Presentation Topic

Analog and Digital Electronics

Yogesh K Sharma
yogesh.sharma@viit.ac.in
Department of Computer Engineering

BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

Sequential Logic
Unit-3

Course Objectives

1. To learn and understand basic digital circuit design techniques
2. To study the implementation of digital circuits using combinational logic.
3. To study the implementation of digital circuits using sequential logic.
4. To illustrate the concept of PLD's.
5. To study the implementation of digital circuits using VHDL.
6. To understand basics of Logic Families and IOT circuit boards in development of mini digital circuits.

Course Outcome

1. Simplify Boolean algebraic expressions for designing digital circuits using K-Maps.(Analyzing)
2. Apply digital concepts in designing combinational circuits.(Applying)
3. Apply digital concepts in designing sequential circuits. (Applying)
4. Implement digital circuits using PLA and PAL. (Applying)
5. Design digital circuits using VHDL. (Applying)
6. Design and implementation of Mini digital circuit applications. (Applying)

.

Content

Unit-3: Sequential Logic.

- **Flip-flop:** SR, JK, D, T; Preset & Clear, Master and Slave Flip Flops, Truth Tables and Excitation tables, Conversion from one type to another type of Flip Flop.

- **Registers:** Buffer register, shift register (SISO, SIPO, PISO & PIPO), Applications of shift registers.

Sequential Logic

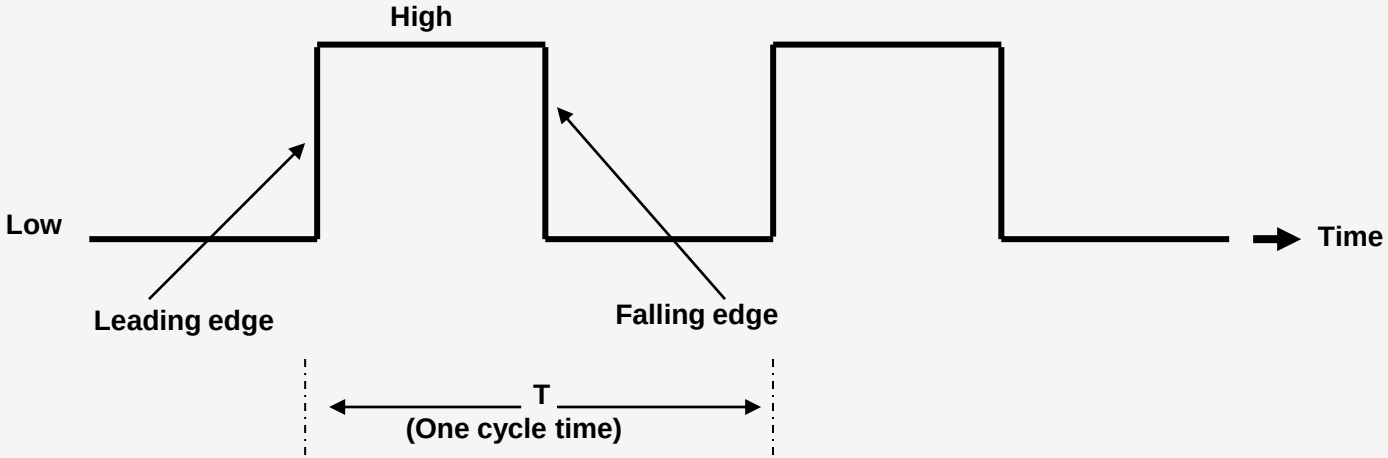
Introduction:

- Timing parameter comes into picture.
- Output depends on present time inputs, the previous output & the sequence in which the inputs are applied.
- To store a previous input or output, a memory element is required.
- The data stored by the memory element at any given instant of time is called as the present state of the sequential circuit.
- Combinational circuit operates on the external inputs and the present state to produce new outputs. Some of these new outputs are stored in the memory element and called as the next state of the sequential circuit.
- Memory element is the most important part of sequential circuits and is known as flip flop (FF). It is the basic memory element.

Sequential Logic

Clock Signal:

- **It is timing signal. Every sequential signal will have this timing signal applied.**
- **It is rectangular signal, with a duty cycle equal to 50%.**
- **Clock signal repeats itself after every T seconds. Hence the clock frequency if f = 1 / T.**



Sequential Logic

Clock Skew:

- **It is defined as the difference in time between the clock edges arriving at a pair of clock inputs.**
- **In a perfect system, all clock signals arrive at all the various clock input pins of the system at exactly the same time & the skew is zero.**
- **In real time systems, the edges do not arrive at exactly the same time & there is some skew.**
- **Maximum allowable clock skew for the system is the difference between the shortest and longest path delays.**
- **It is an important design parameter in high-speed clock systems.**
- **It appears due to different delays on different paths from the clock generator to various circuits. The major reasons for this are:**
  - **Different length of wires**
  - **Gates (buffers) on the paths.**
  - **Flip-flops that clock on different edges**
  - **Gating the clock to control loading of registers**

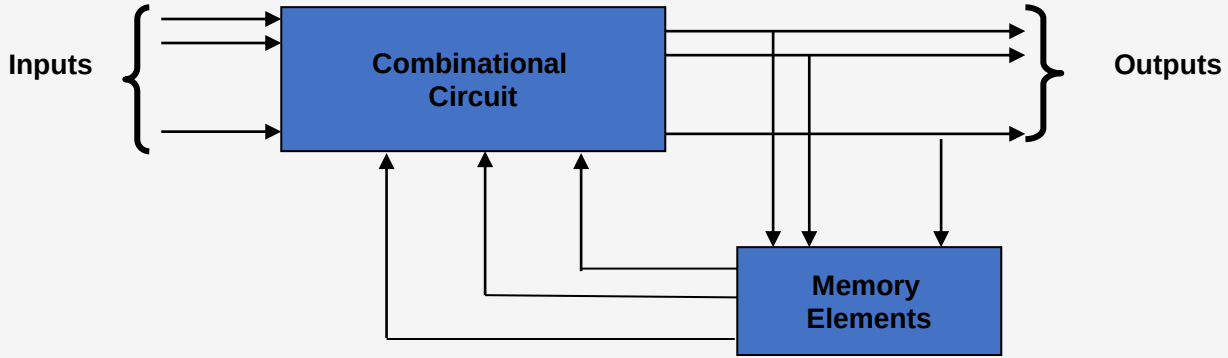
Sequential Logic

Comparison of Combinational & Sequential Circuits:

No.	Parameter	Combinational	Sequential
1.	Output depend on	Inputs present at that instant of time	Present inputs & past inputs/outputs
2.	Memory	Not necessary	Necessary
3.	Clock Input	Not necessary	Necessary
4.	Examples	Adders, Subtractors, Code converters	Flip flops, shift registers, counters

Sequential Logic

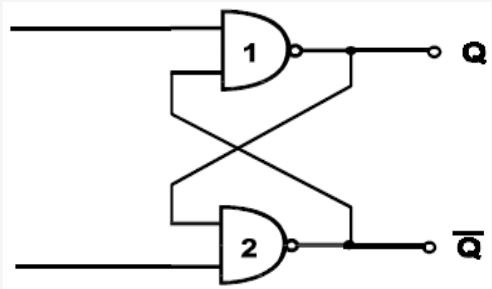
- Sequential circuit consists of combinational circuit along with the memory element.
- Memory element will always appear in the feedback path.
- Output of sequential circuit depends on the present state as well as the past state.
- Present state is stored in the memory elements.
- Present states are applied to the combinational circuit along with the external inputs.
- Thus outputs & the next state of the sequential circuit are decided by the external inputs & the present state.



Sequential Logic

1-Bit Memory Cell:

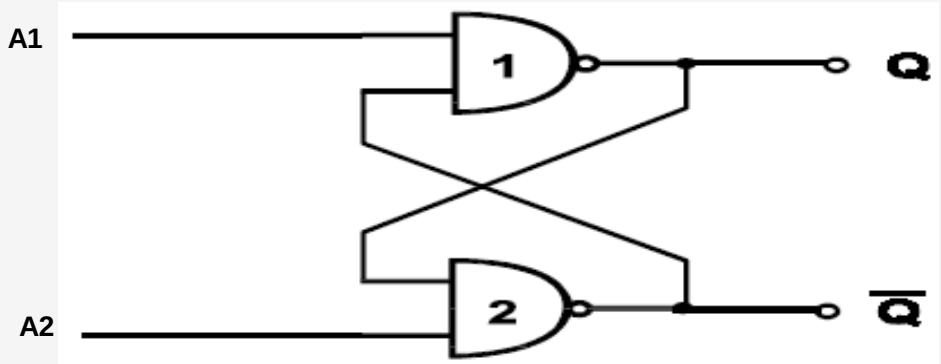
- **Basic digital memory circuit is also known as Flip-flop.**
- **It has two stable states namely logic 1 state and logic 0 state. It can be designed either using NOR gates or NAND gates.**
- **A Flip-flop can be designed using NAND gates 1 and 2, where these two gates G1 & G2 act as inverters. Hence this circuit is called as a cross coupled inverter.**
- **Output of G1 is connected to the input of G2 & output of G2 is connected to input of G1.**
- **Assume Q=1 then input for G2 i.e. B2=1**
- **As B2=1, output of gate-2 ie. G2 i..e $\overline{Q}$=0. This makes B1=0.**
- **Hence Q continues to be equal to 1.**
- **Similarly, if we start with Q=0, then we end up obtaining Q=0 and $\overline{Q}$=1.**



Sequential Logic

Cross Coupled Inverters:

- Outputs Q & Q bar are always complimentary.
- The circuit has two stable states, in one stable state Q=1 & Q bar=0 and it is called as 1 state or set state. Whereas the other Q=0 & Q bar=1 is called as 0 state or reset state.
- If circuit is in 1 reset state, then it will continue to be in the reset state and if it is in the set state then it will continue to remain in the set state.
- This property of circuit shows that it can store 1 bit of digital information and hence known as 1 bit memory cell.



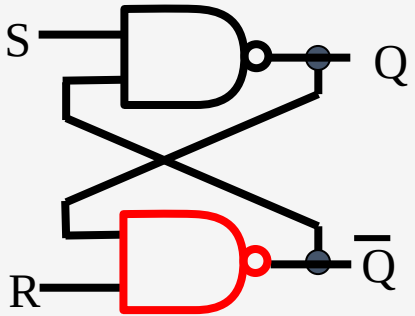
Sequential Logic

Latch:

- The cross coupled inverter is capable of locking or latching the information, hence is known as a latch.
- Disadvantage of this circuit is that we cannot enter the desired digital data into it.
- This disadvantage can be overcome by modifying the circuit, which allows us to enter the desired digital data into the circuit.
- In latch there is no way of entering the desired digital information to be stored in it. Since when the circuit is switched ON, the circuit switches to one of the stable states. Thus additional gates are added to enter the desired digital information.
- The two input terminals are designated as set (S) and reset (R) because S=1 brings the circuit in set state and R=1 brings it to reset or clear state.

Bistable- Flip Flops made with NANDs

- While analyzing this circuit, as soon as one input is a Low or zero (either a direct input or a feedback input) a NAND has High (1) as an output.
- On the other hand for NORs if an input is High (1) then the output must be a Low (0).



S ‘sets’ Q HI when it is LO

R ‘resets’ Q LO when it is HI

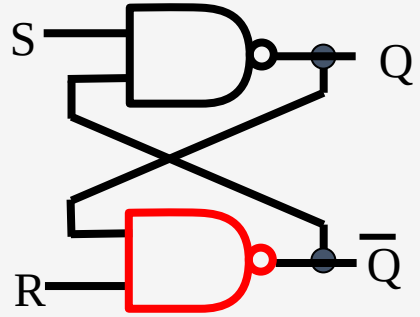
S = R = LO is a ‘disallowed’ state
what happens when S = R = HI?

State Table

S	R	Q	Q'
0	1	1	0
1	0	0	1
0	0	1	1
1	1	?	?

The last state depends on the previous state, since there are two possible output states - thus sequential logic.
What happens if you go from 00 to 11?

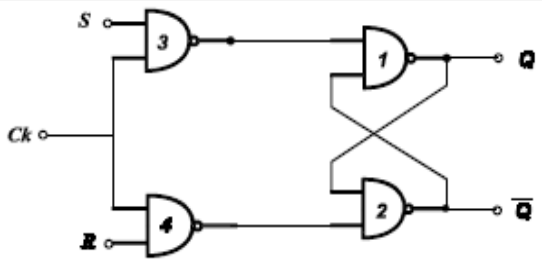
Sequential Logic



Operation:

Case I: S = R = 0:

- **Outputs of gates 1 & 2 will become 1.**
- **Let Q=0 & Q bar=1 initially, hence both the inputs to gate 3 are 1 & the inputs to gate 4 are (01).**
- **Hence gate-3 output i.e. Q=0 & gate-4 output Q bar=1.**
- **Thus with S=R=0, there is no change in the state of outputs.**



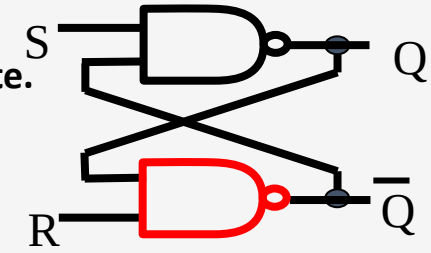
Case II : S = 1, R = 0:

- **Since S=1 & R=0, one of the inputs to gate-3 will be 0. This will force Q output to 1.**
- **Hence both the inputs to gate-4 will be 1. This forces Q to 0.**
- **Hence for S=1, R=0 the outputs are Q=1 & Q=0. This is set state.**

Sequential Logic

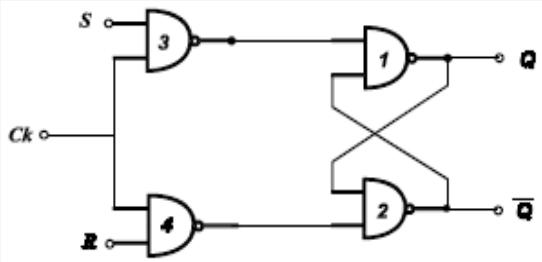
Case III: S = 0, R = 1:

- **If S=0, R=1 then one of the inputs to gate-4 will be 0. This will force the Q bar output to 1.**
- **Hence both the inputs to gate-3 will be 1. This forces Q bar to 0.**
- **Thus for S=0, R=1, the outputs are Q=0, Q bar=1. This is the reset state of clear state.**



Case IV: S = R = 1:

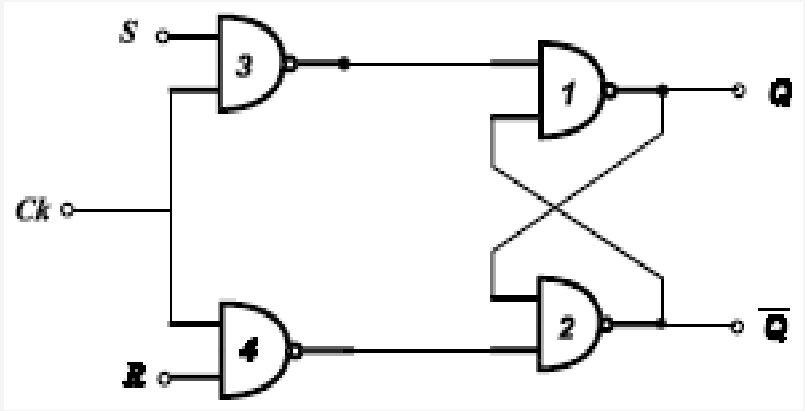
- **If S=R=1 then outputs of gates-1 and 2 will be zero.**
- **Hence one of the inputs to gate-3 and 4 will be 0.**
- **So outputs Q & Q bar both will try to become 1. It is not allowed as Q and Q bar should be complementary. Hence S=R=1 condition is prohibited.**



Sequential Logic

Clocked S-R Flip-Flop:

- **It is required to set or reset the memory cell in synchronous with a train of pulses known as clock (known as CK), & such circuit is referred to as a clocked set-reset (S-R) Flip-Flop.**
- **If CK=1, then it performs the operation exactly the same as that of S-R Flip-Flop, and if CK=0 the gates 3 & 4 are inhibited. i.e. their outputs are 1 irrespective of the values of S or R.**
- **The circuit responds to the inputs S & R only when the clock is present.**



The Truth Table for the S-R clocked flip-flop is shown below.

S_n	R_n	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	?

Sequential Logic

Preset & Clear:

- **In clocked S-R Flip-Flop, when the power is switched on, the state of the circuit is uncertain. It may come to set (Q=1) of reset (Q bar=0) state.**
- **In many cases the FF is to be set or reset i.e. the initial state of the FF is to be assigned. This is achieved by asynchronous inputs, referred to as preset (Pr) and Clear (Cr) inputs.**
- **These inputs may be applied at any time between clock pulses and are not in synchronism with the clock.**
- **If Pr=Cr=1 the Ckt operates in accordance with the T.T of S-R FF.**
- **If Pr=0 & Cr=1, the output of G1(Q) will be 1. Thus all the 3 inputs to G2 will be 1 which make Q bar=0, hence Pr=0 sets the FF.**
- **If Pr=1 & Cr=0, the FF is reset, Once the state of the FF is established asynchronously, the asynchronous inputs Pr & Cr must be connected to logic 1 before the next clock is applied.**
- **The condition Pr=Cr=0 must not be used, since this leads to an uncertain state.**
- **Thus in logic symbol, bubbles are used for Pr & Cr inputs, which means these are active low, i.e. the intended function is performed when the signal applied to Pr & Cr is LOW.**

Sequential Logic

Summary of Operation of S-R Flip-Flop:

	Inputs		Output	Operation Performed
CK	Cr	Pr	Q	
1	1	1	Qn+1	Normal FF Operation
0	0	1	0	Clear
0	1	0	1	Preset

Note: The circuit can be set or reset even in the presence of the clock pulse.

Sequential Logic

J-K FLIP-FLOP:

- **The uncertainty in the state of an S-R FF, when Sn=Rn=1 can be eliminated by converting it into a J-K FF.**
- **The data inputs are J & K which are ANDed with Q & Q bar respectively, to obtain S & R inputs. i.e. S=J.Q bar & R = K. Q**
- **Truth Table for J-K Flip-Flop:**

Inputs		Output
Jn	**Kn**	**Qn+1**
0	0	Qn
1	0	1
0	1	0
1	1	Qn bar

Sequential Logic

J-K Flip-Flop:

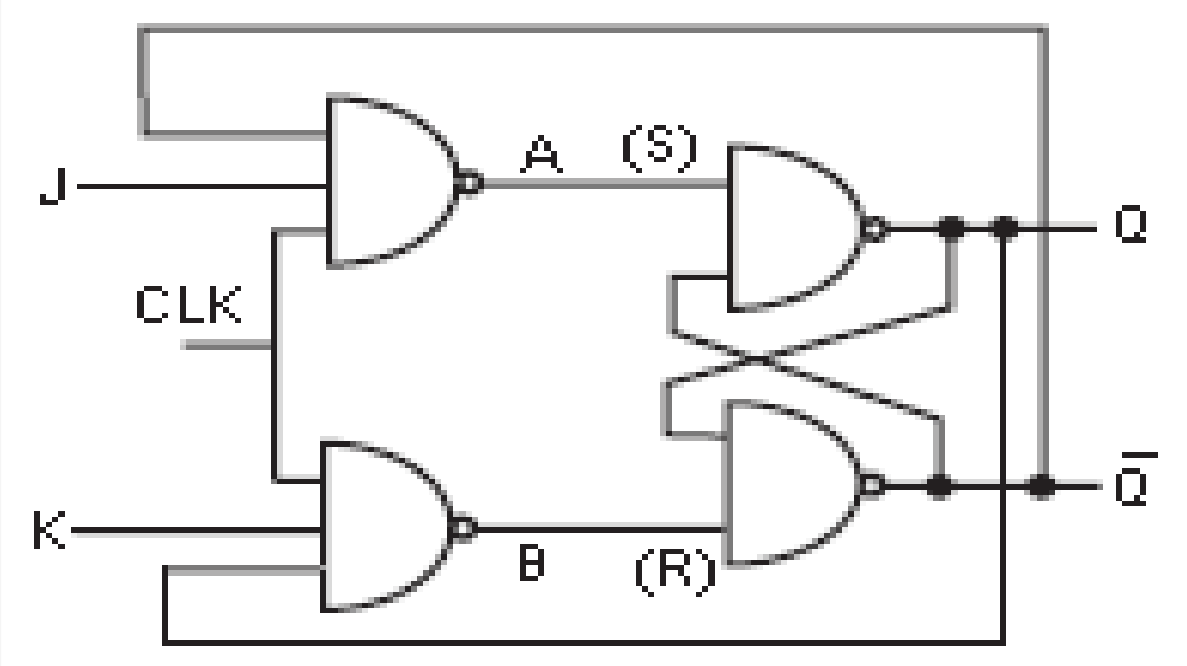


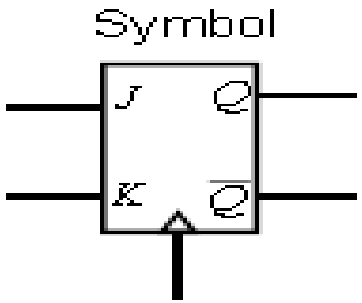
Fig. S-R Flip-Flop Converted into J-K Flip-Flop

J-K Flip-Flop

- The J-K FF is similar in function to a clocked RS FF, but with the illegal state replaced with a new 'toggle' state

J	K	Q'	$\overline{Q}'$	comment
0	0	Q	$\overline{Q}$	hold
0	1	0	1	reset
1	0	1	0	set
1	1	$\overline{Q}$	Q	toggle

Where Q' is the next state and Q is the current state



Logic Symbols of J-K FF

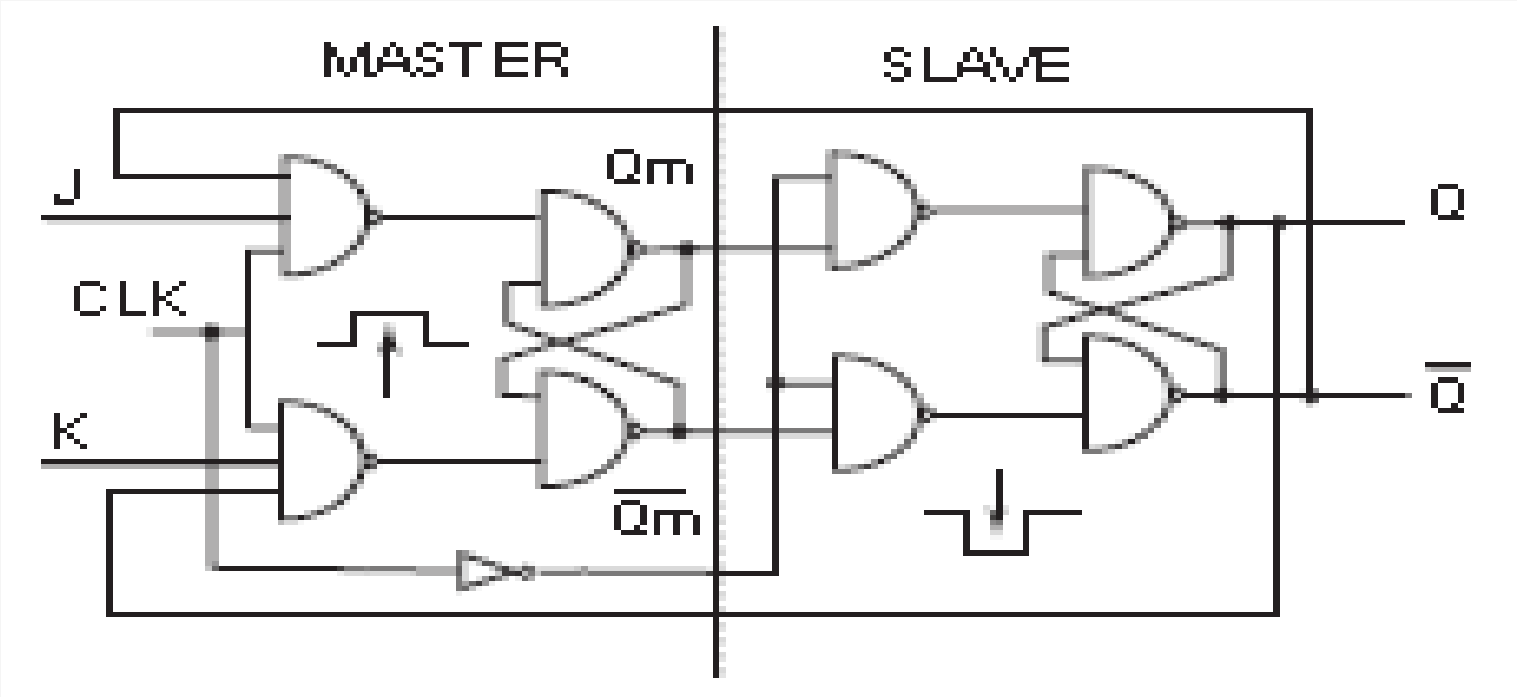
Sequential Logic

Race- Around Condition:

