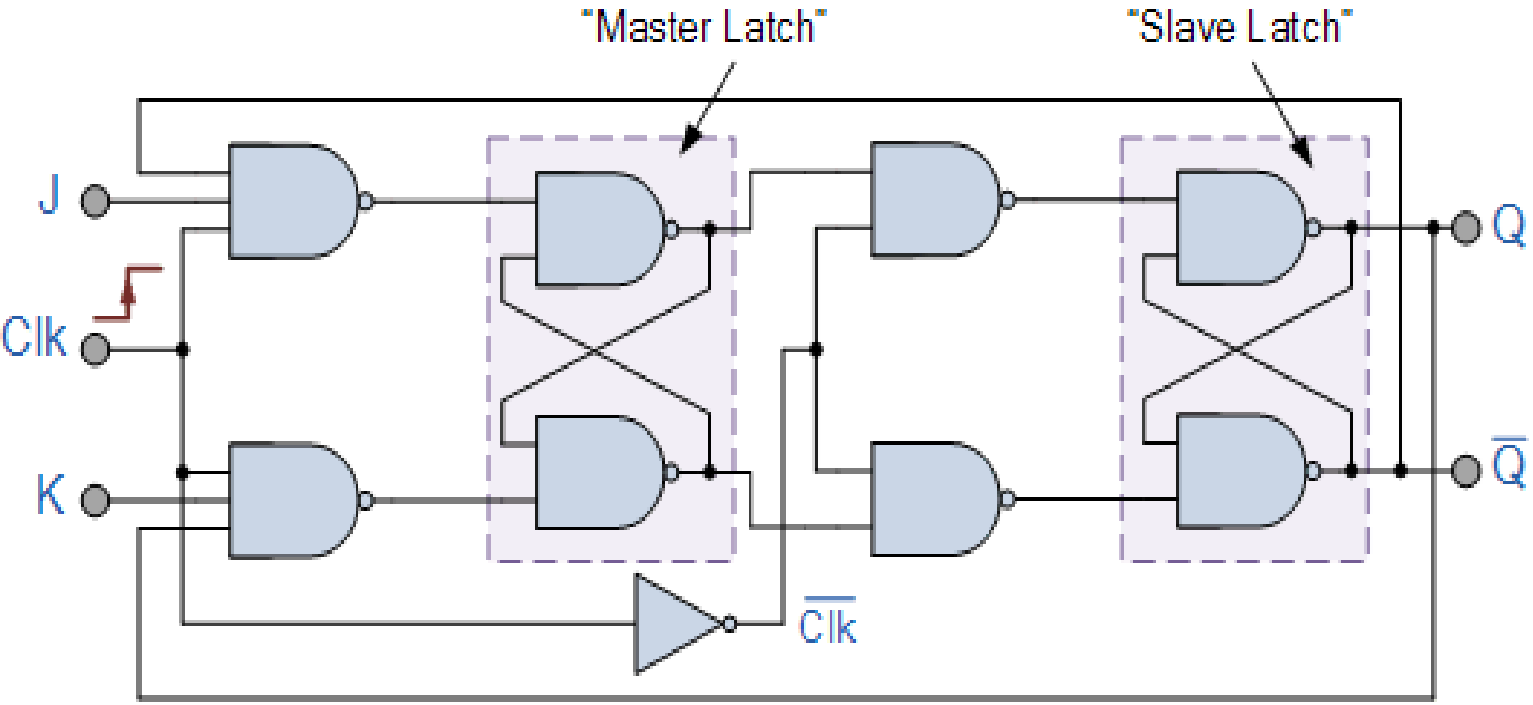
Negative Edge Triggering



Positive Edge triggering

Master Slave JK FF

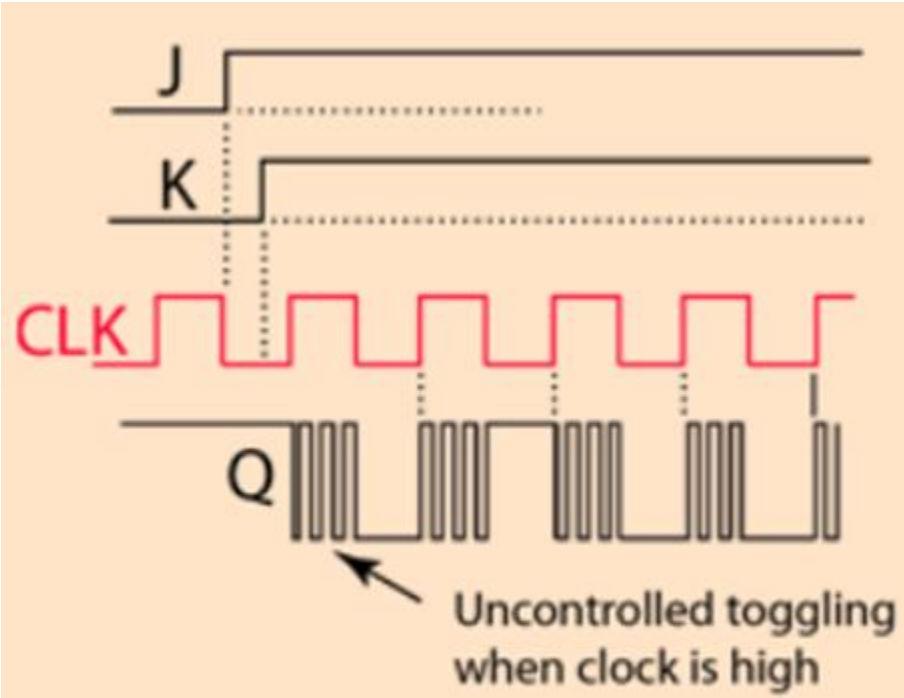




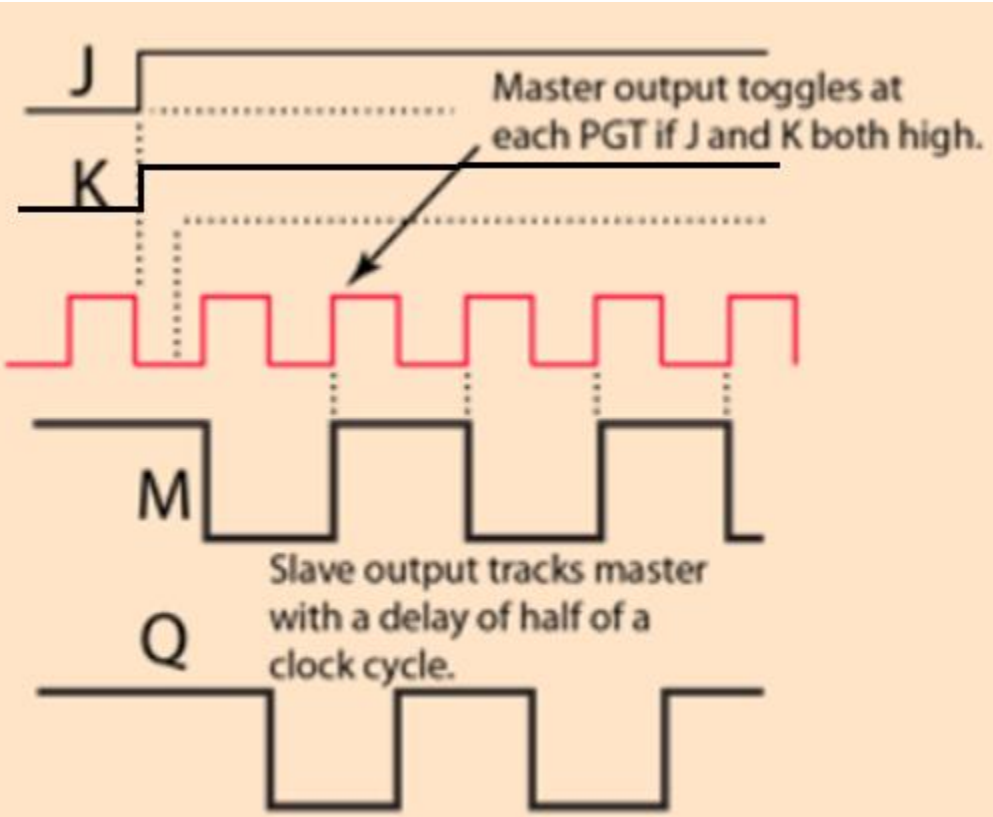
J	K	Q_n	Q_{n+1}	description
0	0	0	0	hold
0	0	1	1	
0	1	0	0	clear (reset)
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	**1**	**toggle**
1	1	1	**0**	

Comparison of JK and MS JK flip-flop

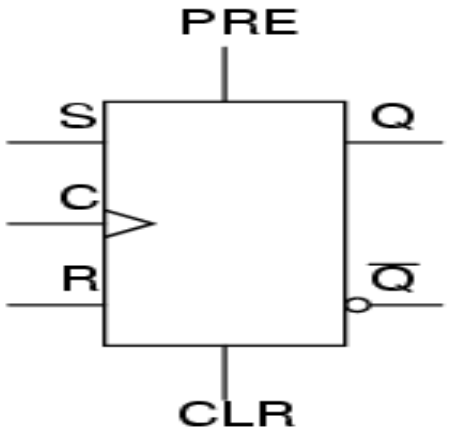
J K Flip Flop

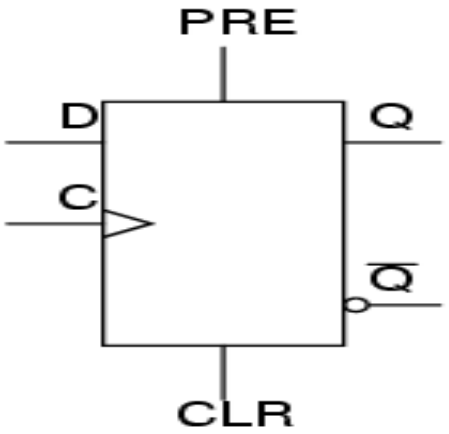


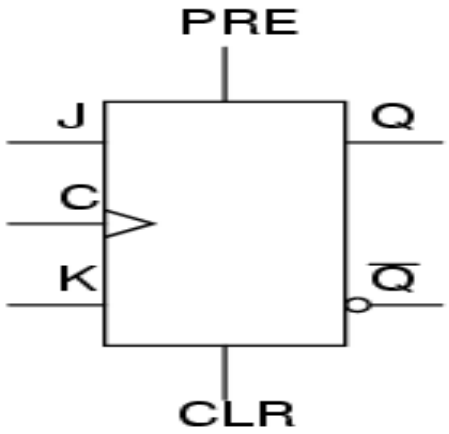
MS JK Flip Flop



Flip Flops (An SR Flip-flop with Preset and Clear)



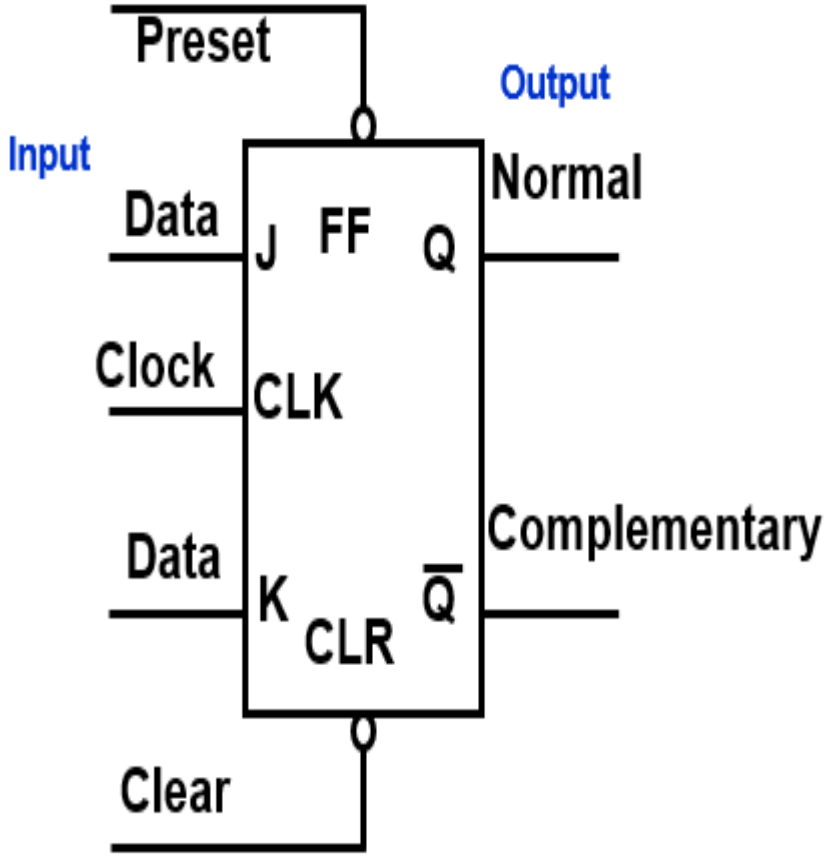




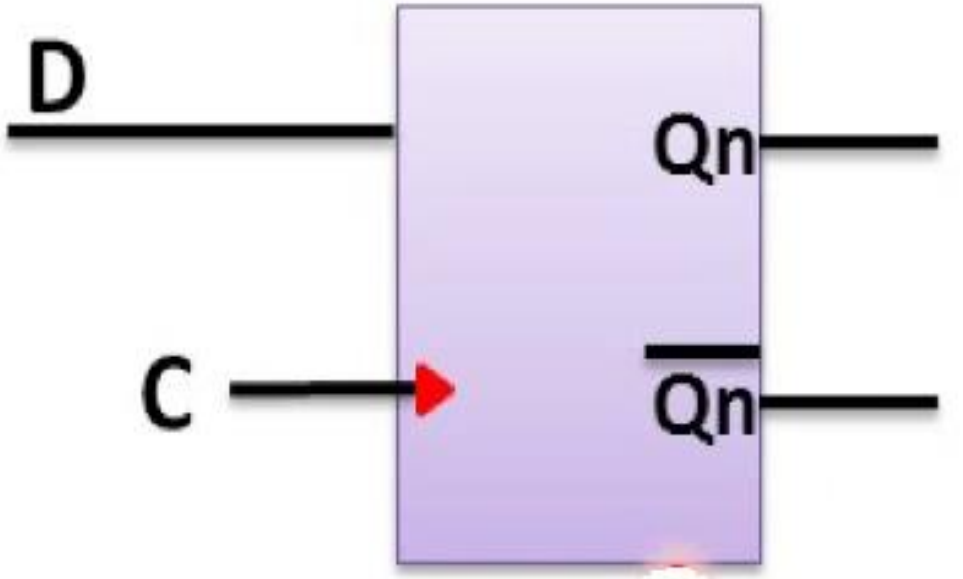
PRE	Clr	C	Q	Operation Performed
1	1	1	Qn+1(SR)	Normal SR Flip flop
1	0	0	0	Clear(reset)
0	1	0	1	Preset(Set)

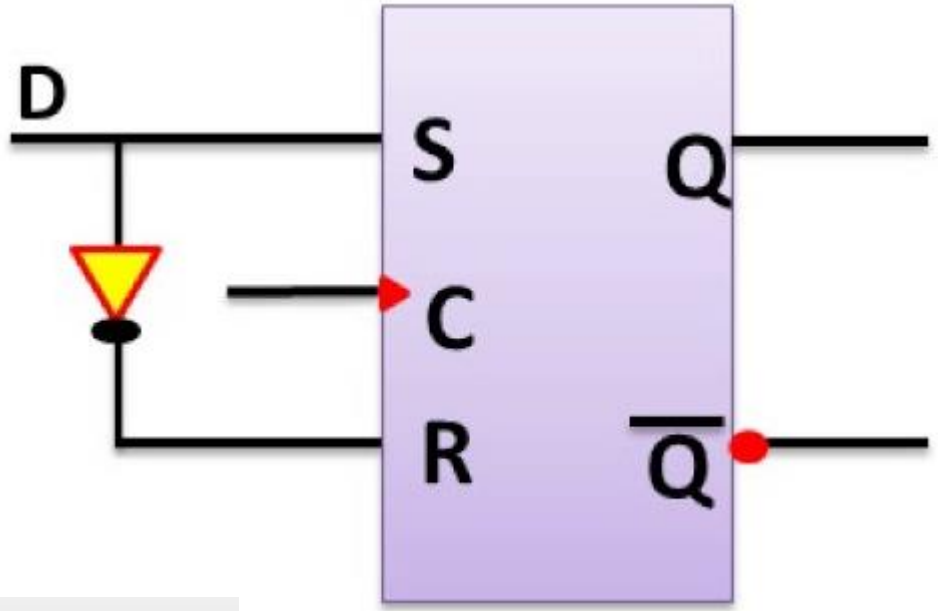
Usage of MS-JK Flip Flop: IC 7476

- 7476 uses the entire pulse to transfer data from J & K data inputs to Q & Q' outputs.
- It is a Universal Flip Flop. It has all features of other FF.
- When being used only in the toggle mode, commonly called a T Flip Flop.
- CMOS examples: 74HC76, 74AC109, 4027.
- Most commonly used as counters.



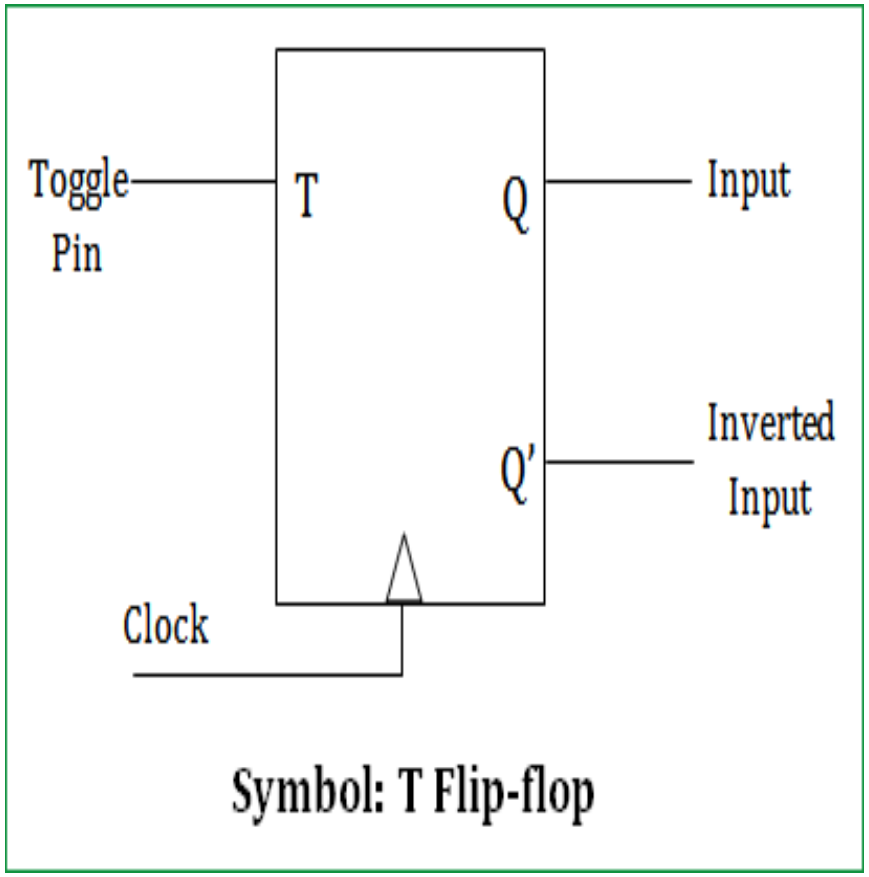
D Flip Flop: Truth Table and Symbol

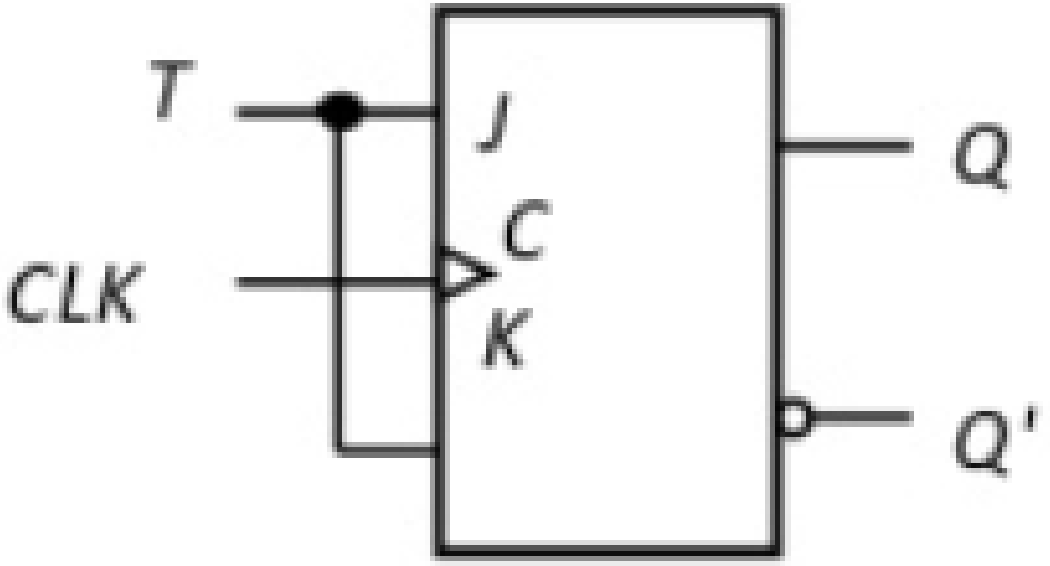




D	Q(n+1)
X	Q(n)
0	0
1	1

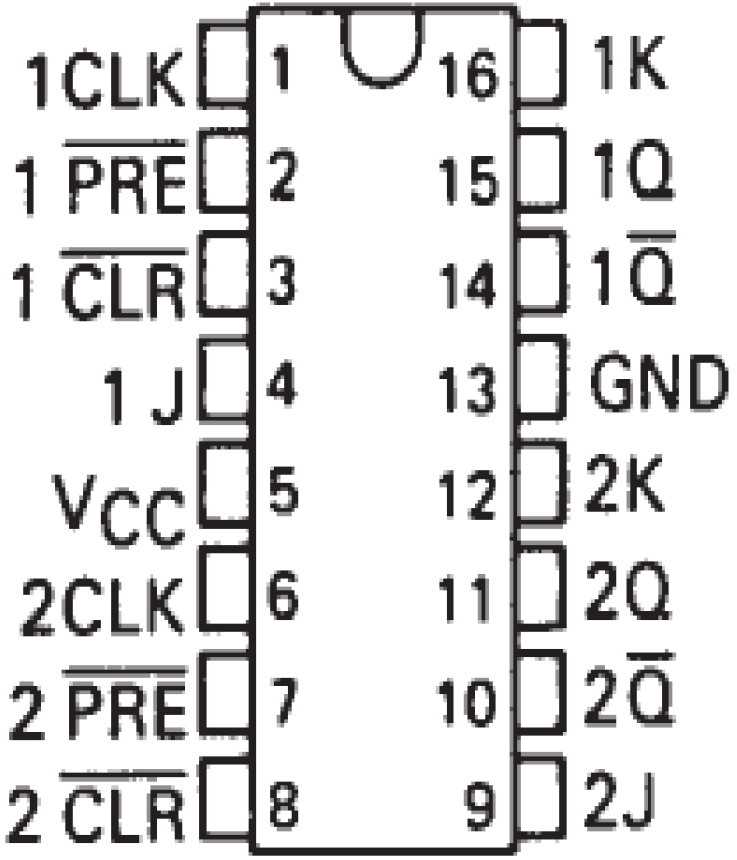
T Flip Flop: Truth Table and Symbol





T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

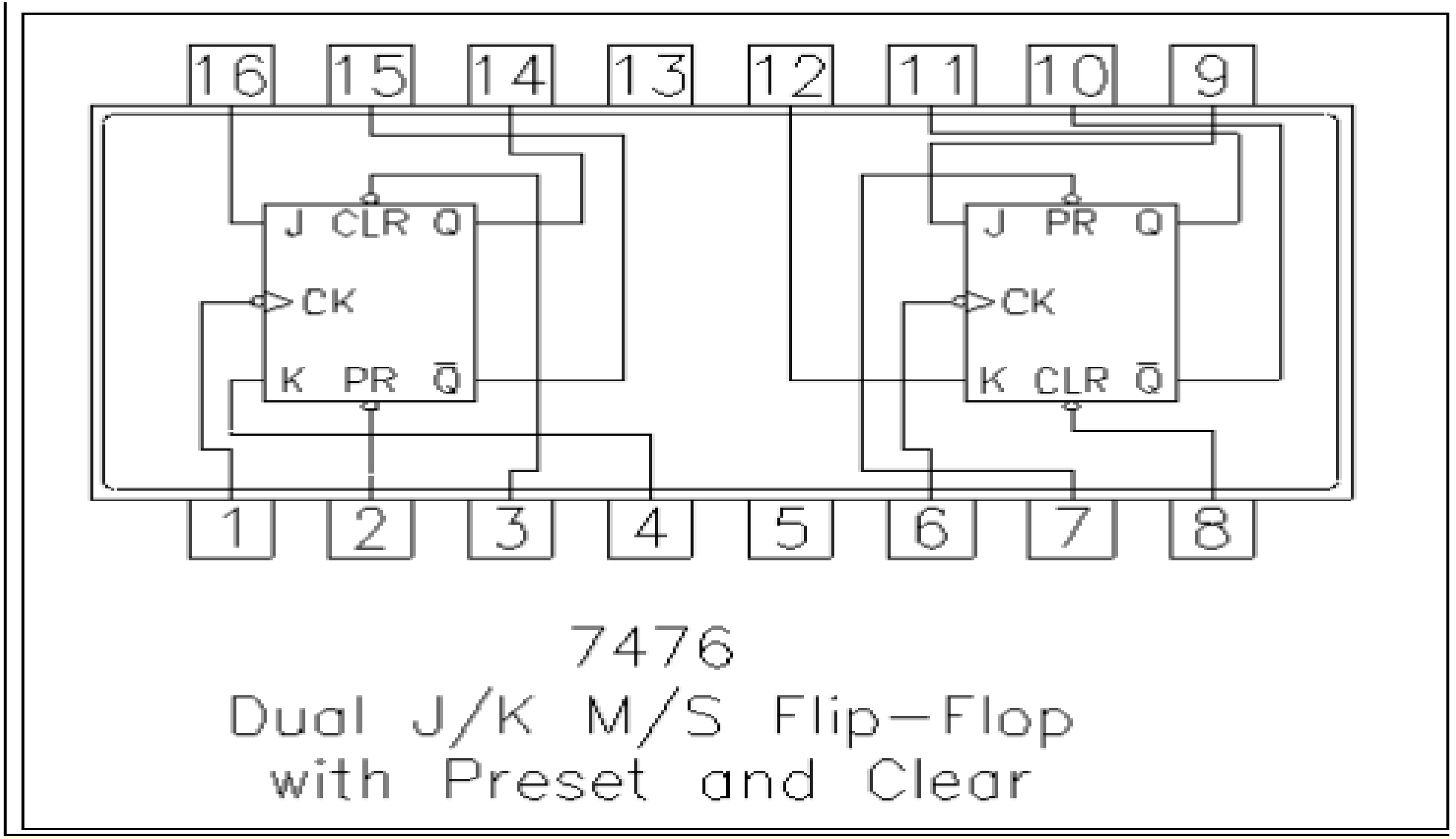
Study of IC7476(DUAL J-K FLIP-FLOPS WITH PRESET AND CLEAR)



FUNCTION TABLE

INPUTS					OUTPUTS	
$\overline{PRE}$	$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H†	H†
H	H	⎍	L	L	Q_0	$\overline{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	TOGGLE	

Study of IC 7476



Introduction to Flip-Flop Excitation Table

- We have described flip-flop behavior by using a characterizes table which gives the new state as a function of the present state and inputs.
- What we need if we know the state and inputs and was to determine the new state.
- Sometimes, though, we know the state we are in and the state we wish to be in, and need to know the inputs required.
- For this purpose, we can consult an excitation table

Important Definition

- **Characteristic Table:** The truth table of flip flop is also referred to as characteristic table and it specifies the operational characterizes of flip-flop.
- **Excitation Table:** In design of sequential circuit, present state and next state of the circuit are specifies.
- **Example:** The output of SR flip-flop before clock pulse is Qn = 0 and it is desired that the output does not change when clock pulse is applied.