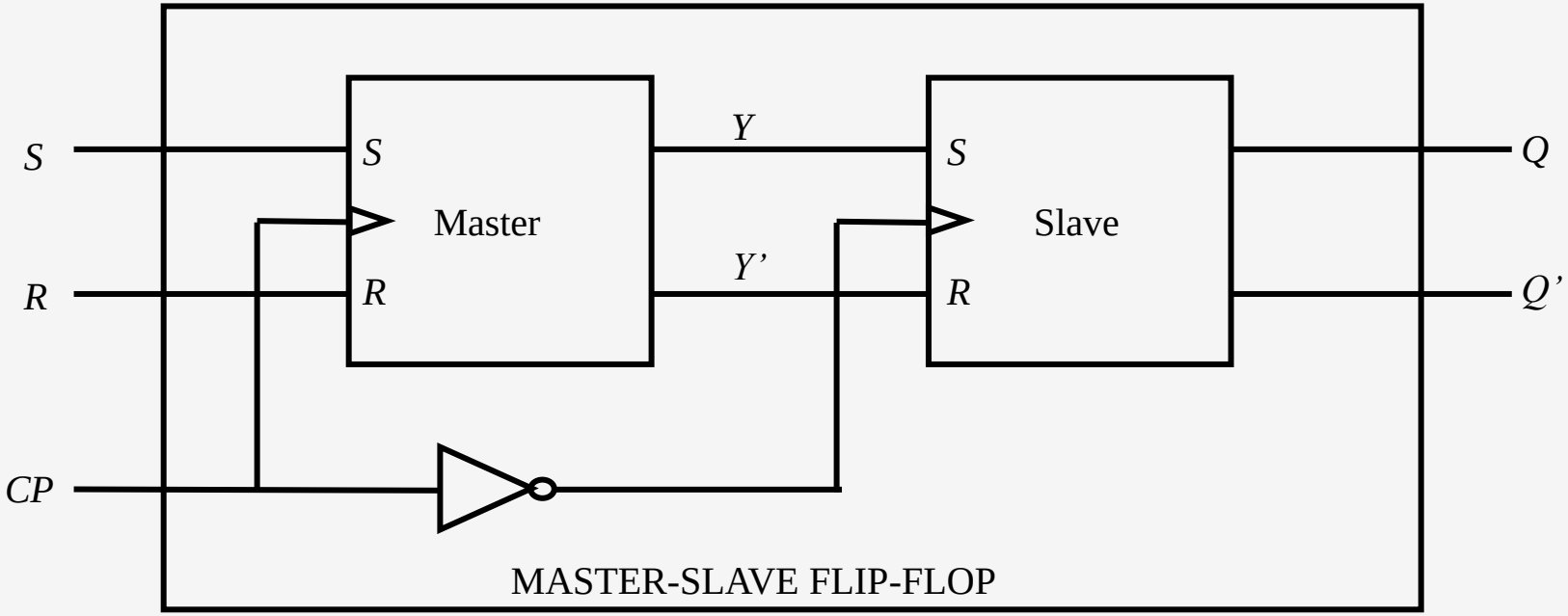
- **In J-K Flip-Flop, condition for both the inputs as 1 i.e. S=R=1 is taken into consideration by using the feedback connection from outputs to the inputs of the gates.**
- **In J-K FF, the assumption that the inputs do not change during the clock pulse (CK=1), which is not true because of the feedback connections.**
- **For e.g. the inputs J=K=1 & Q=0 and a pulse is applied at the clock input.**
- **After a time interval Δt equal to the propagation delay through two NAND gates in series, the output will change to Q=1.**
- **Now J=K=1 & Q=1 and after another time interval of Δt the output will change back to Q=0. Thus for the duration tp of clock pulse, the output will oscillate back and forth between 0 & 1, & at the end of the clock pulse, the value of Q is uncertain. This situation is called as race around condition.**
- **It can be avoided if tp< Δt <T. The best way is to use a master-slave J-K Flip-Flop.**

Sequential Logic

Master-Slave J-K Flip-Flop:

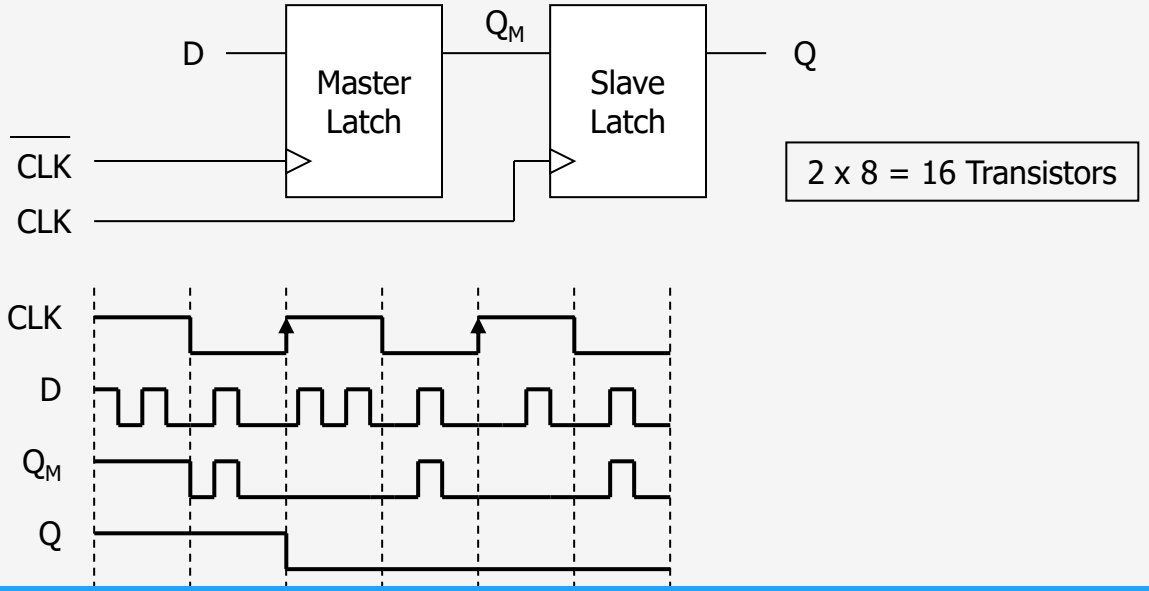


Master-Slave Flip-Flop



Master-Slave Edge-Triggered Flip-Flop

- **Can connect two level-sensitive latches in Master-Slave configuration to form edge-triggered flip-flop.**
- **Master latch "catches" value of "D" at "Q_M" when CLK is low.**
- **Slave latch causes "Q" to change only at rising edge of CLK.**



Sequential Logic

D- Type Flip-Flop:

- **It has only one input referred to as D-input or data input.**
- **The output Qn+1 at the end of the clock pulse equals the input Dn before the clock pulse.**
- **Input data appears at the output at the end of the clock pulse. Thus transfer of data from the input to the output is delayed and hence the name delay (D) FF.**
- **It is either used as a delay device or as a latch to store 1 bit of binary information.**

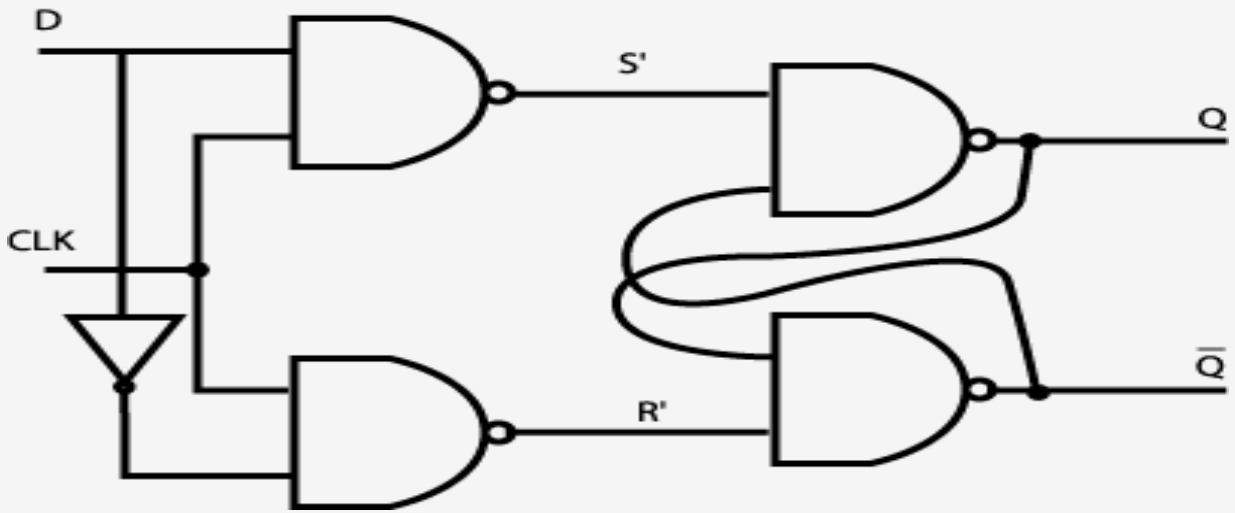
Truth Table:

Input	Output
Dn	Qn+1
0	0
1	1

Sequential Logic

D Flip-Flop:-

Preset	Clear	D	Clock	Qn+1	$\overline{Qn+1}$
1	1	0	⎍	0	1
1	1	1	⎍	1	0



Sequential Logic

T- Type Flip-Flop:

- **In J-K FF, if J=K, the resulting FF is referred to as a T-type FF.**
- **It has only one input, referred to as T-input.**
- **From the truth table of T-type FF, if T=1 it acts as a toggle switch. For every clock pulse, the output Q changes.**
- **An S-R FF cannot be converted into a T-type FF since S=R=1 is not allowed. But its logic symbol can be used as a toggle switch. i.e. the output of Q changes with every clock pulse.**

Truth Table:

Input	Output
Tn	Qn+1
0	Qn
1	Qn bar

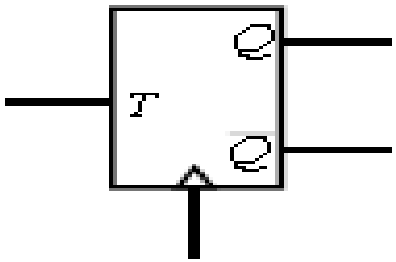
T Flip-Flop

- This is essentially a J-K FF with its J and K inputs connected together and renamed as the T input

T	Q'	$\overline{Q'}$	comment
0	Q	$\overline{Q}$	hold
1	$\overline{Q}$	Q	toggle

Where Q' is the next state and Q is the current state

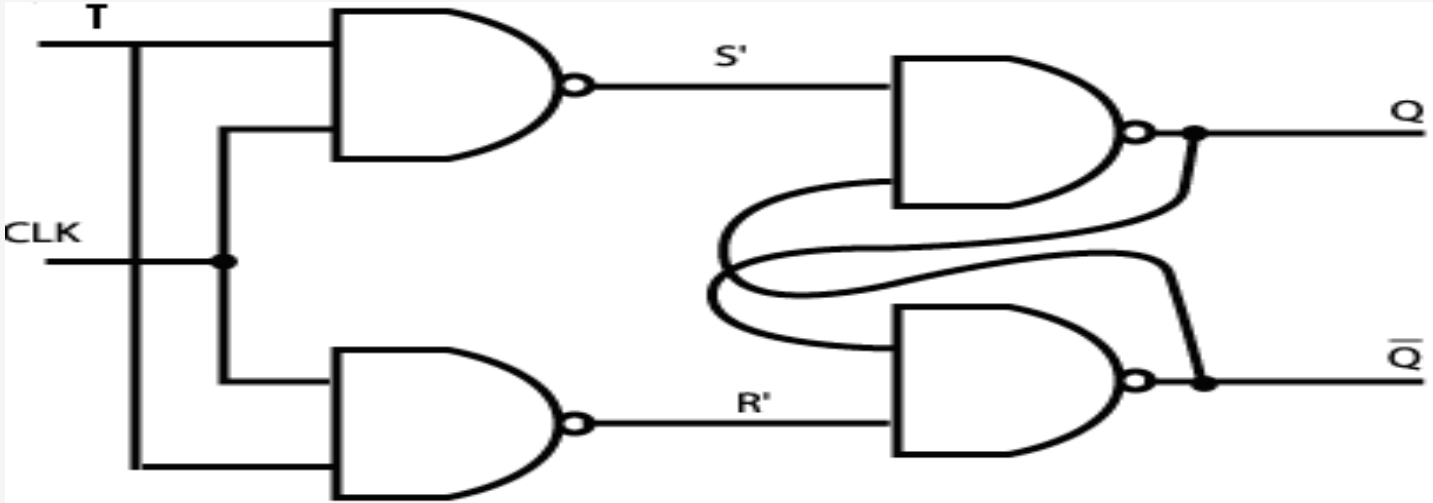
Symbol



Sequential Logic

T Flip-Flop:-

Preset	Clear	T	Clock	Qn+1	$\overline{Qn+1}$
1	1	0	⎍	Qn	$\overline{Qn}$
1	1	1	⎍	$\overline{Qn}$	Qn



Characteristic Table

- In general, a characteristic table for a FF gives the next state of the output, i.e., Q' in terms of its current state Q and current inputs

Q	D	Q'
0	0	0
0	1	1
1	0	0
1	1	1

Which gives the characteristic equation,

$$Q' = D$$

i.e., the next output state is equal to the current input value

Since Q' is independent of Q the characteristic table can be rewritten as

D	Q'
0	0
1	1

Excitation Table

- The characteristic table can be modified to give the excitation table. This table tells us the required FF input value required to achieve a particular next state from a given current state

Q	Q'	D
0	0	0
0	1	1
1	0	0
1	1	1

As with the characteristic table it can be seen that Q', does not depend upon, Q , however this is not generally true for other FF types, in which case, the excitation table is more useful. Clearly for a D-FF, $D = Q'$

Characteristic and Excitation Tables

- Characteristic and excitation tables can be determined for other FF types.
- These should be used in the design process if D-type FFs are not used

Sequential Logic

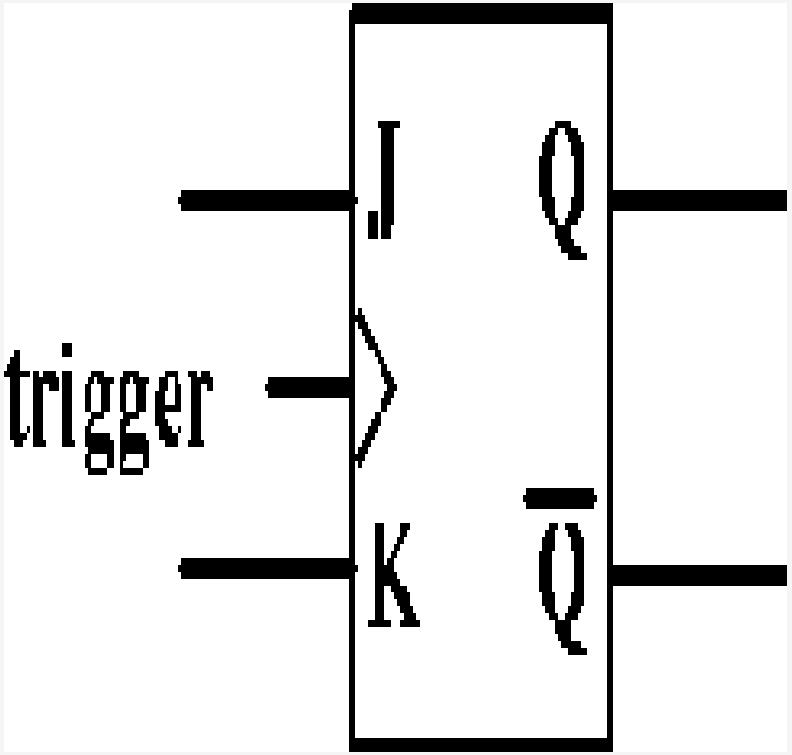
Excitation Table of Flip-Flop:

- **The truth table of a FF is referred as the characteristics table & specifies the operational characteristics of the FF.**
- **To design sequential circuits, it is needed that the input conditions corresponding to present state & next state is specified.**
- **Thus we have to find the input conditions that must prevail to cause the desired transition of the state.**
- **Present state & Next state means circuit prior to & after the clock pulse applied.**

Sequential Logic

Excitation Table of Flip-Flops:							
Present State	Next State	S-R FF		J-K FF		T-FF	D-FF
		Sn	Rn	Jn	Kn	Tn	Dn
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	1	0
1	1	x	0	x	0	0	1

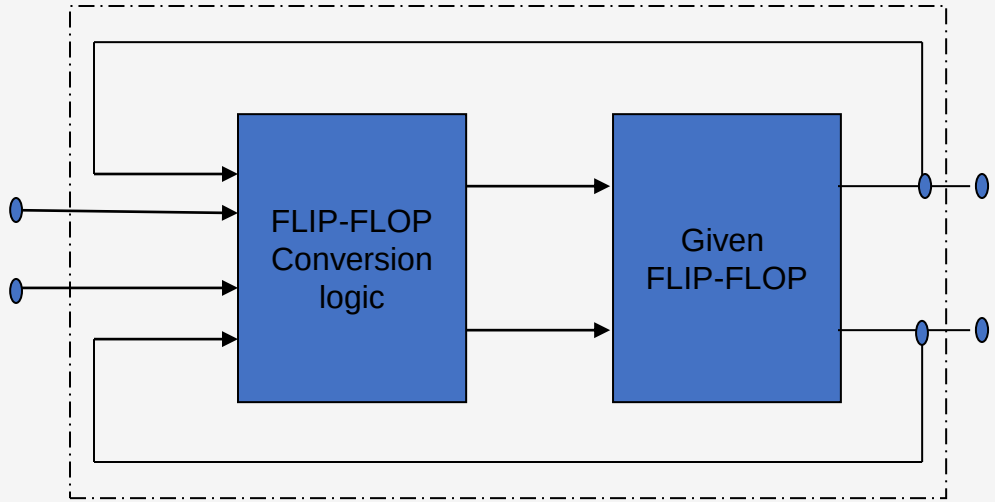
Sequential Logic



J	K	Q (t)	Q (t+1)	
0	0	0	0	no change
0	0	1	1	
0	1	0	0	clears
0	1	1	0	
1	0	0	1	sets
1	0	1	1	
1	1	0	1	toggles
1	1	1	0	

Sequential Logic

- **In general model for conversion from one type of FF to another type, we design the combinational logic decoder (conversion logic) for converting new input definitions into input codes which will cause the given FF to perform as desired.**
- **To design the conversion logic we need to combine the excitation tables for both FF's & make a truth table with data inputs & Q as the inputs and the inputs of the given FF as the outputs.**



Sequential Logic

Converting One Type of FF to another:

- **S-R FF to J-K FF:**

Steps for Conversion:

- **Prepare the excitation table**
- **Prepare K-Maps & derive minimized expression.**
- **Draw the logic diagram**

Data Inputs		Outputs		inputs to S-R FF		Output	
Jn	Kn	Qn	$\overline{Qn}$	Sn	Rn		
0	0	0	1	0	0	0	Qn
0	0	1	0	0	0	1	
1	0	0	1	1	0	1	1
1	0	1	0	0	0	1	
0	1	0	1	0	0	0	0
0	1	1	0	0	1	0	
1	1	0	1	1	0	1	$\overline{Qn}$
1	1	1	0	0	1	0	

Example 7.2

Convert an *S–R* FLIP-FLOP to a *J–K* FLIP-FLOP.

Solution

The excitation tables of *S–R* and *J–K* FLIP-FLOPs are given in Table 7.6 from which we make the truth table given in Table 7.8.

Table 7.8 *Truth Table of Conversion Logic*

	FF data inputs		Output	*S–R FF* inputs	
Row	*J*	*K*	*Q*	*S*	*R*
1	0	0	0	0	×
2	0	1	0	0	×
3	1	0	0	1	0
4	1	1	0	1	0
5	0	1	1	0	1
6	1	1	1	0	1
7	0	0	1	×	0
8	1	0	1	×	0

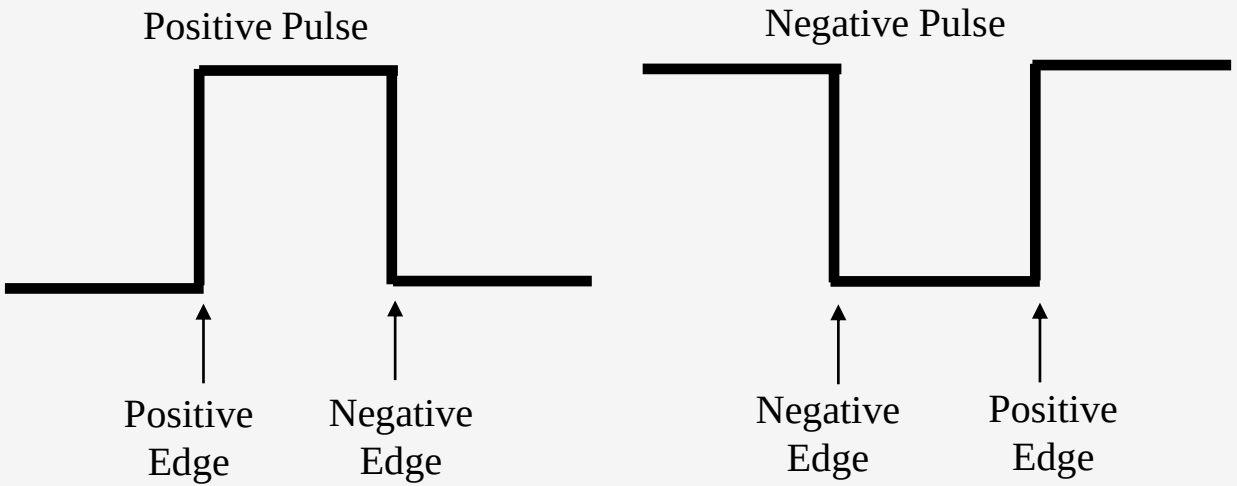
Fig. 7.20 *K-maps for Ex. 7.2*

The *K*-maps are given in Fig. 7.20, which give

$$S = J \cdot \overline{Q} \qquad \text{and} \qquad R = K \cdot Q$$

Thus, we see that the circuit resulting from this design is the same as that shown in Fig. 7.8.

Clock Pulse Definition



Edges can also be referred to as leading and trailing.

Sequential Logic

Flip-Flops

- **Flip-flops are the fundamental element of sequential circuits**
  - **– bistable**
  - **– (gates are the fundamental element for combinational circuits)**
- **Flip-flops are essentially 1-bit storage devices**
  - **– outputs can be set to store either 0 or 1 depending on the inputs**
  - **– even when the inputs are de-asserted, the outputs retain their prescribed value**
- **Flip-flops have (normally) 2 complimentary outputs Q & $\overline{Q}$.**
- **Three main types of flip-flop**
  - **– R-S J-K D-type**

Flip-flop

- Flip-flop is also known as the basic digital memory circuit.
- It has two stable states, memory logic 1 state & logic 0 state.
- We can design it, by using either NOR or NAND gates.
- A flip-flop can be designed by using the fundamental circuit which is also called as cross coupled inverter circuit, as the NAND gates in this circuit basically act as inverter.
- The cross coupled inverter is capable of clocking or batching the information, & hence this circuit is also called as latch.
- The disadvantage of the circuit is that we can't enter the disturbed digital signal as data into it.
- Latch is a distable element with two stable states.
- It has two outputs, Q & $\overline{Q}$ which are complement of each other.
- Latch is a sequential logic circuit which checks all its inputs continuously & will change its output as soon as the input changes without waiting for the clock signal.
- Generally an enable signal is provided for a latch when the enable signal is active the output will change as soon as there is a change in input.
- S-R (Set-Reset) latch is the simplest type of latch.