TRUTH TABLE

S	R	Qn+1
0	0	Qn =0
0	1	0
1	0	1
1	1	*

EXCITATION TABLE

Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation Table and Truth Table of Flip Flops

Excitation Table of D flip-flop

D	Q_{n+1}
0	0
1	1

Truth Table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table

Excitation Table of JK flip-flop

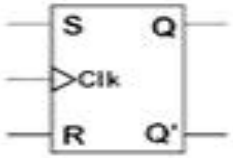
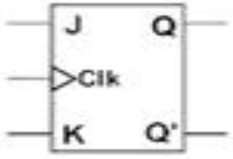
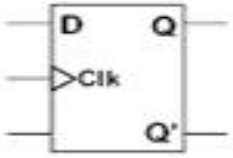
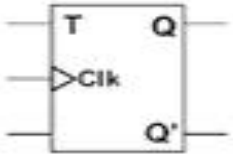
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q_n'

Truth Table

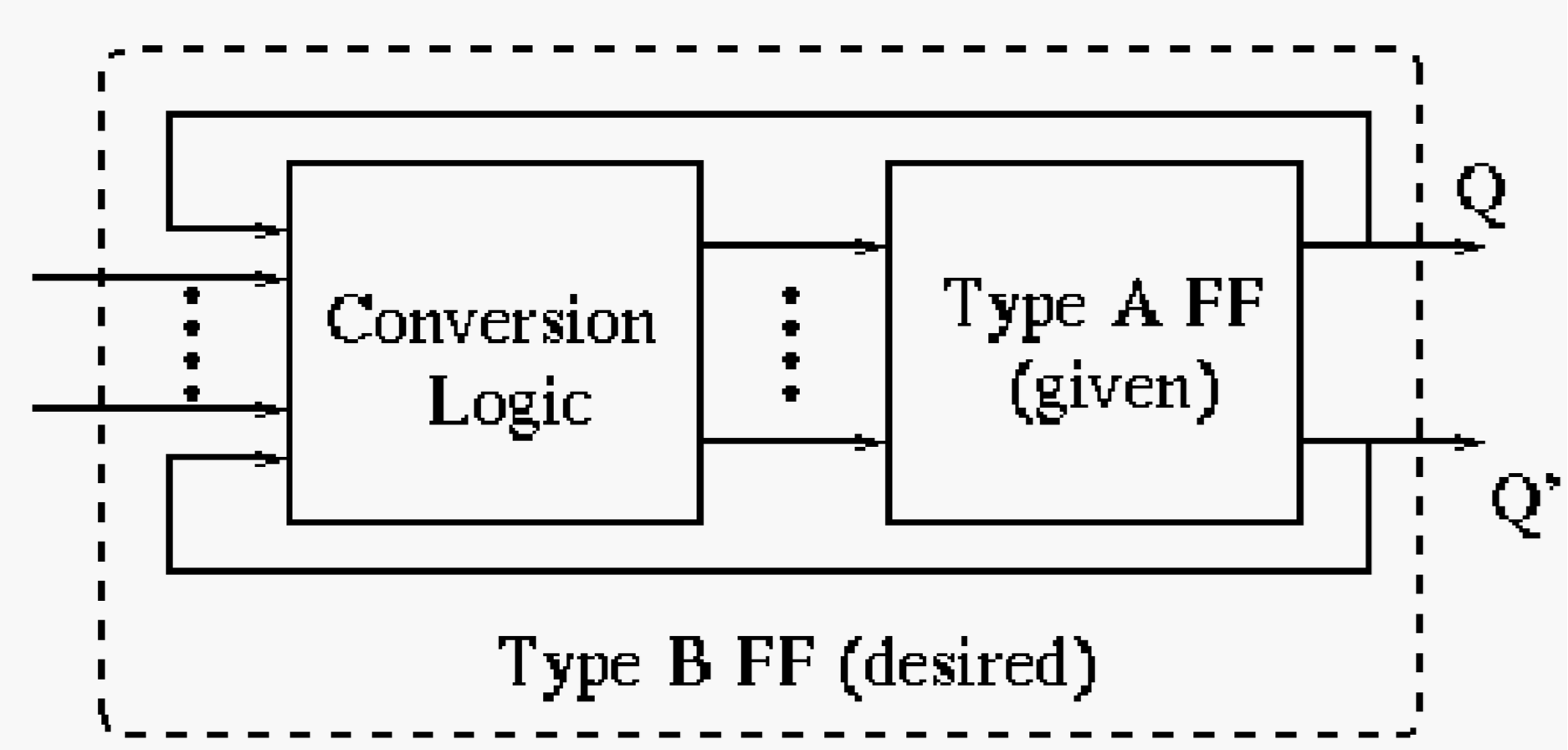
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation Table

Excitation Table of Flip Flops

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		S R $Q_{(next)}$ 0 0 Q 0 1 0 1 0 1 1 1 ?	$Q_{(next)} = S + R'Q$ $SR = 0$	Q $Q_{(next)}$ S R 0 0 0 X 0 1 1 0 1 0 0 1 1 1 X 0
JK		J K $Q_{(next)}$ 0 0 Q 0 1 0 1 0 1 1 1 Q'	$Q_{(next)} = JQ' + K'Q$	Q $Q_{(next)}$ J K 0 0 0 X 0 1 1 X 1 0 X 1 1 1 X 0
D		D $Q_{(next)}$ 0 0 1 1	$Q_{(next)} = D$	Q $Q_{(next)}$ D 0 0 0 0 1 1 1 0 0 1 1 1
T		T $Q_{(next)}$ 0 Q 1 Q'	$Q_{(next)} = TQ' + T'Q$	Q $Q_{(next)}$ T 0 0 0 0 1 1 1 0 1 1 1 0

Conversion from one type to another type of Flip Flop



Convert SR to JK FF

- Write the Excitation table of Available F/F(SR)
- Write the Truth table of Required F/F (JK)
- Prepare a table of inputs of required flip flop along with present state Qn (e.g. J,K and Qn) showing all the possible states
- Add the column of next state Qn+1 considering inputs (J,K) and Qn
- Add the column of inputs of available flip flops considering Qn and Qn+1. Use excitation table of available F/F
- Solve k maps for inputs of available F/F (S,R) using inputs of required F/F and Qn as inputs (J, K, Qn)

J	K	Qn	Qn+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

J \ K, Qn	00	01	11	10
0	0	X	0	0
1	1	X	0	1

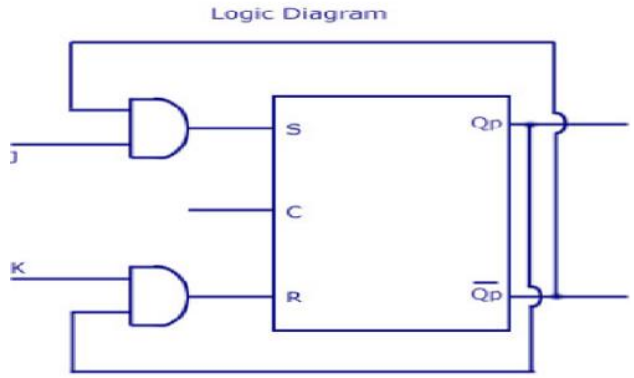
S

S = J.Qn'

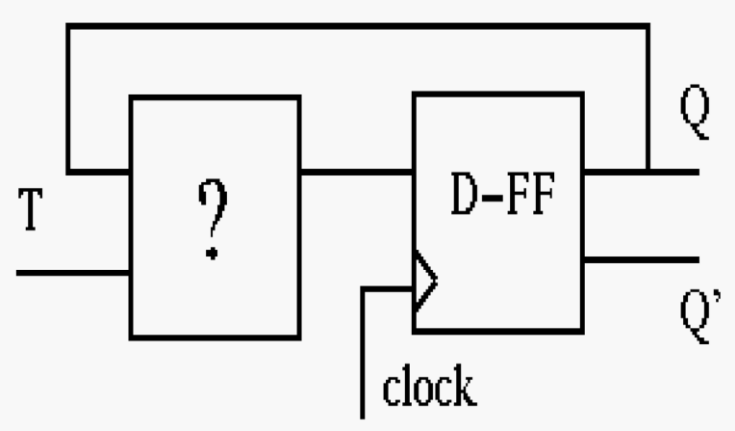
J \ K, Qn	00	01	11	10
X	X	0	1	X
0	0	0	1	0

R

R = K.Qn

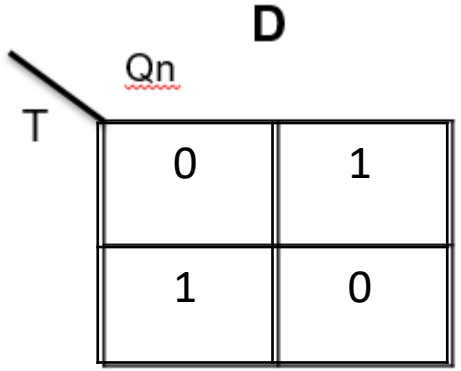


Convert D FF to T FF

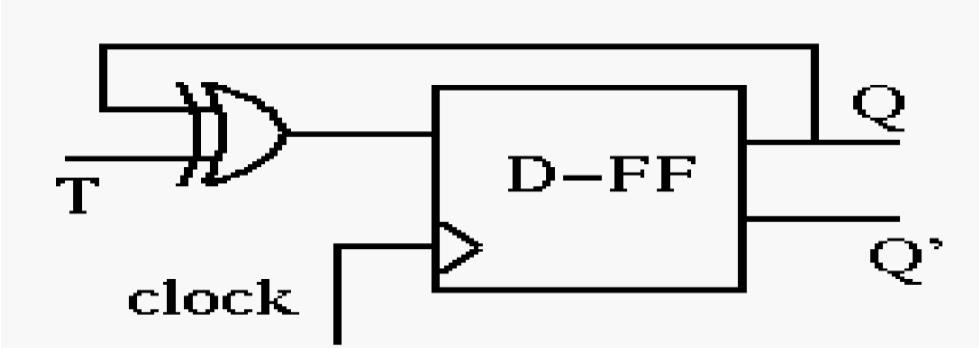


- Consider the excitation table of T and D Flip flops.
- Write Down Excitation Table of T, Qn and Qn+1, D.
- For the K-map, consider T and Qn as Input and D as output.

T	Qn	Qn+1	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0



$$D = T'Q + TQ' = T \oplus Q$$



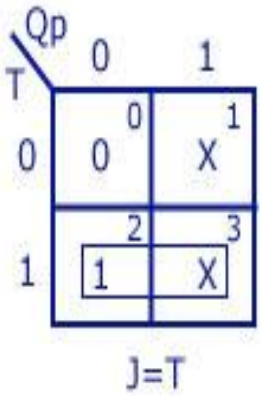
Convert JK FF to T FF

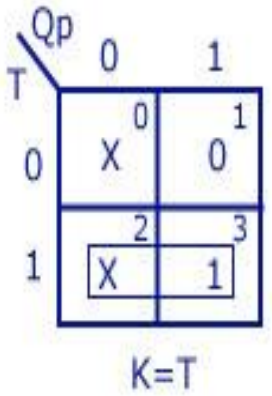
J-K Flip Flop to T Flip Flop

Conversion Table

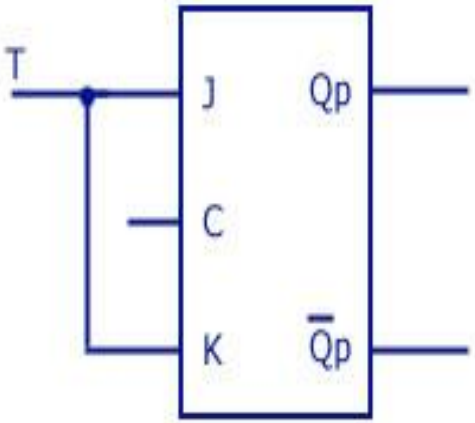
T Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

K-maps





Logic Diagram



www.CircuitsToday.com

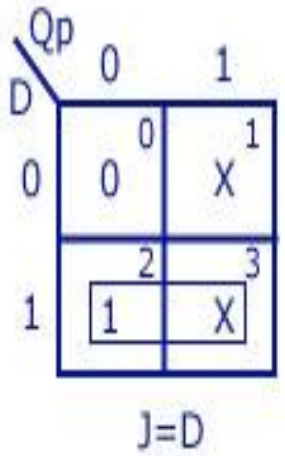
Convert JK FF to D FF

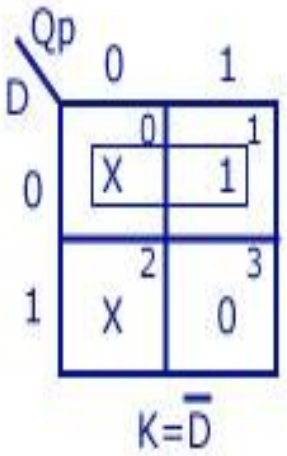
J-K Flip Flop to D Flip Flop

Conversion Table

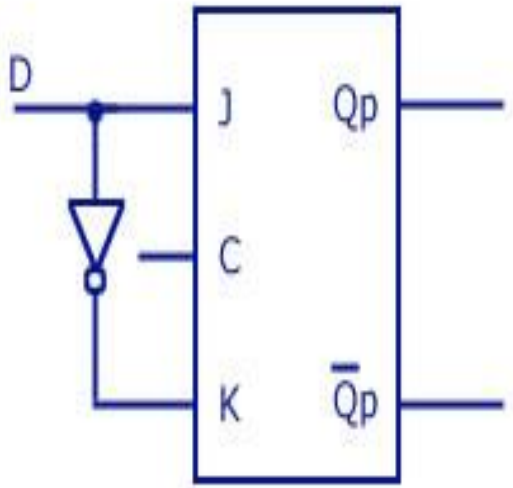
D Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

K-maps





Logic Diagram

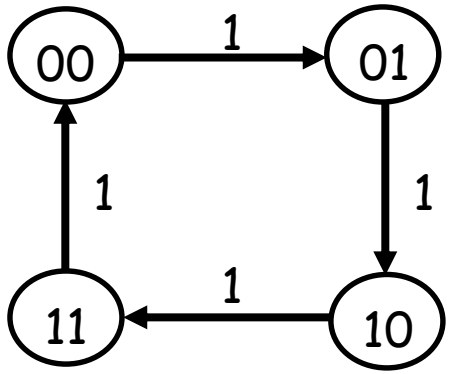


www.CircuitsToday.com

Counters

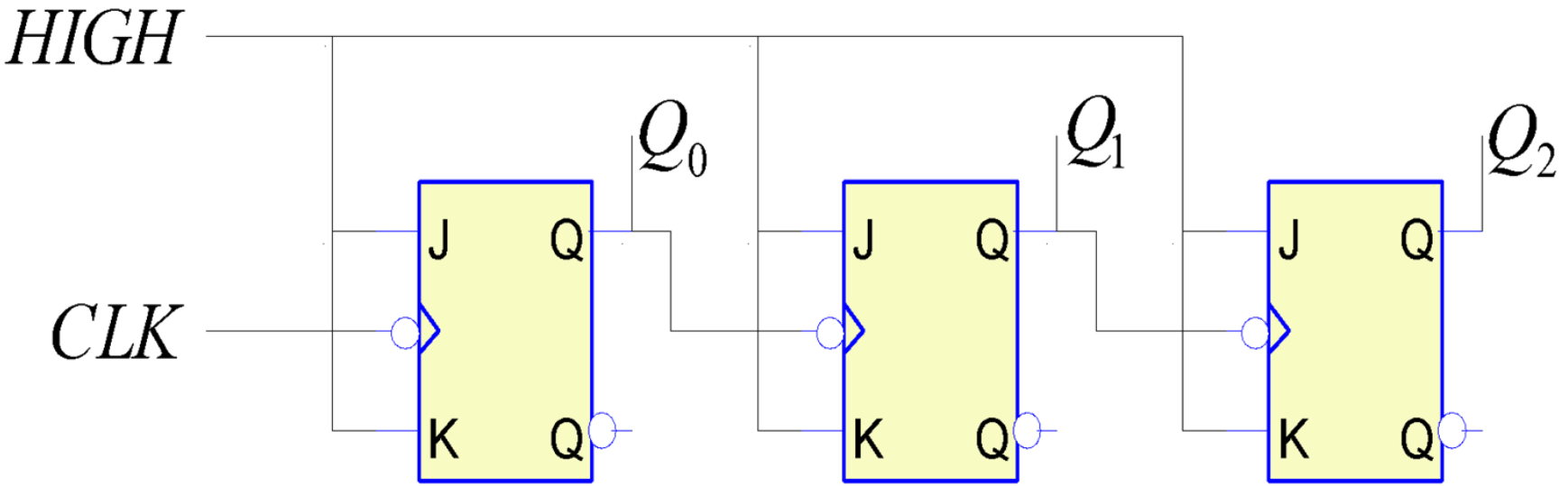
- Counters are a specific type of sequential circuit.
- The state, or the flip-flop values themselves, serves as the "output."
- The output value increases by one on each clock cycle.
- After the largest value, the output "wraps around" back to 0.

Using two bits, we'd get something like that



Asynchronous Counters

- Only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop.
- Asynchronous counters are also called *ripple-counters* because of the way the clock pulse ripples it way through the flip-flops.
- The flip-flop output transition serves as a source for triggering other flip-flops

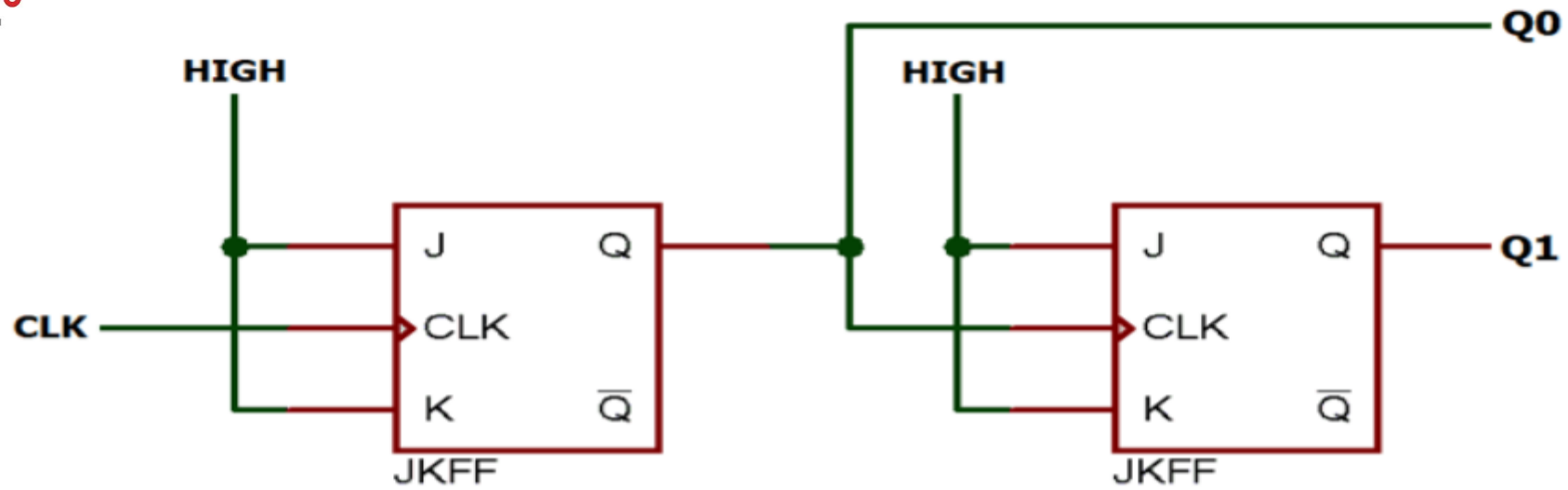


Counting in Binary

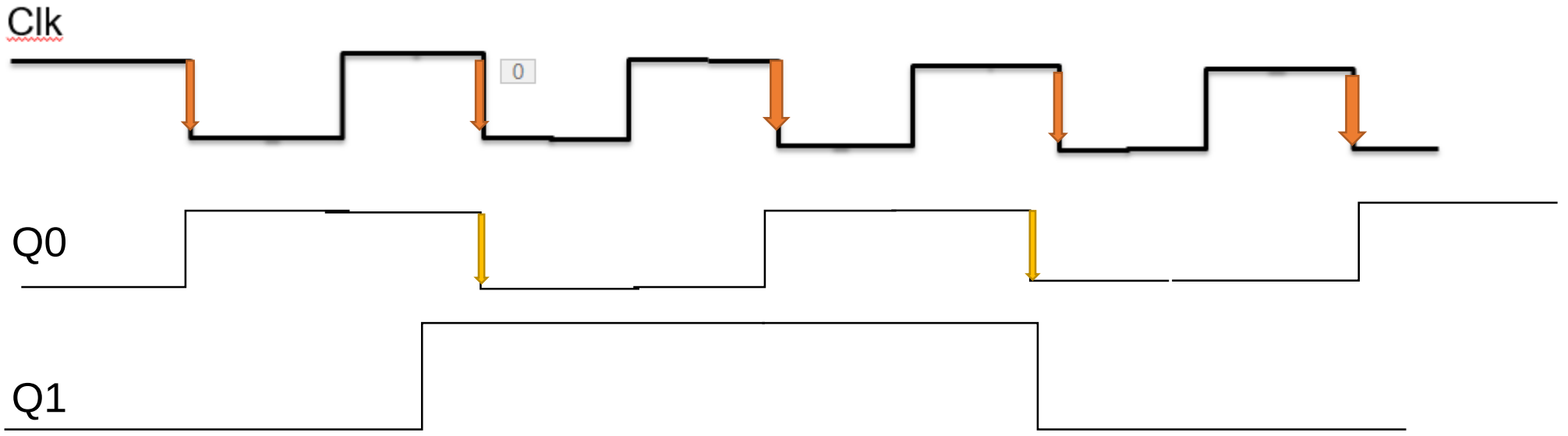
- The binary counting sequence is shown on the right
- Look at the column labelled $\mathbf{Q_0}$. It reads 0, 1, 0, 1, 0, 1, etc. If this were drawn as a timing diagram, it would look like a clock. Assume that this clock has period T
- Now look at the column labelled $\mathbf{Q_1}$. It reads 0, 0, 1, 1, 0, 0, 1, 1. It looks like a clock too! However, it stays 0 twice as long, then 1 twice as long. In fact, it looks like a clock that has a period of 2T
- Now look at the column labelled $\mathbf{Q_2}$. It reads 0, 0, 0, 0, 1, 1, 1, 1. Again, this looks like a clock, except it's going twice as slow! It has a period of 4T!

Row	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

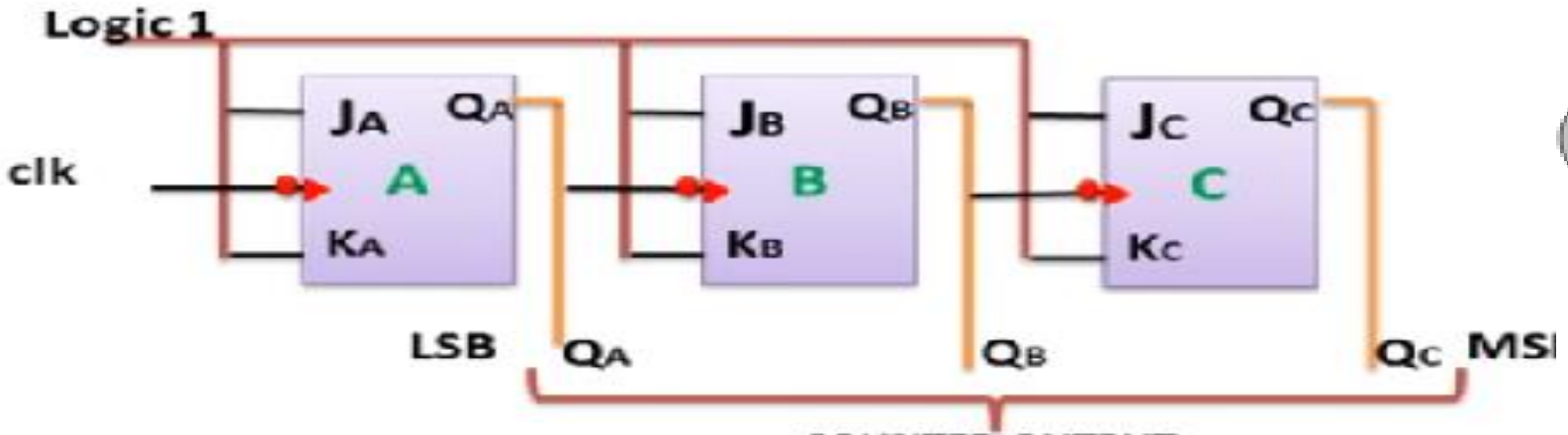
2 Bit Asynchronous UP Counter

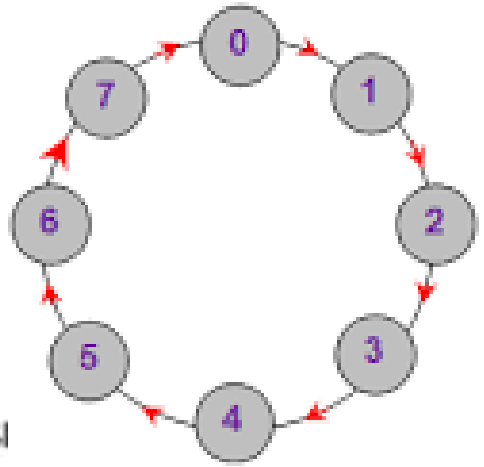


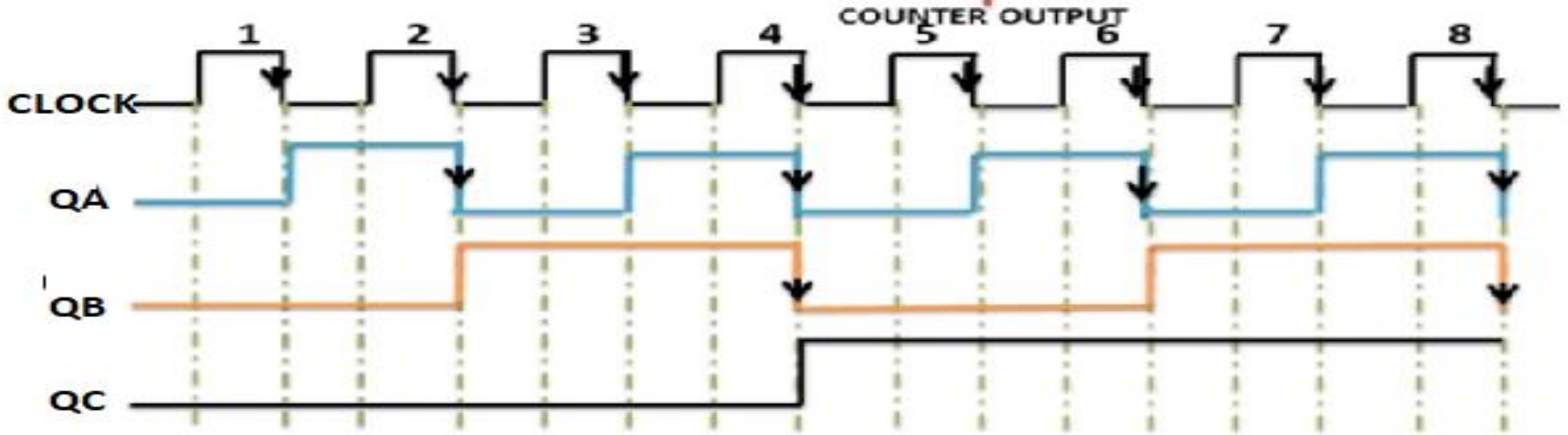
J	K	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	Qn’