Flip-flop

- Difference between a latch & a Flip-flop.
- Latch & Flip-flop both are basically the bistable elements.
- Latch is a dual triggered flip-flop but flip-flop is a sequential circuit which generally samples inputs & changes its outputs only at particular instants of times & not continuously.
- Five types of flip-flops are as follows: S-R Flip-flop, J-K Flip-flop, Master Slave J-K Flip-flop, T- Flip-flop, D- Flip- flop
- In S-R flip-flop output is uncertain and is not according to assumption when both S & R are equal to 1.
- When initial conditions of flip-flops is to be set then we use the preset.
- Disadvantages of output being indeterminate for S-R flip-flop is overcome using J-K flip-flop.
- For $\triangle$t time period which is the propagation delay time of flip-flop, the output fluctuates between 0 and 1 & therefore is not certain. Fluctuation of output reduces life time of IC.
- In computer memory master slave principle is used to drive is slave while the other is master.
- D- Flip-flop can be used as delay or latch element in the circuit to delay the output.
- T- Flip-flop is used for toggling input. In this case output is always the complement of input.

Sequential Logic

Application of Flip-Flop:

- Bounce Elimination Switch
- Latch
- Registers
- Counters
- Memory Elements

Bounce Elimination Switch: (Fig 7.23 on Pg.256 R. P. Jain)

- In sequential circuit, if a 1 is to be entered through switch, then the switch is thrown to the corresponding position.
- But when the switch is thrown to position 1, the output oscillates between 0 & 1 for some time due to make & break (bouncing) of the switch at the point of contact before coming to rest.
- This changes the output of the circuit and creates difficulties in the operation of the system. This problem is eliminated by using bounce-elimination switch.

Sequential Logic

Registers:

- It is composed of a group of FF's to store a group of bits (word).
- FF can store 1-bit of digital information & also called as 1-bit register.
- The number of FF required is equal to the number of bits in binary word.
- E.g. 8085 have seven 8-bit registers called as general purpose register & five 1-bit register called as flags.
- The data can be entered in serial (one bit at a time) or in parallel form (all the bits simultaneously) and can be retrieved in the serial or parallel form.
- Data in serial form is also called as temporal code & in parallel form as spacial code.
- For serial input/output it requires only one line for data input & one line for data ouput.
- Whereas for parallel input/output it requires lines equal to the number of bits.

Sequential Logic

Registers:

- Registers are classified depending upon the way in which data are entered & retrieved.
- There are four possible modes of operation:

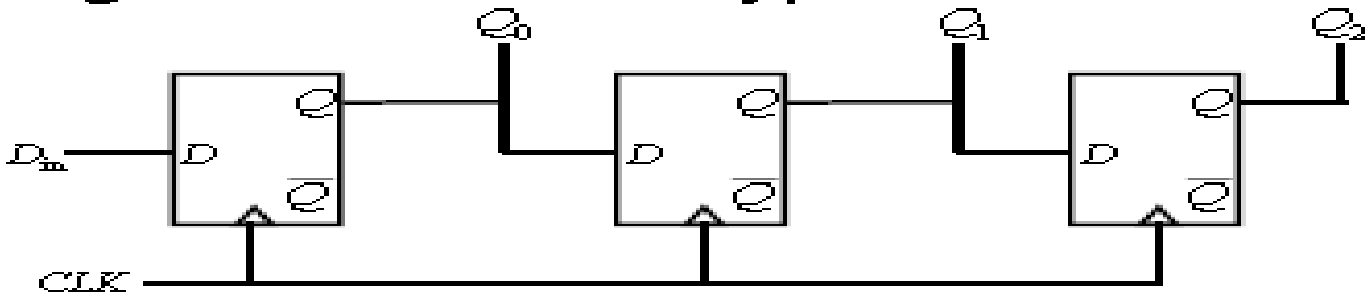
1. Serial-In, Serial-Out (SISO)
2. Serial-In, Parallel-Out (SIPO)
3. Parallel-In, Serial-Out (PISO)
4. Parallel-In, Parallel-Out (PIPO)

Counters:

- A circuit used for counting the pulses is known as a counter.
- Most widely used sequential circuits are registers & counters.
- Counters are composed of Flip-Flops.
- E.g. A circuit with n FF's has 2^n possible states. Therefore a 3-bit counter can count from decimal 0 to 7.
- Flip-Flop's are cleared by applying logic 0 at the clear input terminal momentarily.
- For normal counting operation, it is maintained at logic 1.

Shift Register

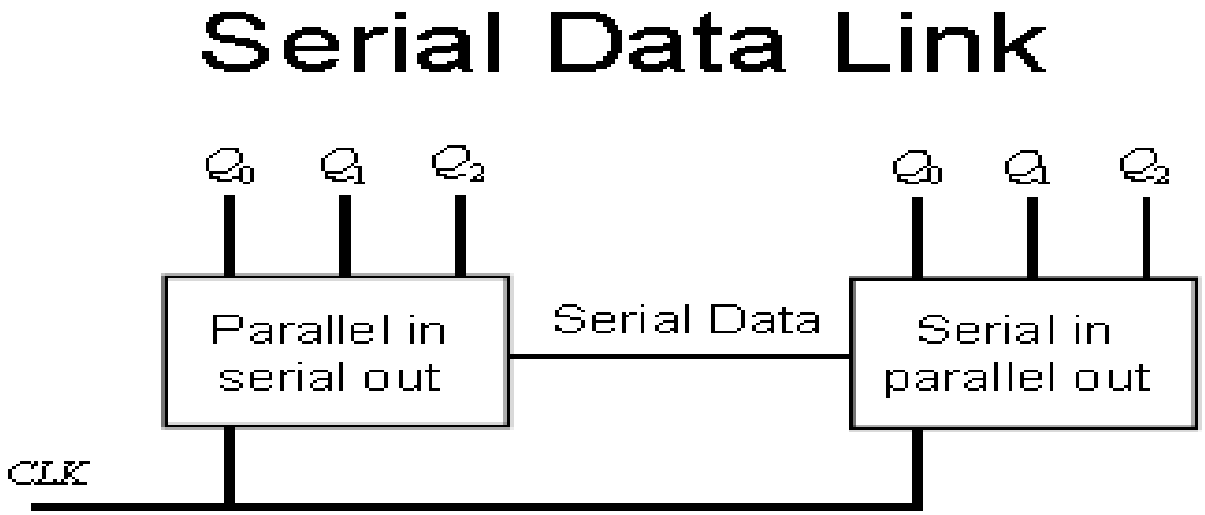
- A shift register can be implemented using a chain of D-type FFs



- Has a serial input, D_{in} and parallel output Q_0, Q_1 and Q_2.
- See data moves one position to the right on application of clock edge

Shift Register

- Preset and Clear inputs on the FFs can be utilised to provide a parallel data input feature
- Data can then be clocked out through Q_2 in a serial fashion, i.e., we now have a parallel in, serial out arrangement
- This along with the previous serial in, parallel out shift register arrangement can be used as the basis for a serial data link



- One data bit at a time is sent across the serial data link
- See less wires are required than for a parallel data link

Sequential Logic

Applications of Shift Registers:

1. Time Delays:

- **The time delay can be adjusted by controlling the number of stages in the register.**
- **By using serial-in & parallel-out registers and by taking the serial output at any one of the intermediate stages, we can delay output by any number of clock pulses.**

2. Serial/Parallel Data Conversion:

- **Data can be available in serial/parallel form.**
- **The transfer of data in parallel is much faster than serial form.**
- **Shift registers are used for converting serial data to parallel form, so that a serial input can be processed by a parallel system.**
- **A serial-in, serial-out, shift register can be used to perform serial to parallel conversion & a parallel in serial out, shift register can perform parallel-to-serial conversion.**

Sequential Logic

3. Ring Counters:

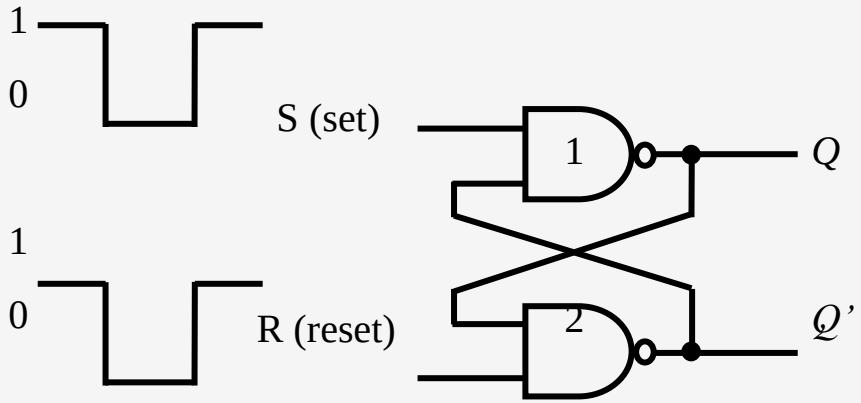
- **Ring counters are constructed by modifying the serial-in, serial-out, shift registers. There are two types of ring counters- basic ring counter & johnson counter.**
- **The basic ring counter can be obtained from a serial-in, serial-out, shift register by connecting Q output of the last FF to D input of first FF.**
- **The ring counter is the decimal counter. It is divide-by-N counter, where N is the number of stages. The keyboard encoder is an application of a ring counter.**

4. UART (Universal Asynchronous Receiver Transmitter):

- **Computers & microprocessor-based systems often send & receive data in a parallel format.**
- **They frequently communicate with external devices that send and/or receive serial data.**
- **An interfacing device used to accomplish these conversions is the UART.**

A Review
of
Sequential Logic

Basic RS Flip-Flop (NAND)



(a) Logic diagram

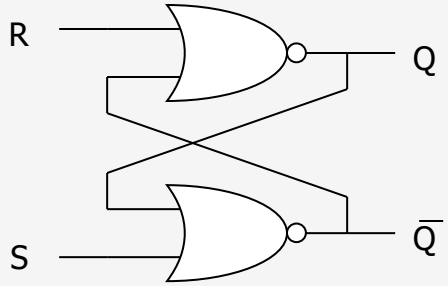
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after S = 1, R = 0)
0	1	1	0	
1	1	1	0	(after S = 0, R = 1)
0	0	1	1	

(b) Truth table

A flip-flop holds 1 "bit".
"Bit" ::= "binary digit."

RS-Latch as Cross-Coupled NOR Gates

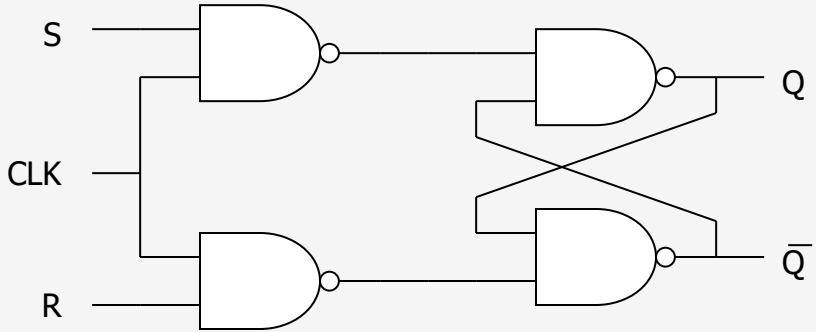
- If R = 1, Q resets to 0
- If S = 1, Q sets to 1
- If RS = 00, no change
- RS = 11 is not allowed because leads to oscillation



S R	Q
0 0	No change
0 1	0
1 0	1
1 1	Undefined

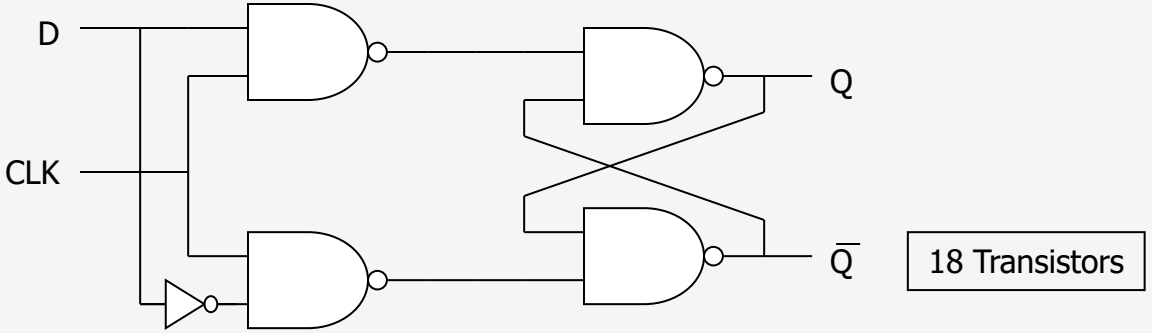
Level-Sensitive RS-Latch

- **"Q" only changes when CLK is high (i.e. level-sensitive)**
- **When CLK is high, behavior same as RS latch**



CLK	S R	Q
0	X X	No change
1	0 0	No change
1	0 1	0
1	1 0	1
1	1 1	Undefined

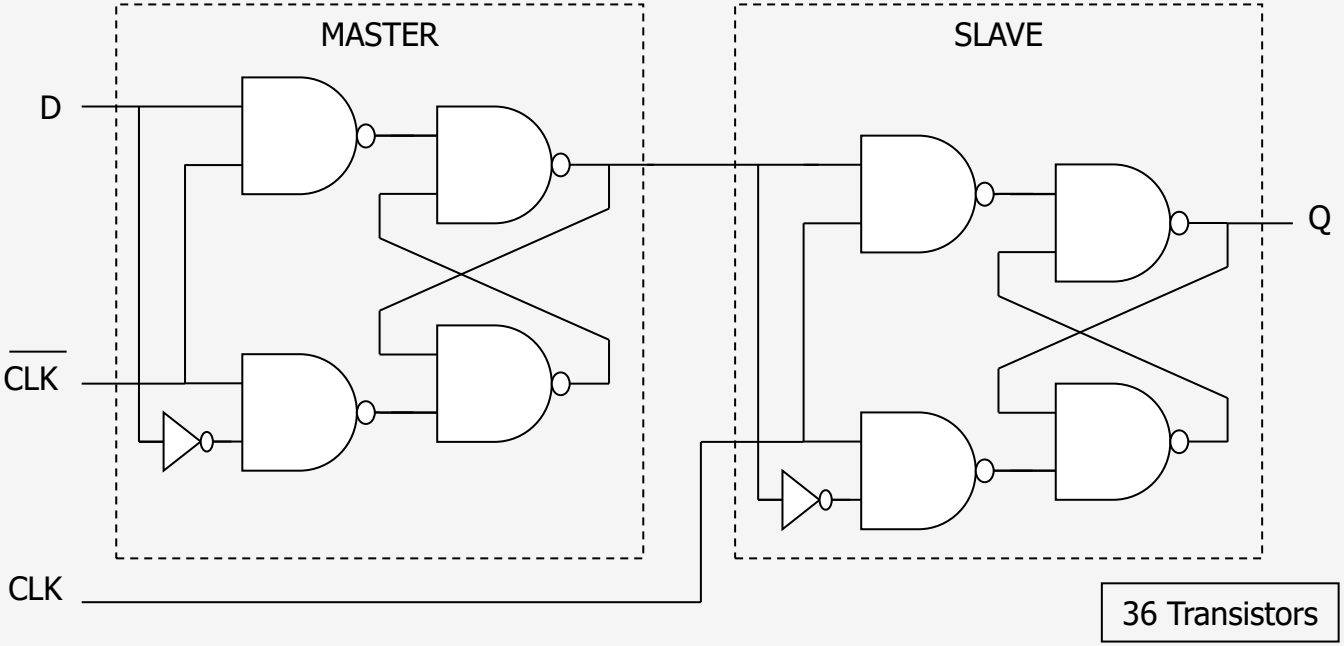
Level-Sensitive D-Latch



- Make level-sensitive D-latch from level-sensitive RS-latch by connecting S = D and R = not D

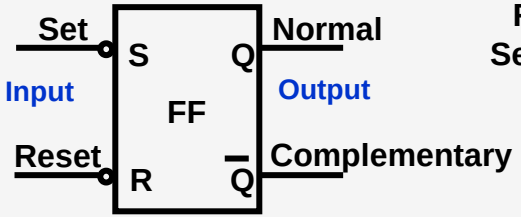
Master-Slave Edge-Triggered Flip-Flop

- Master-Slave configuration



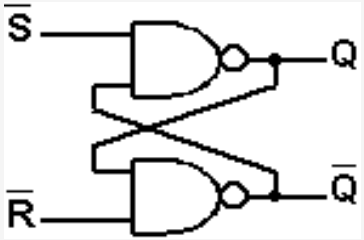
R-S Flip Flop

Logic Symbol



Also called:
R-S Latch
Set-Reset FF

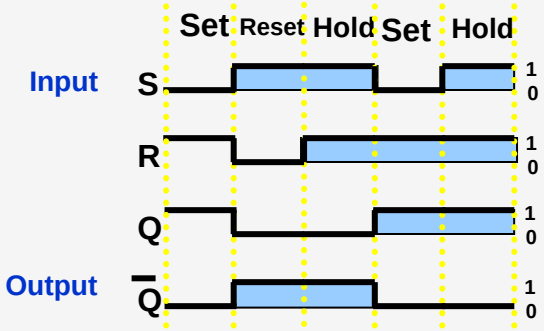
Wiring Diagram



Truth Table

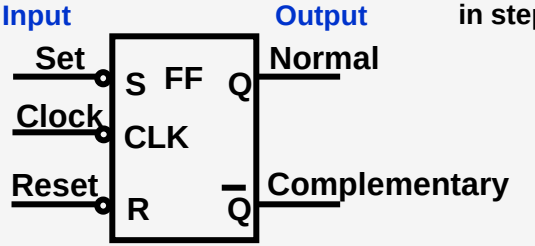
Mode of Operation	Input S	Input R	Output Q	Output Q	Effect
Prohibited	0	0	1	1	Prohibited Do not use
Set	0	1	1	0	For setting Q to 1
Reset	1	0	0	1	Resetting Q to 0
Hold	1	1	Q	Q	Depends Previous State

Waveform Diagram



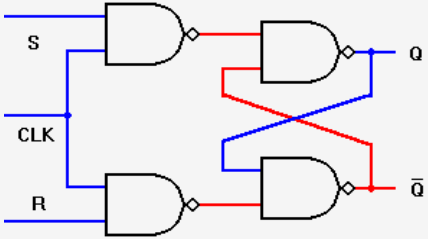
Clocked R-S Flip Flop

Logic Symbol



Output FF operates Synchronously in step with clock.

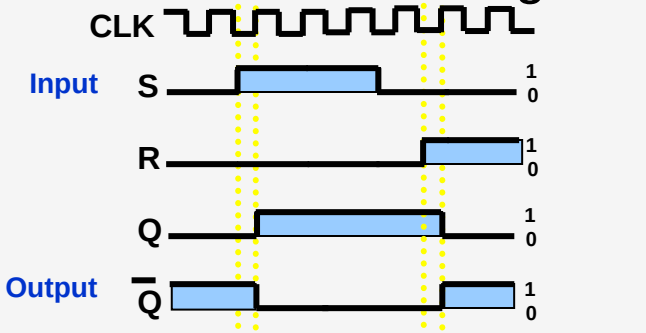
Wiring Diagram



Truth Table

Mode of Operation	Input CLK	Input S	Input R	Output Q	Output Q	Effect
Hold	⊓	0	0	1	1	No Change
Reset	⊓	0	1	0	1	Reset or cleared to 0
Set	⊓	1	0	1	0	Set to 1
Prohibited	⊓	1	1	1	1	Do not use

Waveform Diagram



Questions

Q. What type of waveform is used in flip flops?

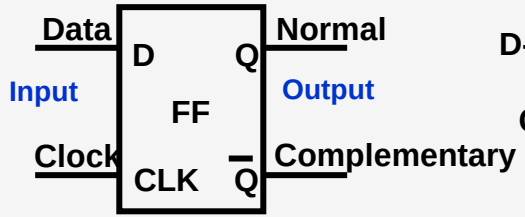
A. Square Waves.

Q. What does the RS stand for in the RS Flip Flop?

A. Reset Set.

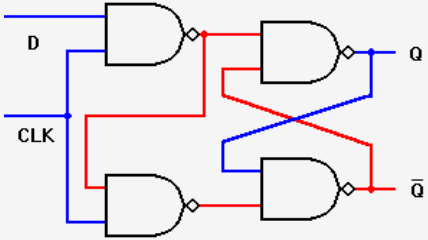
D Flip Flop

- Logic Symbol



Also called:
Delay FF
Data FF
D-type Latches
'Delayed 1 Clock Pulse'

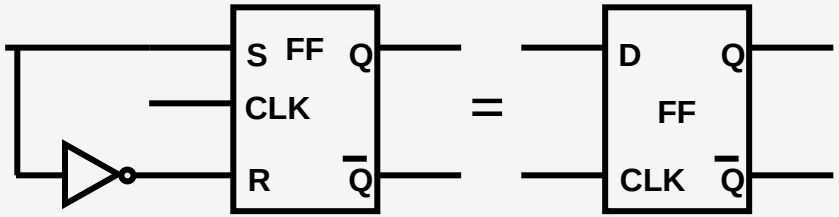
Wiring Diagram



- Truth Table

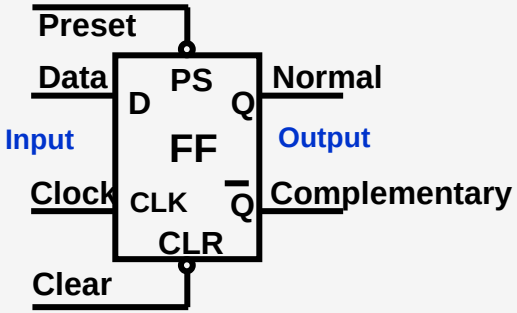
Input CLK	Input D	Output Q^{n+1}	Output $\overline{Q}$
⊓	0	0	1
⊓	1	1	0

Similar Wiring



7474 D Flip Flop

- Logic Symbol



Note: The asynchronous inputs (PS & CLR) Override the synchronous inputs (D & CLK) .

- Truth Table

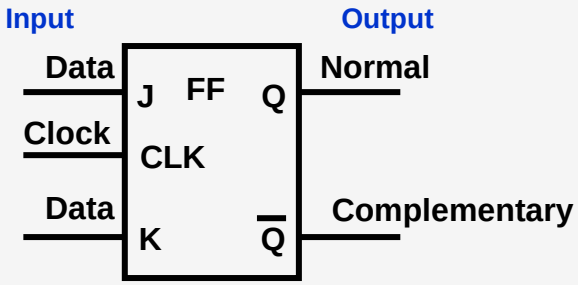
Mode of Operation	INPUTS				OUTPUTS	
	Asynchronous		Synchronous			
	PS	CLR	CLK	D	Q	$\overline{Q}$
Asynchronous Set	0	1	X	X	1	0
Asynchronous Reset	1	0	X	X	0	1
Prohibited	0	0	X	X	1	1
Set	1	1	↑ L to H	1	1	0
Reset	1	1	↑ L to H	0	0	1

D (Delay) Flip Flop Uses

- Sequential logic devices used in temporary memory devices.
- Wired together to form shift registers and storage registers.
- Delays data from reaching output Q one clock pulse.

J-K Flip Flop

Logic Symbol

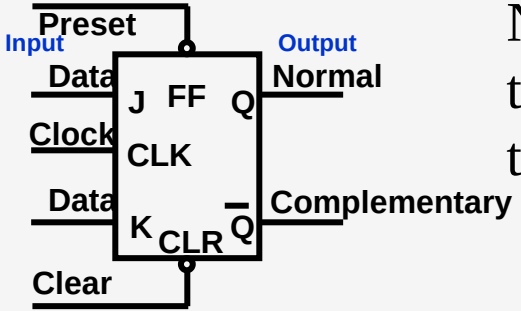


Truth Table

Mode of Operation	INPUTS			OUTPUTS		Effect
	Input CLK	Input J	Input K	Output Q	Output Q	
Hold	⊓	0	0	No Change	No Change	No Change
Reset	⊓	0	1	0	1	Reset or cleared to 0
Set	⊓	1	0	1	0	Set to 1
Toggle	⊓	1	1	Toggle	Toggle	Changed to Opposite State

7476 J-K Flip Flop

- Logic Symbol



Note: 7476 uses the entire pulse to transfer data from J & K data inputs to Q & $\overline{Q}$ outputs.

- Truth Table

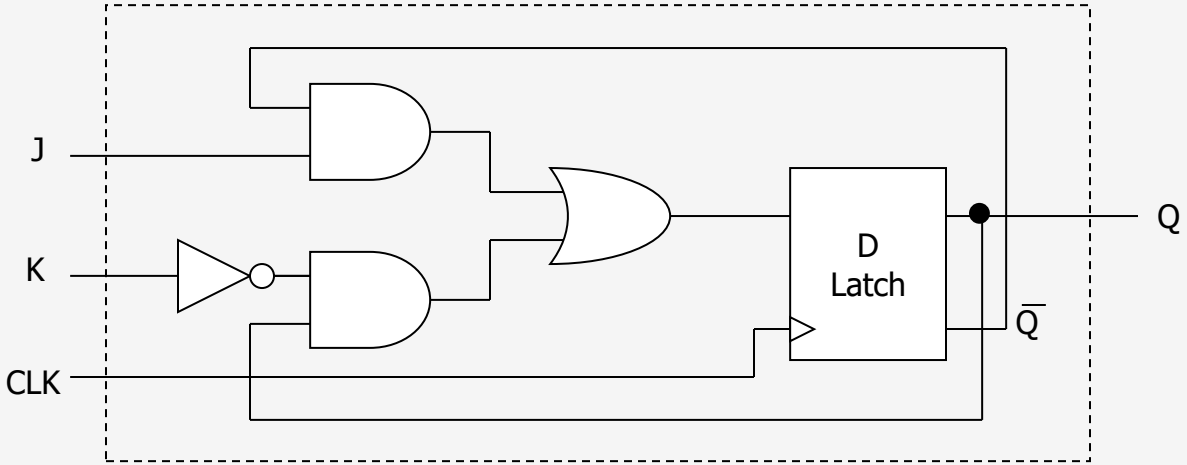
Mode of Operation	INPUTS					OUTPUTS	
	Asynchronous		Synchronous				
	PS	CLR	CLK	J	$\overline{K}$	Q	$\overline{Q}$
Asynchronous Set	0	1	X	X	X	1	0
Asynchronous Reset	1	0	X	X	X	0	1
Prohibited	0	0	X	X	X	1	1
Hold	1	⊓		0	0	No Change	No Change
Reset	1	⊓		0	1	0	1
Set	1	⊓		1	0	1	0
Toggle	1	⊓		1	1	Opposite State	

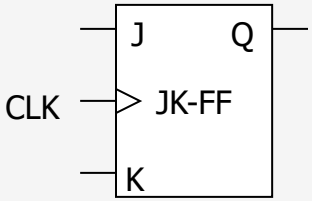
J-K Flip Flop Uses

- Universal Flip Flop. Has all features of other FF.
- When being used only in the toggle mode, commonly called a T Flip Flop.
- Most commonly used as counters.

JK Flip-Flop from D-Latch

- Same as RS-Latch except “toggle” on 11

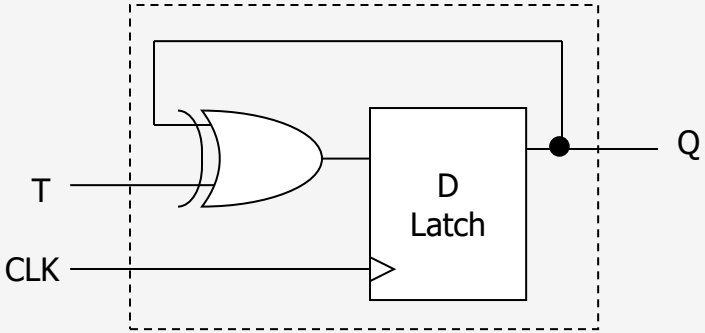


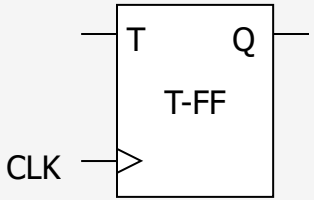


CLK	J K	Q
0	X X	No change
1	0 0	No change
1	0 1	0
1	1 0	1
1	1 1	**Toggle**

Toggle Flip-Flop from D-Latch

- Toggles stored value if T = 1 when CLK is high





CLK	T	Q
0	X	No change
1	0	No change
1	1	**Toggle**

Presentation Topic

Analog and Digital Electronics

Yogesh K Sharma
yogesh.sharma@viit.ac.in
Department of Computer Engineering

BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

Counters

Unit - 3

Courtesy: R. P. Jain & J. S. Katre (Modern Digital Electronics)

Contents

Unit-3: Counters

- **Counters:** Asynchronous counter. Synchronous counter, ring counters, Johnson Counter, Modulus of the counter (IC 7490).
- **Synchronous Sequential Circuit Design:** Models – Moore and Mealy, State diagram and State Tables, Design Procedure, Sequence generator and detector.
- **Asynchronous Sequential Circuit Design:** Difference with synchronous circuit design, design principles and procedure, applications.

Counters

- A circuit used for counting the pulses is known as a counter.
- Most widely used sequential circuits are registers & counters.
- Counters are composed of Flip-Flops.
- E.g. A circuit with n FF's has 2^n possible states. Therefore a 3-bit counter can count from decimal 0 to 7.
- Flip-Flop's are cleared by applying logic 0 at the clear input terminal momentarily.
- For normal counting operation, it is maintained at logic 1.

Counters

- There are two types of counters Asynchronous (ripple) & Synchronous counters.
- Asynchronous counter is easy to design & requires the least amount of homework.
- Asynchronous counter is not triggered simultaneously & is also called as serial or series counter.
- Design of Synchronous counter requires some amount of homework & each Flip-Flop is triggered simultaneously.
- Each count of the counter is called as a state of the counter.
- The number of states through which the counter passes before returning to the starting state is called the modulus of the counter.
- E.g. A 2-bit counter has 4 states, it is called a mod-4 counter & as it divides the clock signal frequency by 4, it is called as divide-by-4 counter.
- An n-bit counter will have n FF's & 2^n states, and divides the input frequency by 2^n. Hence it is called as divide-by-2^n counter. The LSB of ripple counters is the Q output of the FF to which the external clock is applied.

Counters

Classification of Sequential Circuits:

- Synchronous sequential circuits: Contents of memory elements can be changed only at the rising or falling edges of clock signal.
- Asynchronous sequential circuits: Contents of memory elements can be changed at any instant of time.

	Asynchronous	Synchronous
1.	Depends upon the sequence in which the input signals change.	1. Behaviour can be defined from the knowledge of its signal at discrete instants of time.
2.	Commonly used memory elements are time delays.	2. Memory elements used are Flip-Flops.
3.	Design is tedious.	3. Design is easy.
4.	They are combinational circuit with feedback.	4. Synchronization is achieved by system clock.
		5. Also called as clocked seq. ckt

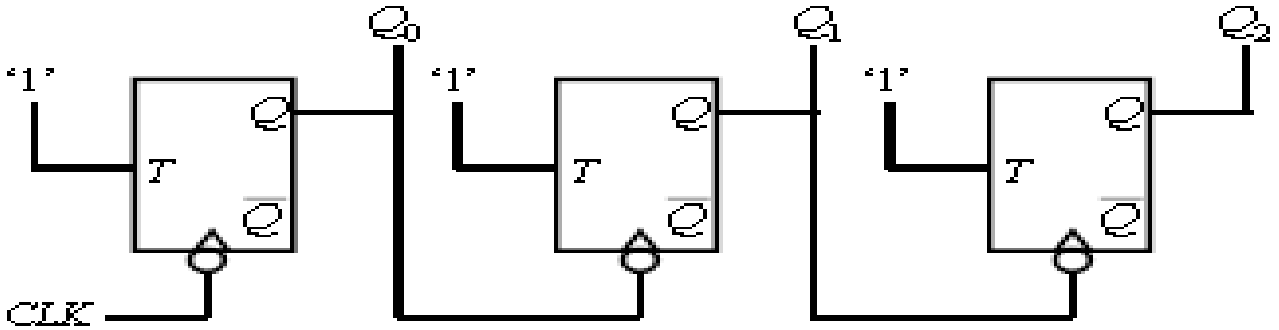
Counters

Types of Counters:

1. Ripple/Asynchronous Counters
2. Synchronous Counters

Ripple Counters

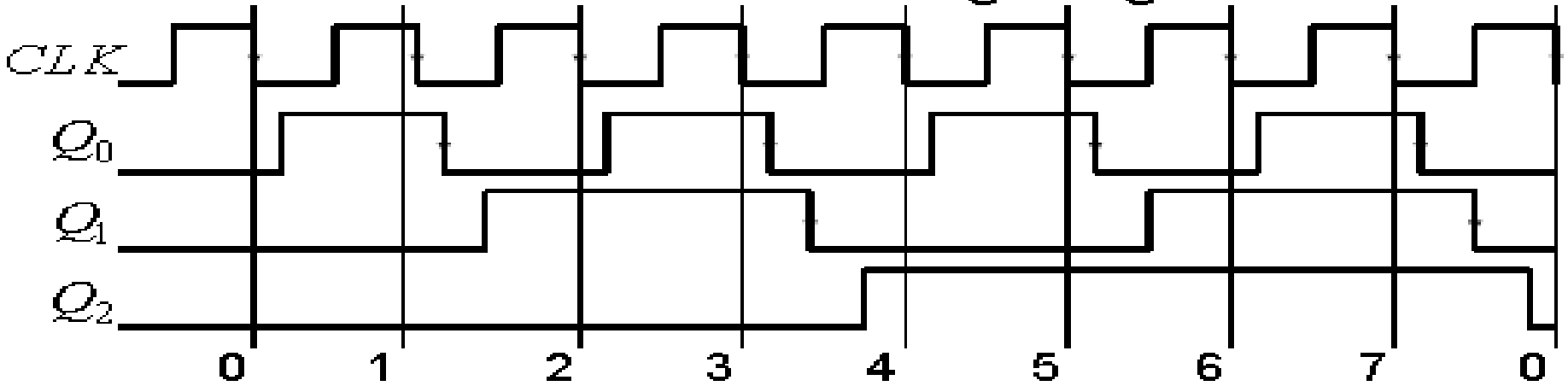
- A ripple counter can be made be cascading together negative edge triggered T-type FFs operating in 'toggle' mode, i.e., $T=1$



- See that the FFs are not clocked using the same clock, i.e., this is **not** a synchronous design. This gives some problems....

Ripple Counters

- We will now draw a timing diagram



- Problems:

See outputs do not change at the same time, i.e., synchronously. So hard to know when count output is actually valid.

Propagation delay builds up from stage to stage, limiting maximum clock speed before miscounting occurs.

Ripple Counters

- If you observe the frequency of the counter output signals you will note that each has half the frequency, i.e., double the repetition period of the previous one. This is why counters are often known as dividers
- Often we wish to have a count which is not a power of 2, e.g., for a BCD counter (0 to 9). To do this:
  - use FFs having a Reset/Clear input
  - Use an AND gate to detect the count of 10 and use its output to Reset the FFs

Synchronous Counters

- Owing to the problems identified with ripple counters, they should not usually be used to implement counter functions
- It is recommended that *synchronous* counter designs be used
- In a synchronous design
  - all the FF clock inputs are directly connected to the clock signal and so all FF outputs change at the same time, i.e., *synchronously*
  - more complex combinational logic is now needed to generate the appropriate FF input signals (which will be different depending upon the type of FF chosen)

Synchronous Counters

- We will now investigate the design of synchronous counters
- We will consider the use of D-type FFs only, although the technique can be extended to cover other FF types.
- As an example, we will consider a 0 to 7 up-counter

Synchronous Counters

- To assist in the design of the counter we will make use of a modified *state transition table*. This table has additional columns that define the required FF inputs (or *excitation* as it is known)
  - Note we have used a state transition table previously when determining the state diagram for an RS latch
- We will also make use of the so called '*excitation table*' for a D-type FF
- First however, we will investigate the so called *characteristic table* and *characteristic equation* for a D-type FF

Synchronous Counters

- Also note that if we are using D-type FFs, it is not necessary to explicitly write out the FF input columns, since we know they are identical to those for the next state
- To complete the design we now have to determine appropriate combinational logic circuits which will generate the required FF inputs from the current states
- We can do this from inspection, using Boolean algebra or using K-maps.

Register

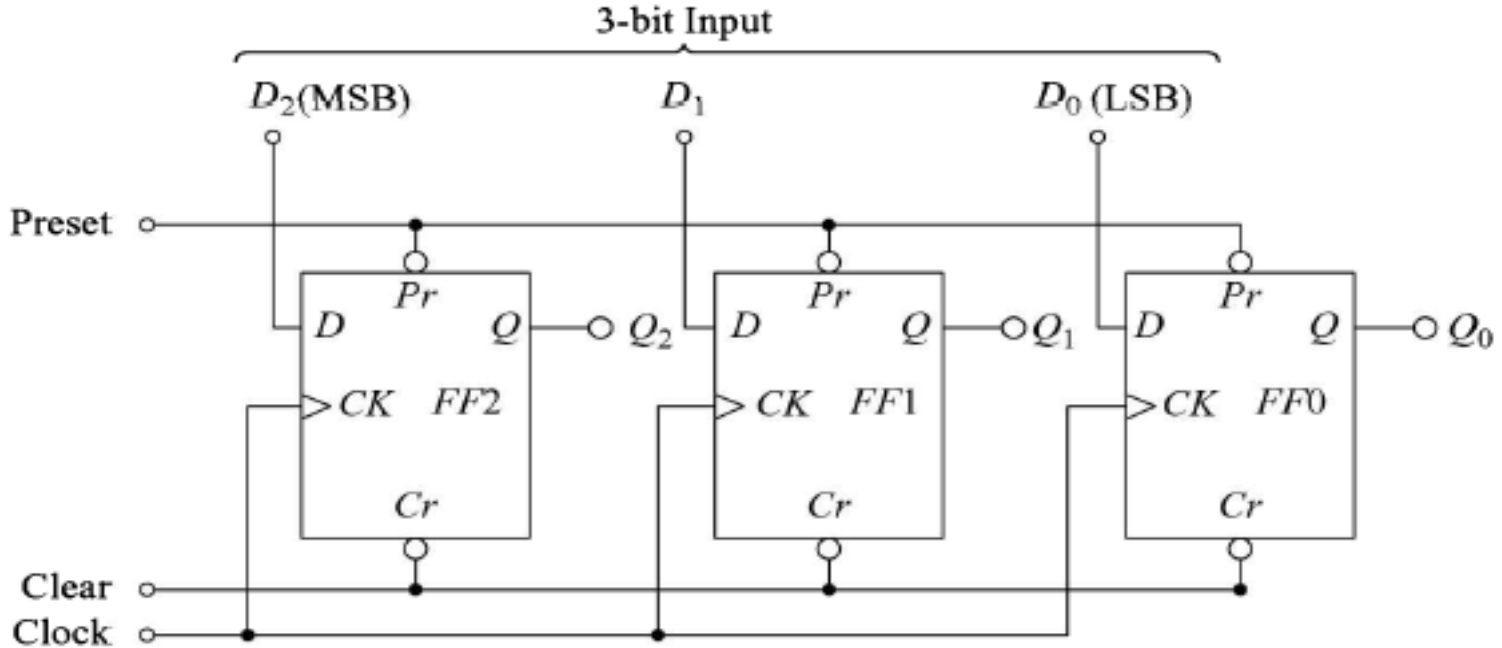


Fig. 7.24 ***A 3-bit Register Using*** `FLIP-FLOPs`

Counter

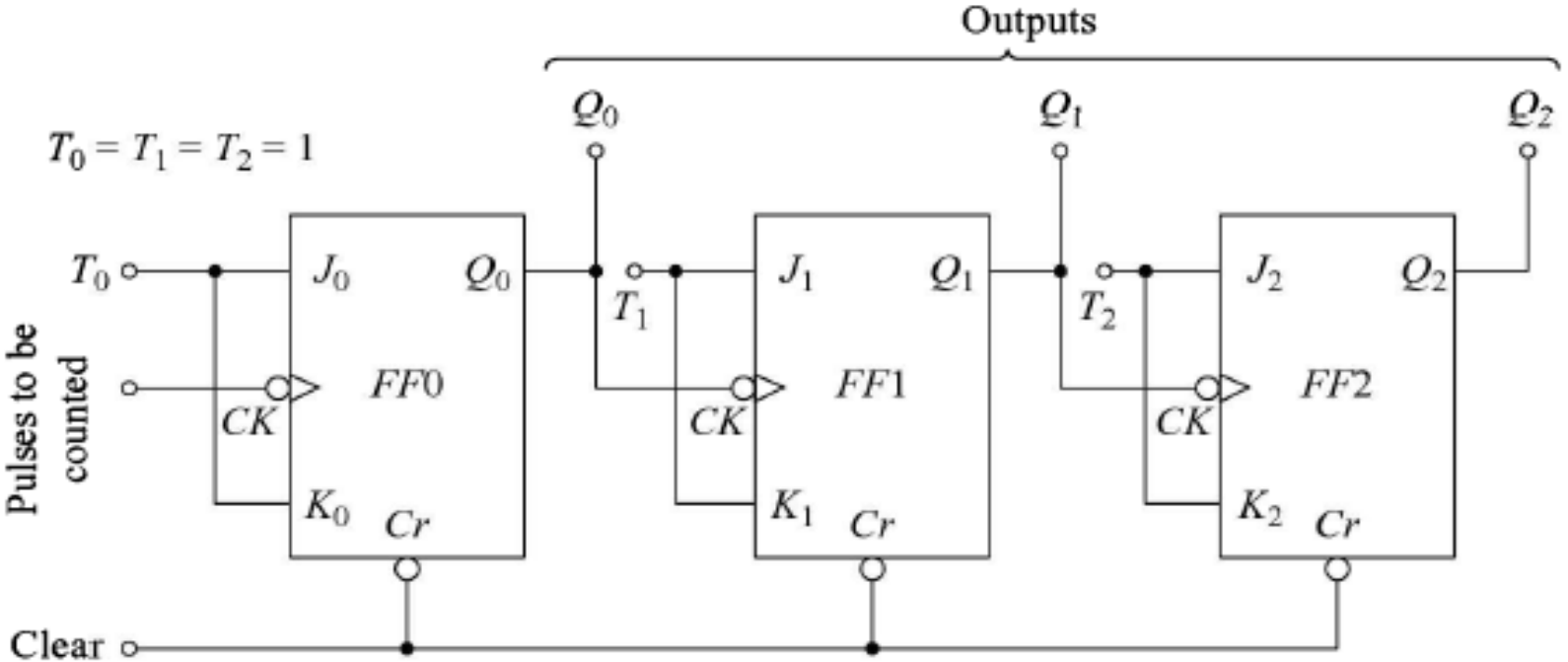


Fig. 7.25 ***A 3-bit Counter Using*** FLIP-FLOP***s***

Counters

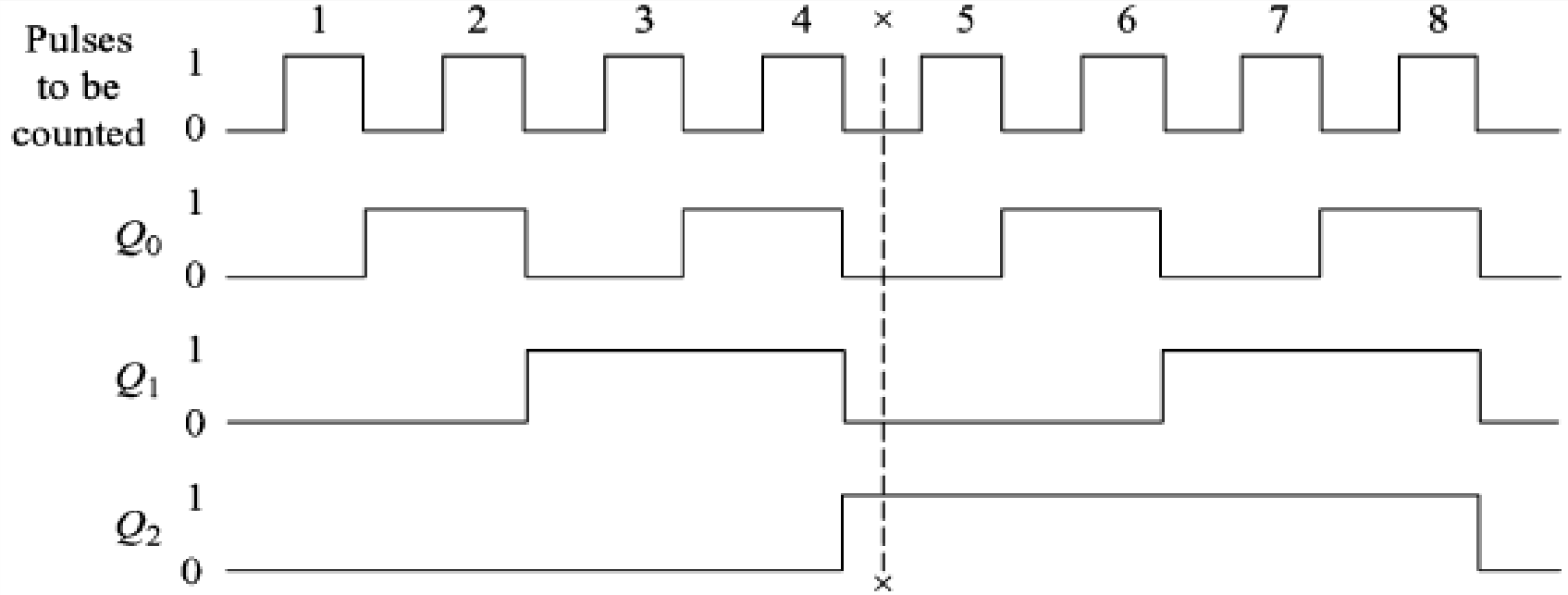
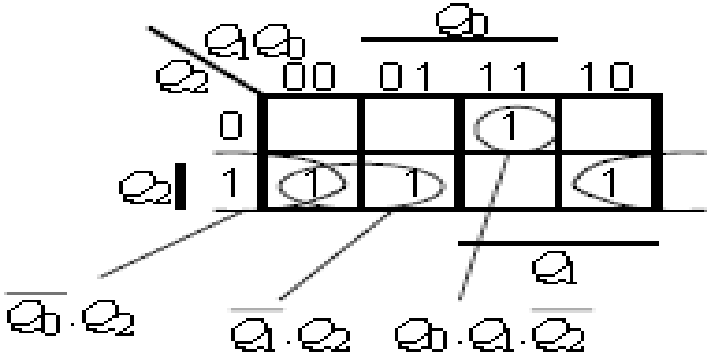


Fig. 7.26 *Waveforms of Counter of Fig. 7.25*

Synchronous Counters

- A similar procedure can be used to design counters having an arbitrary count sequence
  - Write down the state transition table
  - Determine the FF excitation (easy for D-types)
  - Determine the combinational logic necessary to generate the required FF excitation from the current states – **Note:** remember to take into account any unused counts since these can be used as don't care states when determining the combinational logic circuits

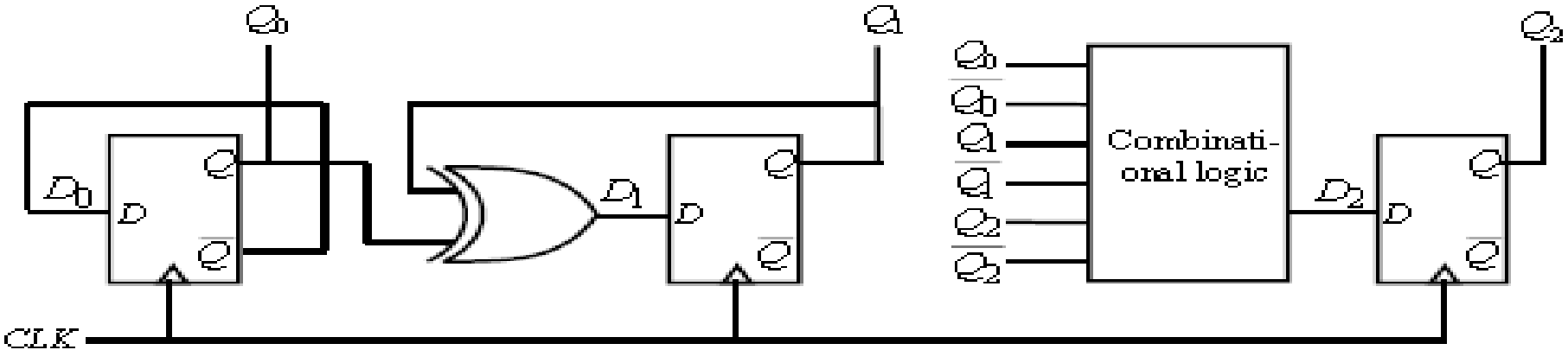
Synchronous Counters



So,

$$D_2 = \overline{Q_0}\cdot Q_2 + \overline{Q_1}\cdot Q_2 + Q_0\cdot Q_1\cdot\overline{Q_2}$$

$$D_2 = Q_2\cdot(\overline{Q_0}\cdot + \overline{Q_1}) + Q_0\cdot Q_1\cdot\overline{Q_2}$$



3 Bit Synchronous Counter

Example 8.9

Design a 3-bit synchronous counter using *J-K* FLIP-FLOPs.