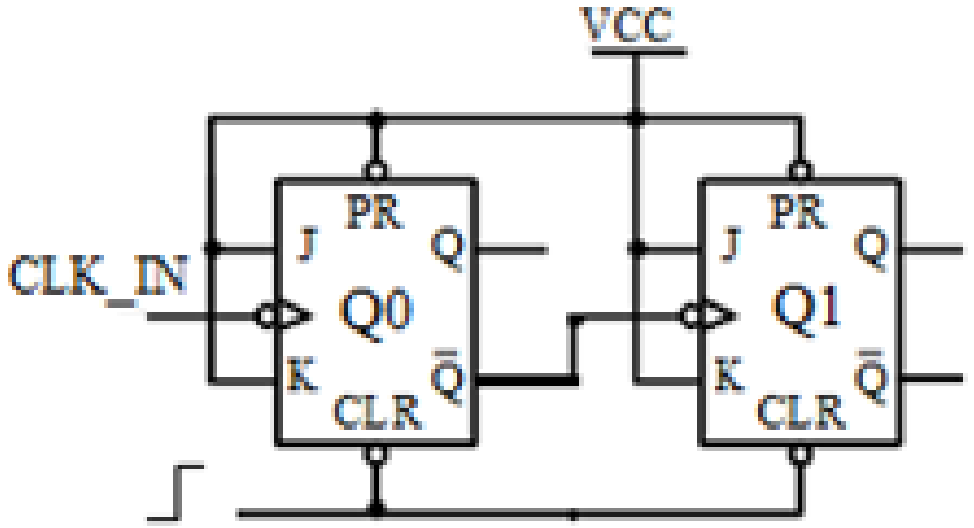
The Timing Diagram of 3 bit Asynchronous UP counter



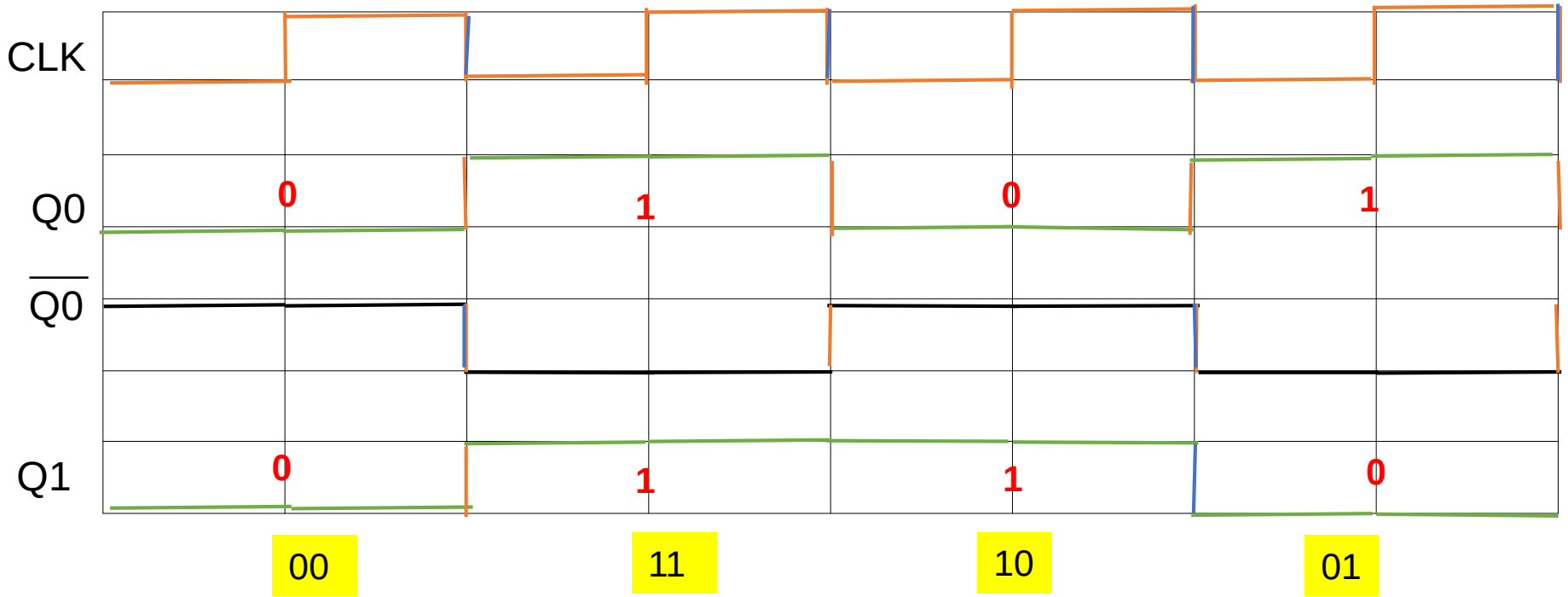




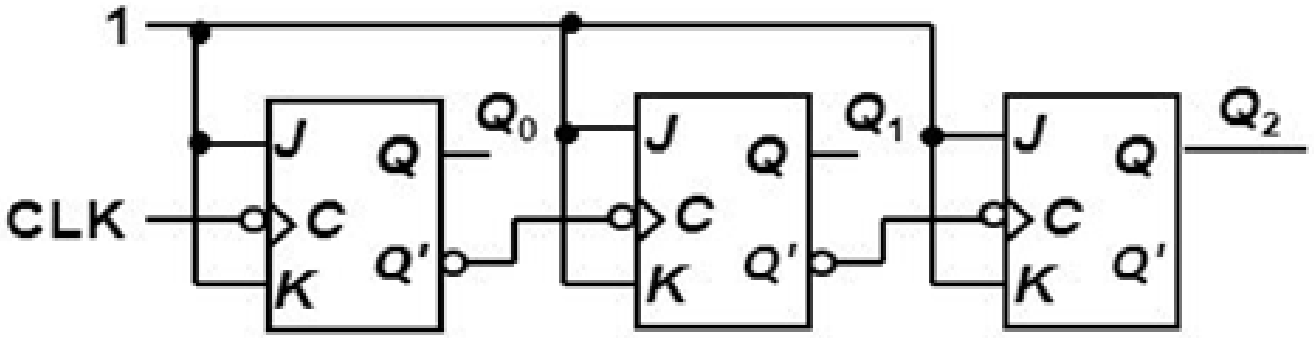
2 Bit Asynchronous Down Counter

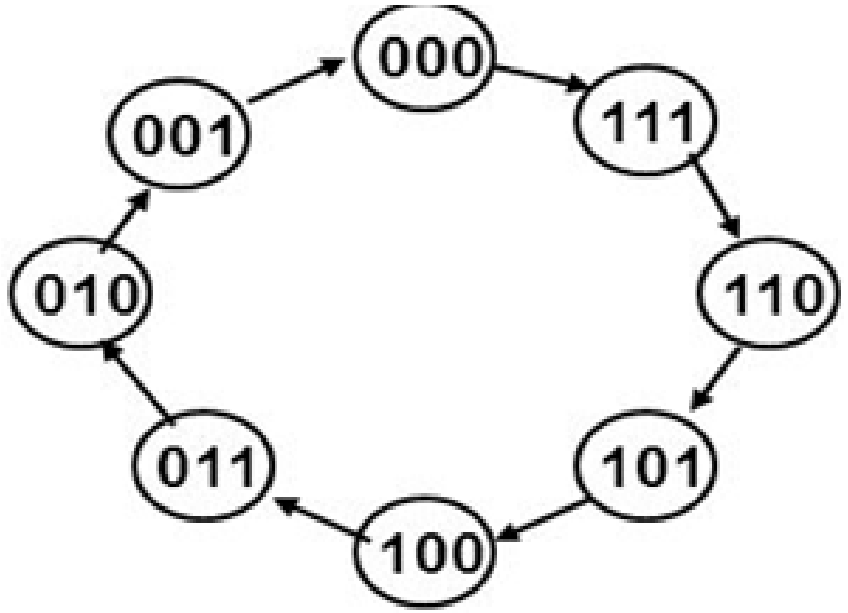


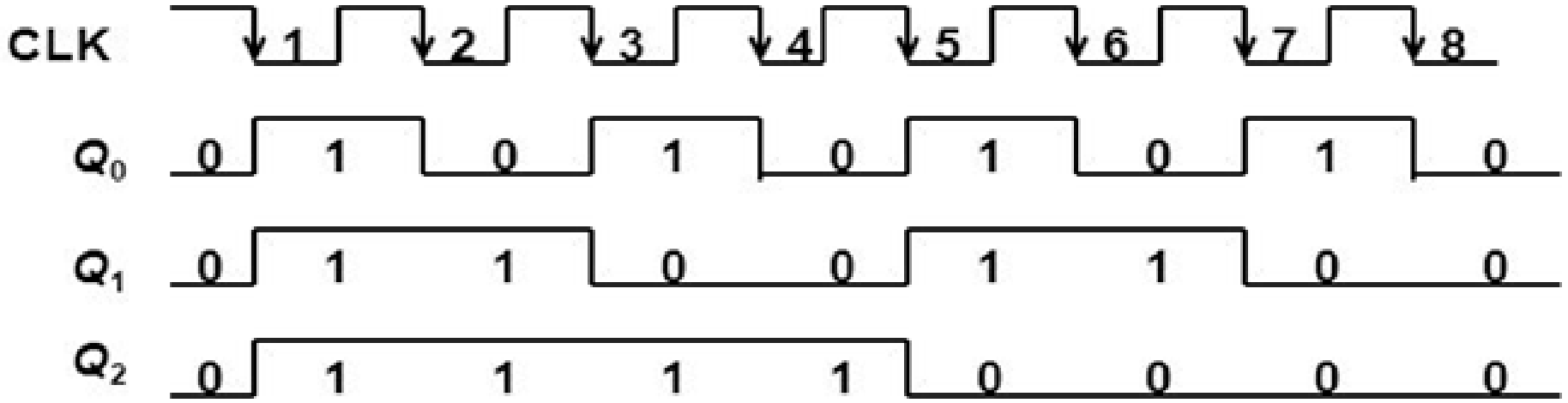
Q1	Q0
0	0
1	1
1	0
0	1



3 bit Asynchronous Down Counter







Modulus ‘n’ counter

- The number of flip-flops determines the count limit or number of states. If N flip-flops are connected:

  No. of states = 2^N where N = number of flip-flops

- The number of states used is called the MODULUS

- Modulus Counters, or simply MOD counters, are defined based on the number of states that the counter will sequence through before returning back to its original value
- For example, a Modulus-12 counter (Mod-12) would count from 0 (0000_2) to 11 (1011_2) and would require four flip-flops (2^4 = 16 states; 12 are used)

Modulus ‘n’ counter

Power-of-2 counters use all states

- Example: 4-bit Binary / Hex / Mod-16 Counter
- 0000, 0001, 0010, … 1110, 1111, 0000, 0001, … all states used

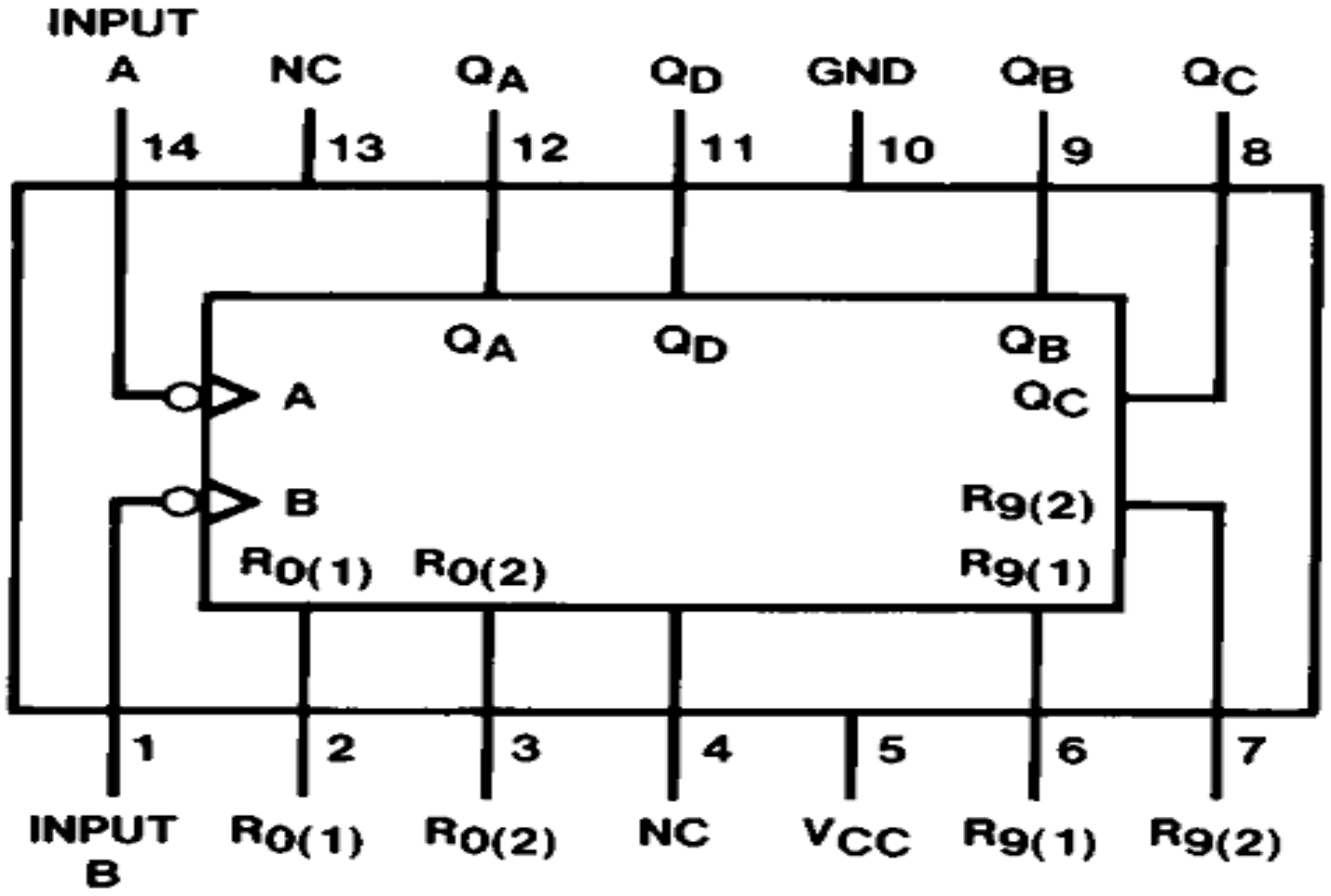
Non-power-of-2 counters have extra, unused states

- Example: 4-bit BCD / Decade / Mod-10 Counter
- 0000, 0001, 0010, … 1000, 1001, 0000, 0001, … six unused states

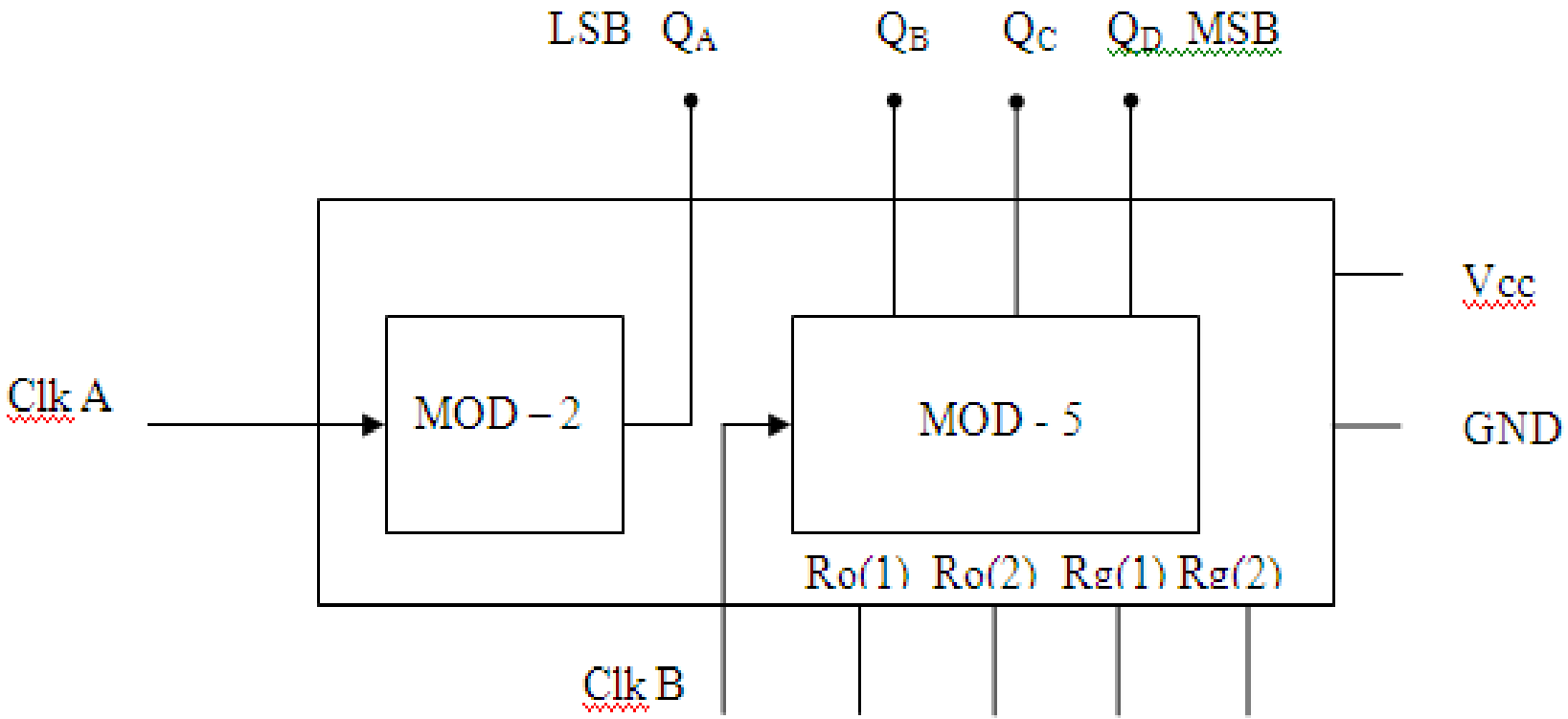
Feature of IC 7490 MOD

- IC 7490 is a TTL MSI (medium scale integration) decade counter.
- It is an asynchronous counter
- It contains 4 master slave flip flops internally connected to provide MOD-2 i.e. divide by 2 and MOD-5 i.e. divide by 5 counters.
- MOD-2 and Mod-5 counters can be used independently or in cascading.
- It is a 4-bit ripple type decade counter.
- It is also called as Divide by 10 counter as frequency at the MSB (QD) is 1/10*fclk

Pin Out of 7490



Internal Diagram of IC 7490



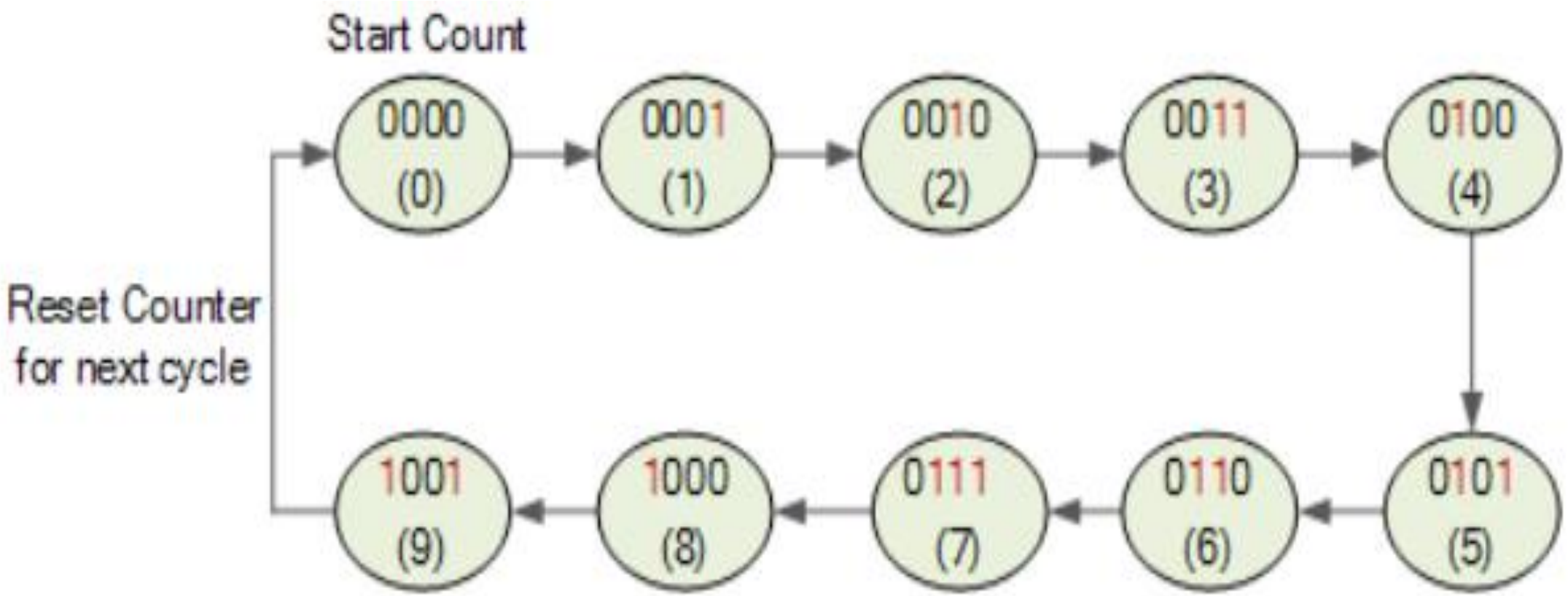
Function Table of MOD-2 and MOD - 5 counter

- **Function Table**

Input A clock	Output QA	Count
↓	0	0
↓	1	1

Input B clock	Output			Count
	QD	QC	QB	
↓	0	0	0	0
↓	0	0	1	1
↓	0	1	0	2
↓	0	1	1	3
↓	1	0	0	4

BCD Decade(MOD-10) Counter using IC 7490
State Diagram



Design of BCD Decade Counter using IC 7490

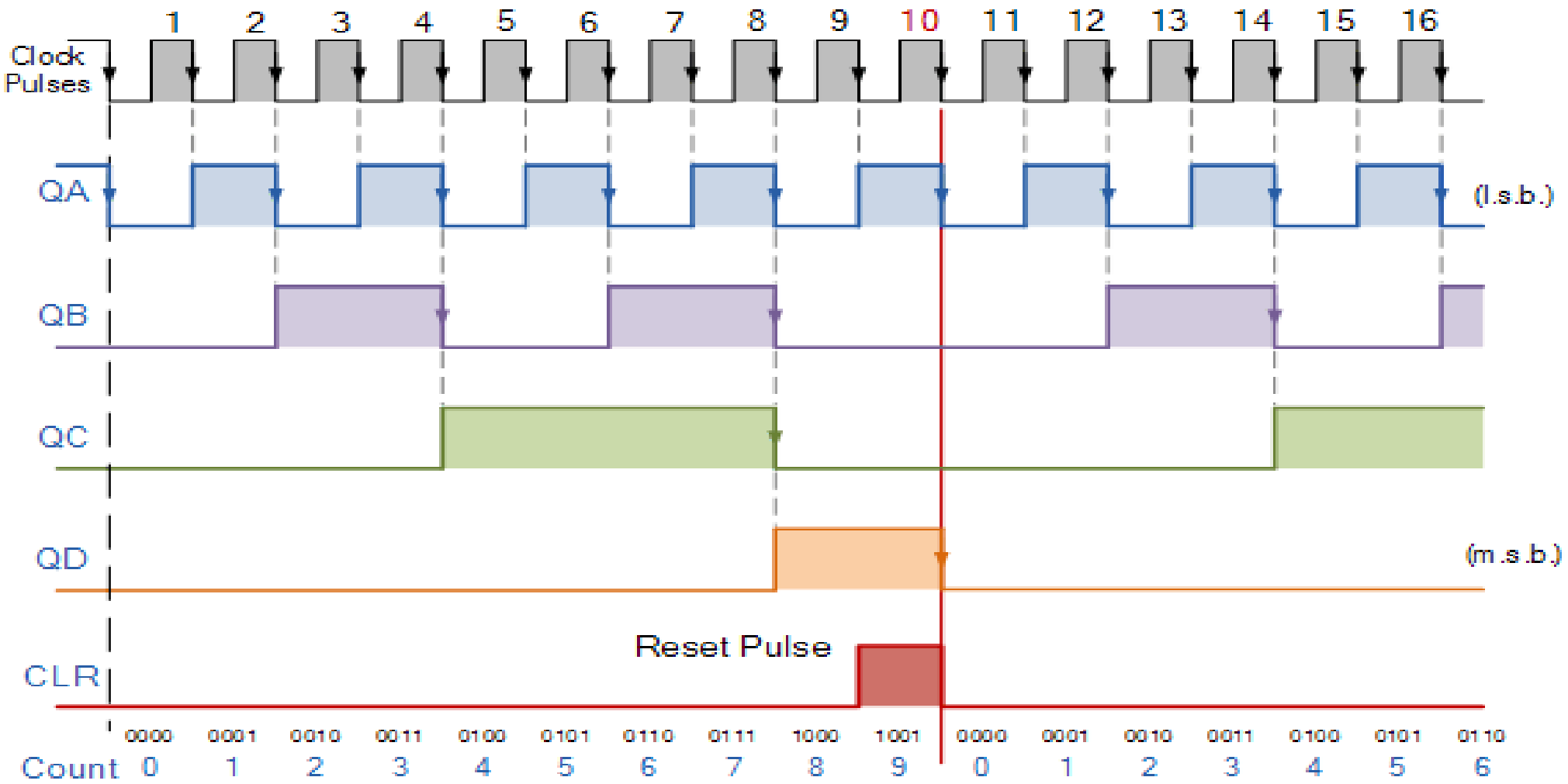
I/p clock	Output				Count
	QD	QC	QB	QA	
↓	0	0	0	0	0
↓	0	0	0	1	1
↓	0	0	1	0	2
↓	0	0	1	1	3
↓	0	1	0	0	4
↓	0	1	0	1	5
↓	0	1	1	0	6
↓	0	1	1	1	7
↓	1	0	0	0	8
↓	1	0	0	1	9

The QA o/p the first flip flop is connected to the input B which is clock i/p of internal MOD-5 ripple counter.

Due to cascading of Mod-2 and Mod-5 counters, the overall configuration is of a decade counters which counts from 0000 to 1001.

After 1001 mod-5 resets to 000 and mod 2 resets to 0. So next count after 1001 is 0000.