Solution

The number of FLIP-FLOPs required is 3. Let the FLIP-FLOPs be FF0, FF1, and FF2 and their inputs and outputs are given below:

FLIP-FLOP	Inputs	Output
FF0	J_0, K_0	Q_0
FF1	J_1, K_1	Q_1
FF2	J_2, K_2	Q_2

3 Bit Synchronous Counter

Table 8.10

Counter state			FLIP-FLOP inputs					
			FF0		FF1		FF2	
Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2
0	0	0	1	×	0	×	0	×
0	0	1	×	1	1	×	0	×
0	1	0	1	×	×	0	0	×
0	1	1	×	1	×	1	1	×
1	0	0	1	×	0	×	×	0
1	0	1	×	1	1	×	×	0
1	1	0	1	×	×	0	×	0
1	1	1	×	1	×	1	×	1
0	0	0						

The count sequence and the required inputs of FLIP-FLOPs are given in Table 8.10. The inputs to the FLIP-FLOPs are determined in the following manner:

Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	1	1
1	×	×	×	×

$J_0 = 1$ (a)

Q_0 \ Q_2Q_1	00	01	11	10
0	×	×	×	×
1	1	1	1	1

$K_0 = 1$ (b)

Q_0 \ Q_2Q_1	00	01	11	10
0	0	×	×	0
1	1	×	×	1

$J_1 = Q_0$ (c)

Q_0 \ Q_2Q_1	00	01	11	10
0	×	0	0	×
1	×	1	1	×

$K_1 = Q_0$ (d)

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	×	×
1	0	1	×	×

$J_2 = Q_0Q_1$ (e)

Q_0 \ Q_2Q_1	00	01	11	10
0	×	×	0	0
1	×	×	1	0

$K_2 = Q_0Q_1$ (f)

3 Bit Synchronous Counter

Consider one column of the counter state at a time and start from the first row, for example, consider Q_0. Before the first pulse is applied, $Q_0 = 0$ and it is required to be 1 at the end of the first clock pulse. Therefore, to achieve this condition, the values of J_0 and K_0 are 1 and × respectively (from the excitation Table 7.6). These are entered in the table in the row corresponding to 0 pulse. When the second clock pulse is applied Q_0 is to change from 1 to 0, therefore, the required inputs are

$$J_0 = \times, K_0 = 1$$

In a similar manner inputs of each FLIP-FLOP are determined.

Now, we prepare the *K*-maps (Fig. 8.23) with Q_2, Q_1, and Q_0 as input variables and FLIP-FLOP inputs as output variables. We then minimize the *K*-maps and the resulting minimized expressions are:

$$J_0 = 1, \quad K_0 = 1$$
$$J_1 = Q_0, \quad K_1 = Q_0$$
$$J_2 = Q_0 Q_1, \quad K_2 = Q_0 Q_1$$

The resulting counter circuit is same as the circuit of Fig. 8.22.

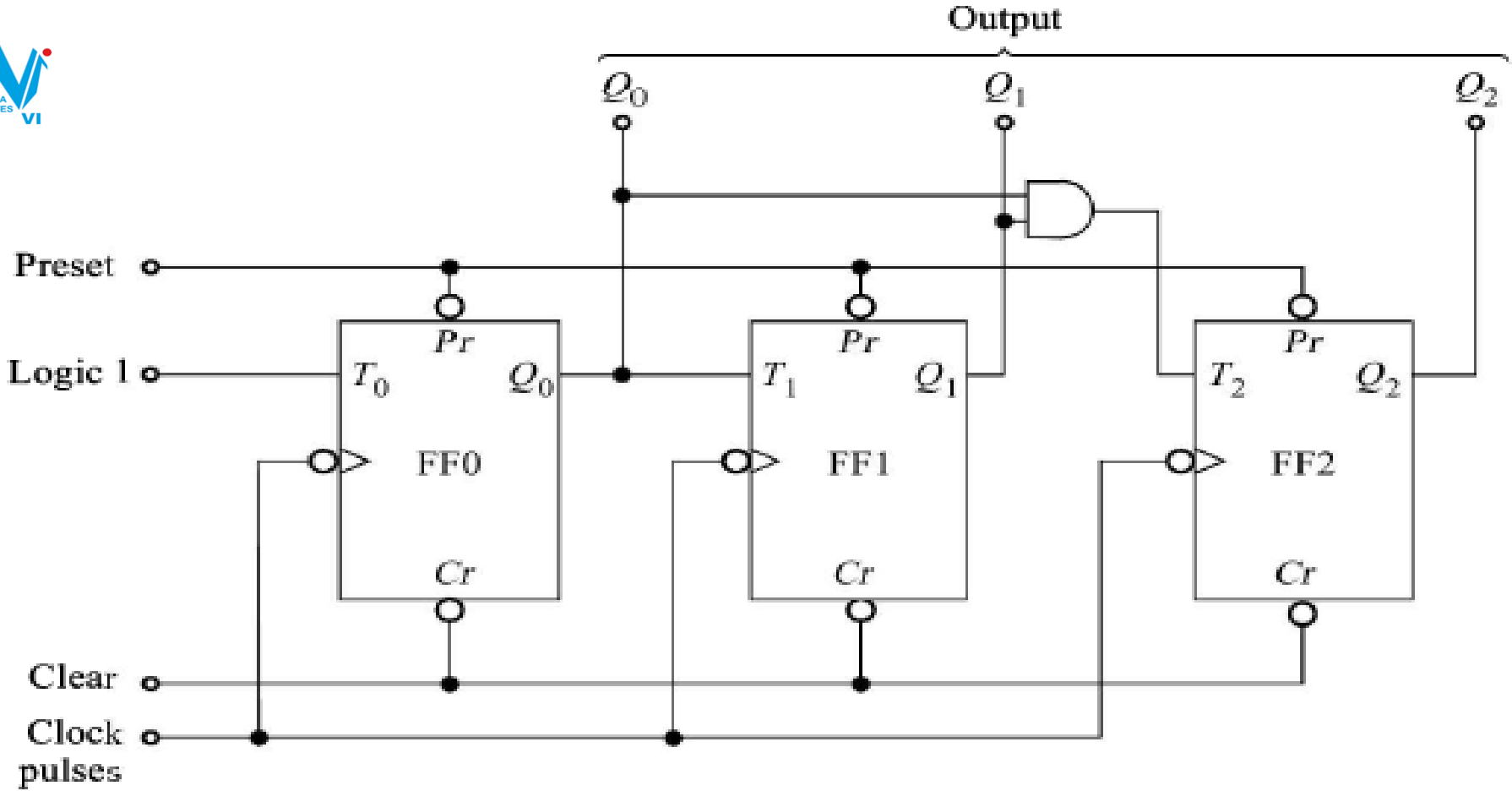


Fig. 8.22 ***A 3-bit Synchronous Counter***

3 Bit Binary UP/Down Counter

Example 8.10

Design a 3-bit binary UP/DOWN counter with a direction control M. Use J-K FLIP-FLOPs.

Solution

The count sequence is given in Table 8.11. For $M = 0$, it acts as an UP counter and for $M = 1$ as a DOWN counter. The number of FLIP-FLOPs required is 3. The inputs of the FLIP-FLOPs are determined in a manner similar to the one employed in Ex. 8.9.

Table 8.11

Direction control	Counter state			FLIP-FLOP inputs					
M	Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2
0	0	0	0	1	×	0	×	0	×
0	0	0	1	×	1	1	×	0	×
0	0	1	0	1	×	×	0	0	×
0	0	1	1	×	1	×	1	1	×
0	1	0	0	1	×	0	×	×	0
0	1	0	1	×	1	1	×	×	0
0	1	1	0	1	×	×	0	×	0
0	1	1	1	×	1	×	1	×	1
1	0	0	0	1	×	1	×	1	×
1	1	1	1	×	1	×	0	×	0
1	1	1	0	1	×	×	1	×	0
1	1	0	1	×	1	0	×	×	0
1	1	0	0	1	×	1	×	×	1
1	0	1	1	×	1	×	0	0	×
1	0	1	0	1	×	×	1	0	×
1	0	0	1	×	1	0	×	0	×
	0	0	0						

3 Bit Binary UP/Down Counter

From Table 8.11, we obtain

$$J_0 = K_0 = 1$$

The K-maps for J_1, K_1, J_2, and K_2 are shown in Fig. 8.24. From the K-maps, the minimized expressions are obtained as

$$J_1 = K_1 = Q_0\overline{M} + \overline{Q}_0 M$$
$$J_2 = K_2 = \overline{M}\,Q_1 Q_0 + M\,\overline{Q}_1\,\overline{Q}_0$$

The counter circuit can be drawn using the above expressions

Q_1Q_0 \ MQ_2	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	×	×	×	×
10	×	×	×	×

J_1

Q_1Q_0 \ MQ_2	00	01	11	10
00	×	×	×	×
01	×	×	×	×
11	1	1	0	0
10	0	0	1	1

K_1

Q_1Q_0 \ MQ_2	00	01	11	10
00	0	×	×	1
01	0	×	×	0
11	1	×	×	0
10	0	×	×	0

J_2

Q_1Q_0 \ MQ_2	00	01	11	10
00	×	0	1	×
01	×	0	0	×
11	×	1	0	×
10	×	0	0	×

K_2

Decade UP Counter

Example 8.11

Design a decade UP counter. Use *J-K* FLIP-FLOPs.

Solution

There are ten states in a decade counter, which requires four FLIP-FLOPs. The remaining six states are unused states. The count sequence and the FLIP-FLOP inputs are given in Table 8.12.

Table 8.12

Counter state				FLIP-FLOP inputs							
Q_3	Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	1	×	0	×	0	×	0	×
0	0	0	1	×	1	1	×	0	×	0	×
0	0	1	0	1	×	×	0	0	×	0	×
0	0	1	1	×	1	×	1	1	×	0	×
0	1	0	0	1	×	0	×	×	0	0	×
0	1	0	1	×	1	1	×	×	0	0	×
0	1	1	0	1	×	×	0	×	0	0	×
0	1	1	1	×	1	×	1	×	1	1	×
1	0	0	0	1	×	0	×	0	×	×	0
1	0	0	1	×	1	0	×	0	×	×	1
0	0	0	0								

The *K*-maps are shown in Fig. 8.25 from which the minimized expressions are obtained as

$$J_0 = 1, \qquad K_0 = 1$$
$$J_1 = Q_0\,\overline{Q}_3, \qquad K_1 = Q_0$$
$$J_2 = Q_0 Q_1, \qquad K_2 = Q_0 Q_1$$
$$J_3 = Q_0 Q_1 Q_2, \qquad K_3 = Q_0$$

The counter circuit can be drawn using the above expressions.

MOD-8 Synchronous Counter

Example 8.12

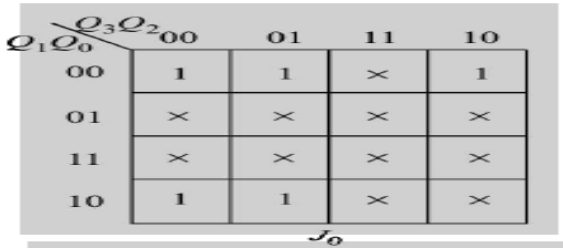
Design a natural binary sequence mod-8 synchronous counter using *D* FLIP-FLOPs.

Solution

The number of FLIP-FLOPs required is 3. Let the FLIP-FLOPs be FF0, FF1 and FF2 with inputs D_0, D_1 and D_2 respectively. Their outputs are Q_0, Q_1, and Q_2 respective. The count sequence and the corresponding FLIP-FLOPs input required are given in Table 8.13. Using the excitation Table 7.6, the FLIP-FLOPs inputs are determined in the same way as determined for the *J-K* FLIP-FLOPs.

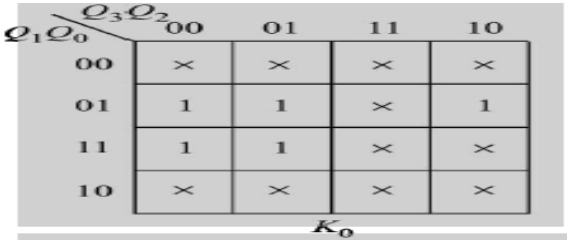
Table 8.13 ***Counter States and D FLIP-FLOPs Input***

Counter state			FLIP-FLOP inputs		
Q_2	Q_1	Q_0	D_0	D_1	D_2
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1
1	1	1	0	0	0
0	0	0			



Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	1	1	×	1
01	×	×	×	×
11	×	×	×	×
10	1	1	×	×

J_0

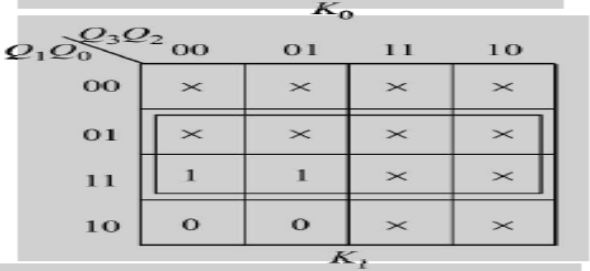


Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	×	×	×	×
01	1	1	×	1
11	1	1	×	×
10	×	×	×	×

K_0

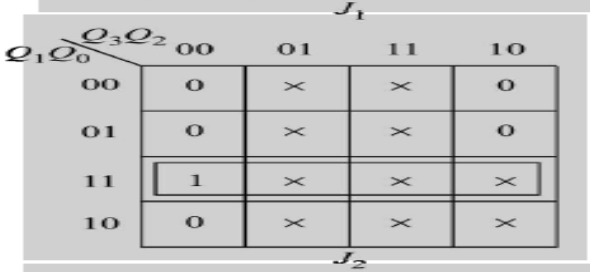
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	×	0
01	1	1	×	0
11	×	×	×	×
10	×	×	×	×

J_1



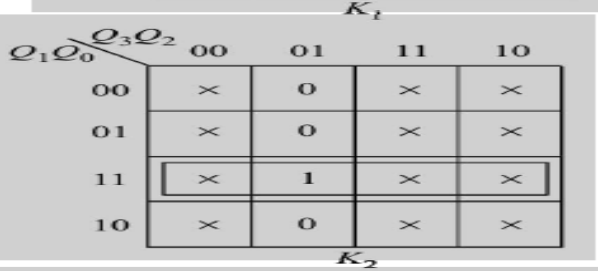
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	×	×	×	×
01	×	×	×	×
11	1	1	×	×
10	0	0	×	×

K_1



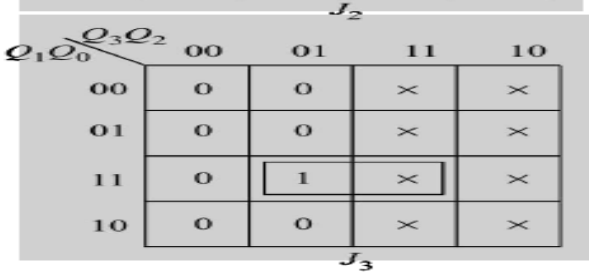
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	×	×	0
01	0	×	×	0
11	1	×	×	×
10	0	×	×	×

J_2



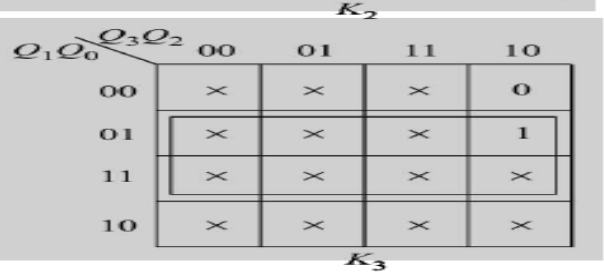
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	×	0	×	×
01	×	0	×	×
11	×	1	×	×
10	×	0	×	×

K_2



Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	×	×
01	0	0	×	×
11	0	1	×	×
10	0	0	×	×

J_3



Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	×	×	×	0
01	×	×	×	1
11	×	×	×	×
10	×	×	×	×

K_3

The *K*-maps for D_0, D_1, and D_2 are given in Fig. 8.26.

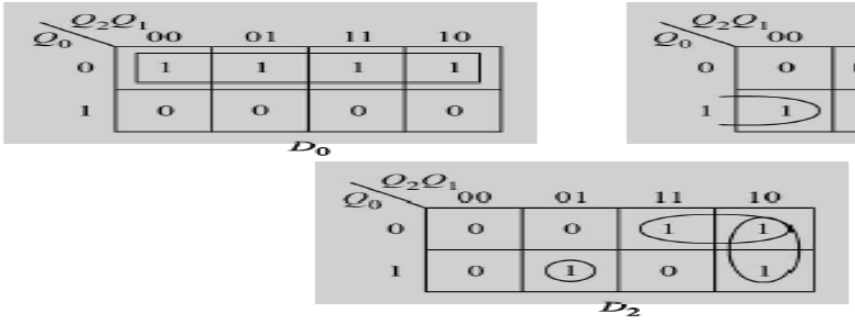


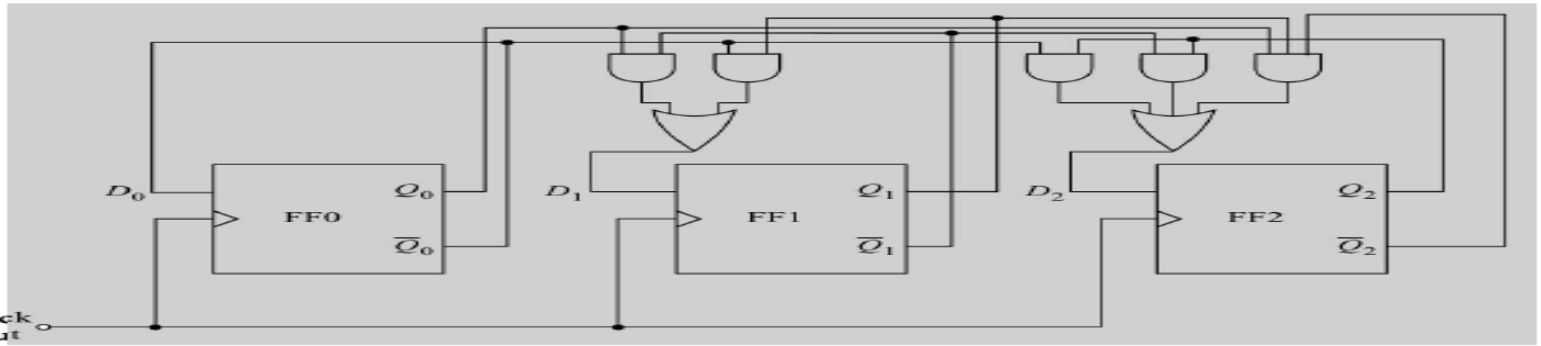
Fig. 8.26 ***K-Maps of Ex. 8.12***

The minimised expressions for D_0, D_1, and D_2 are:

$$D_0 = \overline{Q}_0 \qquad D_1 = Q_1\overline{Q}_0 + \overline{Q}_1 Q_0$$

$$\begin{aligned} D_2 &= Q_2\overline{Q}_0 + Q_2\overline{Q}_1 + \overline{Q}_2 Q_1 Q_0 \\ &= Q_2(\overline{Q}_0 + \overline{Q}_1) + \overline{Q}_2 Q_1 Q_0 \\ &= Q_2(\overline{Q_0 \cdot Q_1}) + \overline{Q}_2(Q_1 Q_0) \\ &= Q_2 \oplus Q_1 \cdot Q_0 \end{aligned}$$

The complete circuit of the synchronous counter using positive edge triggered *D* FLIP–FLOPs is shown in Fig. 8.27.



Counters

Ring Counter:

- If the serial output Q_0 of the shift register is connected back to the serial input, then an injected pulse will keep circulating. This circuit is referred to as a ring counter.
- The pulse is injected by entering 00001 in the parallel form after clearing the Flip-Flop's.
- When Clock pulses are applied, this 1 circulates around the circuit.
- The outputs are sequential non-overlapping pulses which are useful for control-state counters, for stepper motor (which rotates in steps) which require sequential pulses to rotate it from one position to the next etc.
- This circuit can also be used for counting the number of pulses.
- The number of pulses counted is read by noting which FF is in state 1.
- Since there is one pulse at the output for each of the N clock pulses, this circuit is referred to as a divide-by-N counter or an N:1 scalar.
- If $\overline{Q_0}$ is connected to the serial output, the resulting circuit is referred to as a twisted ring, johnson, or moebius counter.
- If the clock pulses are applied after clearing the FF's, square waveform is obtained at the Q outputs.
- It is useful for the generation of multiphase clock.

Counters

Twisted Ring Counter (Johnson Counter):

- It is obtained from a serial-in, serial-out shift register by providing a feedback from Q bar of last FF to the D input of first FF, called as twisted ring counter.
- The output Q is connected to the D input of next stage.
- It produces the unique sequence of states.
- Initially all the FF's are reset i.e. the state of counter is 0000.
- After each clock pulse, the level of Q1 is shifted to Q2, the level of Q2 to Q3, Q3 to Q4 & finally Q4 bar to Q1. The sequence is repeated after every 8 clock pulse.
- N Flip-Flop or n-bit Johnson counter can have 2n unique states and can count up to 2n states. So it is mod-2n counter.