Timing diagram of mod10 Counter



Design of Mod-7 Counter using IC 7490

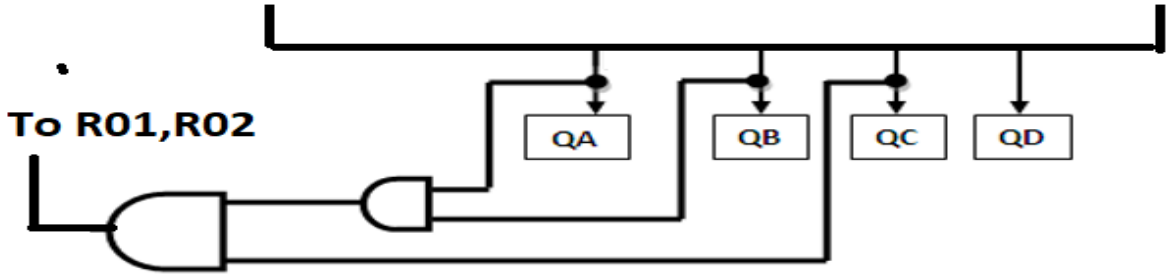
Construct MOD-10 counter and then truncate the count to 7 using RESET pins

Mod-7 counter counts through seven states from 0 to 6 It should reset as soon as the count becomes 7.

The o/p of reset logic should be 1 corresponding to invalid states.

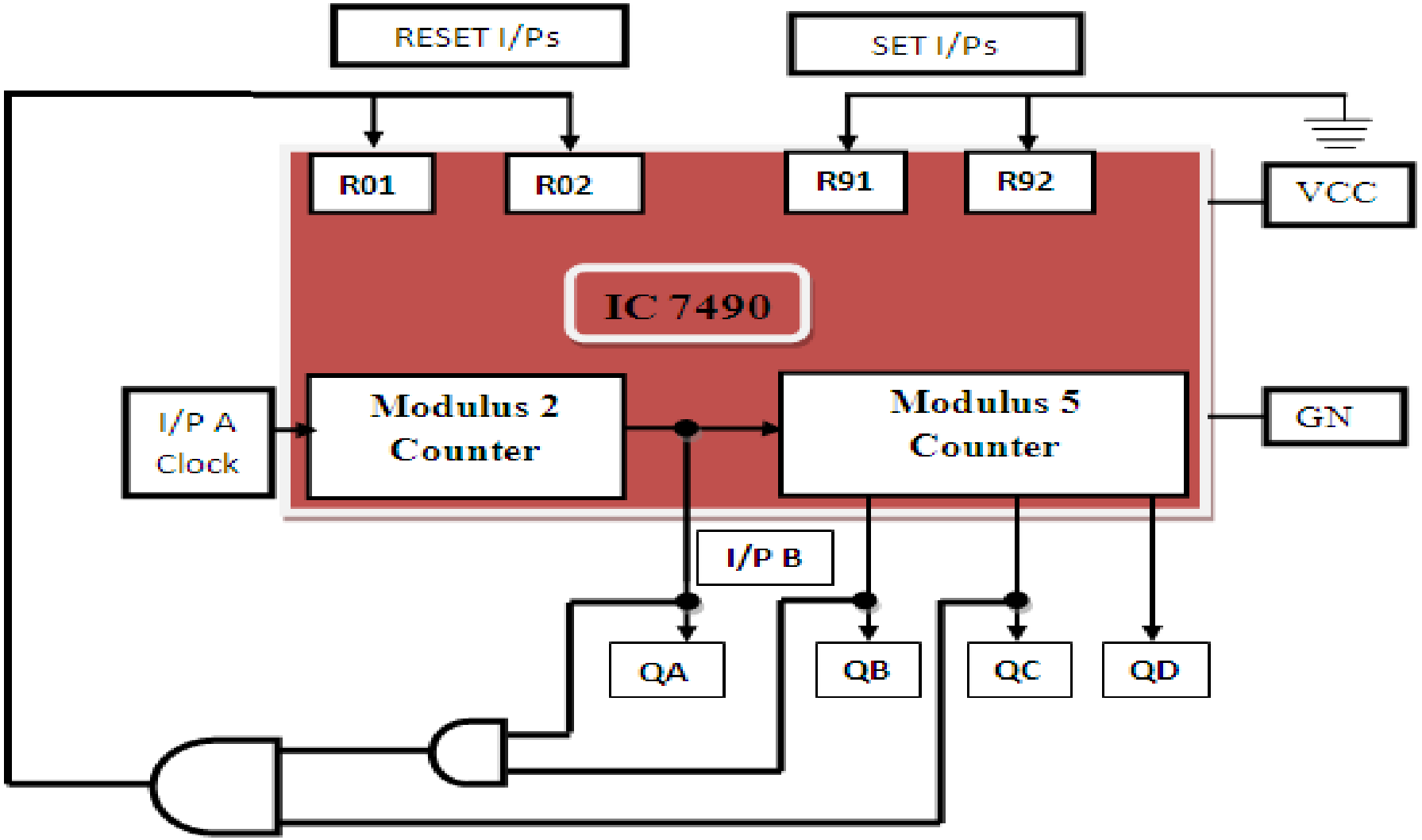
The reset logic o/p should be applied to pin 2 and 3(RESET pins).

To design RESET logic, connect the outputs going one in first unwanted state (here 7) to input of AND gate and comnnect AND gate output to R01 and R02



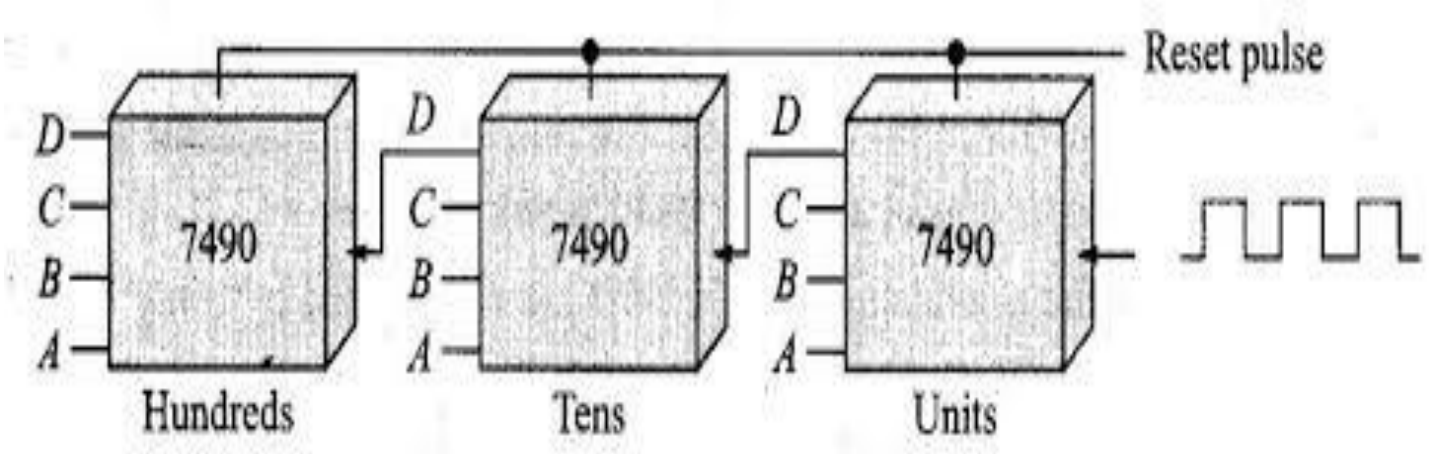
QD	QC	QB	QA	Count	RESET I/P
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	0
0	1	0	1	5	0
0	1	1	0	6	0
0	1	1	1	7	1
1	0	0	0	8	1
1	0	0	1	9	1

Design of Mod-7 Counter using IC 7490



Cascading of IC 7490

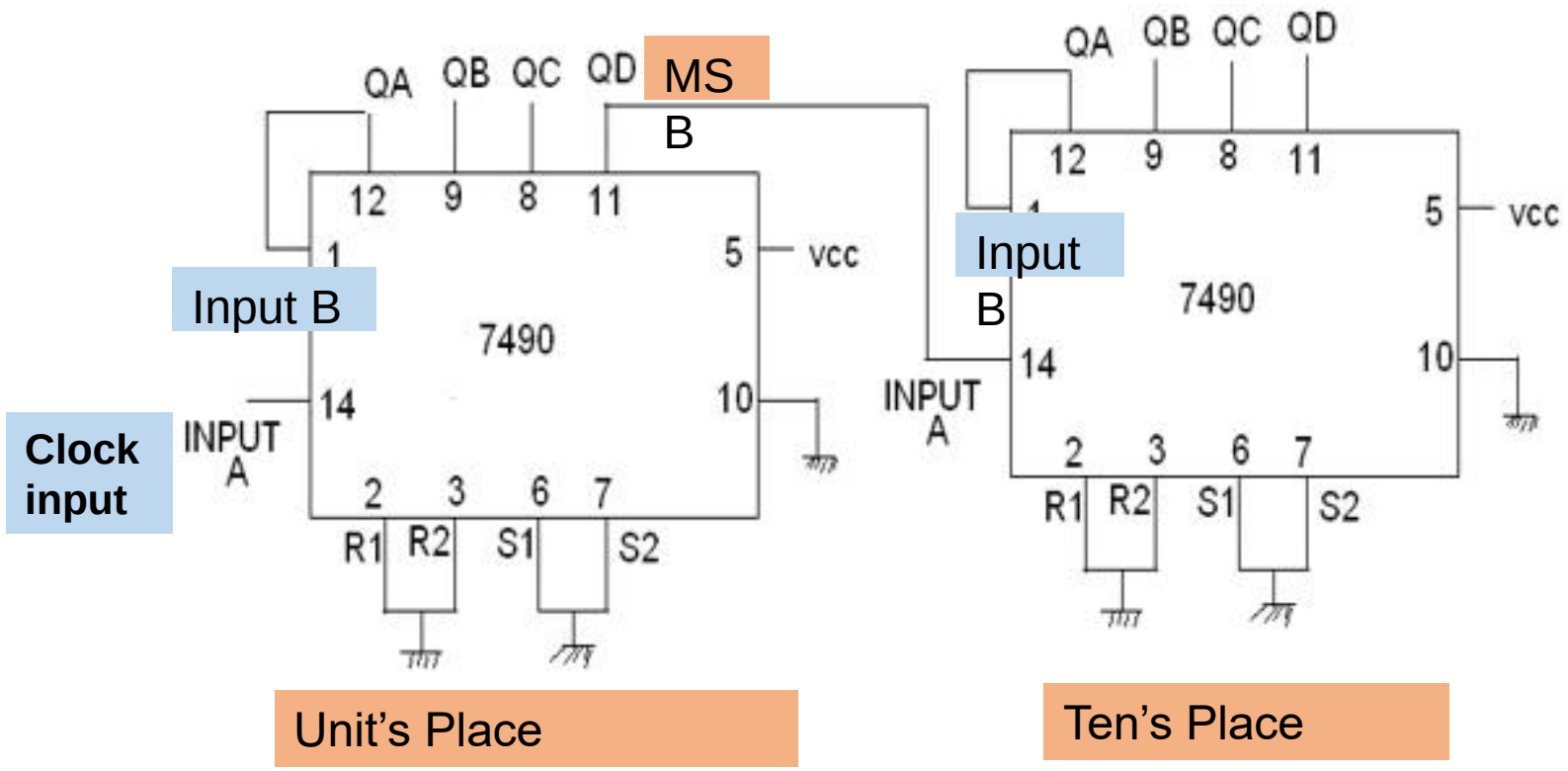
- To get the BCD counter which can count more than 10 pulses, cascading of IC 7490 is done.
- Multiple ICs can be cascaded together to get required count.
- Each cascaded IC is a decade counter (QA connected to CLK B)
- MSB output (QD) of first IC is connected as CLK A of second IC
- Two cascaded ICs can count up to 100 states (0 to 99)
- Three cascaded ICs can count up to 1000states (0 to 999) and so on



Design of Mod-100 Counter using IC 7490

- The MOD-100 counter will count 100 states from 0 to 99.
- One IC 7490 can count from 0 to 9.
- For Mod-100 counter, two ICs of 7490 will be required. One for Unit's place and one for Ten's place
- Cascading two BCD counters in such a way that after the Unit's place IC counts from 0 to 9, ten's place IC will increment once will suffice the requirement.
- No separate reset logic is required

Design of Mod-100 Counter using IC 7490

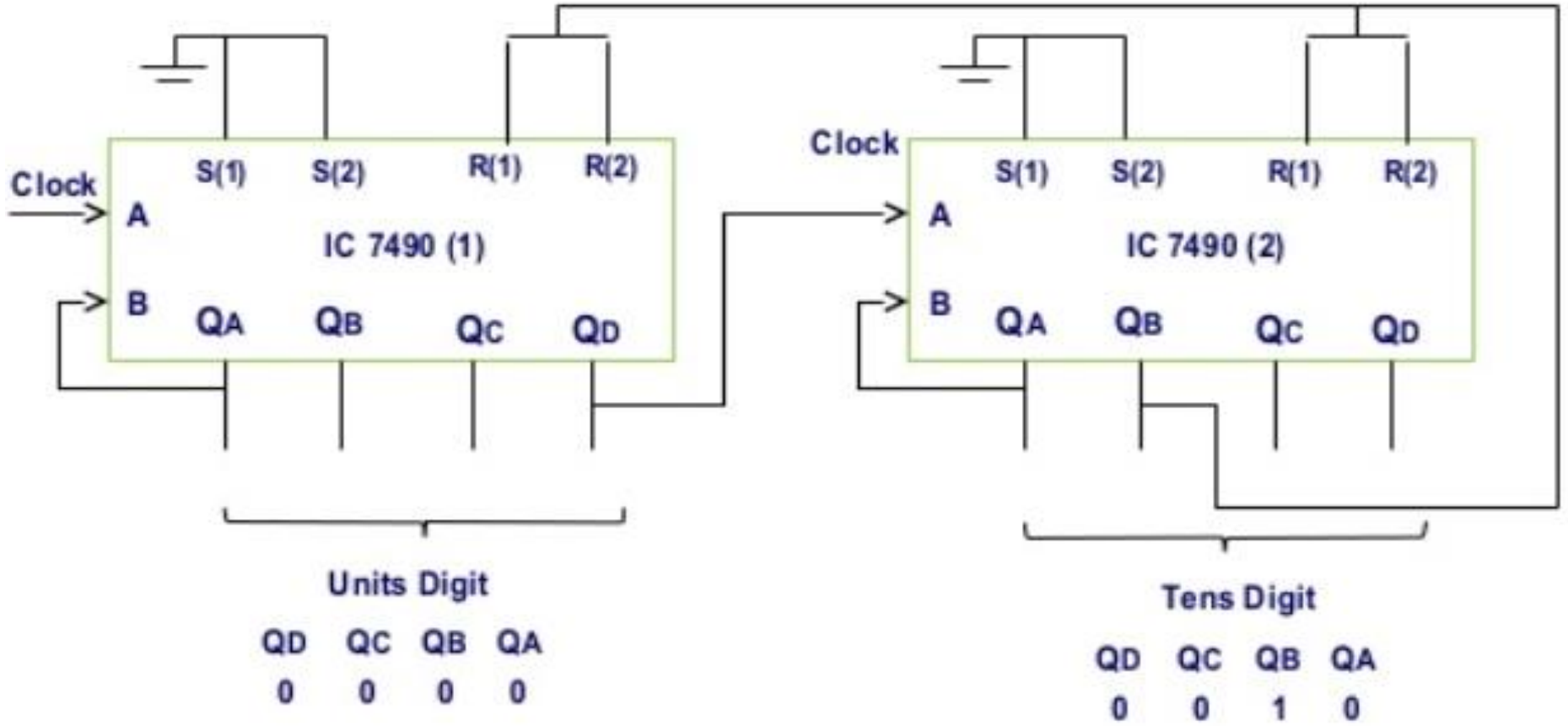


R1 = R0 1
R2 = R0 2
S1 = R9 1
S2 = R9 2

As input clock is given to Unit Place IC, it will count from 0 to 9.
When QD of IC 1 will go from 1 to 0 in next clock pulse, IC 2 will get clock at its input A and second IC will count 1.

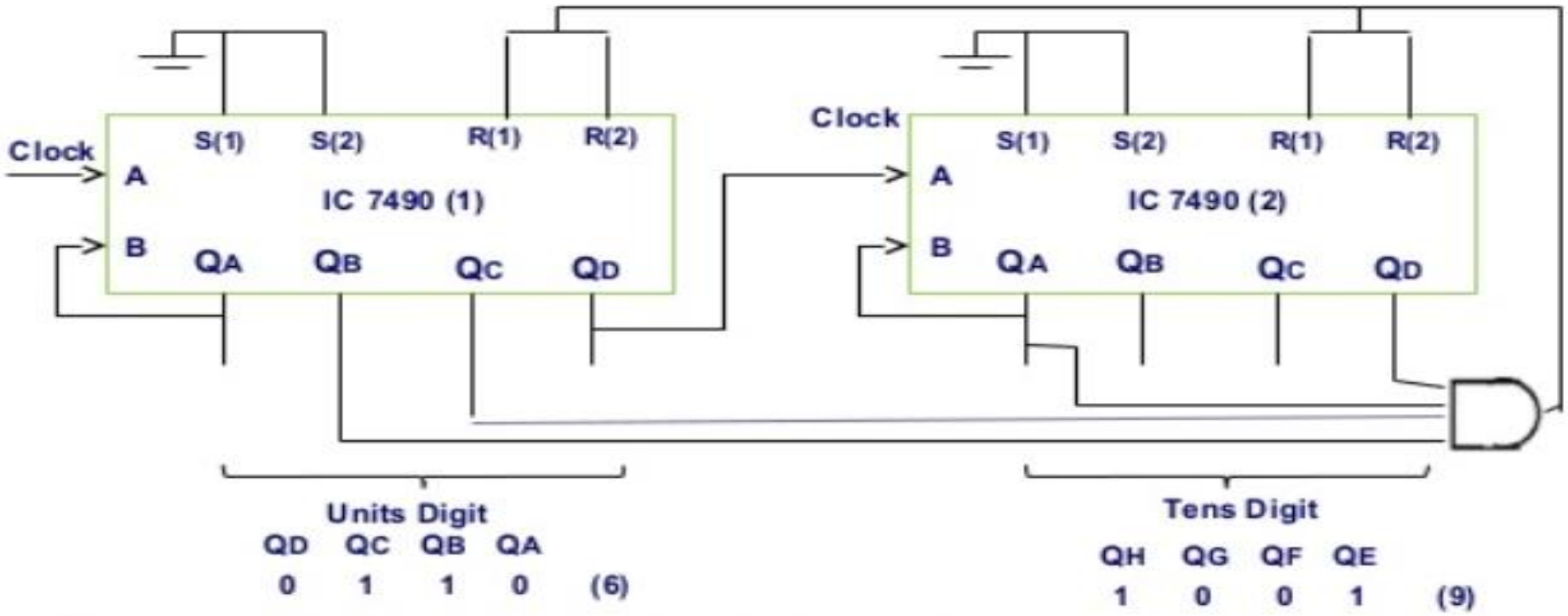
Second IC will get clock only when first IC will count from 0 to 9 (i.e. after every 10 clock pulses).
Thus the counter will reach to 99 before resetting to 00

MOD 20 Counter



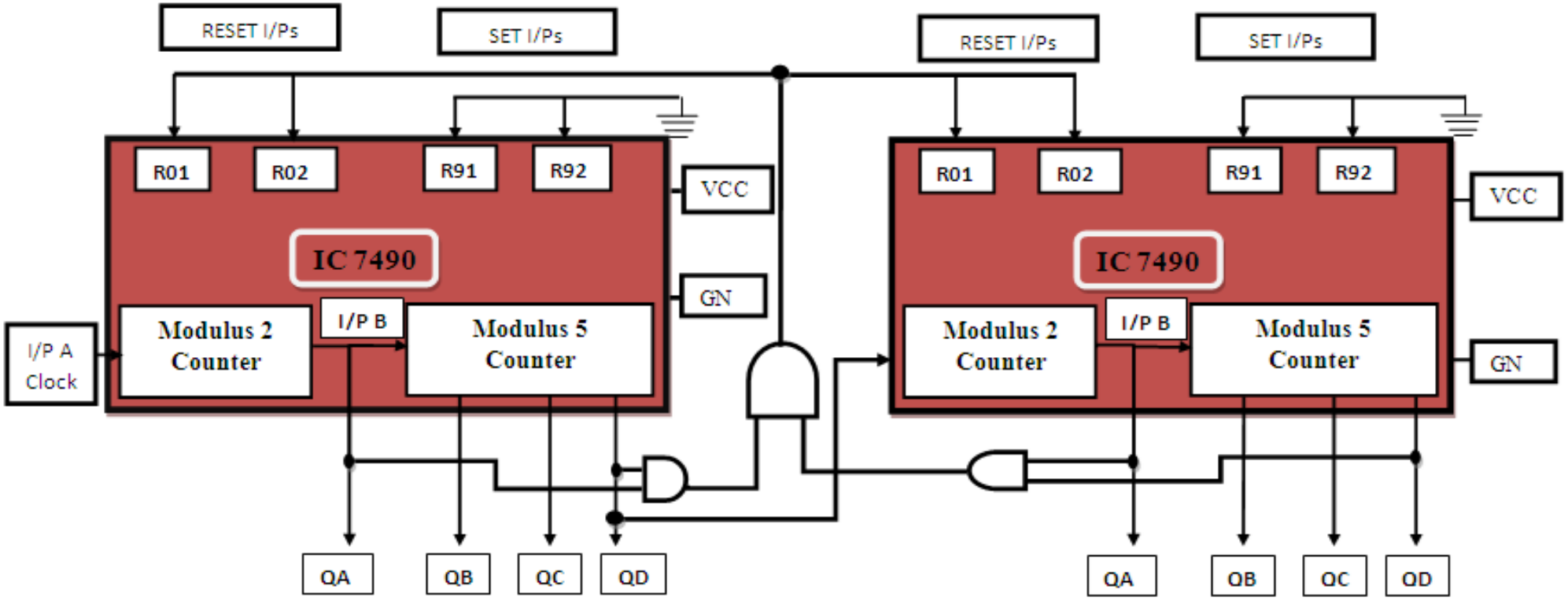
- In MOD 20 Counter, states will be from 00 to 19.
- The counter should get reset when it will try to go to state 20
- So design a resetting logic by considering the pins which are at logic 1 when count will be 20
- Connect the output of the reset logic to RESET input pins R0 1, R0 2 of both the Ics
- As in only QB pin will be high in count 20, connecting it to reset pins will suffice the purpose.

MOD 96 Counter



- In MOD 96 Counter, states will be from 00 to 95.
- The counter should get reset when it will try to go to state 96
- So design a resetting logic by considering the pins which are at logic 1 when count will be 96
- Connect the output of the reset logic to RESET input pins R0 1, R0 2 of both the ICs

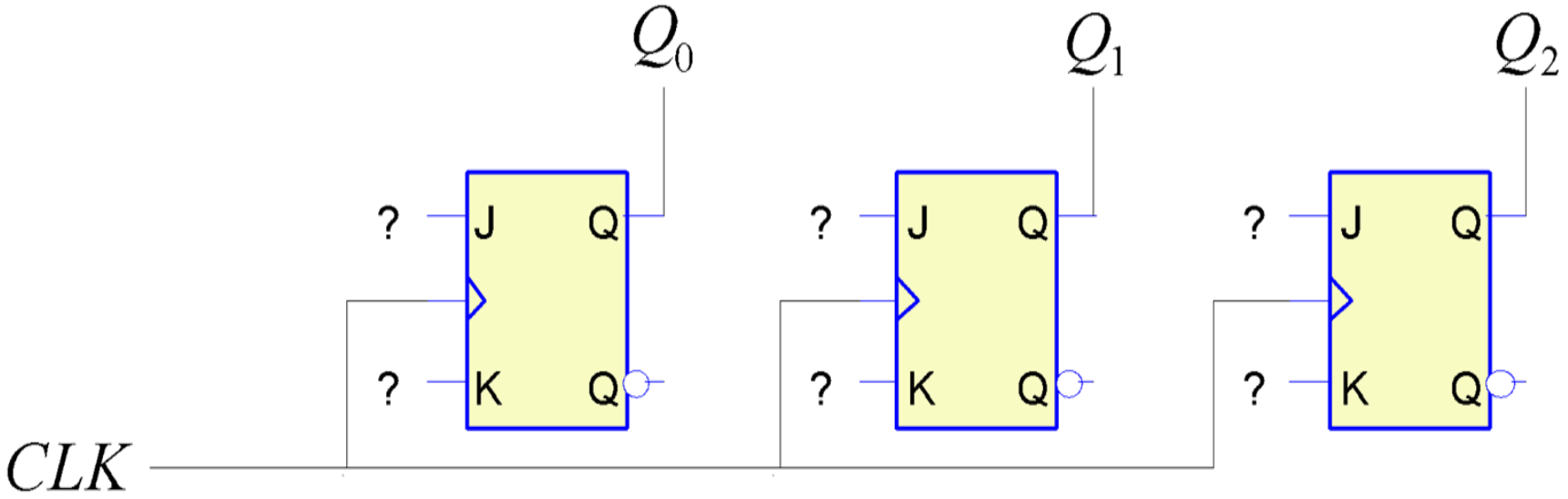
Design of Mod-99 using IC 7490



- In MOD 99 Counter, states will be from 00 to 98.
- The counter should get reset when it will try to go to state 98
- So design a resetting logic by considering the pins which are at logic 1 when count will be 98
- Connect the output of the reset logic to RESET input pins R0 1, R0 2 of both the ICs

Build a 3-bit Synchronous Up Counter

- A synchronous counter, in contrast to an asynchronous counter, is one whose output bits change state simultaneously, with no ripple
- The only way we can build such a counter circuit from J-K flip-flops is to connect all the clock inputs together, so that each and every flip-flop receives the exact same clock pulse at the exact same time



Build a 3-bit Synchronous Up Counter

- Draw the state diagram for the given counter
- Identify the number of flip flops required
- Write the state table using Excitation table of the flip flop
- Find the equation for each input of flip flop using K Map simplification technique
- Connect the circuit by connecting the inputs as per the equations obtained and connect clock inputs of all the flip flops to main clock input

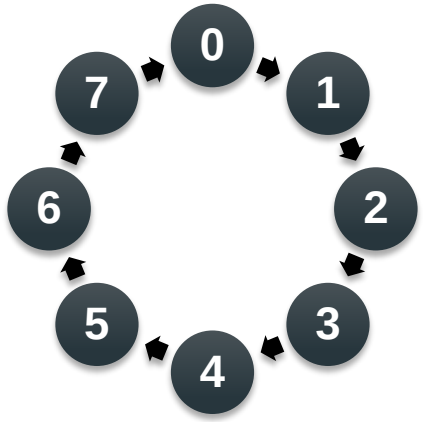
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q_n'

Truth Table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation Table

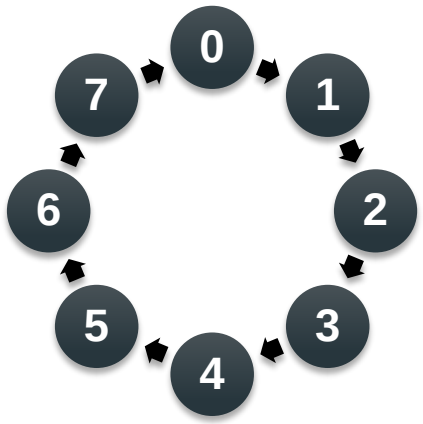
3 Bit Synchronous Counter(State Table)



Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

No	Present State			Next State			Flip Flop Inputs					
	Q_2	Q_1	Q_0	Q_{2+1}	Q_{1+1}	Q_{0+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
2	0	1	0	0	1	1	0	X	X	0	1	X
3	0	1	1	1	0	0	1	X	X	1	X	1
4	1	0	0	1	0	1	X	0	0	X	1	X
5	1	0	1	1	1	0	X	0	1	X	X	1
6	1	1	0	1	1	1	X	0	X	0	1	X
7	1	1	1	0	0	0	X	1	X	1	X	1