Counters

Advantages:

1. It is more economical than ring counter but less than ripple counter.

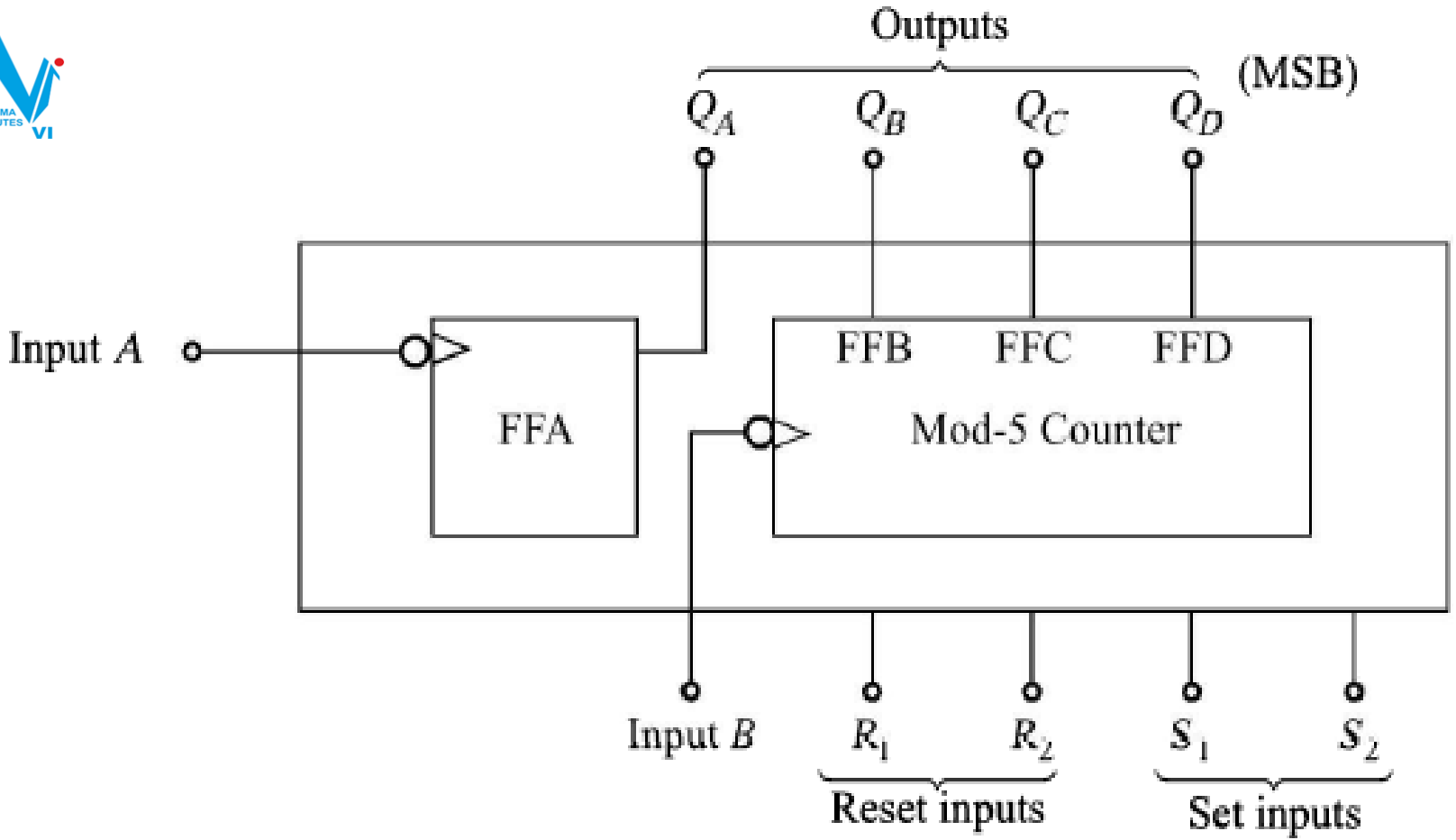
Disadvantages:

1. It requires 2 input gates for decoding regardless of number of Flip-Flops.
2. Both the ring counters suffer from problem of lock out i.e. if the counter finds itself in an unused state, it will persist in moving from one unused state to another and will never find its way to a used state.

Counters

7490 Asynchronous Counter IC: (Modulus of the counter)

- IC's available are divided into 3 groups A,B,C depending on its features. The IC consists of four MS-FF. The load, set & reset (clear) operations are asynchronous. i.e. independent of clock pulse.
- It consists of four FF's internally connected to provide mod-2 counter & a mod-5 counter. These counters can be used independently or in combination.
- FFA operates as a mod-2 counter whereas the combination of FFB, FFC & FFD form a mod-5 counter. The two reset inputs R1 & R2 both are connected to logic 1 level for clearing all FF's.
- The two set inputs S1 & S2 when connected to logic 1 level, are used for setting the counter to 1001.
- E.g. In a 7490 if QA output is connected to B input and the pulses are applied at A input, find the count sequence. Ans. It is a decade counter.


Outputs
(MSB)
Q_A
Q_B
Q_C
Q_D
Input A
FFA
FFB
FFC
FFD
Mod-5 Counter
Input B
R_1
R_2
S_1
S_2
Reset inputs
Set inputs

Divide by 6 Counter

Example 8.5

Design a divide-by-6 counter using 7490.

Solution

Connect the counter as divide-by-10 for normal binary sequence (Ex. 8.3). Outputs Q_B and Q_C are connected to the reset inputs. Therefore as soon as Q_B and Q_C both become 1, the counter is reset to 0000. Figure 8.17 shows the divide-by-6 ripple counter.

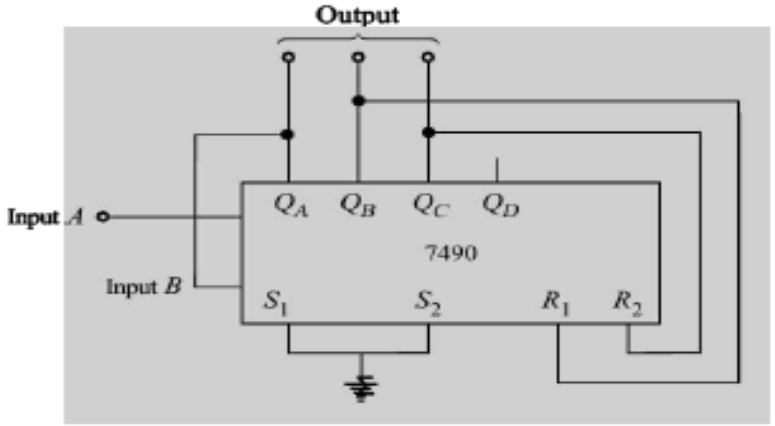


Fig. 8.17 ***A Divide-by-6 Ripple Counter Using 7490***

Applications of Flip-Flops

- Memories, e.g.,
  - Shift register
    - Parallel loading shift register : can be used for parallel to serial conversion in serial data communication
    - Serial in, parallel out shift register: can be used for serial to parallel conversion in a serial data communication system.

Synchronous Sequential Circuit Models

8.6.1 Synchronous Sequential Circuits Models

A general block diagram of a clocked sequential circuit is shown in Fig. 7.1. This is also known as a finite state machine (FSM). Depending upon the way the external outputs are obtained from the circuit, there are two different models of sequential circuits. These are Mealy model and Moore model.

Mealy Model

In the case of a Mealy model, the next state is a function of the present state and the present inputs. Its output is also a function of the present state and the present inputs. Figure 7.1 represents Mealy model.

In general, the next state and the output of a Mealy model are uniquely defined by

$$\text{Next State} = F_1 \text{ (Present state, inputs)}$$
$$\text{Outputs} = F_2 \text{ (Present state, inputs)}$$

Moore Model

The block diagram of a Moore model of circuit is shown in Fig. 8.38. Similar to the Mealy model, the next state in a Moore model is also a function of the present state and the inputs but the outputs of Moore model are functions of only the present state and are independent of the inputs. Therefore, the Moore model is defined as:

$$\text{Next state} = F_1 \text{ (Present state, inputs)}$$
$$\text{Outputs} = F_2 \text{ (Present state)}$$

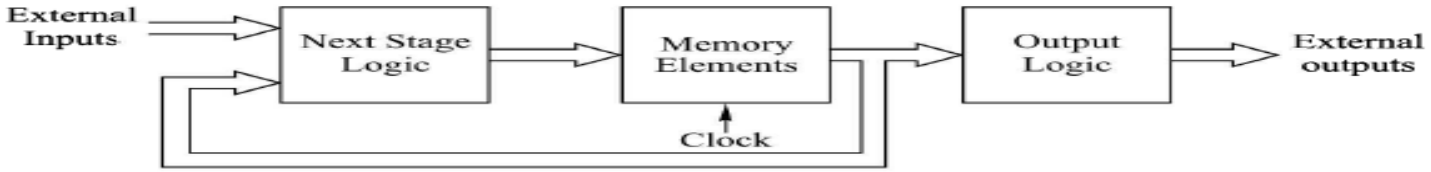
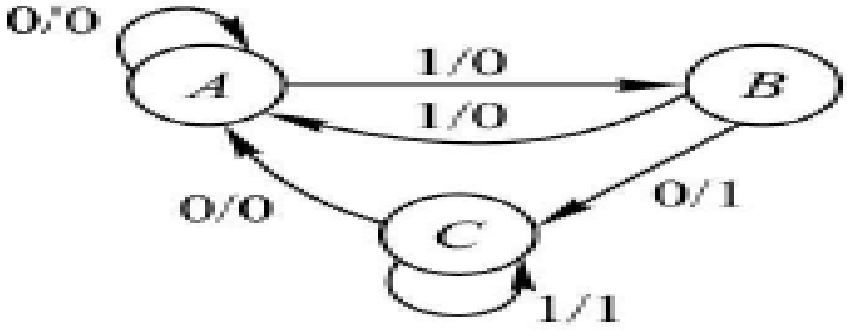


Fig. 8.38 ***Block Diagram of a Moore Model***

State Diagram

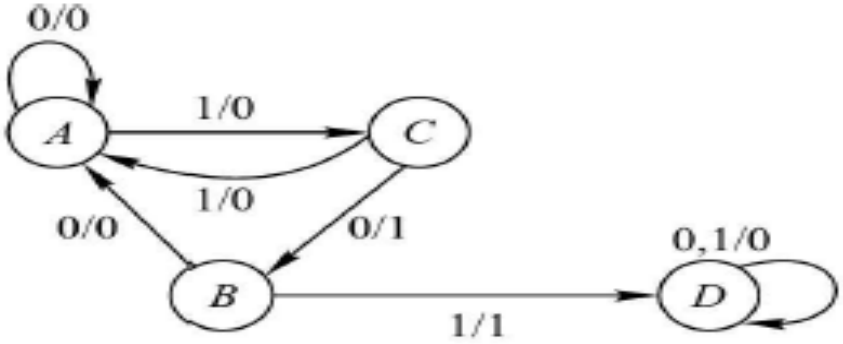
- It is a directed graph consisting of vertices and directed arcs between the nodes.
- Each state of circuit is represented by a node in the graph.
- A node is represented by a circle with the name of the state written in the circle and the directed arcs represent the state transitions.
- A state transition to next state will take place at the occurrence of a clock pulse representing an output.

(a)

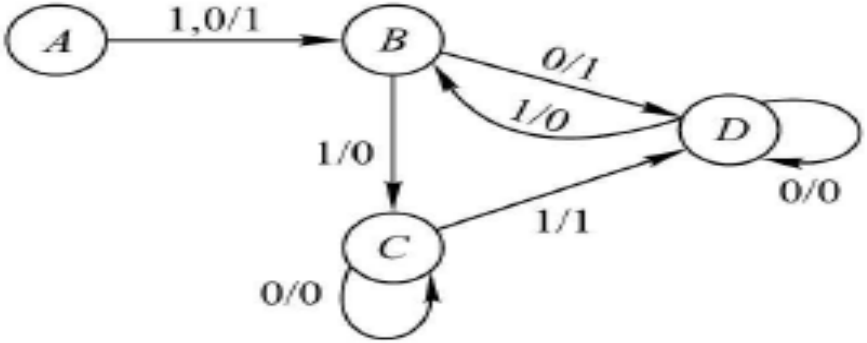


(b)

Fig. 8.40 *Illustrations of Directed Graph*



(a)



(b)

State Diagram of a D-Type FF

Example 8.16

Draw the state diagram of a *D* type FLIP-FLOP shown in Fig. 8.42.

Solution

This circuit has one input (D), two states ($Q = 0$ and $Q = 1$), and a positive edge-triggered clock (CK) terminal.

The two states 0 and 1 are represented by two nodes as shown in Fig. 8.43. Let us assume the circuit to be in state 0 and $D = 0$, when a clock pulse occurs. At the end of the clock pulse, the circuit remains in the same state. This is indicated by a directed arc emanating from the state 0 and terminating on the same state. If $D = 1$, while the circuit's present state is 0, state transition takes place taking the circuit to another state 1 when a clock pulse occurs. Similarly, if the circuit is in state 1, for $D = 0$ a clock pulse applied will cause state to change to 0, whereas for $D = 1$ it remains in the same state. All these operations are clearly indicated in the state diagram shown in Fig. 8.43.

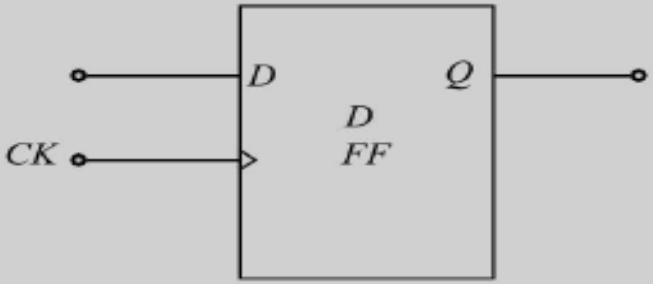


Fig. 8.42 ***D-Type*** FLIP-FLOP ***for Ex. 8.16***

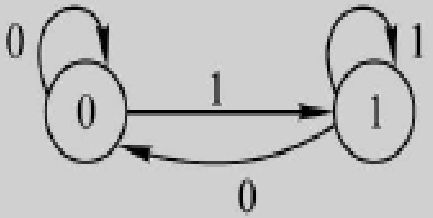


Fig. 8.43 ***State Diagram of a D*** FLIP-FLOP

State Diagram of a J-K FF

Example 8.17

Draw the state diagram of a *J-K* FLIP–FLOP.

Solution

A *J-K* FLIP–FLOP has two inputs (*J* and *K*) and one clock input (*CK*). There are two states ($Q = 0$ and $Q = 1$). Its state diagram is shown in Fig. 8.44.

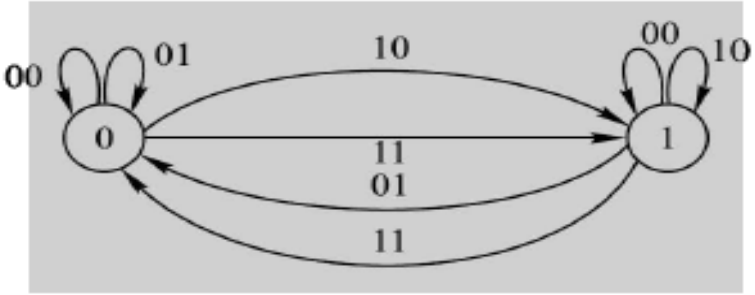


Fig. 8.44 ***State Diagram of a J-K*** FLIP–FLOP

Since, there are two inputs (*J* and *K*), therefore, there are $2^2 = 4$ possible input conditions and consequently, four directed arcs emanate from each state. The two inputs are indicated in *JK* order, i.e., the first input is represented by the first bit and the second input is represented by the second bit.

State Table

- It is a tabular form of describing the operation of a synchronous circuit.
- Each row of state table represents a state of the circuit and each column corresponds to a combination of external inputs.
- Entries in this table represents state transitions and outputs associated with these transitions.
- Number of rows is equal to the number of states of the circuit and the number of columns will be equal to combinations of the inputs.
- It can be easily constructed from a state diagram.

State Table

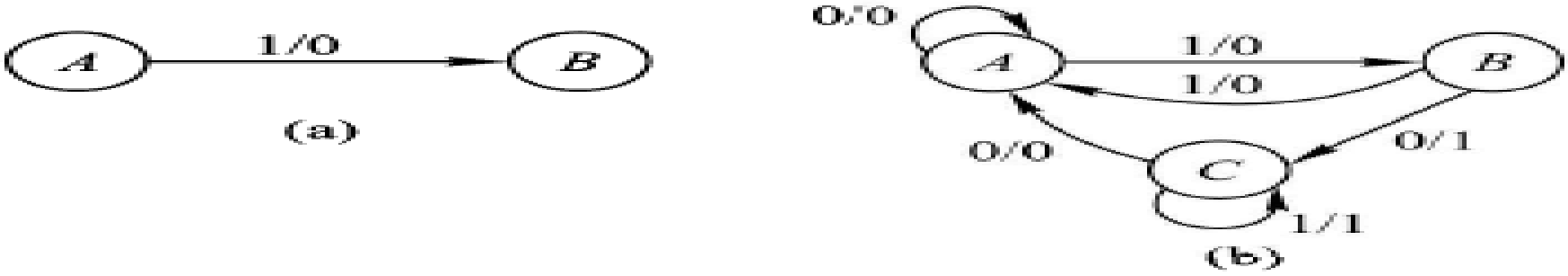


Fig. 8.40 *Illustrations of Directed Graph*

Example 8.18

Construct state table for the state diagram of Fig. 8.40*b*.

Solution

There are three states in this circuit A, B, and C, therefore, its state table will contain three rows. There is one input (X) and one output (Y) variable. There is one column for each value of X, i.e., $X = 0$ and $X = 1$. Each entry of the table contains two pieces of information: next state and output separated by a comma. The first entry is for the next state and after it is output.

Table 8.15 ***State Table for the State Diagram of Fig. 8.40b***

Present state	Next state, Output *NS, Y*	
PS	***X* = 0**	***X* = 1**
A	*A*, 0	*B*, 0
B	*C*, 1	*A*, 0
C	*A*, 0	*C*, 1

Rules for State Assignment

In the design of a sequential circuit, the complexity of the combinational circuit obtained depends on the chosen state assignment. There is no general procedure for the state assignment which produces the most cost effective design of the resulting combinational circuit, however, trial and error attempts at making state assignments are not practically feasible. The following thumb rules will help in generating simple combinational circuits.

Rule 1

Adjacent codes should be assigned to the states having the same next state for:

(a) each input combination
(b) different input combinations, if the next state can also be assigned adjacent codes
(c) some of the input combinations, not all

Rule 2

Adjacent codes should be assigned to the next state (s) of every present state.

Rule 3

Adjacent codes should be assigned to states that have the same outputs.

For a given state table, it may not be possible to achieve all the adjacencies of the above rules. Application of some of these rules may lead to conflicting state assignments, in such cases, the higher-priority rules should be given precedence. Even if the rules can be fully implemented, they do not guarantee an optimal assignment, i.e., these rules do not constitute an optimal algorithm.

Example 8.19

Partial state diagram of a sequential machine is shown in Fig. 8.45. Make suitable state assignment for obtaining minimal logic expression. Assume the machine with a single input and the complete state diagram contains seven states.

Solution

The portion of the state diagram shown has state transitions from the state A to C and from B to C for both the values of the input X and Y is the output. Since, there are 7 states, therefore, the number of FLIP-FLOPs required will be $\log_2^7 = 3$ (next higher integer). Its state assignment map can be prepared giving the state transitions from the present state to the next state. It is similar to a K-map. It is shown in Fig. 8.46. Q_3, Q_2 and Q_1 are the FLIP-FLOP outputs and the states will be assigned binary numbers in the order $Q_3 Q_2 Q_1$.

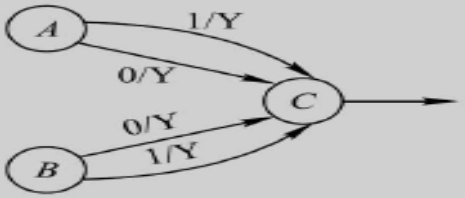


Fig. 8.45 *State Diagram for Ex. 8.19*

The two states, A and B can be assigned two adjacent cells according to Rule 1. One possible assignment is shown in the Fig. 8.46, for which $A = 100$ and $B = 110$. When the logic function is simplified using K-map or Quine-McCluskey method, combination of two adjacent cells will give a term with only two literals. Whereas if the states are assigned as 010 and 101, the expression will contain two minterms which cannot be combined.

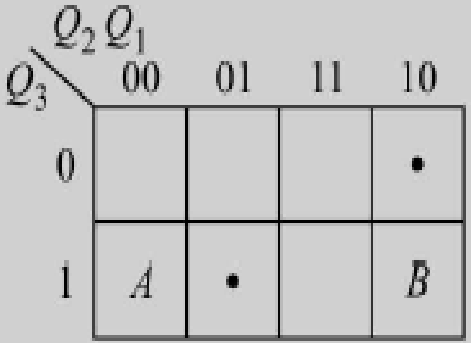


Fig. 8.46 *State Assignment Map of Ex. 8.19*

Example 8.20

For the partial state diagram of a sequential circuit shown in Fig. 8.47 make suitable state assignment for obtaining minimal logic expression. Assume one single input and a total of 6 states in the state diagram.

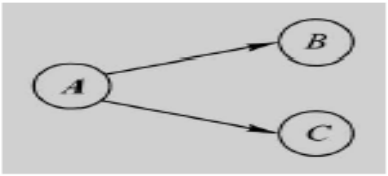


Fig. 8.47 ***State Diagram for Ex. 8.20***

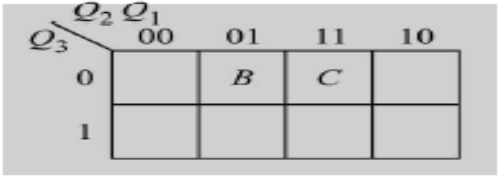


Fig. 8.48 ***State Assignment Map of Ex. 8.20***

Solution

State assignment map can be prepared similar to the Ex. 8.19. It is shown in Fig. 8.48.

One possible state assignment shown in the map gives $B = 001$ and $C = 011$.

Example 8.21

Table 8.16 gives state table of a sequential circuit. Give a good state assignment scheme.

Table 8.16 ***State Table for Ex. 8.21***

Present state *PS*	Next state, Output *NS, Y*	
	$X = 0$	$X = 1$
A	$B, 1$	$B, 0$
B	$C, 0$	$D, 1$
C	$E, 1$	$F, 0$
D	$F, 0$	$E, 1$
E	$G, 0$	$A, 0$
F	$A, 0$	$G, 0$
G	$B, 0$	$B, 0$

Solution

Since there are seven states in this circuit, therefore, it will have three FLIP–FLOPs FF3, FF2, and FF1 with state variables Q_3, Q_2, and Q_1 respectively. Application of the rules of state assignment is given below.

Rule 1

(a) Next state from the present states *A* and *G* is same. It is *B*. Therefore, the states *A* and *G* should be assigned adjacent codes.

(b) From the present state *C*, the next state is *E* for $X = 0$ and *F* for $X = 1$, whereas from the present state *D*, the next states are *F* and *E* for $X = 0$ and $X = 1$ respectively. Therefore, *C* and *D* may be assigned adjacent codes. Similarly, *E* and *F* may be assigned adjacent codes.

Rule 2

From the state *B*, the next states are *C* and *D*, therefore, *C* and *D* may be assigned adjacent states.
Similarly, *E* and *F*; *G* and *A* may be assigned adjacent states.

Rule 3

The outputs are same for the present states *A* and *C*; *B* and *D*, therefore, *A* and *C*; *B* and *D*, may be assigned adjacent codes.

Now, the state assignment is to be made so that as many as possible of these adjacencies can be accommodated. For this assignment map is prepared. Let us assign the state 000 to *A*, then *G* must be assigned an adjacent state. There are three possible adjacent states. Any one of them can be assigned to *G*, since *G* is not required to be adjacent to any other state. Similarly, all the other assignments are made and the resulting assignment map is shown in Fig. 8.49. The binary states are given below.

A-000
B-111
C-100
D-101
E-001
F-011
G-010

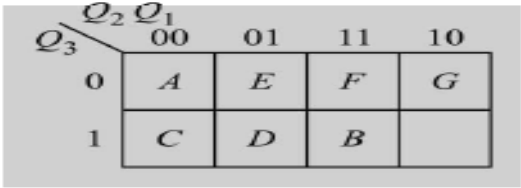


Fig. 8.49 ***State Assignment Map of Ex. 8.21***

8.6.7 State Equivalence and Minimisation

Two states are said to be equivalent if, for each input condition, they give exactly the same output and go to the same next state. In a state table, when two states are found to be equivalent, one of them is eliminated without altering the input-output relations. This process is referred to as *state-reduction*. Using the concept of equivalent states and state reduction, all possible equivalent states are determined and the reduced state table is obtained which will contain the minimum number of states. This process minimises the number of states.

Example 8.22

For the state table given in Table 8.17, obtain reduced state table with minimum number of states.

Table 8.17 ***State Table for Ex. 8.22***

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

Solution

(i) By observing Table 8.17, we see that from the initial state e, the next state is f for $X = 0$ and it is a for $X = 1$. The output is 0 and 1 for $X = 0$ and $X = 1$ respectively. Similarly, from the initial state g also, the next state is f and a for $X = 0$ and $X = 1$ respectively; and the output is 0 and 1 for $X = 0$ and $X = 1$ respectively. From this, we conclude that the two states e and g are equivalent, since for each input condition, they go to the same next state and give same output. We can eliminate one of them, say g. Thus g is replaced by e wherever it occurs.

(ii) After replacing g by e in the state table, we notice that the states d and f are equivalent. Thus, one of them, say d, can be eliminated.

The effect of eliminating equivalent states is illustrated in Table 8.18 and the reduced state table is given in Table 8.19. The reduced state table contains 5 states which is the minimum number of states required to implement the circuit represented by the given state table.

Table 8.18 ***Effect of Eliminating Equivalent States***

Present state	Next state		Output	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
a	c	b	0	0
b	$(d)f$	c	0	0
c	$(g)e$	$(d)f$	1	1
d	e	f	1	0
e	f	a	0	1
f	$(g)e$	f	1	0
g	f	a	0	1

Table 8.19 ***Reduced State Table***

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

Example 8.23

Design a minimal clocked sequential circuit for the reduced state table given in Table 8.19.