3 Bit Synchronous Counter(State Table)



No	Present State			Next State			Flip Flop Inputs					
	Q_2	Q_1	Q_0	Q_{2+1}	Q_{1+1}	Q_{0+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
2	0	1	0	0	1	1	0	X	X	0	1	X
3	0	1	1	1	0	0	1	X	X	1	X	1
4	1	0	0	1	0	1	X	0	0	X	1	X
5	1	0	1	1	1	0	X	0	1	X	X	1
6	1	1	0	1	1	1	X	0	X	0	1	X
7	1	1	1	0	0	0	X	1	X	1	X	1

Simplification using K- MAP

J_2	K_2
0	X
0	X
0	X
1	X
X	0
X	0
X	0
X	1

Q2 \ Q1Q0	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_2 = Q1Q0$

Q2 \ Q1Q0	00	01	11	10
0	X	X	X	X
1	0	0	1	0

$K_2 = Q1Q0$

Simplification using K- MAP

J_1	K_1
0	X
1	X
X	0
X	1
0	X
1	X
X	0
X	1

Q2 \ Q1Q0	00	01	11	10
0	0	1	X	X
1	0	1	X	X

$J_1 = Q0$

Q2 \ Q1Q0	00	01	11	10
0	X	X	1	0
1	X	X	1	0

$K_1 = Q0$

Simplification using K- MAP

J_0	K_0
1	X
X	1
1	X
X	1
1	X
X	1
1	X
X	1

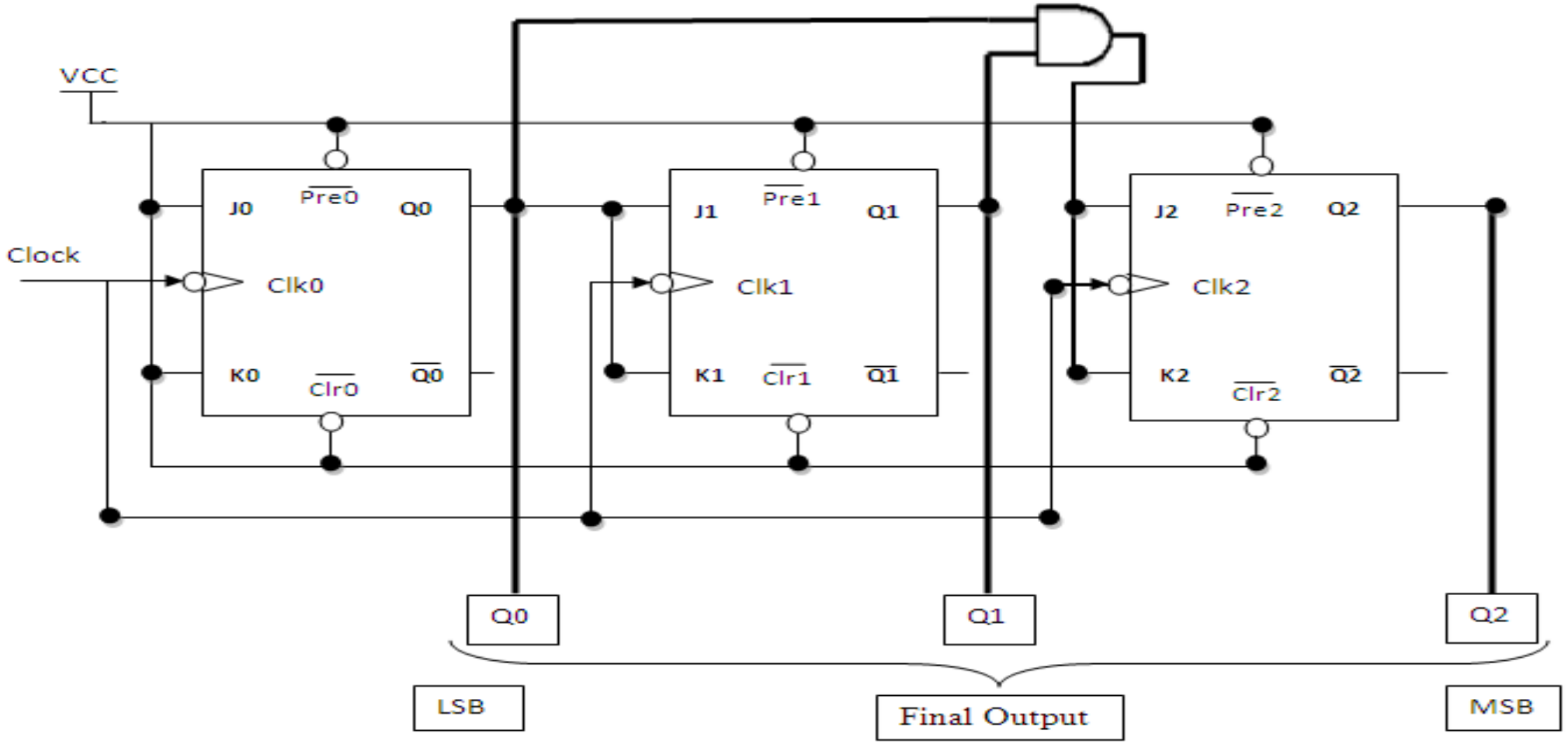
Q2 \ Q1Q0	00	01	11	10
0	1	X	X	1
1	1	X	X	1

$J_0 = 1$

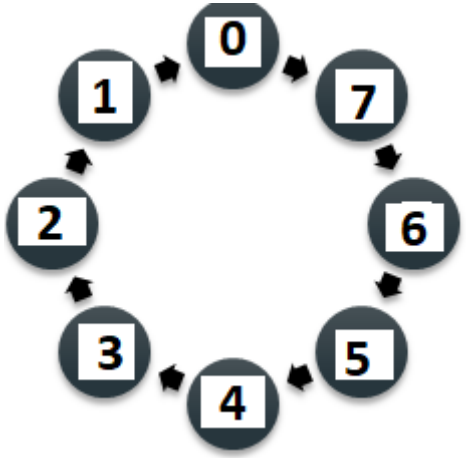
Q2 \ Q1Q0	00	01	11	10
0	X	1	1	X
1	X	1	1	X

$K_0 = 1$

3 bit Synchronous UP Counter



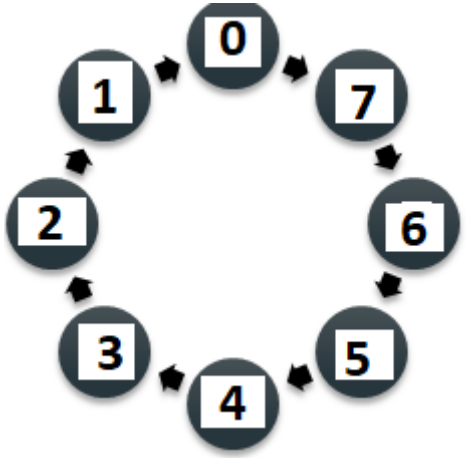
3 bit Synchronous down counter(State Table)



Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Present state			Next state			Flip flop 3		Flip flop 2		Flip flop 1	
Q2	Q1	Q0	Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
1	1	1	1	1	0	X	0	X	0	X	1
1	1	0	1	0	1	X	0	X	1	1	X
1	0	1	1	0	0	X	0	0	X	X	1
1	0	0	0	1	1	X	1	1	X	1	X
0	1	1	0	1	0	0	X	X	0	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	0	1	0	0	0	0	X	0	X	X	1
0	0	0	1	1	1	1	X	1	X	1	X

3 bit Synchronous down counter(State Table)



	Present state			Next state			Flip flop 3		Flip flop 2		Flip flop 1	
	Q2	Q1	Q0	Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
7	1	1	1	1	1	0	X	0	X	0	X	1
6	1	1	0	1	0	1	X	0	X	1	1	X
5	1	0	1	1	0	0	X	0	0	X	X	1
4	1	0	0	0	1	1	X	1	1	X	1	X
3	0	1	1	0	1	0	0	X	X	0	X	1
2	0	1	0	0	0	1	0	X	X	1	1	X
1	0	0	1	0	0	0	0	X	0	X	X	1
0	0	0	0	1	1	1	1	X	1	X	1	X

Simplification using K- MAP for J2,K2,J1,K1,J0,K0

J2

Q2 \ Q1Q0	00	01	11	10
0	1	0	0	0
1	X	X	X	X

J2 =Q1'Q0'

K2

Q2 \ Q1Q0	00	01	11	10
0	X	X	X	X
1	1	0	0	0

K2 = Q1'Q0'

J1

Q2 \ Q1Q0	00	01	11	10
0	1	0	X	X
1	1	0	X	X

J1 =Q0'

K1

Q2 \ Q1Q0	00	01	11	10
0	X	X	0	1
1	X	X	0	1

K1 =Q0'

J0

Q2 \ Q1Q0	00	01	11	10
0	1	X	X	1
1	1	X	X	1

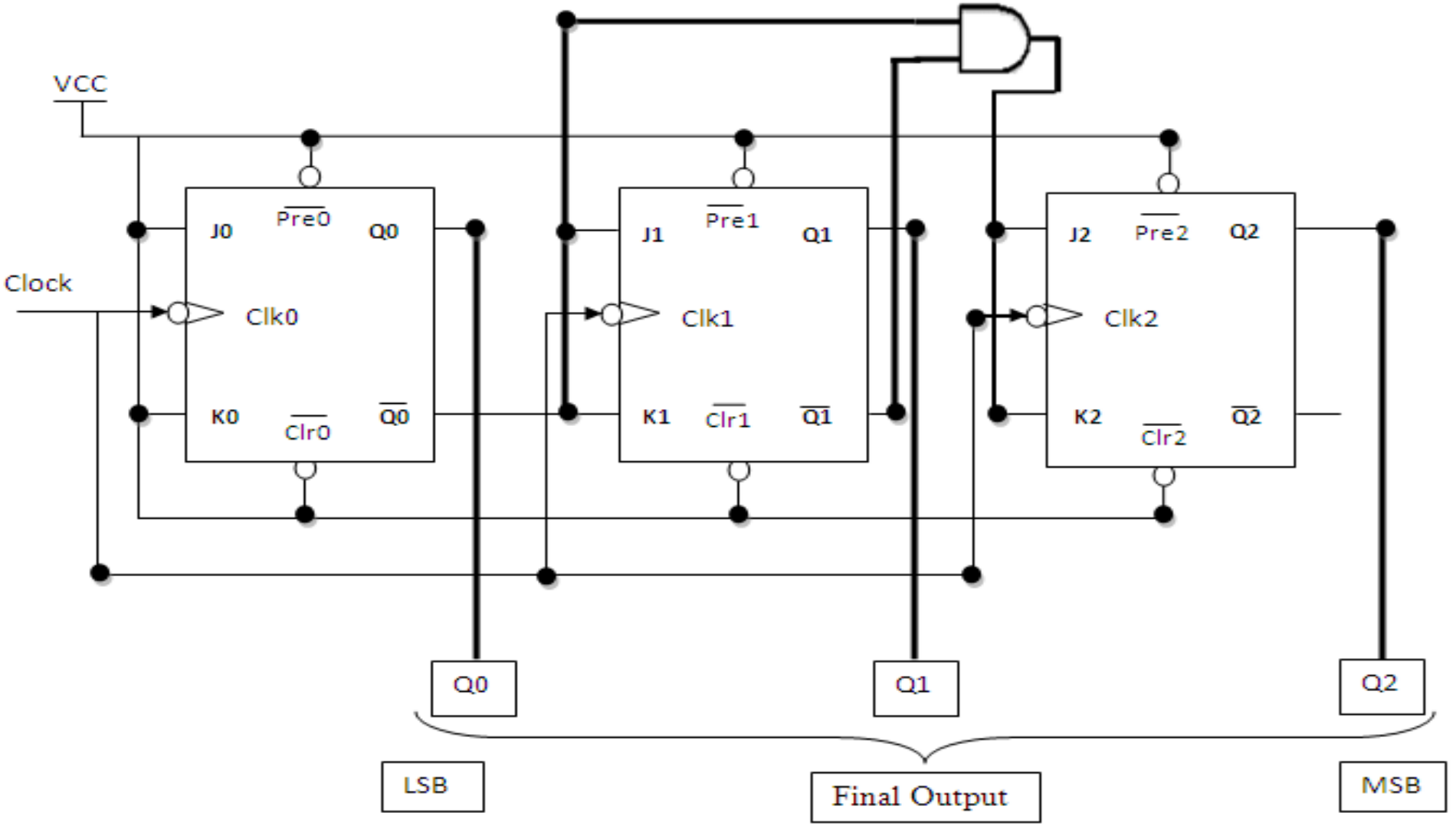
J0 = 1

K0

Q2 \ Q1Q0	00	01	11	10
0	X	1	1	X
1	X	1	1	X

K0 =1

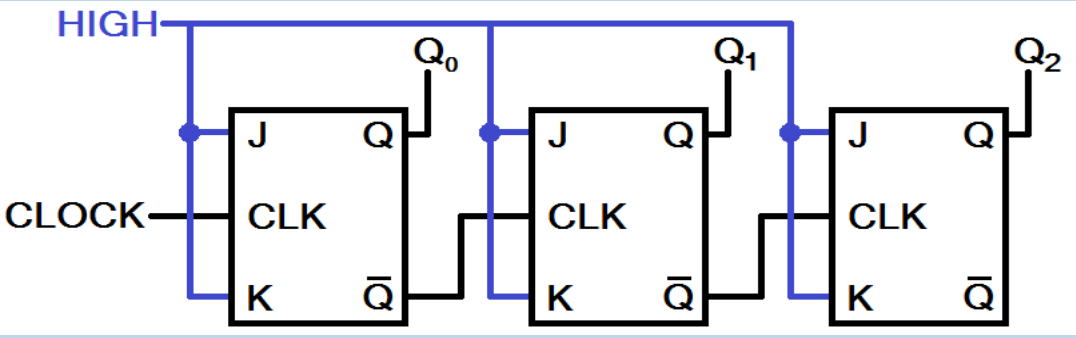
3 bit Synchronous down counter



Comparison of Counters

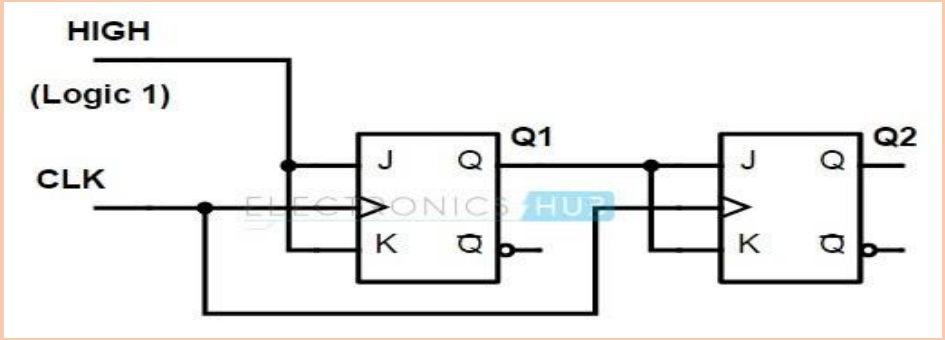
Asynchronous/Ripple Counter

- Flip flops are connected in such a way that the o/p of first flip-flop drives the clock of next flip-flop.
- Flip-flops are *not clocked* simultaneously.
- Circuit is *simple* for more number of states.
- Speed is *slow* as clock is propagated through number of stages.
- Asynchronous Counter are also known as “Ripple Counter” because of the way the clock pulses or ripples, its way through the flip-flop.



Synchronous Counter

- There is no connection between o/p of first flip-flop and clock of next flip-flop.
- Flip-flops are *clocked* simultaneously.
- Circuit becomes *complicated* as number of states increases.
- Speed is *high* as clock is given at a same time.

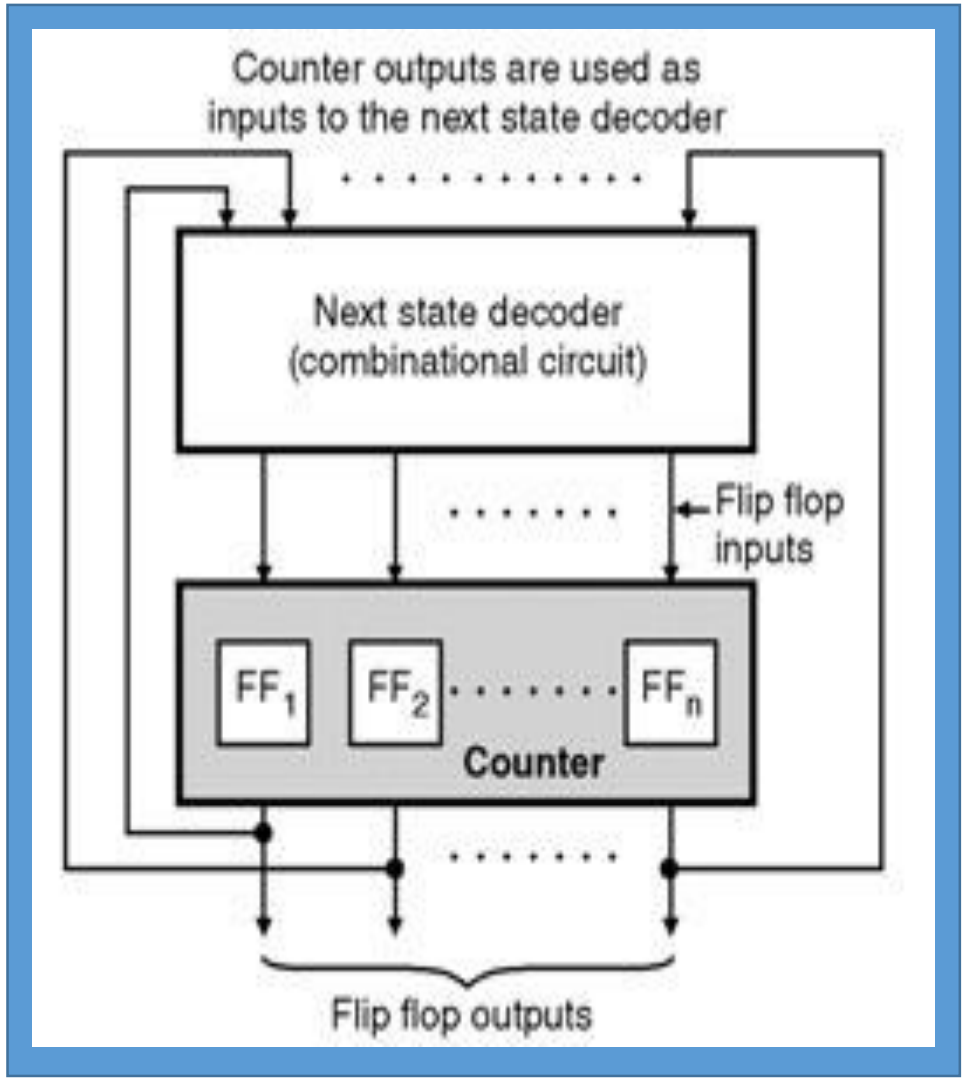


Design of sequence generator

Definition: A sequence generator is one kind of digital logic circuit. The main function of this is to generate a set of outputs.

A special kind of sequence generator is a binary counter.

The sequence generator circuit is used to generate a prescribed series of bits in synchronization through a CLK.



Design of sequence generator

The sequence generator can be classified as

1) Sequence generator without bushing (*Not taking care of unwanted states*)
2) Sequence generator with bushing (*Taking care of unwanted states*)

Step 1

To find number of flip flops required,

1. Find the no. of 1‘s in the sequence.
2. Find the no. of 0‘s in the sequence.
3. Take the maximum out of two.
4. If ‘n’ is the required no. of flip flops, choose minimum value of ’n’ to satisfy equation given below.

$$\text{Max}\,(0\text{'s}\ ,\ 1\text{'s}) \leq 2^{n-1}$$

Design of sequence generator

Step 2:

- Once the no. of flip flops are decided assign unique states corresponding to each bit in the given sequence.
- Copy the given sequence in the column under the LSB output
- Usually the o/p of flip flop representing the least significant bit is used to generate the given sequence but any other flip flop in the circuit also can be used to give the output.

Step 3:

Draw a state diagram for states

Step 4:

Prepare a state table using excitation table of the flip flop

Step 5:

Prepare and solve K Maps for inputs of flip flops (e.g. J2, K2, J1, K1 etc)

Step 6:

Draw the circuit diagram

Example: Design a sequence generator for sequence 10101 (Without Bushing) Ex. 10110

Step 1

Design a sequence generator to generate a sequence of bit 10101.

For finding the no. of flip flops we use the formula

$$M \leq 2^{n-1}$$

where m= maximum count of 0s and 1s

n= no. of flip flops required.

Here, No. of 1s = 3, No. of 0s = 2 Hence M= 3

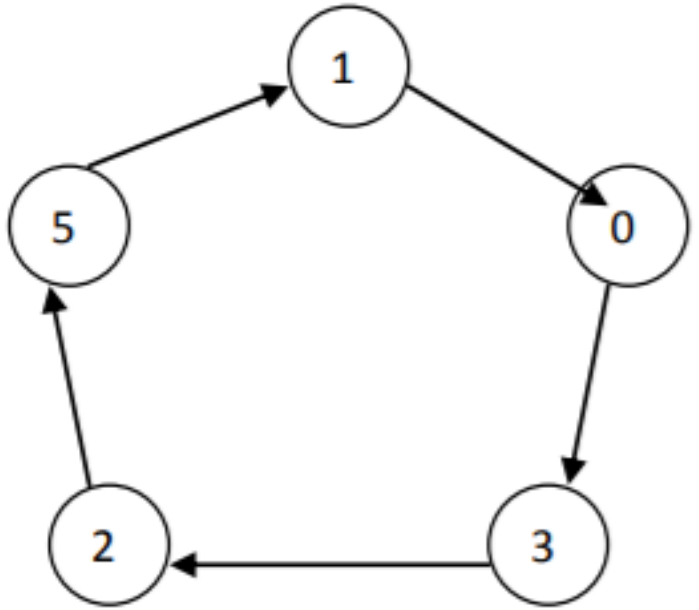
$3 \leq 2^{n-1}$ Hence, n = 3

Design of sequence generator : Step 2

- For the given sequence 10101, we know 3 flip flops are required. Lets call the flip flops as FF2,FF1,and FF0 with the corresponding outputs Q2, Q1 and Q0
- Copy the given sequence in Q0 column
- Fill the values in column of Q2 and Q1 in such a way that no state repeats to prepare a truth table

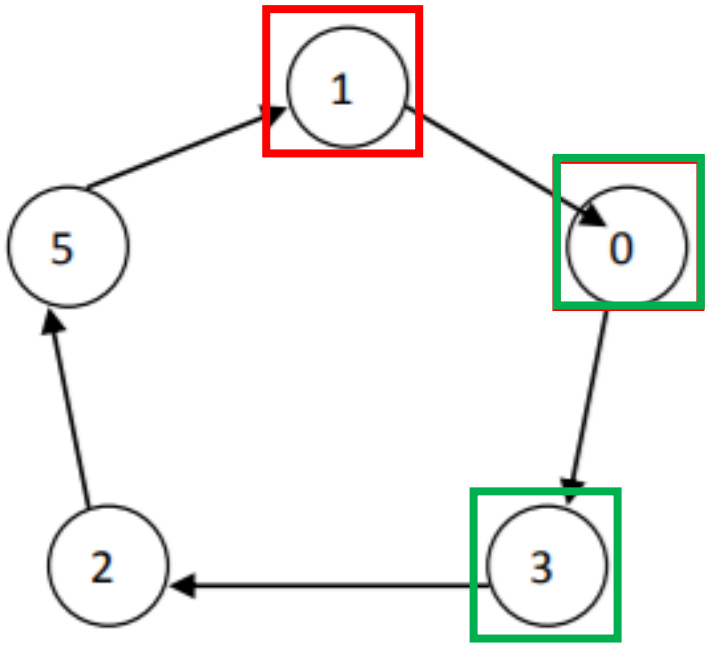
Q2	Q1	Q0	State Assigned
0	0	1	1
0	0	0	0
0	1	1	3
0	1	0	2
1	0	1	5

Step 3
Prepare a state diagram (Without Bushing)



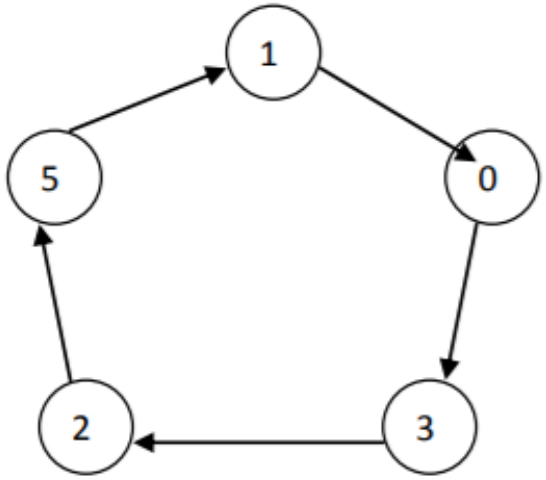
Design of sequence generator : Step 3

Prepare a state table using state diagram. Consider all the possible states as present state. Write Don't care condition (X) as a next state for states not present in the state diagram.



Present State			Next State		
Q2	Q1	Q0	Q2	Q1	Q0
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	X	X	X
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

Design of sequence generator : Step 3

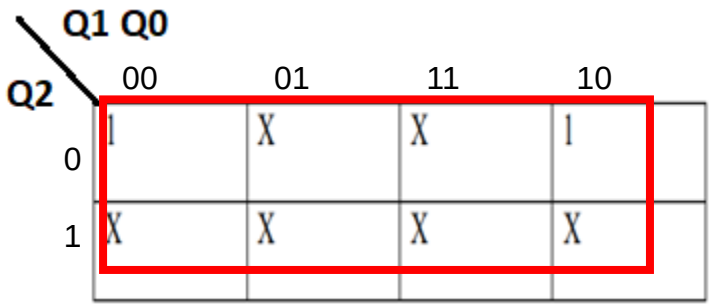


Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

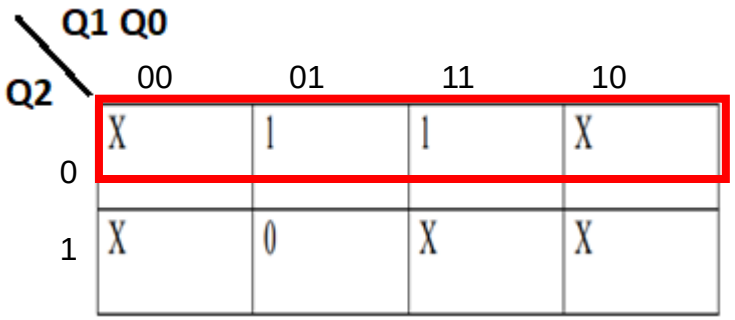
Complete the state table using Excitation table of flip flop

Present states			Next states			Flip flop input					
Q2	Q1	Q0	Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
0	0	0	0	1	1	0	X	1	X	1	X
0	0	1	0	0	0	0	X	0	X	X	1
0	1	0	1	0	1	1	X	X	1	1	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	X	X	X	X	X	X	X	X	X
1	0	1	0	0	1	X	1	0	X	X	0
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

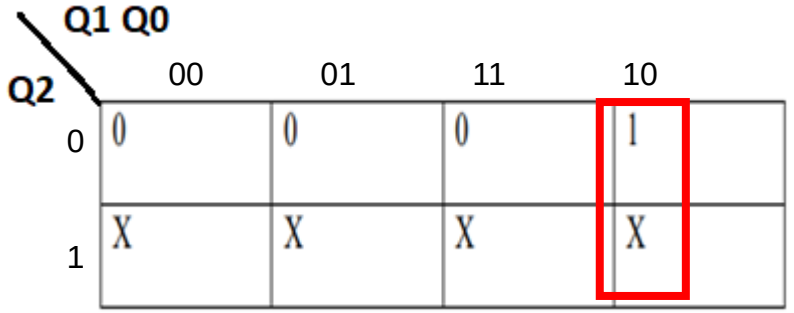
Design of sequence generator : Step 4 K Maps



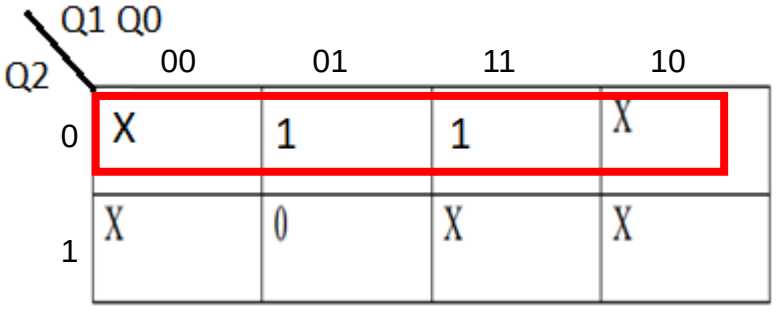
J0 = 1



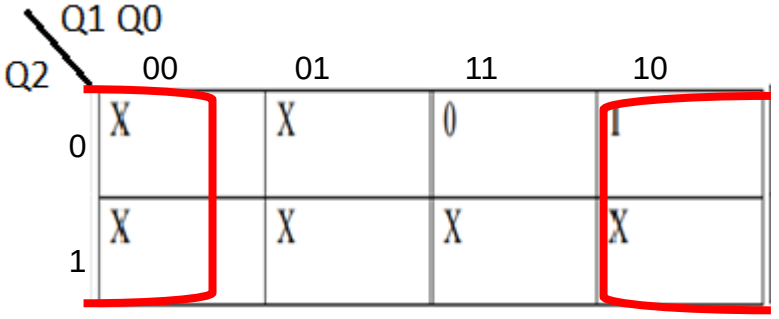
J1 = Q2'



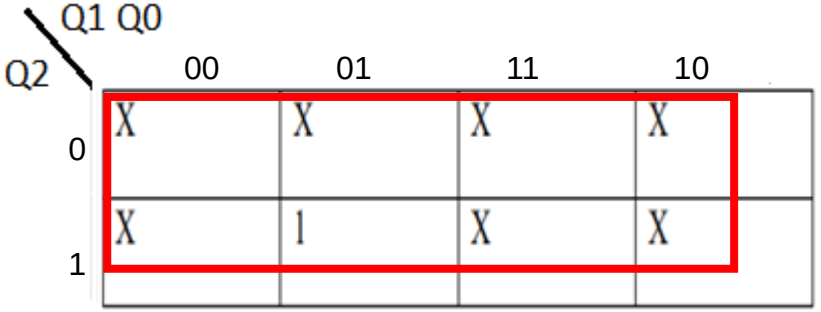
J2 = Q1Q0'



K0 = Q2'

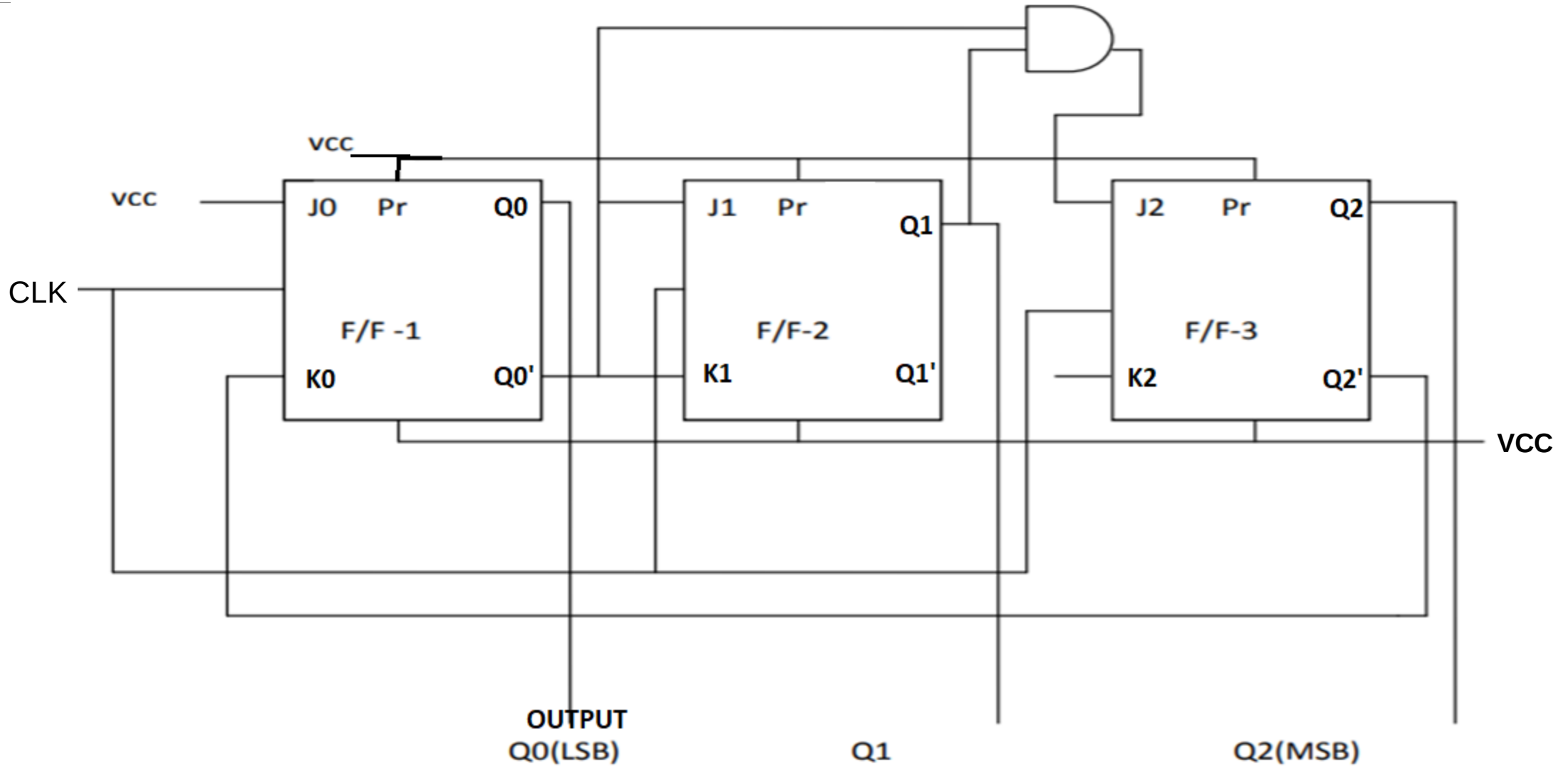


K1 = Q0'



K2 = 1

Design of sequence generator : Step 5 Circuit Diagram



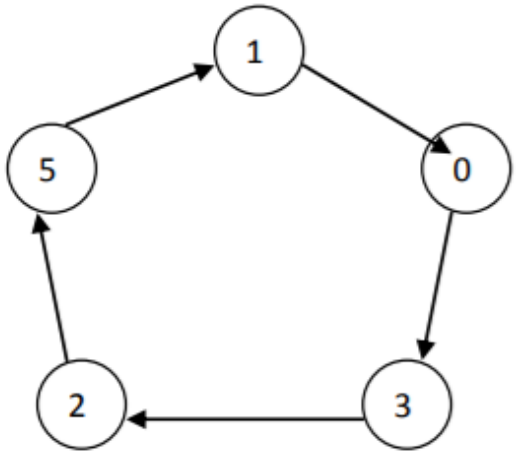
Design of sequence generator : With Bushing

In the given example, (sequence 10101) as we have three flip flops, maximum 8 states can be generated. Out of eight, 5 states (1,0,3,2,5) are wanted states where as remaining 3 states (4,6,7) are unwanted states.

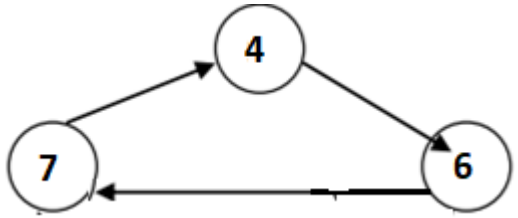
Lock out condition will occur if circuit enters in any unwanted state and keeps on moving in unwanted states.

With bushing, design takes care of these unwanted or unassigned states by connecting them to one of the valid state to avoid lock out condition

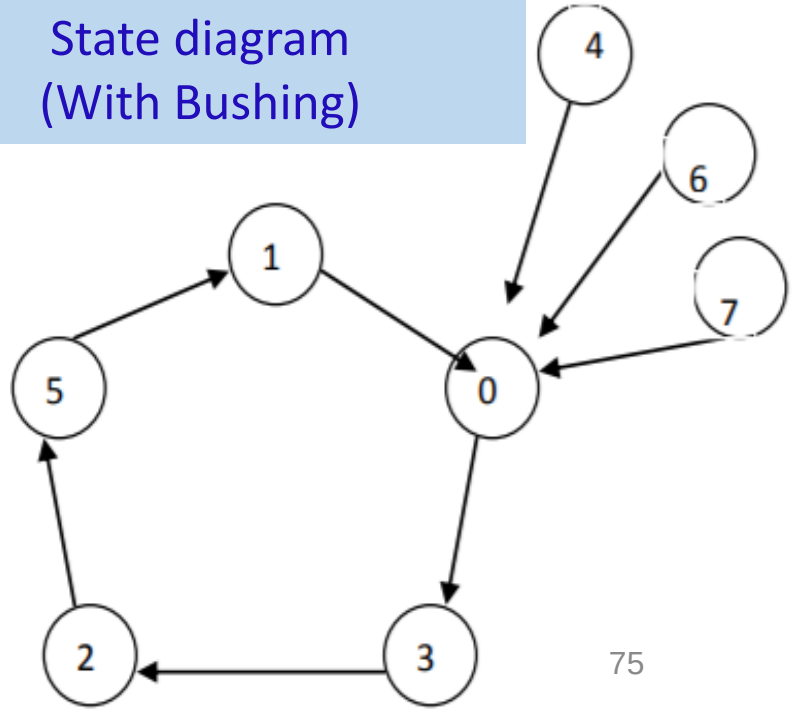
State diagram
(Without Bushing)



State diagram
(Lockout Condition)



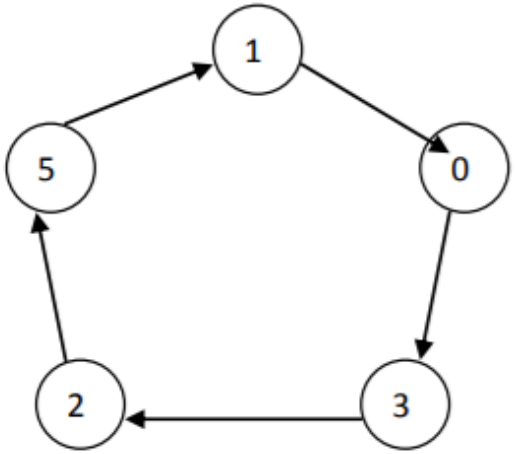
State diagram
(With Bushing)



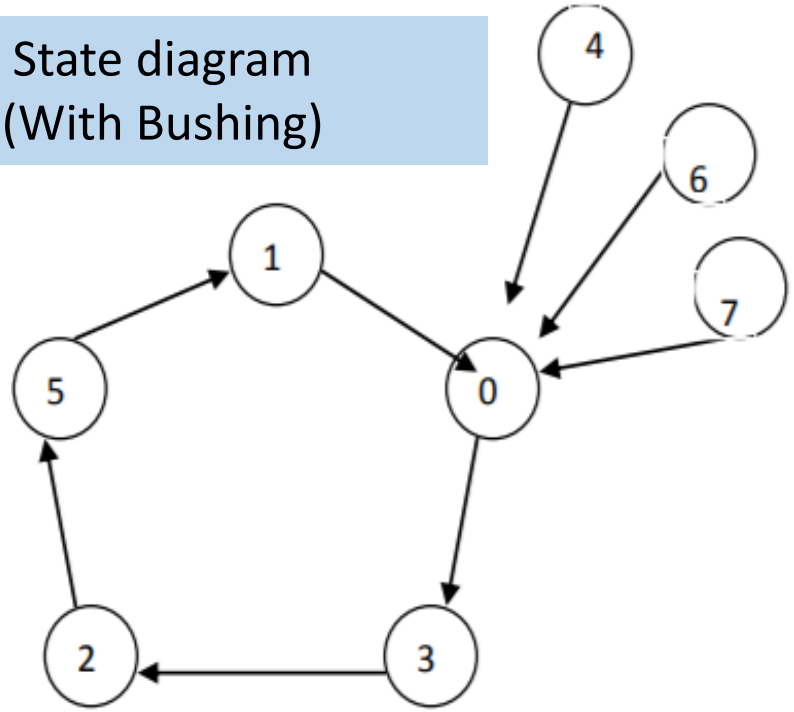
Design of sequence generator : With Bushing

Q2	Q1	Q0	State Assigned
0	0	1	1
0	0	0	0
0	1	1	3
0	1	0	2
1	0	1	5

State diagram
(Without Bushing)



State diagram
(With Bushing)



- The unwanted states can be connected to any wanted state.
- Unwanted states are shown connected to state 0 as an example.
- The three unwanted states can be connected to three different wanted states.
- Remaining steps are same as that followed in sequence generator without bushing.

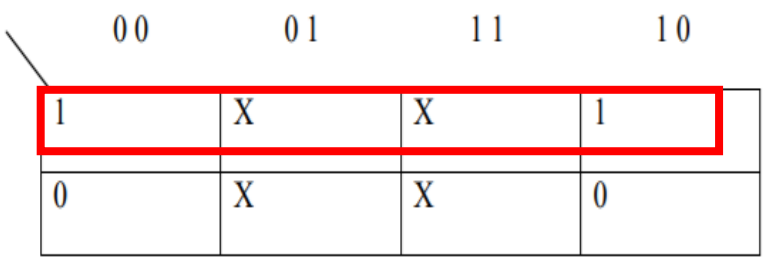
Design of sequence generator : Step 3

Prepare a state table using Excitation table of flip flop

Present state			Next state			Flip flop inputs					
Q2	Q1	Q0	Q2	Q1	Q0	J0	K0	J1	K1	J2	K2
0	0	0	0	1	1	1	X	1	X	0	X
0	0	1	0	0	0	X	1	0	X	0	X
0	1	0	1	0	1	1	X	X	1	1	X
0	1	1	0	1	0	X	1	X	0	0	X
1	0	0	0	0	0	0	X	0	X	X	1
1	0	1	0	0	1	X	0	0	X	X	1
1	1	0	0	0	0	0	X	X	1	X	1
1	1	1	0	0	0	X	1	X	1	X	1

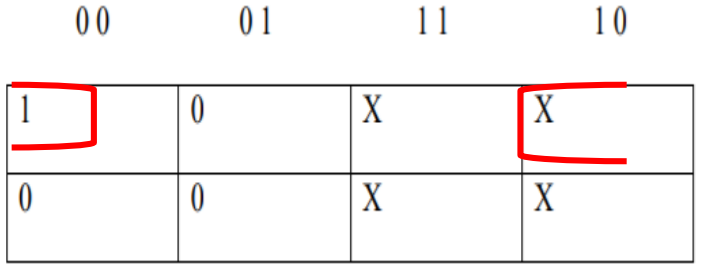
Unwanted states 4,6,7

Design of sequence generator : Step 4 K Maps



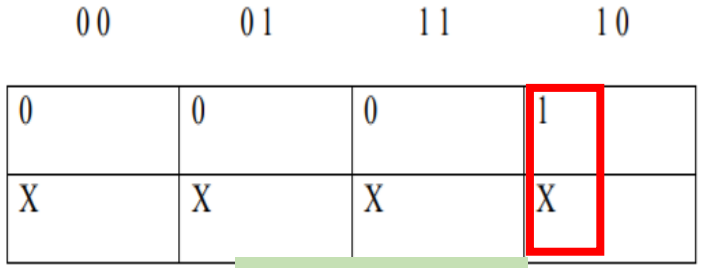
	00	01	11	10
	1	X	X	1
	0	X	X	0

J0 = Q2'



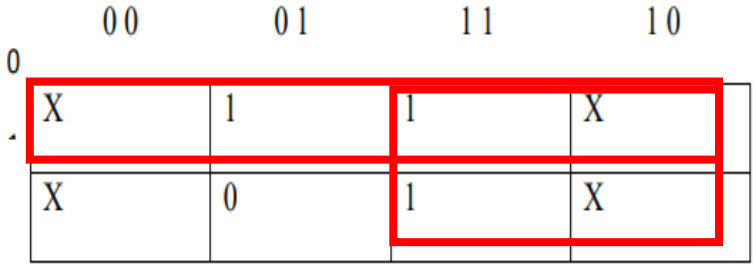
	00	01	11	10
	1	0	X	X
	0	0	X	X

J1 = Q2'Q0'



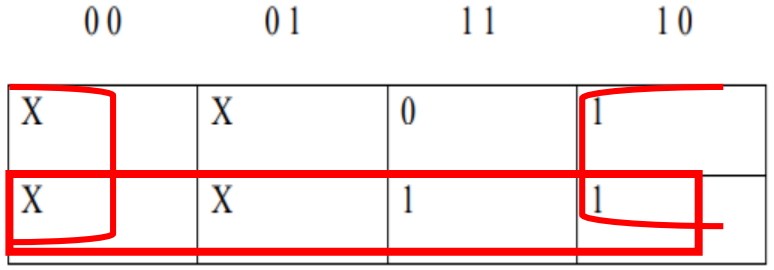
	00	01	11	10
	0	0	0	1
	X	X	X	X

J2 = Q1Q0'



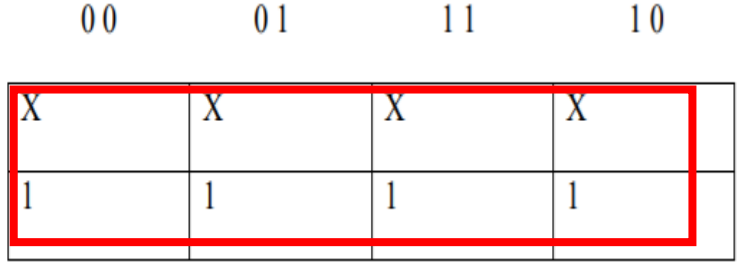
0	00	01	11	10
	X	1	1	X
	X	0	1	X

K0 = Q2' + Q1



	00	01	11	10
	X	X	0	1
	X	X	1	1

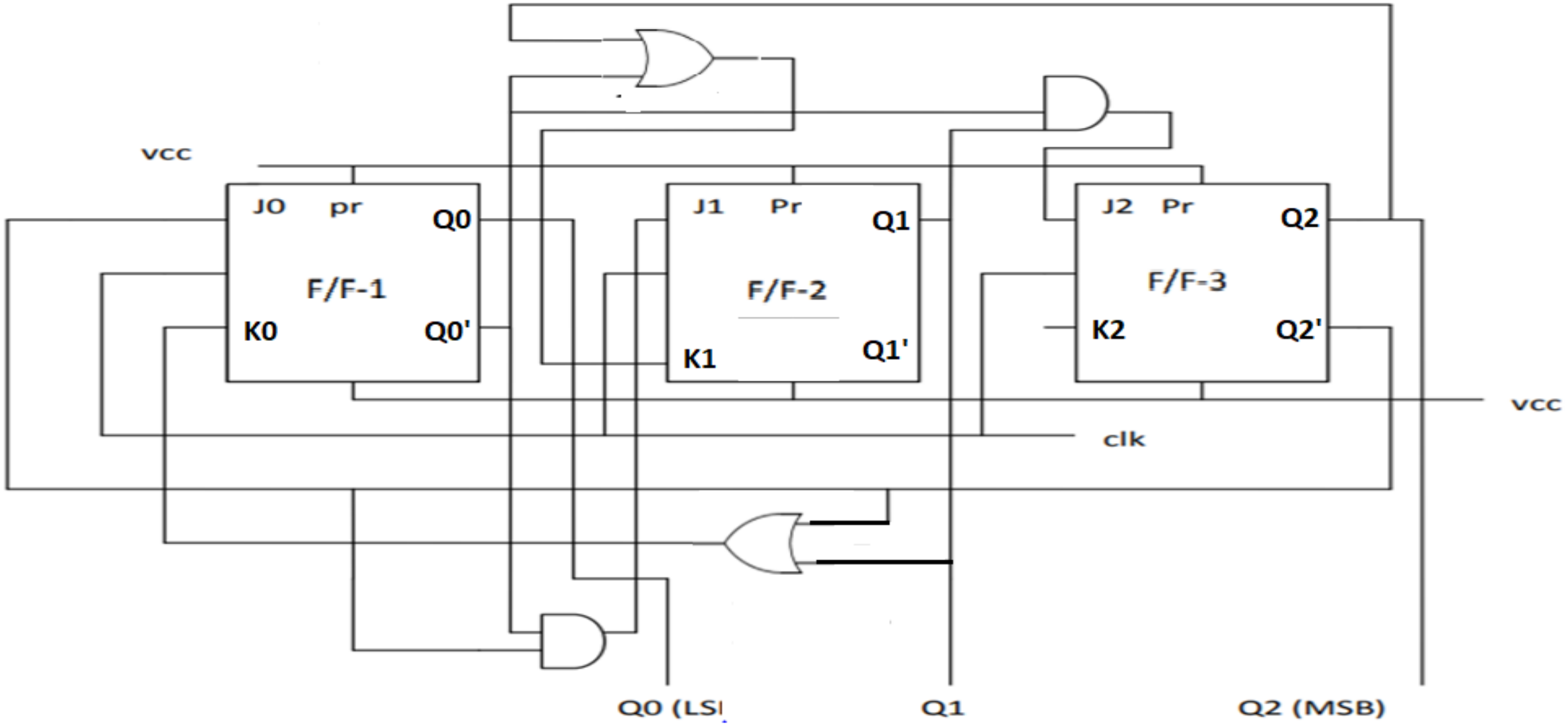
K1 = Q2 + Q0'



	00	01	11	10
	X	X	X	X
	1	1	1	1

K2 = 1

Design of sequence generator : Step 5 Circuit Diagram



Complex sequential circuits - Applications

Complex sequential circuits – An Example

Any Vending machine is an example of state machine and have:

- **Different States**
  - **Dispensing product, accept money, dispense change, error, Jam, initial stage**
- **Sequential traversing between states**
- **For orderly traversal,** control sequencing, next state decoding and branching to next state is required

Introduction to Synchronous Sequential Circuit Design

- A **state machine is a digital device** that traverses through a predetermined sequence of states in an orderly fashion
- A **state is a set of values** measured at different parts of the circuit.
- A simple state machine can consist of PAL device based combinatorial logic, output registers, and buried (state) registers.
- **State machine designs are widely used for sequential control logic, which forms the core of many digital systems**

Introduction to Synchronous Sequential Circuit Design contd.

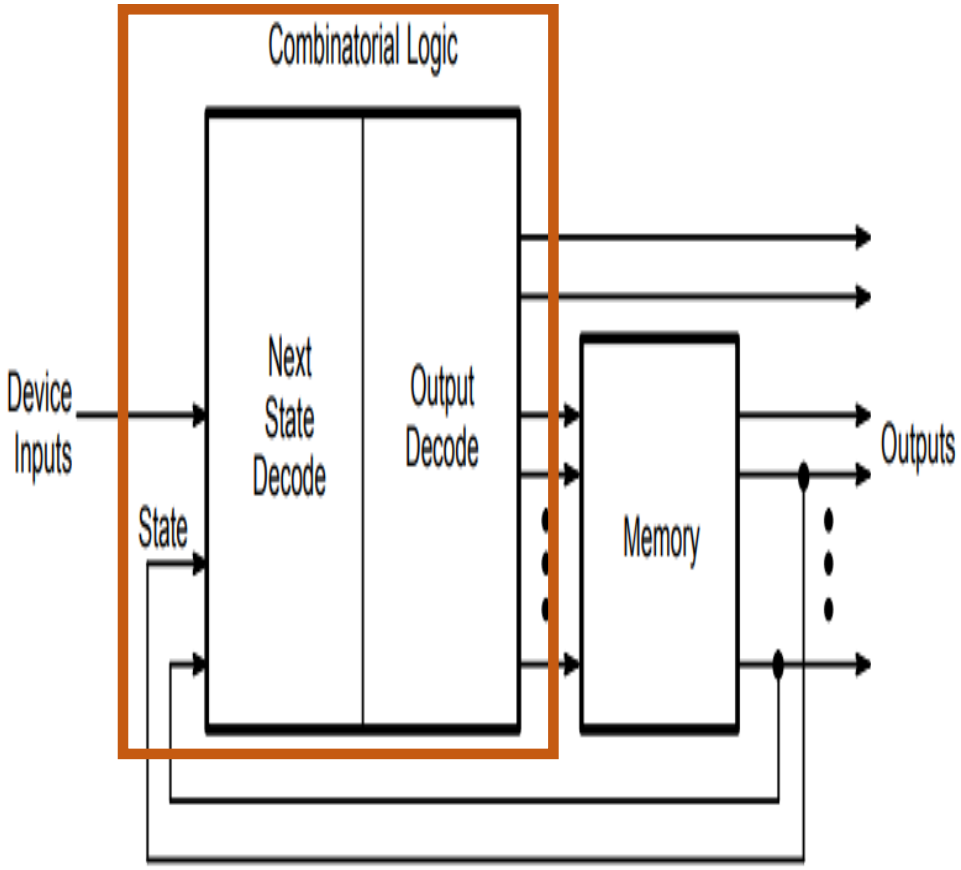
- Sequential Circuits with finite number of states are called as **Finite State Machines (FSM)**

Different ways to express the FSMs w.r.t outputs:

- **Moore Machines** - Sequential circuits in which outputs are function of only **present state**

- **Mealy Machines** - Sequential circuits in which outputs are function of **present state as well as input**

Block Diagram of a Simple State Machine



- In addition to the device **inputs and outputs**, a state machine consists of **two essential elements: combinatorial logic** and **memory** (registers)
- The **memory is used to store the state** of the machine
- The **combinatorial logic** can be viewed as two distinct functional blocks: the **next state decoder** and the **output decoder**

- The **next state decoder determines** the **next state of the state machine** while the **output decoder generates the actual outputs**

- Although Combinational logic perform two distinct functions, these are usually combined into one combinatorial logic array

Operation of a state machine

The basic operation of a state machine is two fold:

1. It **traverses through a sequence of states**, where the next state is determined by **next state decoder**, **depending upon the present state and input conditions.**

2. It **provides sequences of output signals based upon state transitions**. The outputs are generated by the **output decoder** based upon present state and input conditions.

Control sequencing: The transitions from one state to another.