Solution

The reduced state table was obtained, using state equivalence and minimisation concepts, from Table 8.17. The next step in the design is to assign binary values to the states. Since, there are 5 states in this machine, therefore, the number of FLIP–FLOPs required is three. Let the three FLIP–FLOPs be designated as FF2, FF1, and FF0; and their outputs are Q_2, Q_1, and Q_0 respectively. Therefore, every state will be a 3-bit number with Q_2 value as the MSB and Q_0 as LSB. Assume D FLIP–FLOPs.

From application of the rules of state assignment, we find that

(i) the next states are same for the states c and f for $X = 0$ and $X = 1$, therefore, the states c and f should be assigned adjacent codes (Rule 1 (a)).

(ii) from the states b and e, the next state is f for $X = 0$, therefore, the states b and e should be assigned adjacent codes (Rule 1 (c)).

State assignment map can then be constructed. Let us assume $a = 000$ as the initial state. The state assignment map is shown in Fig. 8.50.

Therefore, the states assigned are:

$$a = 000$$
$$b = 011$$
$$c = 010$$
$$e = 111$$
$$f = 110$$

Q_0 \ Q_2Q_1	00	01	11	10
0	a	c	f	
1		b	e	

Fig. 8.50 ***State Assignment Map***

The state transition and output table is constructed next. From the state a (000) the next state is $c = 010$. When $X = 0$. The output corresponding to this is $Y = 0$, these entries are made in the first row. Similarly all the entries are made in this table.

Since D-type of FLIP–FLOPs have been chosen here, therefore, it is not necessary to construct excitation table separately. For a D-type FLIP–FLOP for next state to be 0, the D input must be 0 just before the clock pulse. Similarly, for the next state to be 1, its D input must be 1. Therefore Q_2^*, Q_1^* and Q_0^* values are the same as the excitation values of D_2, D_1, and D_0 respectively. The state transition and output table is given in Table 8.20.

Table 8.20 ***State Transition and Output Table***

Present state			Input	Next state			Output
Q_2	Q_1	Q_0	X	Q_2^*	Q_1^*	Q_0^*	Y
0	0	0	0	0	1	0	0
0	1	1	0	1	1	0	0
0	1	0	0	1	1	1	1
1	1	1	0	1	1	0	0
1	1	0	0	1	1	1	1
0	0	0	1	0	1	1	0
0	1	1	1	0	1	0	0
0	1	0	1	1	1	0	1
1	1	1	1	0	0	0	1
1	1	0	1	1	1	0	0

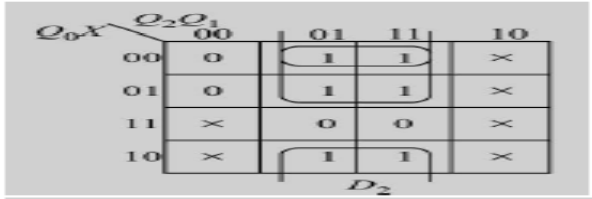
The next step involves finding out the minimised logic expressions for D_2, D_1, D_0, and Y in terms of Q_2, Q_1, Q_0, and X. This is a combinational logic design problem. For this K-maps are constructed for D_2, D_1, D_0, and Y and are minimised. The K-maps are shown in Fig. 8.51.

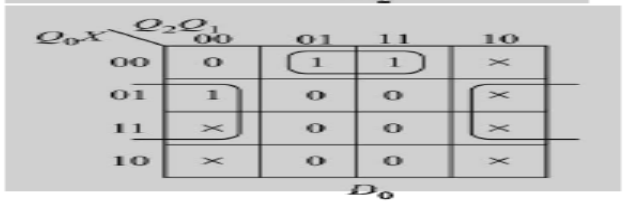
Since, there are eight states possible using three bits, out of which only five states are required for this circuit. The other three states do not exist and are therefore, taken as don't cares (X's).

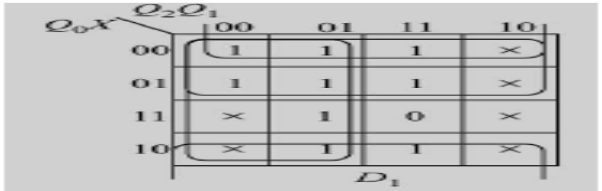
The minimised expressions are:

$$D_2 = Q_1\overline{Q}_0 + Q_1\overline{X}$$
$$D_1 = \overline{Q}_2 + \overline{Q}_0 + \overline{X}$$
$$D_0 = \overline{Q}_1 X + Q_1\overline{Q}_0\overline{X}$$
$$Y = Q_1\overline{Q}_0\overline{X} + \overline{Q}_2 Q_1\overline{Q}_0 + Q_2 Q_0 X$$

The complete circuit is given in Fig. 8.52.







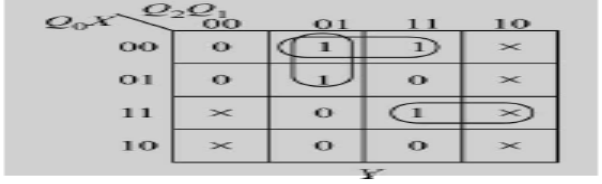


Fig. 8.51 ***K-Maps for D_2, D_1, D_0 and Y***

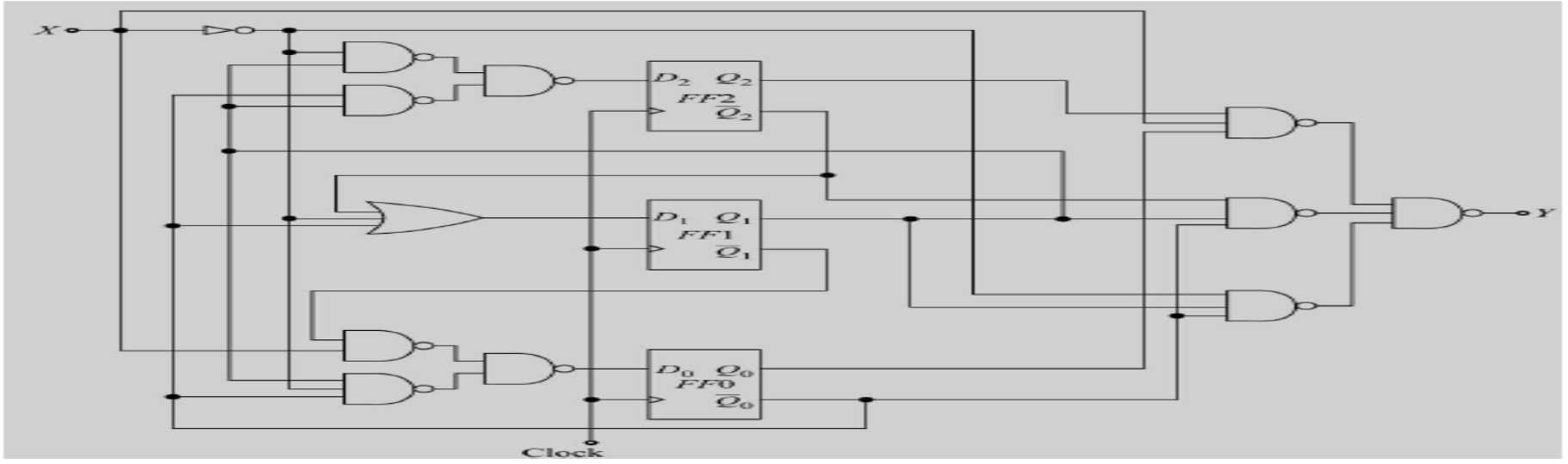


Fig. 8.52 ***Sequential Circuit for Ex. 8.23***

Example 8.24

For the reduced state table given in Table 8.19, design the circuit assuming the state assignment given below. Compare the hardware requirements of this state assignment with the state assignment done in Example 8.23.

$$a = 000,\ b = 001,\ c = 010,\ e = 011, \quad \text{and} \quad f = 100.$$

Solution

The state transition and output table is given in Table 8.21, and the *K*-maps are given in Fig. 8.53. The unused states have been taken as don't care conditions.

Table 8.21 ***State Transition and Output Table for Ex. 8.24***

Present state			Input	Next state			Output
Q_2	Q_1	Q_0	X	Q_2^*	Q_1^*	Q_0^*	Y
0	0	0	0	0	1	0	0
0	0	0	1	0	0	1	0
0	0	1	0	1	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	1
0	1	1	0	1	0	0	0
0	1	1	1	0	0	0	1
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0

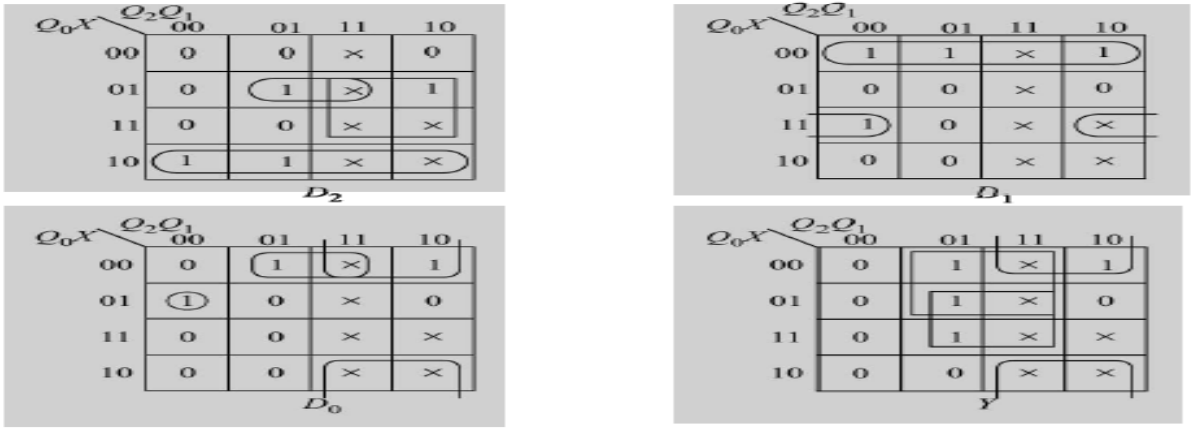


Fig. 8.53 ***K-Maps for Table 8.21***

The minimised expressions are:

$$D_2 = Q_0 \cdot \overline{X} + Q_2 \cdot X + Q_1 \cdot \overline{Q}_0 \cdot X$$
$$D_1 = \overline{Q}_0 \cdot \overline{X} + \overline{Q}_1 \cdot Q_0 \cdot X$$
$$D_0 = Q_2 \cdot \overline{X} + Q_1 \cdot \overline{Q}_0 \cdot \overline{X} + \overline{Q}_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0 \cdot X$$
$$Y = Q_1 \overline{Q}_0 + Q_1 \cdot X + Q_2 \cdot \overline{X}$$

The complete circuit is given in Fig. 8.54.

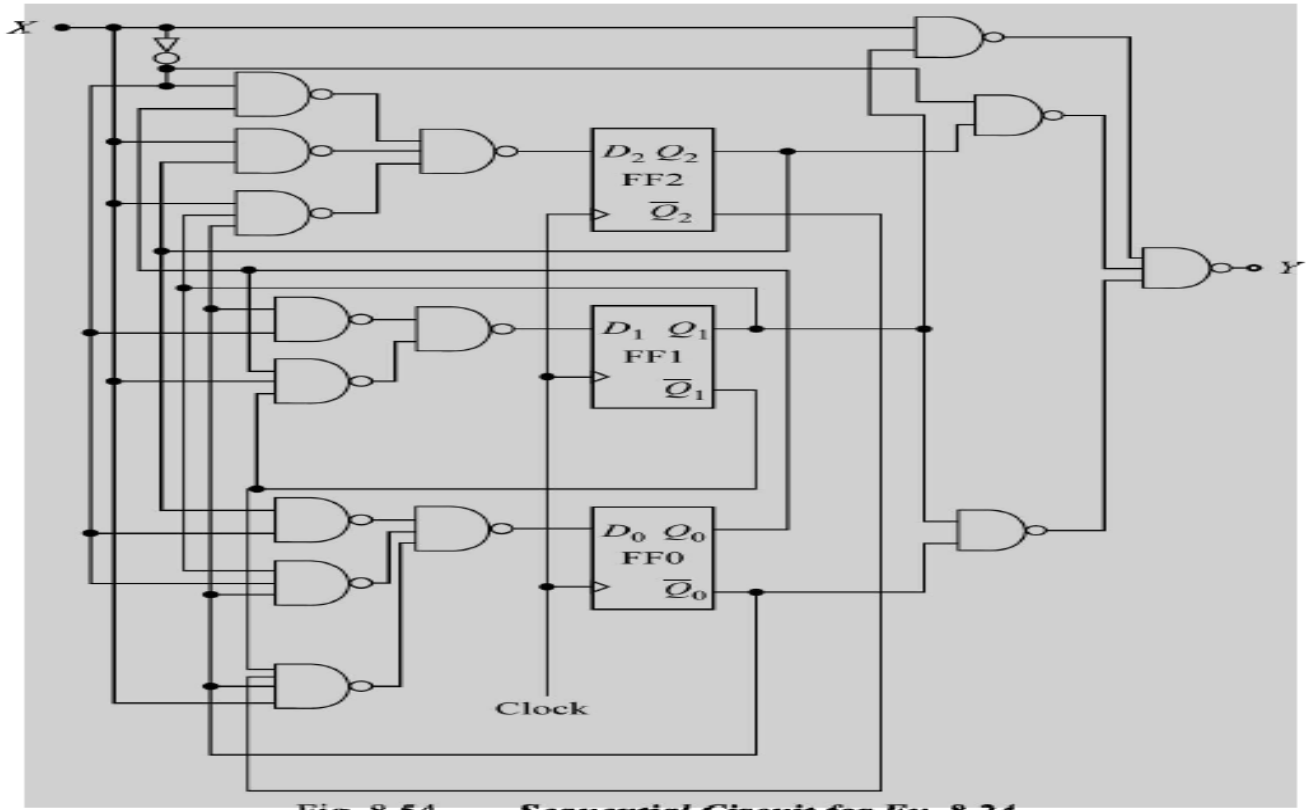


Fig. 8.54 ***Sequential Circuit for Ex. 8.24***

The circuit can be designed using *J-K* FFs also (Prob. 8.30).
The requirement of gates for the circuits of Figs. 8.52 and 8.54 are given below.

For Fig. 8.52

3-input NAND gates-4 (one NAND gate can be shared between D_0 and Y)
2-input NAND gates-5
3-input OR gate-1

For Fig. 8.54

4-input NAND gate-1
3-input NAND gates-6
2-input NAND gates-7 (two NAND gates can be shared between D_0 and Y)

From the requirement of gates by two different state assignments, we can conclude that a carefully done state assignment will lead to a circuit requiring lesser number of gates.

Example 8.25

A synchronous sequential circuit is to be designed having a single input X and a single output Y to detect single change of level (from 0 to 1 or from 1 to 0) in a 3-bit word and produce an output $Y = 1$, otherwise $Y = 0$. When a new 3-bit word is to come, the circuit must be at its initial (reset) state and there should be a time delay of one clock cycle between the words.

Solution

There are eight possible words of three bits. These are: 000, 001, 010, 011, 100, 101, 110, and 111. The number of level change is one in 001, 011, 100, and 110. When any one of these words is applied bit by bit at the X input, Y must be 1 after three clock cycles, i.e., when the third bit is applied. A state diagram is to be constructed first for this. The state diagram is given in Fig. 8.55.

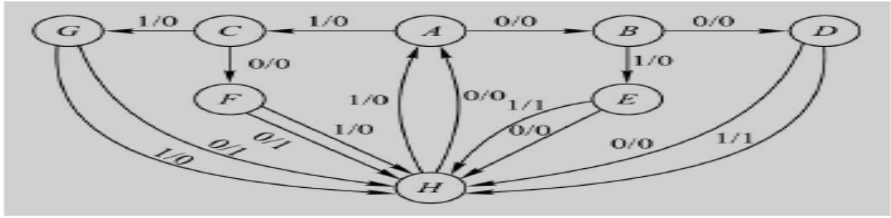


Fig. 8.55 ***State Diagram of Ex. 8.25***

A is the initial state. For each 3-bit word, find the next state and output bit by bit. Let us take the input word 000. When first 0 is applied, the circuit goes to state B and output is 0, next 0 will take the circuit to D with output 0, the third 0 will take the circuit to H with output 0. Therefore, for this 3-bit word, the output is 0 at the end of the third bit. Hence, we conclude that the number of level changes from the first to the third bit is not 1. When the fourth clock pulse appears, the circuit is reset, i.e., it reaches its initial state A irrespective of the value of X (0 or 1).

Now consider input word 110. From the reset state the circuit goes to $C \rightarrow G \rightarrow H$ and produces an output of 1 at the third clock pulse indicating 1 level change. On fourth clock pulse, the circuit resets. Similarly, it can be verified for each input word.

Next a state table (Table 8.22) is constructed from the state diagram.

Table 8.22 ***State Table of Ex. 8.25***

Present State	Next state, Output	
	$X = 0$	$X = 1$
A	B, 0	C, 0
B	D, 0	E, 0
C	F, 0	G, 0
D	H, 0	H, 1
E	H, 0	H, 1
F	H, 1	H, 0
G	H, 1	H, 0
H	A, 0	A, 0

Now the state reduction table is to be constructed. From the state table, we observe the following.

(i) From the present states D and E, the next state and output are same for $X = 0$ and $X = 1$, therefore, these two states are equivalent and state D is eliminated and replaced by state E.

(ii) From the present states F and G, the next state and output are same for $X = 0$ and $X = 1$, therefore, these two states are equivalent and state G is eliminated and replaced by state F.

The reduced state table is given in Table 8.23 and Fig. 8.56 gives reduced state diagram.

Table 8.23 ***Reduced State Table of Ex. 8.25***

Present state	Next state, Output	
	$X = 0$	$X = 1$
A	B, 0	C, 0
B	E, 0	E, 0
C	F, 0	F, 0
E	H, 0	H, 1
F	H, 1	H, 0
H	A, 0	A, 0

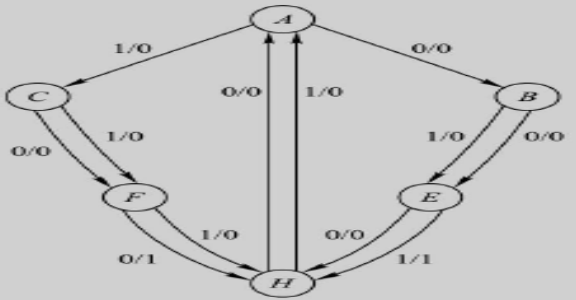


Fig. 8.56 ***Reduced State Diagram of Ex. 8.25***

Since, there are six states, therefore, the number of FLIP–FLOPs required is 3. Let us assume D FLIP–FLOPs.

From the states E and F, the next state is H for both the values of X, therefore, E and F should be assigned adjacent codes.

States having same outputs should be assigned adjacent states. These are:

$$AB, AC, BC, AH, BH, CH$$

The state assignment map is shown in Fig. 8.57.

The states assigned are:

$A = 000$ $\quad$ $E = 100$

$B = 010$ $\quad$ $F = 101$

$C = 001$ $\quad$ $H = 011$

Q_0 \ $Q_2 Q_1$	00	01	11	10
0	A	B		E
1	C	H		F

Fig. 8.57 ***State Assignment Map of Ex. 8.25***

State transition and output table is constructed on the basis of state assignment. It is given in Table 8.24 and K-maps are given in Fig. 8.58.

The minimised logic expressions are:

$$D_2 = Q_1 \overline{Q}_0 + \overline{Q}_2 \overline{Q}_1 Q_0$$

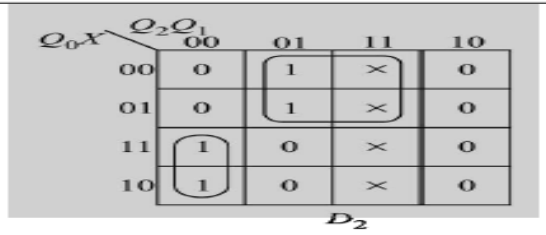
$$D_1 = Q_2 + \overline{Q}_1 \overline{Q}_0 \overline{X}$$

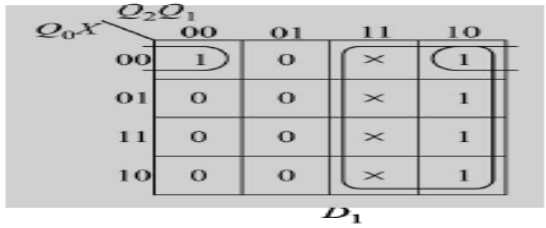
$$D_0 = Q_2 + \overline{Q}_1 X + \overline{Q}_1 Q_0$$

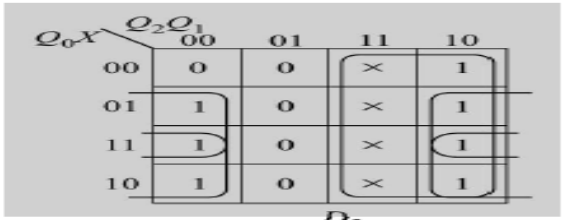
$$Y = Q_2 \overline{Q}_0 X + Q_2 Q_0 \overline{X}$$

Table 8.24 ***State Transition and Output Table of Ex. 8.25***

Present state			Input	Next state			Output
Q_2	Q_1	Q_0	X	Q_2^*	Q_1^*	Q_0^*	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	0	0	0
0	0	1	0	1	0	1	0
1	0	0	0	0	1	1	0
1	0	1	0	0	1	1	1
0	1	1	0	0	0	0	0
0	0	0	1	0	0	1	0
0	1	0	1	1	0	0	0
0	0	1	1	1	0	1	0
1	0	0	1	0	1	1	1
1	0	1	1	0	1	1	0
0	1	1	1	0	0	0	0







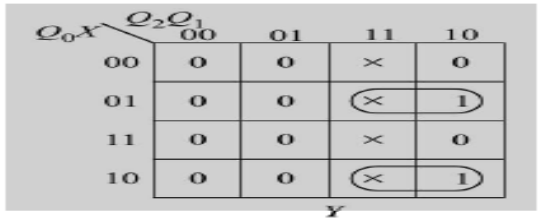


Fig. 8.58 ***K-Maps of Ex. 8.25***

The complete circuit can be drawn using three *D* FLIP–FLOPs and NAND gates.

Example 8.26

Design a serial adder to add two binary numbers.

Solution

We are familiar with the process of decimal addition using paper and pencil. In this, digits in the same significant position are added starting from the LSD alongwith a carry generated from the addition of digits from the previous significant position. A similar process is used for the serial addition of binary numbers.

Let the two binary inputs be X_1 and X_2. At X_1 and X_2 inputs are applied sequentially. Assume

$$\begin{aligned} X_1 &= 10010110 \\ X_2 &= 11011110 \\ \hline Y &= 101110100 \end{aligned}$$

The sum output Y does not depend on the present inputs alone, it also depends on the carry from the previous position. In the LSB (b_0) position both the input bits are 0 and the output is also 0; in the next position b_1, the sum of 1 and 1 is 0 and a carry c_1 is generated; in the next position b_2, the two 1's alongwith the carry c_1 are to be added resulting in sum bit $Y = 1$ and so on. Therefore, a memory device is required to keep track of the carry input. The memory should have two states 0 and 1 corresponding to carry = 0 or 1. We can construct a state transition and output table from the above discussion.

Table 8.25 gives the state transition and output table and Fig. 8.59 gives *K*-maps for the *D* input of *D* FLIP–FLOP and output *Y*.

Table 8.25 ***State Transition and Output Table for Serial Adder***

Present state $PS(Q)$	Inputs X_1	Inputs X_2	Next state $NS(Q^*)$	Output Y
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Fig. 8.59 ***K-Maps for*** FLIP–FLOP ***Input D and Output Y***

The logic expressions for D and Y are:

$$D = X_1X_2 + X_1Q + X_2Q$$
$$Y = \overline{X}_1\,\overline{X}_2Q + \overline{X}_1X_2\,\overline{Q} + X_1X_2Q + X_1\,\overline{X}_2\,\overline{Q}$$

Its circuit is shown in Fig. 8.60.