Transition function: The logic required for deciding the next states

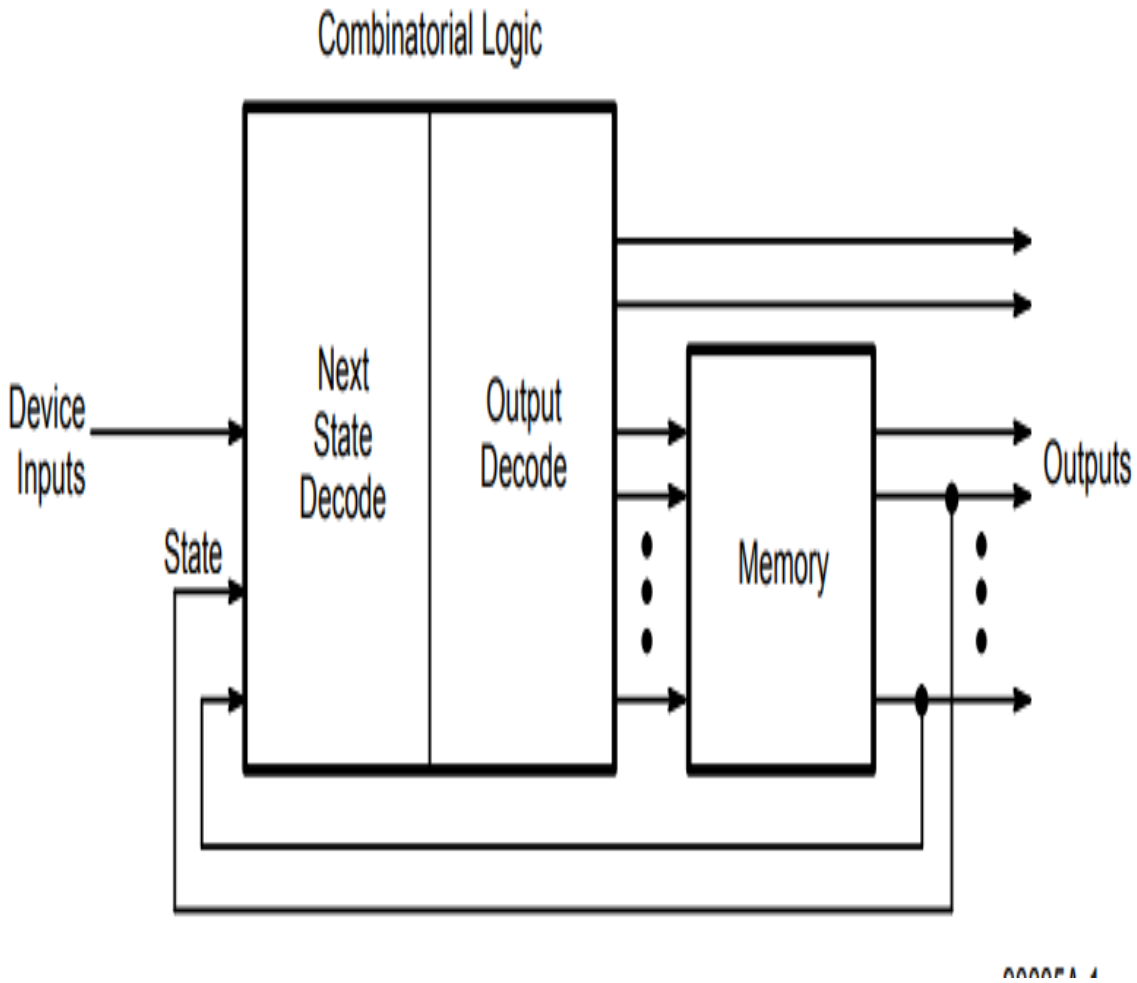
Branching: deciding the next state using input signals.

State Machine Applications

State machines are used in a number of system control applications

- **In DSP:** As sequencers for digital signal processing (DSP) applications, state machines offer speed and sufficient functionality without the overkill of complex microprocessors
- **In video controller:** It generates addresses for scanning purposes, using counters with various sequences and lengths
- **In peripheral control:** Both encoding and decoding can be translated into state machines, which examine the serial data stream as it is read, and generate the output data
- **Industrial control and robotics**: Mechanical positioning of a robot arm, simple decision making, and calculation of a trigonometric function
- **Data encryption and decryption:** A programmable state machine device with a security Bit is ideal for this because memory is internally programmed and cannot be accessed by someone tampering with the system

Finite State Machines

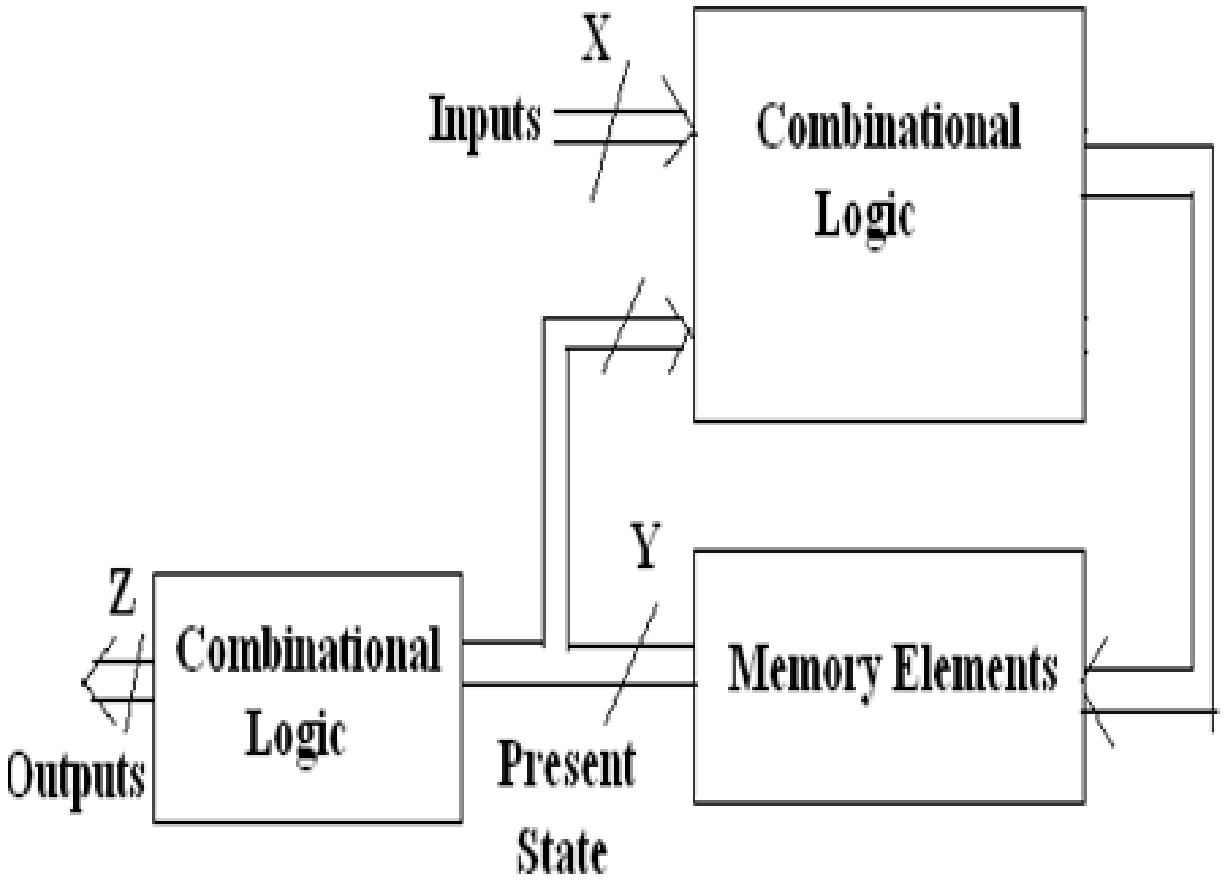


- Sequential Circuits with finite number of states are called as Finite State Machines (FSM)
- Those parts of digital systems whose outputs depend on their past inputs as well as their current ones can be modeled as finite state machines.
- When a new input is presented to the FSM, an output is generated which depends on this input and the present state of the FSM, and the machine is caused to move into new state, referred to as the next state
- The internal state is stored in a block labeled “memory.”

Types of FSMs

- There are **two different ways to express the FSMs** with respect to the output.
- Sequential circuits in which outputs are function of only present state are called **Moore Machines**
- Sequential circuits in which outputs are function of present state as well as input are called **Mealy Machines**
- **For either type, the control sequencing depends upon both states and input signals**

Moore machines

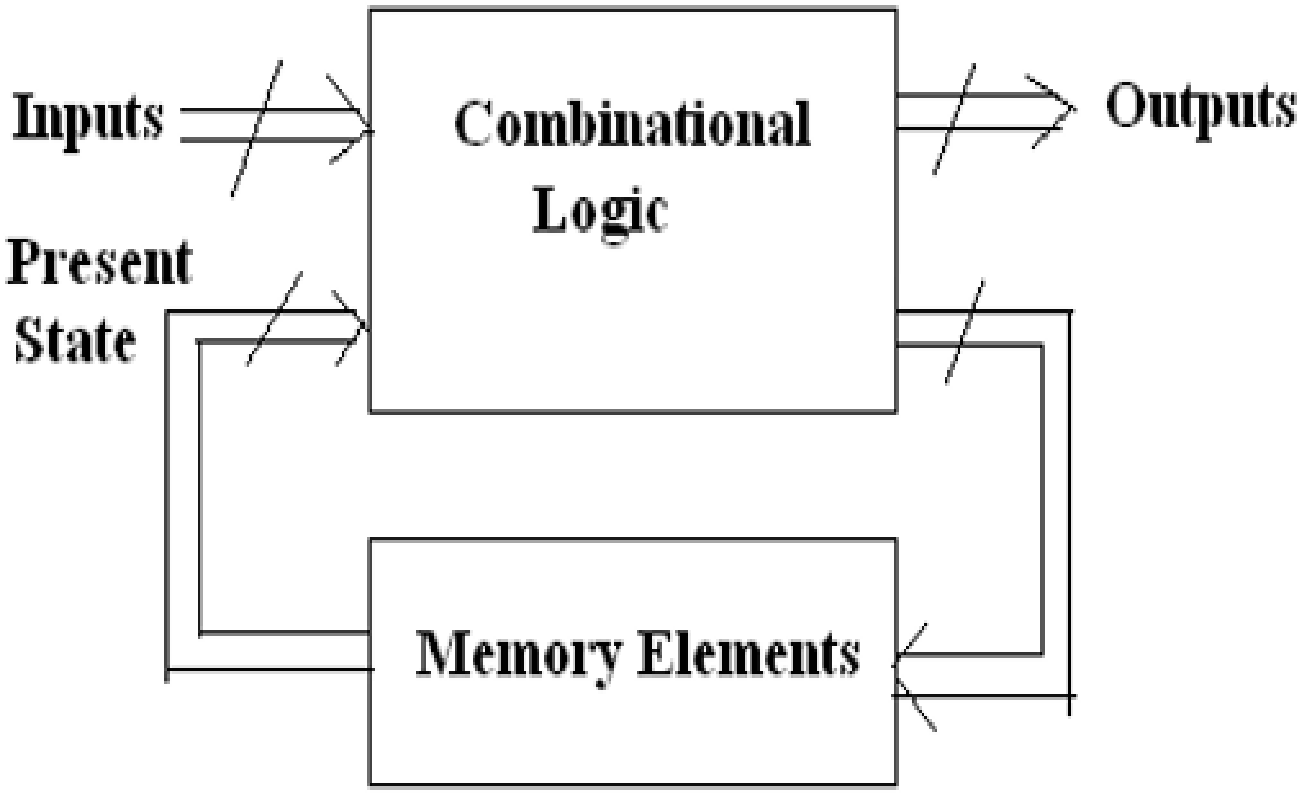


Moore machine

Outputs are a function of current state

Outputs change synchronously with state changes

Mealy machine

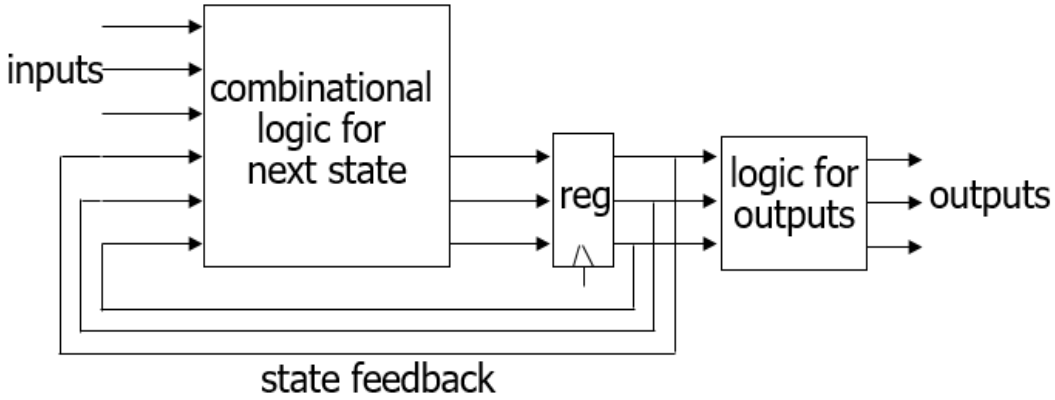


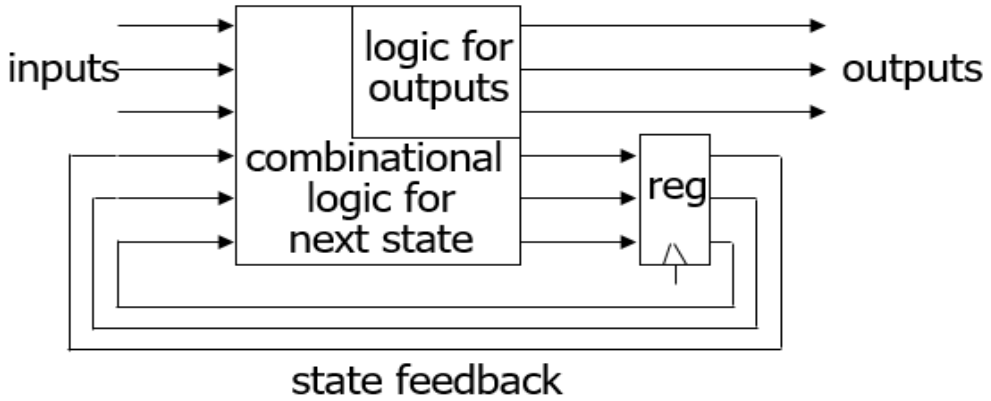
- **Mealy machine**

- Outputs depend on state and on inputs

- Input changes can cause immediate output changes

Moore versus Mealy machines





Moore machine

Outputs are a function of current state

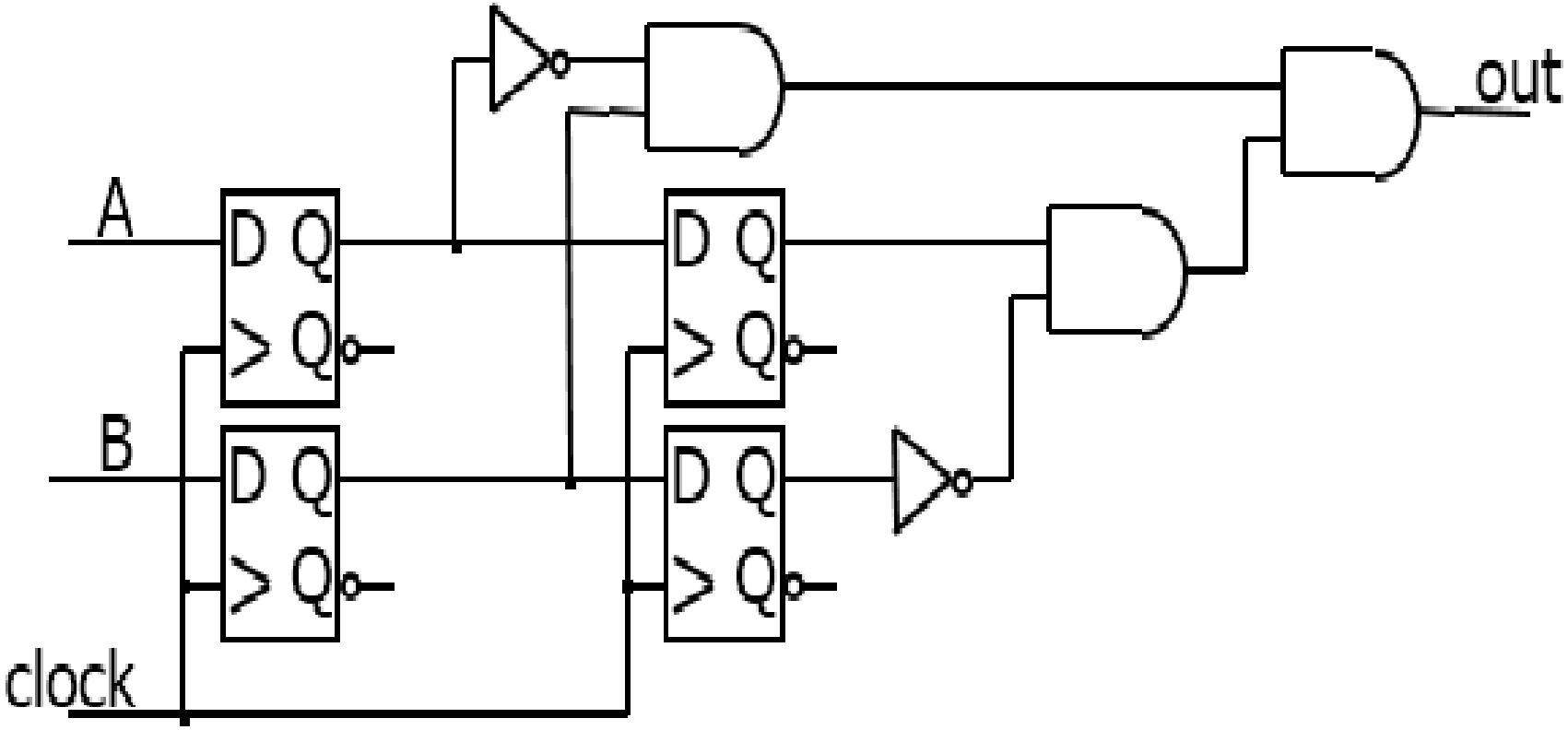
Outputs change synchronously with state changes

Mealy machine

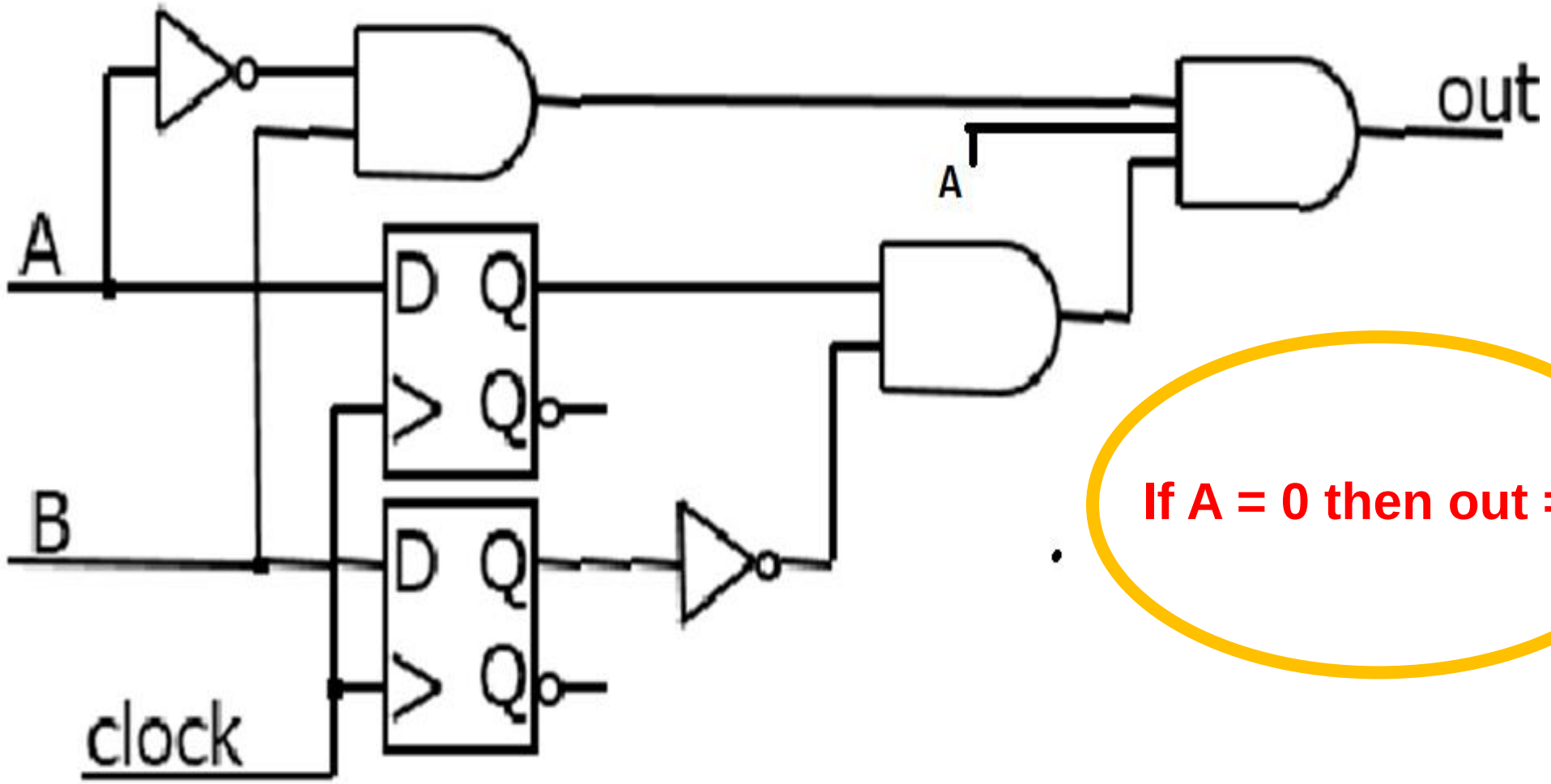
Outputs depend on state and on inputs

Input changes can cause immediate output changes

Example Moore Machine



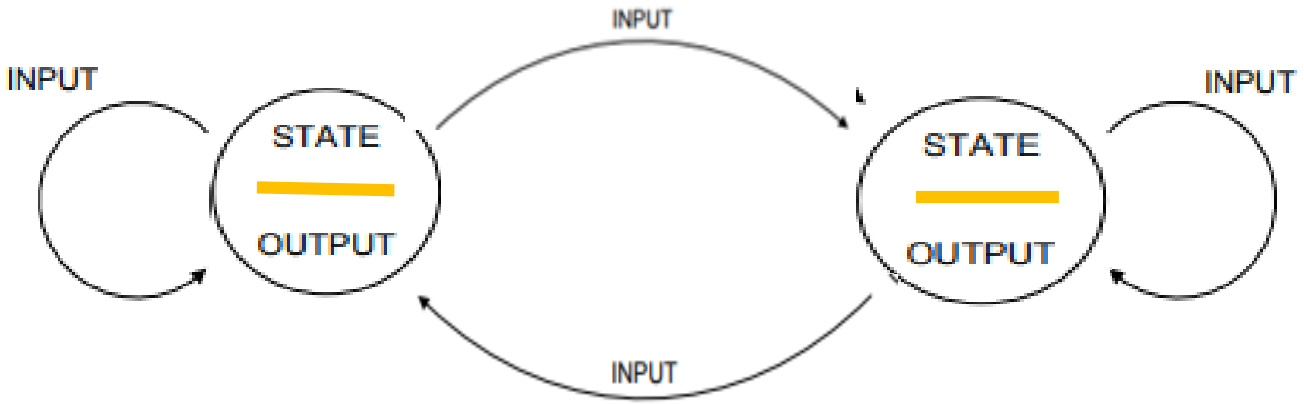
Example of Mealy Machine



If A = 0 then out = 0

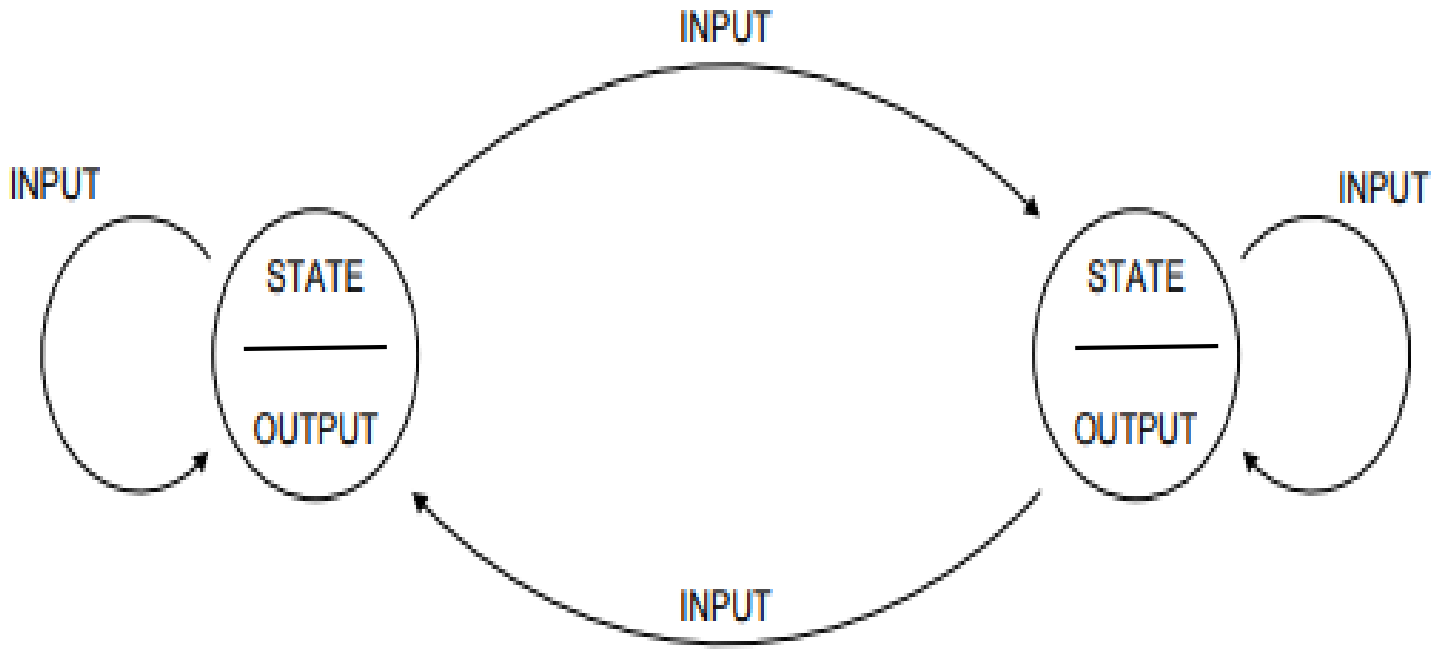
State diagram notations: Moore machine

Each state is labeled by a pair: **state-name / output** or **state-name [output]**



State diagram notations: Moore machine

Each state is labeled by a pair: state-name / output or state-name [output]

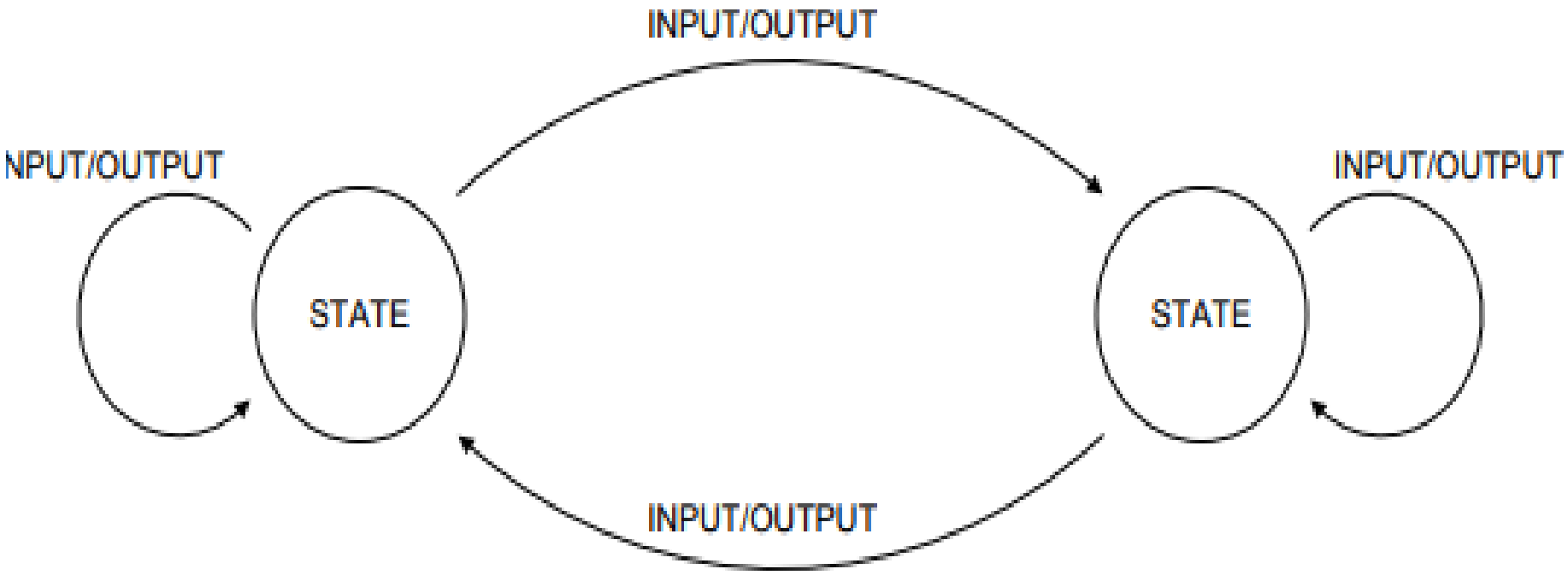


GENERIC MOORE STATE MACHINE

State diagram notations: Mealy machine

Each transition arc is labeled by a pair:

input-condition / output



GENERIC MEALY STATE MACHINE

Sequence detector circuit

To detect the start of the transmission or detection of sender etc, it is required to detect some specific sequence of 1s and 0s.