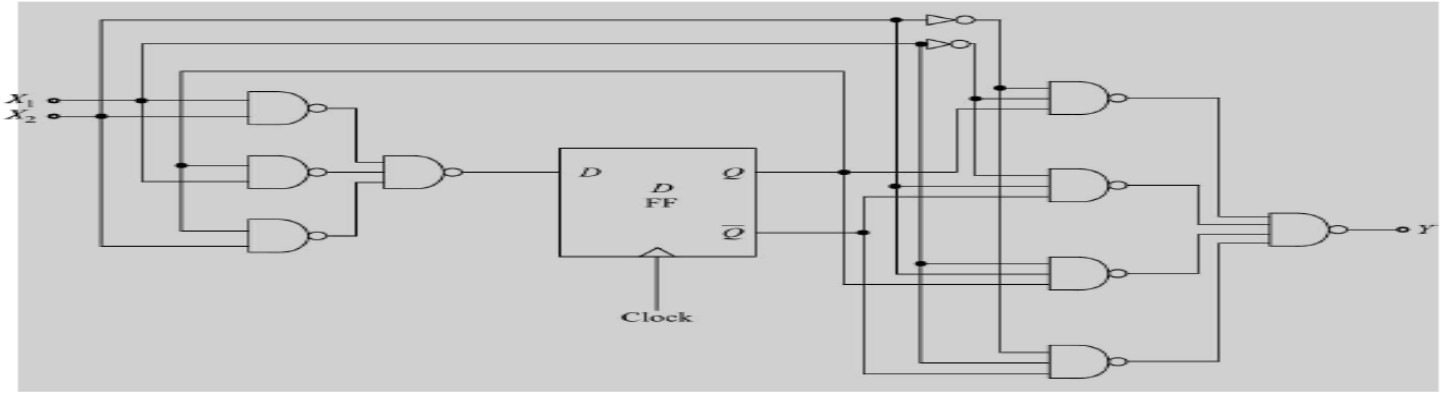


Fig. 8.60 ***Serial Adder Circuit***

Example 8.27

Design a sequence detector circuit to detect a serial input sequence of 1010. It should produce an output 1 when the input pattern has been detected.

Solution

Let the input string be 10101010, the desired output string can be determined, which is given below

Input X	1	0	1	0	1	0	1	0
Output Y	0	0	0	1	0	1	0	1

A state diagram can be constructed for the required input-output relationship.

Let us start with the initial state A. If the input is 0, the detection cycle must not start and therefore, the state remains the same and output is 0. When it is 1, it should go to the next state B with output as 0. In the present state B, if the input is 0 the next state will be C and output 0, while for an input 1, it is to be counted as the first correct bit in the string (since it is the second 1 bit from the start) and the state of the circuit should remain unchanged. In the present state C, if an input bit 0 occurs, the circuit should go back to the initial state (since it is second consecutive 0), while an input 1 causes state transition to next state D. When the circuit is in the state D, a 0 input will detect the correct sequence and will produce an output of 1. If the input is 1, it is a second consecutive 1 which must take the circuit to the state B. The complete state diagram is shown in Fig. 8.61. Its state table is constructed as given in Table 8.26.

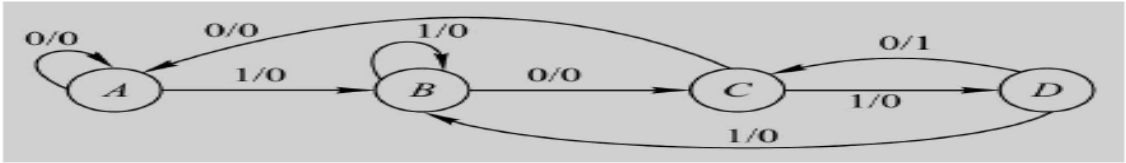


Fig. 8.61 ***State Diagram of Sequence Detector Circuit***

There are four states in this circuit and no equivalent states are present. Therefore, two D-type FLIP–FLOPs can be used for the implementation of this circuit.

Table 8.26 ***State Table of Sequence Detector***

Present state	Next state, Output	
	$X = 0$	$X = 1$
A	A, 0	B, 0
B	C, 0	B, 0
C	A, 0	D, 0
D	C, 1	B, 0

The two states B and D must be assigned adjacent codes, since the next state is same from these states for $X = 0$ and $X = 1$. Therefore, the following state assignment is made.

$$A \rightarrow 00$$
$$B \rightarrow 01$$
$$C \rightarrow 10$$
$$D \rightarrow 11$$

Table 8.27 gives state transition and output table. From this K-maps are constructed for the FLIP–FLOP inputs D_1 and D_0; and output Y. These are given in Fig. 8.62.

Table 8.27 ***State Transition and Output Table***

Present state		Input	Next state		Output
Q_1	Q_0	X	Q_1^*	Q_0^*	Y
0	0	0	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	1	0	1
0	0	1	0	1	0
0	1	1	0	1	0
1	0	1	1	1	0
1	1	1	0	1	0

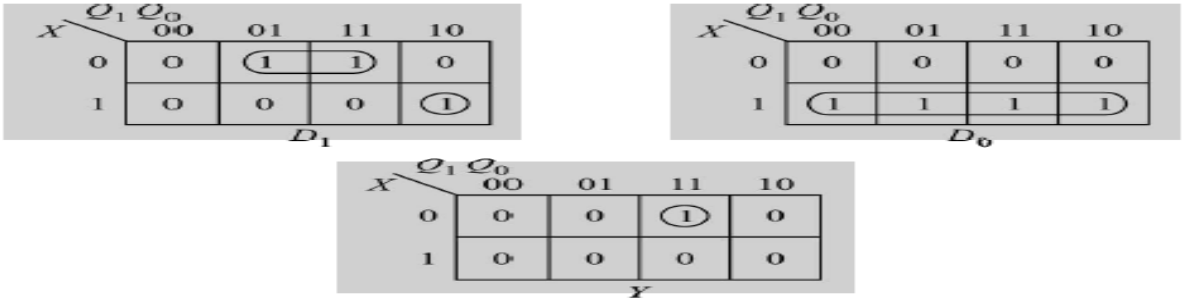


Fig. 8.62 ***K-Maps of Ex. 8.27***

The minimised logic expressions are:

$$D_1 = Q_0 \overline{X} + Q_1 \overline{Q}_0 X$$
$$D_0 = X$$
$$Y = Q_1 Q_0 \overline{X}$$

Its circuit diagram is given in Fig. 8.63.

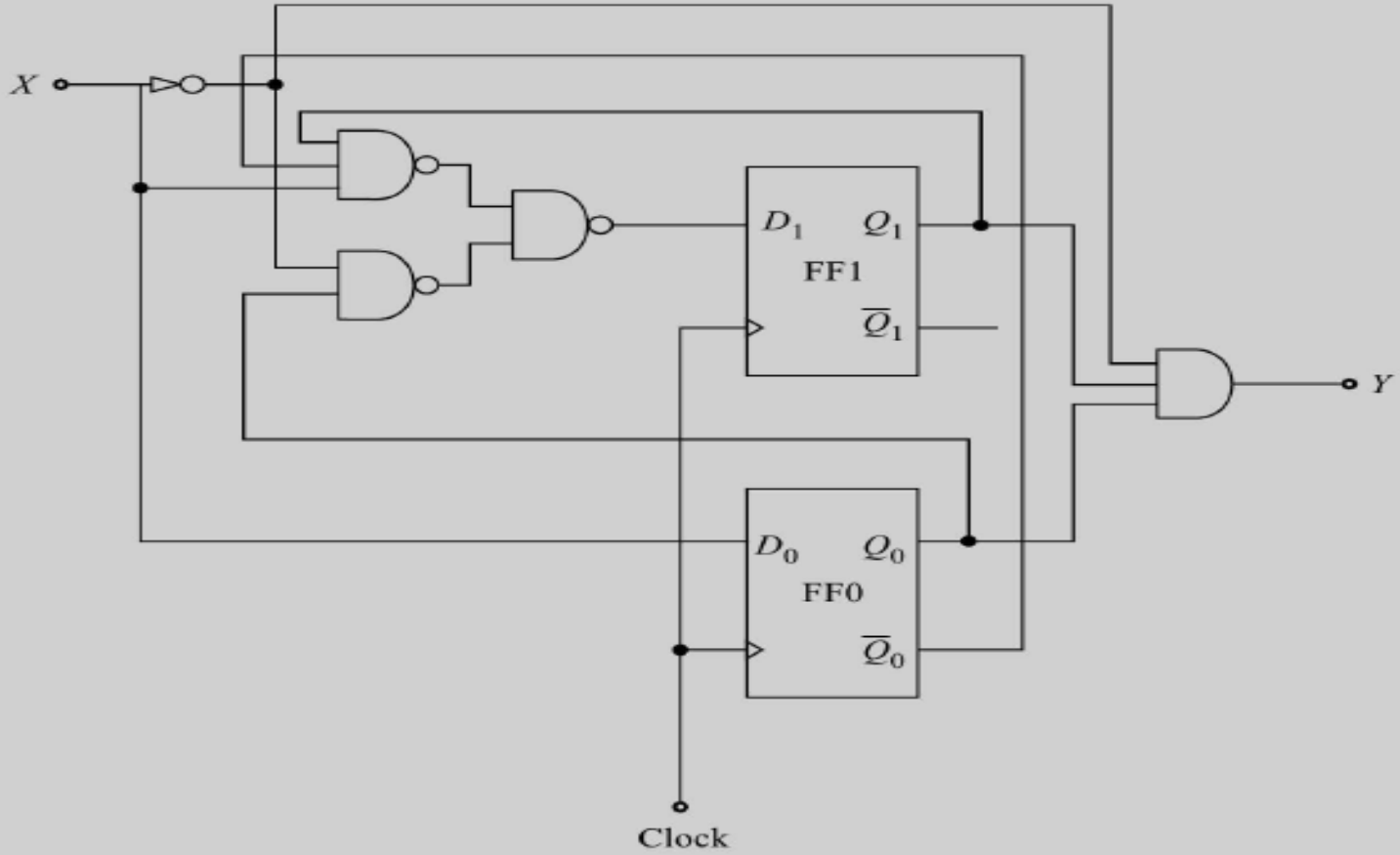


Fig. 8.63 ***Circuit of Sequences Detector***

Example 8.1

Design a sequence generator to generate the sequence ... 1101011...

Solution

The minimum number of FLIP–FLOPs, N, required to generate a sequence of length S is given by

$$S \leq 2^N - 1 \tag{8.2}$$

In this case $S = 7$, therefore, the minimum value of N, which may generate this sequence is 3. However, it is not guaranteed to lead to a solution. If the given sequence leads to seven distinct states, then only three FLIP–FLOPs are sufficient otherwise we have to increase the number of FLIP–FLOPs. We write the states of the circuit as given in Table 8.2. The prescribed sequence is listed under Q_2 and the sequence listed under Q_1 and Q_0 are the same sequence delayed by one and two clock pulses respectively. From the table we observe that all the states are not distinct, which means $N = 3$ is not sufficient. Next we assume $N = 4$ and prepare Table 8.3 in a similar manner as Table 8.2. The last column gives the required serial input for getting the desired change of state when a clock pulse is applied. This is obtained by assuming D-type FLIP–FLOPs and looking at the Q_3 output. For example, at the falling edge of the first clock pulse, $Q_3 = 1$. The second clock pulse must result in $Q_3 = 1$ which requires its D input to be 1. In the same manner, all the entries in column Y are determined. The K-map of Table 8.3 is given in Fig. 8.9 and the simplified expression is given by

$$Y = \overline{Q}_3 + \overline{Q}_1 + \overline{Q}_0 \tag{8.3}$$

Table 8.2 ***State Table of Sequence Generator*** **($N = 3$)**

Number of clock pulses	FLIP–FLOP outputs		
	Q_2	Q_1	Q_0
1	1	1	1
2	1	1	1
3	0	1	1
4	1	0	1
5	0	1	0
6	1	0	1
7	1	1	0

Table 8.3 ***Truth Table of Sequence Generator*** **($N = 4$)**

Number of clock pulses	FLIP–FLOP outputs				Serial input
	Q_3	Q_2	Q_1	Q_0	Y
1	1	1	1	0	1
2	1	1	1	1	0
3	0	1	1	1	1

Table 8.3 (*Continued*)

Number of clock pulses	FLIP-FLOP outputs				Serial input
	Q_3	Q_2	Q_1	Q_0	Y
4	1	0	1	1	0
5	0	1	0	1	1
6	1	0	1	0	1
7	1	1	0	1	1
1	1	1	1	0	1
2	1	1	1	1	0
3	0	1	1	1	1
*	1	0	1	1	0
*	0	1	0	1	1
*	1	0	1	0	1

The sequence generator circuit can be designed using a 4-stage shift register using *D*-type FLIP-FLOPs and the decoder circuit given by Eq. (8.3).

AIM:

To design a circuit which can generate a sequence using J-K flip flop

IC used - 7473 (J-K flip flop)

THEORY -

A circuit which generate a prescribed sequence of bits, in synchronous with a clock, is referred as a sequence generator. It is possible to design a sequence generator using counters or using the shift registers

a) Design the synchronous counter which goes through the states 1 — 3 — 7 — 6

K - Maps

1. J_1

Q_1 \ Q_3Q_2	00	01	11	10
0	X	X	1	X
1	X	X	X	X

$J_1 = 1$

2. K_1

Q_1 \ Q_3Q_2	00	01	11	10
0	X	X	X	X
1			1	X

$K_1 = Q_3$

3 K_2

Q_1 \ Q_3Q_2	00	01	11	10
0	X	X	1	X
1	X	0	0	X

$K_2 = \overline{Q_1}$

4 J_2

Q_1 \ Q_3Q_2	00	01	11	10
0	X	X	X	X
1	1	X	X	X

$J_2 = 1$

Excitation Table

PS	N.S	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Truth Table

PRESENT			NEXT			FLIP FLOP OUTPUT					
Q_3	Q_2	Q_1	Q_3'	Q_2'	Q_1'	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	x	x	x	x	x	x	x	x	x
0	0	1	0	1	1	0	x	1	x	x	0
0	1	0	x	x	x	x	x	x	x	x	x
0	1	1	1	1	1	1	x	x	0	x	0
1	0	0	x	x	x	x	x	x	x	x	x
1	0	1	x	x	x	x	x	x	x	x	x
1	1	0	0	0	1	x	1	x	1	1	x
1	1	1	1	1	0	x	0	x	0	x	1

5 J_3

Q_1 \ Q_3Q_2	00	01	11	10
0	X	X	X	X
1	.	1	X	X

$J_3 = Q_2$

6 K_3

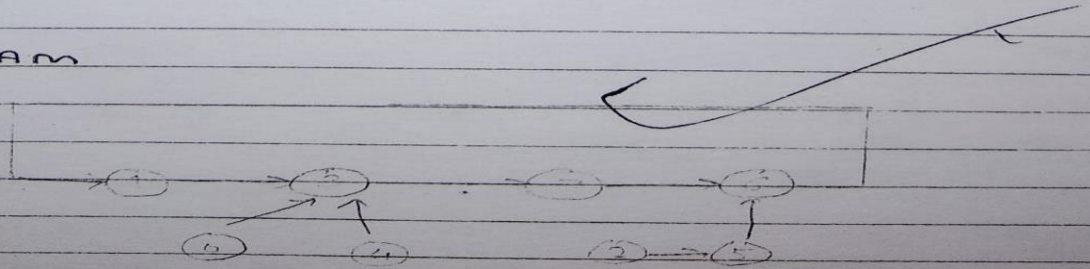
Q_1 \ Q_3Q_2	00	01	11	10
0	X	X	1	X
1	X	X		X

$K_3 = \overline{Q_1}$

BUSH TABLE

Present state			Next state			FLIP FLOP OUTPUTS					
Q_3	Q_2	Q_1	Q_3'	Q_2'	Q_1'	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	1	1	1	0	1	1	0	1
0	0	1	0	1	1	1	0	1	0	0	0
0	1	0	1	0	1	1	0	1	1	1	1
0	1	1	1	1	1	1	0	1	0	1	0
1	0	0	0	1	1	1	1	1	1	0	1
1	0	1	1	1	0	1	1	1	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	0	1	0

BUSH DIAGRAM



BUSH DIAGRAM

b Design the synchronous counter which goes through the states 1 – 2 – 8 – 12 – 1

~~K-Maps~~ Truth Table

PRESENT STATE				NEXT STATE				FLIP FLOP# OUTPUTS							
Q_4	Q_3	Q_2	Q_1	Q_4'	Q_3'	Q_2'	Q_1'	J_1	K_1	J_2	K_2	J_3	K_3	J_4	K_4
0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	1	0	0	1	0	X	1	1	X	0	X	0	X
0	0	1	0	1	0	0	0	0	X	X	1	0	X	1	X
0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X
0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X
0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X
0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X
1	0	0	0	1	1	0	0	0	X	0	X	1	X	X	1
1	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X
1	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X
1	1	0	0	0	0	0	1	1	X	0	X	X	1	X	1
1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

K-Maps

1 J_1

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	1	
01	X	X	X	X
11	X	X	X	X
10		X	X	X

$J_1 = Q_3$

2. K_1

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	X	X
01	1	X	X	X
11	X	X	X	X
10	X	X	X	X

$K_1 = 1$.

3 J_2

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X		
01	1	X	X	X
11	X	X	X	X
10	X	X	X	X

$J_2 = Q_1$

4. K_2

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10	1	X	X	X

$K_2 = 1$

5 J_3

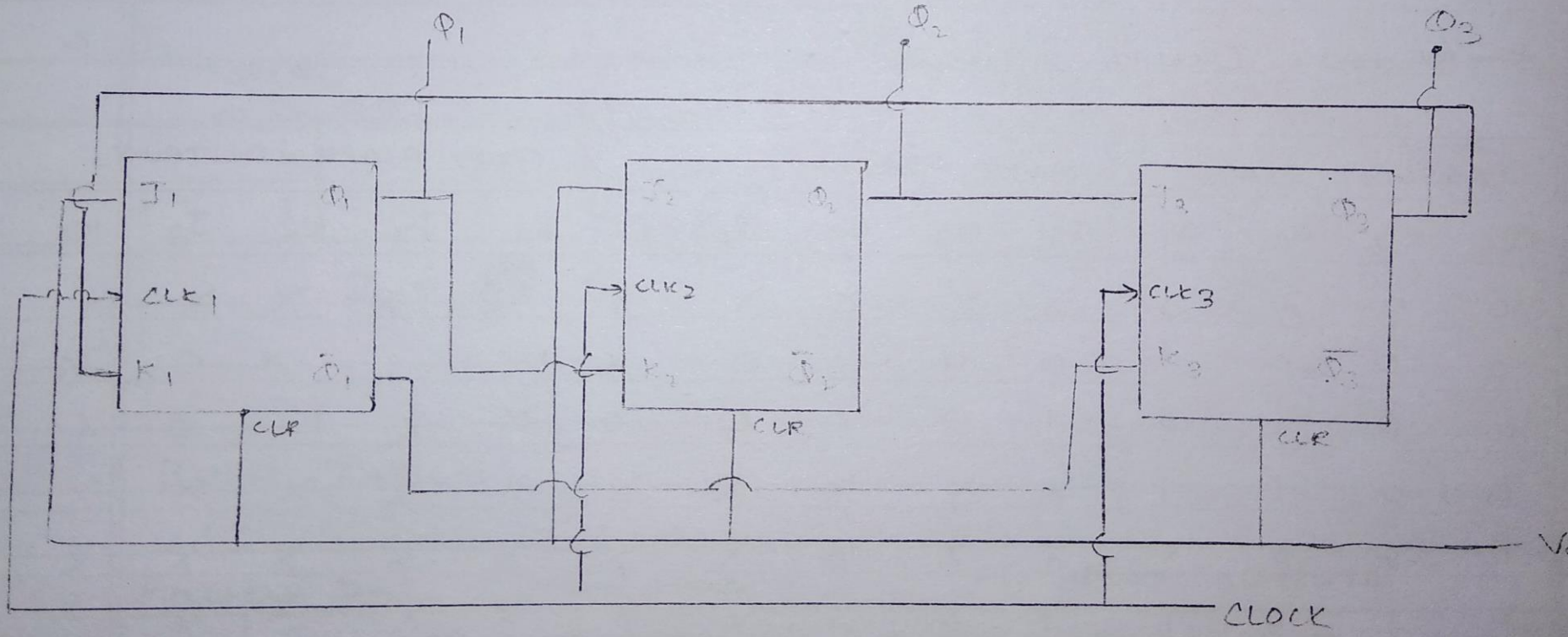
Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	X	1
01		X	X	X
11	X	X	X	X
10		X	X	X

$J_3 = Q_4$

6. K_3

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	1	X
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$K_3 = 1$



SEQUENCE GENERATOR

for 1-3-7-6.

J_4

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	X	X
01		X	X	X
11	X	X	X	X
10	1	X	X	X

$J_4 = \overline{Q_1}$

Q_2Q_1 \ Q_4Q_3	00	01	11	10
00	X	X	1	
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$K_4 = Q_3$

BUSH TABLE

PRESENT TABLE				NEXT TABLE				FLIP-FLOP OUTPUTS							
Q_4	Q_3	Q_2	Q_1	Q_4'	Q_3'	Q_2'	Q_1'	J_1	K_1	J_2	K_2	J_3	K_3	J_4	K_4
0	0	0	0	1	0	0	0	0	1	0	1	0	1	1	0
0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0
0	0	1	0	1	0	0	0	0	1	0	1	0	1	1	0
0	0	1	1	0	0	0	0	0	1	1	1	0	1	0	0
0	1	0	0	1	0	1	0	1	1	0	1	0	1	1	1
0	1	0	1	0	0	1	0	1	1	1	1	0	1	0	1
0	1	1	0	1	0	0	1	1	1	0	1	0	1	1	1
0	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1
1	0	0	0	1	1	0	0	0	1	0	1	1	1	1	0
1	0	0	1	1	1	1	0	0	1	1	1	1	1	0	0
1	0	1	0	1	1	0	0	0	1	0	1	1	1	1	0
1	0	1	1	1	1	0	0	0	1	1	1	1	1	0	0
1	1	0	0	0	0	0	1	1	1	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1	1	1	1	1	1	0	1
1	1	1	0	0	0	0	1	1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1

Asynchronous Vs Synchronous Sequential Circuits

- In a clocked sequential circuit a change of state occurs only in response to a synchronizing clock pulse. All the FLIP-FLOPs are clocked simultaneously by a common clock pulse. In an asynchronous sequential circuit, the state of the circuit can change immediately when an input change occurs. It does not use a clock.
- In clocked sequential circuits input changes are assumed to occur between clock pulses. The circuit must be in the stable state before next clock pulse arrives.

  In asynchronous sequential circuits input changes should occur only when the circuit is in a stable state.

Asynchronous Vs Synchronous Sequential Circuits

- In clocked sequential circuits, the speed of operation depends on the maximum allowed clock frequency.
  Asynchronous sequential circuits do not require clock pulses and they can change state with the input change. Therefore, in general the asynchronous sequential circuits are faster than the synchrous sequential circuits.
- In clocked sequential circuits, the memory elements are clocked FLIP-FLOPs.
  In asynchronous sequential circuits, the memory elements are either unclocked FLIP-FLOPs (latches) or gate circuits with feedback producing the effect of latch operation.
- In clocked sequential circuits, any number of inputs can change simultaneously (during the absence of the clock).

  In asynchronous sequential circuits only one input is allowed to change at a time in the case of the level inputs and only one pulse input is allowed to be present in the case of the pulse inputs. If more than one level inputs change simultaneously or more that one pulse input is present, the circuit makes erroneous state transitions due to different delay paths for each input variable.

Applications of Asynchronous Sequential Circuits

- An asynchronous circuit is preferred over synchronous circuit when high speed of operation is required since asynchronous sequential circuits respond immediately whenever there is a change in any input variable without having to wait for a clock pulse.
- Asynchronous sequential circuits are useful in applications in which input signals may change at any time.
- Asynchronous sequential circuits cost less than the sequential circuits, therefore, for economical reasons, they find useful applications.

Asynchronous Sequential Machine Modes

There are two modes of operations of asynchronous sequential machines depending upon the type of input signals. These are:

- Fundamental Mode
- Pulse Mode

End
of
Unit-3

Thank You