Sequence detector is a circuit whose output becomes 1 whenever the specified sequence is detected.

The sequence can be considered as overlapped or non overlapped.

Example: Let us detect sequence 010.

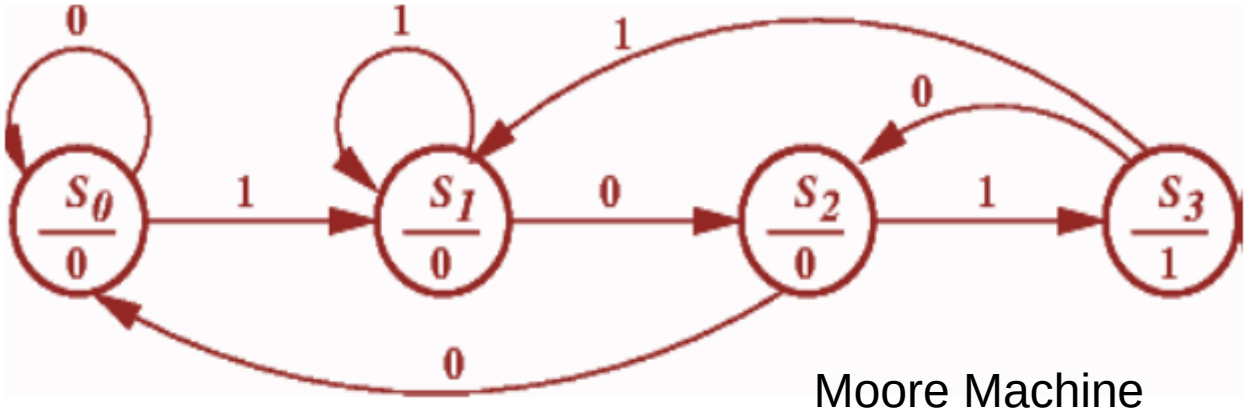
- **Z1 is the output if overlapping condition is considered and**
- **Z2 is output if sequence is considered in non overlapped mode.**
- **X is the input sequence received**

X	=	0	0	1	1	0	1	0	1	0	0	0
Z1	=	0	0	0	0	0	0	1	0	1	0	0
Z2	=	0	0	0	0	0	0	1	0	0	0	0

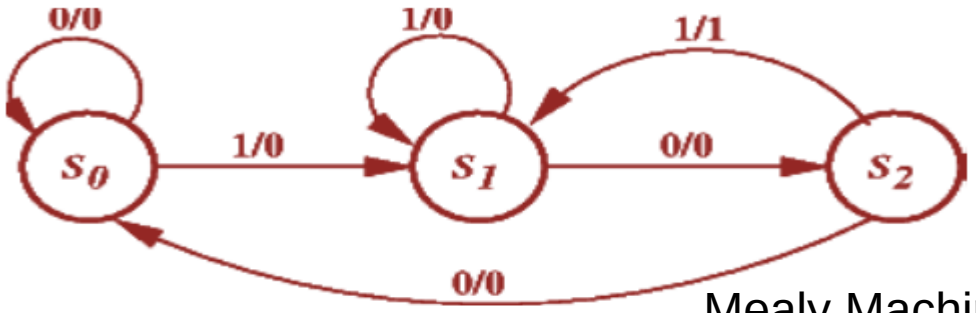
Overlapping

Non Overlapping

Sequence detector for the *sequence 101* using both Mealy and Moore Machine



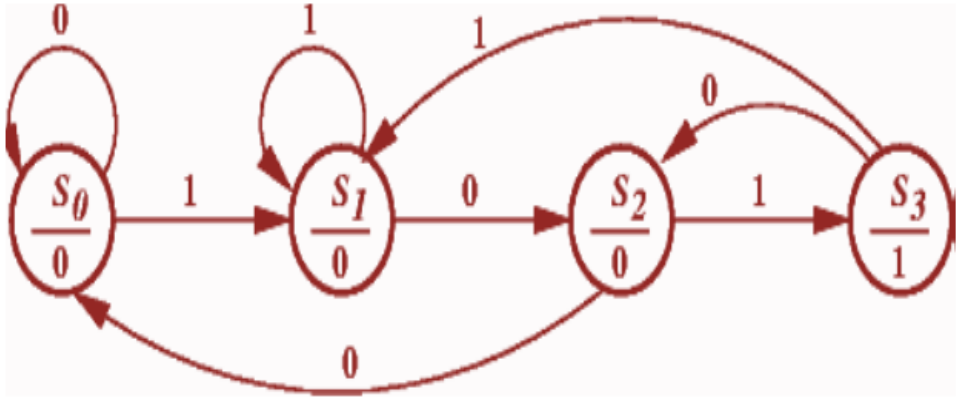
Moore Machine



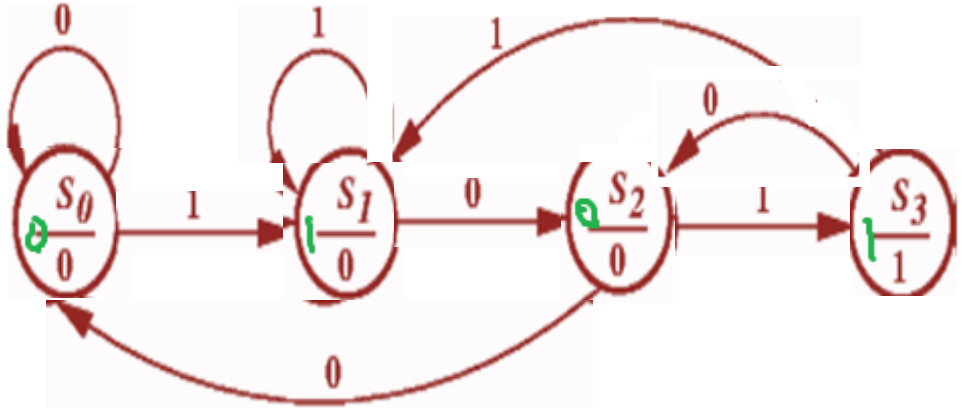
Mealy Machine

Sequence detector for the *sequence 101* using Moore Machine

Sequence: 101; *Length of sequence = 3 bits; No. of states = 4; State Names: S0, S1, S2, S3*



Moore Machine



Values within states with green colour are written only for ref. not to be written as standard convention

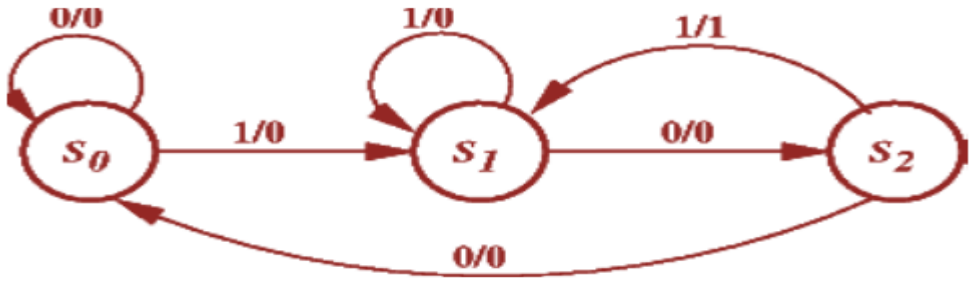
Initial Stage = S0 [S0/0]
At S0 --On occurrence of 1; make transition to S1
At S0 – On Occurrence of 0; Remain in So only

At S1 Stage [S1/0]
At S1 --On occurrence of 0; make transition to S2
At S1 – On Occurrence of 1; Remain in S1 only because S0 is state of 0 (Highlighted with green)

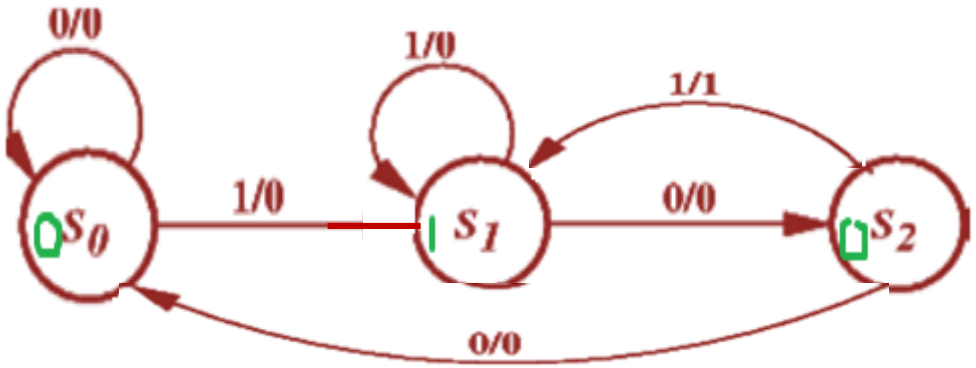
At S2 Stage [S2/0]
At S2 --On occurrence of 1; make transition to S3
At S2 – On Occurrence of 0; either remain in S2 or make transition to S0. Possibility of detecting new sequence is very high if make transition to S0

At S3 Stage [S3/1]
At S3 --On occurrence of 1; make transition to S1 as it is state of 1 and new sequence can be found as 101.
At S3 – On Occurrence of 0; either make a transition to S2 for overlap sequence detection or make transition to S0 for non overlapped sequence.

Sequence detector for the *sequence 101* using Mealy Machine



Mealy Machine



sequence :101

Length of sequence = 3 bits

No. of states = 3

State Names: S0, S1, S2

Types of state machine representations

Three ways of representing state machines:

- **State diagrams**
- **State transition tables**
- **ASM charts: Another popular notation is similar to flowcharts called as Algorithmic State Machines.**
- All these representations are **equivalent and interchangeable**, since they all describe the same hardware structure. Each style has its **own particular advantages.**

State Transition Table Representation

- A second method for state machine representation is the tabular form known as the state transition table

Present State	Inputs	Next State	Outputs Generated
S0 – Sn	I0 – Im	S0 – Sn	O0 – Op

- State transition table has
  - All the possible input bit combinations as present state
  - Each row gives the next state, Next output
- State transition table is not suitable for specifying practical machines in which there is a large number of inputs, since each input combination defines a row of the table. For example, with 10 inputs, 1024 rows would be required!

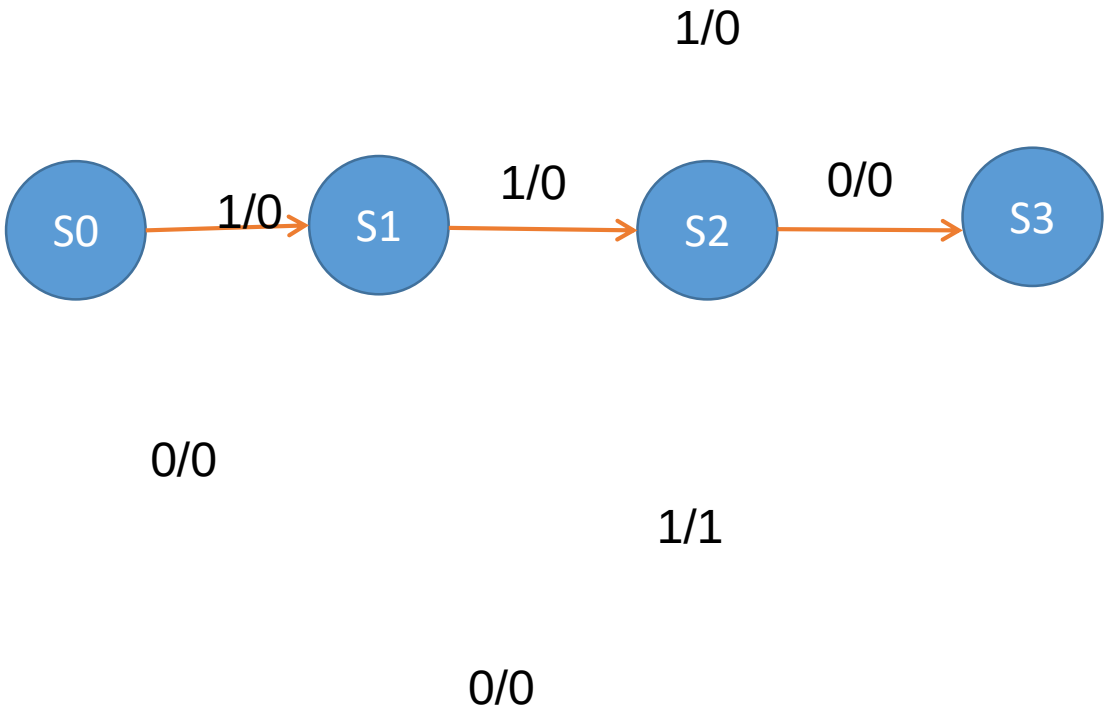
Example:

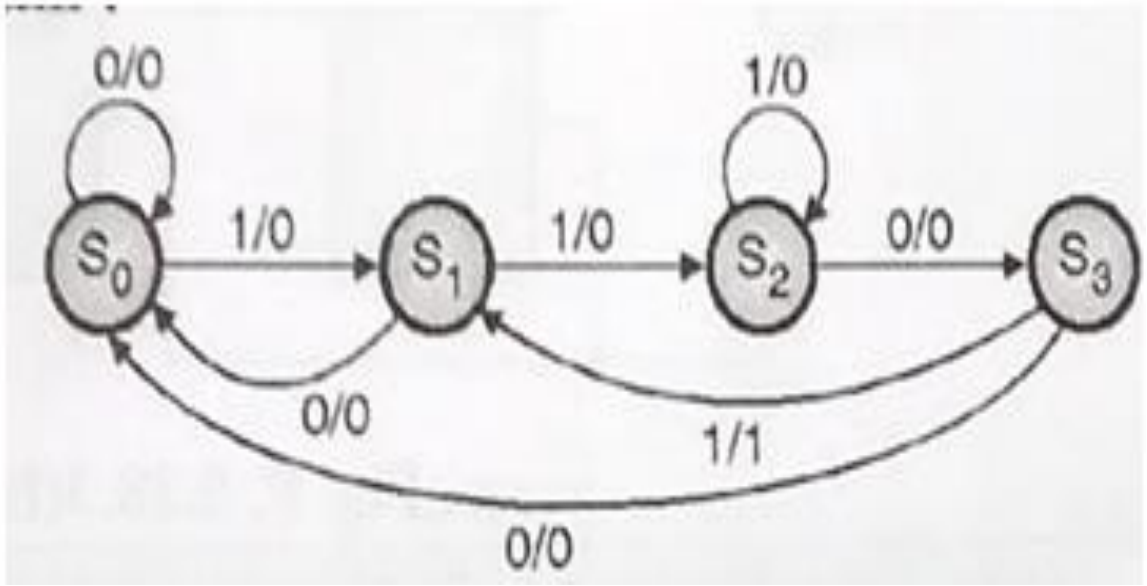
Design a sequence detector todetect the bit sequence **1101** using Mealy Machine.

Sequence detector : sequence 1101using Mealy Machine

1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

State diagram: Sequence detector 1101 Mealy Machine

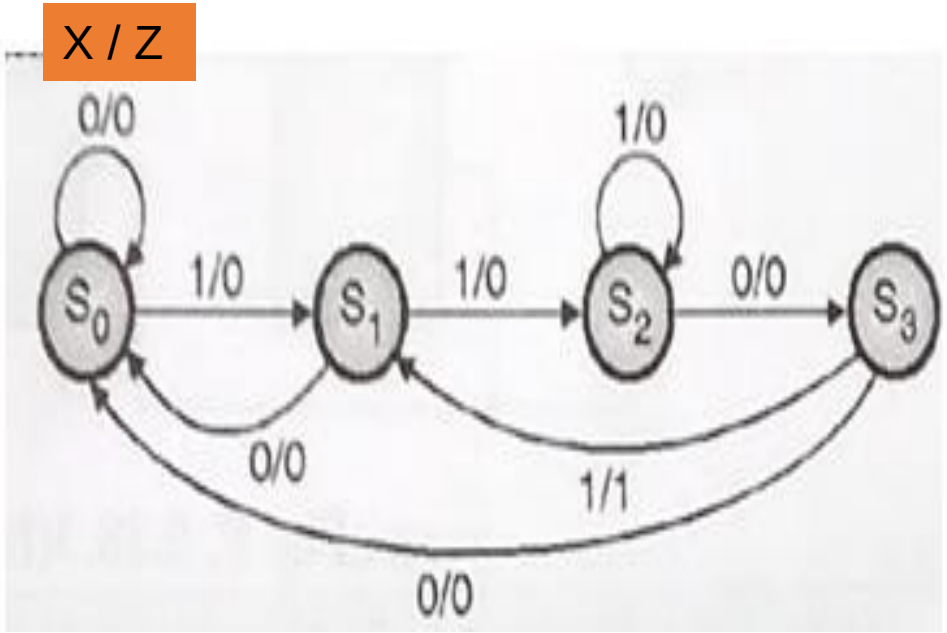




1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

State Table: Sequence detector 1101

Present state	Next State		Output (Z)	
	X = 0	X = 1	X = 0	X = 1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1



1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

State assignment: Sequence detector 1101

State assignments

$S_0 = 00$

$S_1 = 01$

$S_2 = 11$

$S_3 = 10$

1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

State Table with state Assignment

Present state	Next State		Output	
	X = 0	X = 1	X = 0	X = 1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

Present state	Next State		Output	
	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Sequence detector 1101

1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

Excitation table: Sequence detector 1101

Present State QA QB	Next State X = 0 QA QB	Next State X = 1 QA QB	Output Z X = 0 Z	Output Z X = 1 Z
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1

Present State QA	Present State QB	Input X	Next State QA_{n+1}	Next State QB_{n+1}	Flip Flop Inputs D_1	Flip Flop Inputs D_2	Output Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1
1	1	0	1	0	1	0	0
1	1	1	1	1	1	1	0

Excitation Table of D flip flop

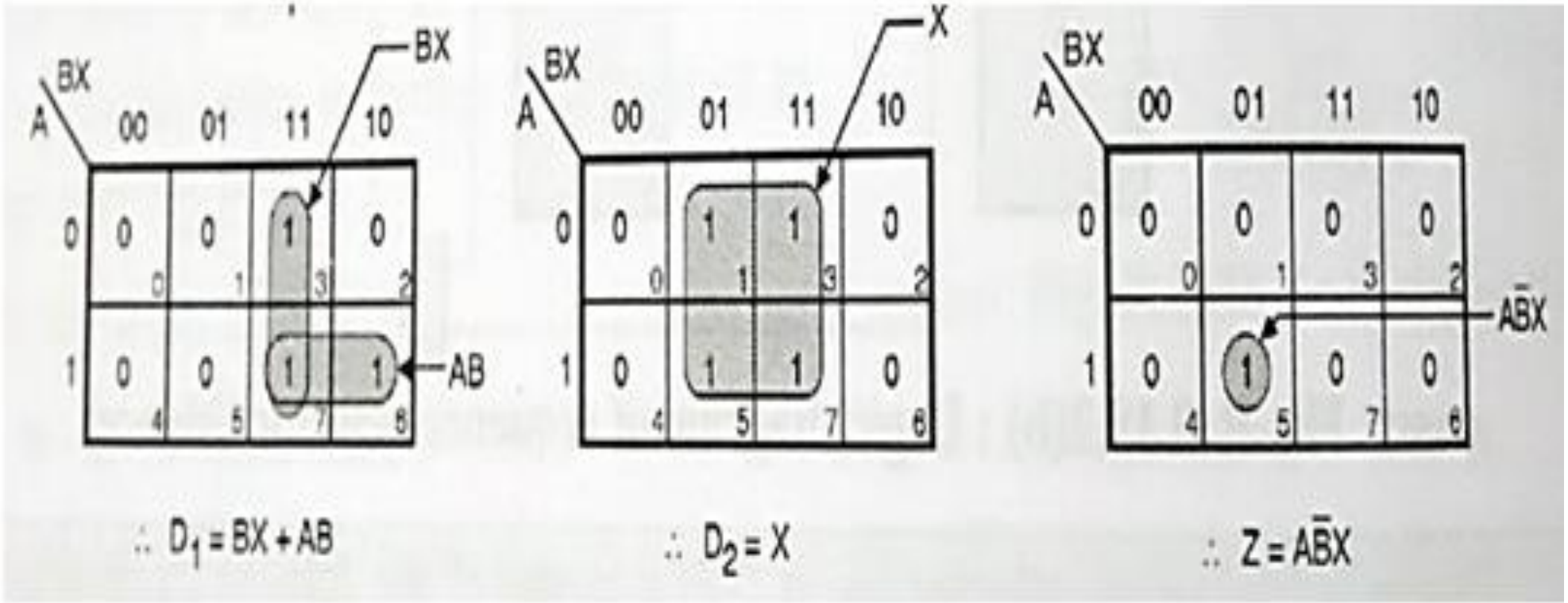
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Sequence detector 1101

1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

K-maps: Sequence detector 1101

Flip Flop Inputs		Output
D_1	D_2	Z
0	0	0
0	1	0
0	0	0
1	1	0
0	0	0
0	1	1
1	0	0
1	1	0

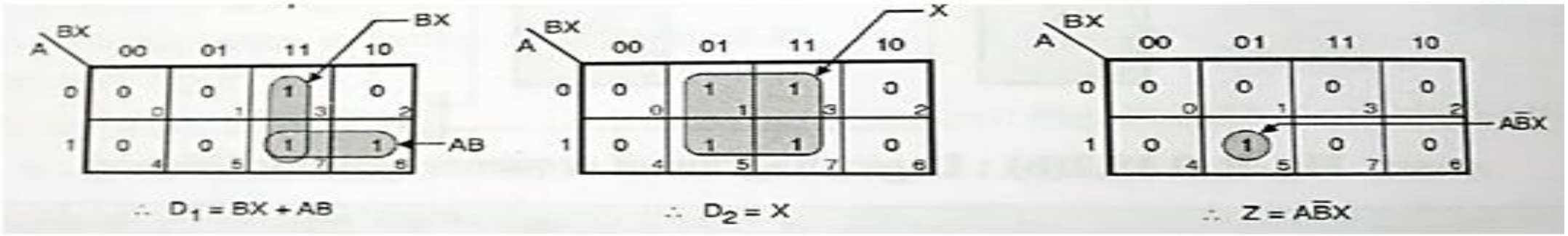


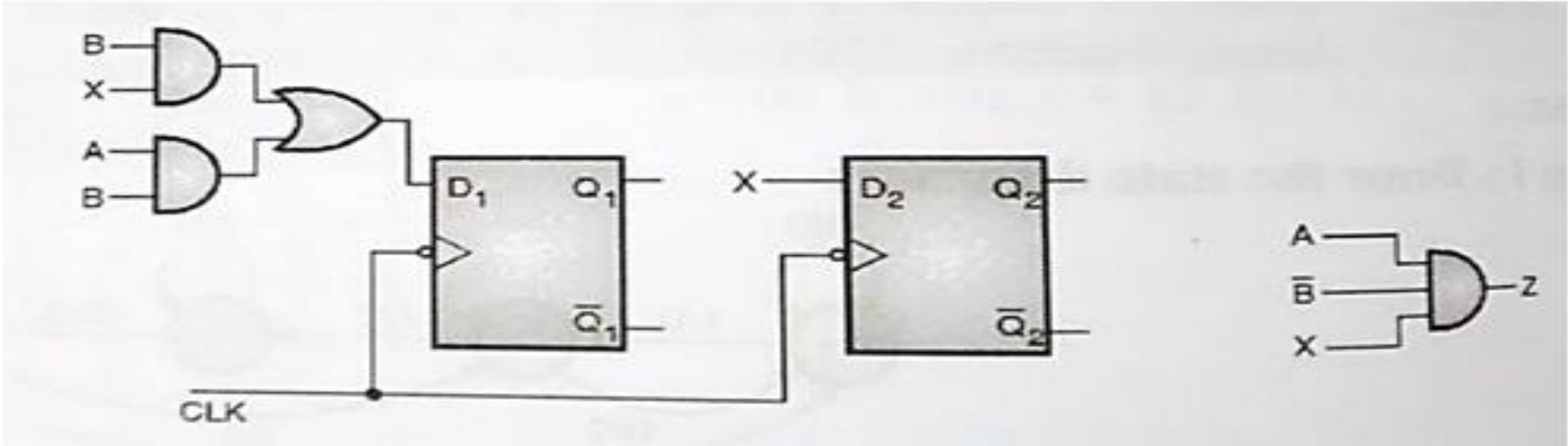
A=QA and B= QB

Sequence detector 1101

1. Draw state diagram
2. Make a state table
3. Assign states
4. Re-write the state table using states assigned.
5. Find number of Flip Flops required
6. Write Excitation table
7. Use k-maps to find inputs to Flip Flop
8. Draw the circuit diagram.

Circuit Diagram: Sequence detector 1101





Comparison of Moore and Mealy Machine

Moore Machine	Mealy Machine
Output of Moore machine only depends on its current state and not on the current input	Mealy machine changes its output based on its current input and present state
Output is placed on state	From presentation point of view, output is placed on transition
Moore machine may be safer to use, because they change states on the clock edge	Mealy will be faster, in the sense that output will change as soon as an input transition occurs
Both output and state change synchronous to the clock edge	Asynchronous output generation though the state changes synchronous to the clock
Predictable	Faster
In general needs more states for synthesis. Advantage of Moore model is simplification of behavior and easy to design	Generally needs less states for synthesis. So less hardware required to design. Less states doesn't always mean simpler to implement